VARIABLES IN NATURAL LANGUAGE: WHERE DO THEY COME FROM?

Jaroslav Peregrin[*]
www.cuni.cz/~peregrin

[*Variable-Free Semantics* (ed. M.Boettner and W.Thümmel), Secolo, Osnabrück, 2000, 46-65]

0. Introduction

What is a variable and in which sense can we say that natural language contains variables? Inspecting the *Oxford Advanced Learner's Dictionary* we learn that a variable is a 'variable thing or quantity'. *The American Heritage Dictionary of the English Language* tells us that a variable is (1) 'something that varies or is prone to variation'; (2) within astronomy, 'a variable star'; and (3) within mathematics, 'a quantity capable of assuming any of a set of values' or 'a symbol representing such a quantity'. *Penguin's Dictionary of Mathematics* states that a variable is (1) 'a mathematical entity that can stand for any of the members of a set' and (2) 'an expression in logic that can stand for any element of a set (called the domain) over which it is said to range'. *The Oxford Dictionary of Philosophy* gives no explicit definition, but explicates a variable informally as something which can replace a word in a sentence and which can then be seen as 'pointing' at different members of a domain.

From this mini-recherché two senses of 'variable' seem to emerge a narrower sense, in which a variable is something which inhabits the realms of logics and mathematics, and a wider sense in which a variable is simply anything that varies. It seems also clear that this latter concept of variable is wide to the point of not being a concept at all (for surely *any* thing can be seen as somehow varying, i.e. changing, evolving, or displaying varying aspects etc.), and therefore the only concept we really have is the narrower one. This means that we should see variables as primarily a matter of logical and mathematical calculi.

Does this mean that it makes no real sense to speak about variables in connection with natural language? Of course not: natural language is in various respects similar to formal calculi; and formal calculi are employed, in various ways, to regiment, analyze or explicate it. It might therefore be both possible and reasonable to transfer the concept from the latter to the former. To find out whether and in which sense this is justified, we must investigate (i) the role of variables within logical calculi, and (ii) the ways in which we can, or should, see natural language through the prism of such calculi.

1. Variables within logic

1.1 The birth of variable

The concept of variable, as nowadays employed within logic and mathematics, is inseparably connected with the concept of quantifier developed by Gottlob Frege and his followers. Frege, when introducing quantifiers, says roughly this: Imagine a sentence decomposed into two

[*] The author is grateful to Vladimír Svoboda for helpful criticism.

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

> In dem Ausdrucke eines Urtheils kann man die rechts von $\vdash$ stehende Verbindung von Zeichen immer als Funktion eines der darin vorkommenden Zeichen ansehen. *Setzt man an die Stelle dieses Argumentes einen deutschen Buchstaben, und giebt man dem Inhaltsstriche eine Höhlung, in der dieser selbe Buchstabe steht, wie in*
>
> $$\vdash\!\overset{a}{\smile}\!- X(a)$$
>
> *so bedeutet dies das Urtheil, daß jene Function eine Thatsache sei, was man auch als ihr argument ansehen möge.*[2]

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), he writes:

> In dem Ausdruck erkennen wir die Funktion dadurch, daß wir ihn zerlegt denken; und eine solche mögliche Zerlegung wird durch seine Bildung nahe gelegt. ... Wenn ich nun z.B. sage „die Funktion $2\bullet x^3+x$", so ist x nicht als zur Funktion gohörig zu betrachten, sondern dieser Buchstabe dient nur dazu, die Art der Ergänzungsbedürftigkeit anzudeuten, indem er die Stelle kenntlich macht, wo das Zeichen des Arguments einzutreten hat[3].

---

[1] This consideration has later come to be seen as ambiguous between the 'substitutional' and the 'objectual' version. According to the 'substitutional' version of the story, we fill the gap literally: we replace the variable by suitable expressions, thus saturating the matrix 'syntactically'. According to the 'objectual' version, on the other hand, we make the variable stand for 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', 'Clinton', ...) and thus turning the the matrix into various sentences ('Caesar conquered Gaul', 'Aristotle conquered Gaul', 'Clinton conquered Gaul', ...), whereas the 'objectual' one would consist in making *x* stand for various individuals (Caesar, Aristotle, Clinton, ...) and thus making the matrix express various propositions (that Caesar conquered Gaul, that Aristotle conquered Gaul, that Clinton conquered Gaul, ...).

[2] „In the expression for a judgement, the complex symbol to the right of $\vdash$ 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.:*

$$\vdash\!\overset{a}{\smile}\!- X(a)$$

This signifies the judgement that the function is a fact whatever we take its argument to be."

[3] „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 $2\bullet x^3+x$', x must not be considered as belonging to the function; this letter only serves to indicate the kind of

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 no names and thus do not denote anything)[4] 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).

1.2 Bound variables

Let us now restrict our attention to *bound* variables, i.e. to 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. A particularly instructive way of reformulating the syntax of the first-order predicate calculus enabling us to rid ourselves of variables was presented by Quine (1960); and this way can be generalized to higher-order calculi as well (for second-order predicate calculus, this has actually been carried out by Došen, 1988). Indeed, that predicate calculus of *any* order can be articulated without variables follows from the proof of the equivalence of lambda calculus and combinatory logic (see Curry and Feys, 1957).

Why are bound variables not essential? Informally speaking, the reason is that their role is merely auxiliary, it is a role which can be played also 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 for 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 (even utilizing things evocative of real clamps, such as the arrows in Bourbaki, 1958).

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., $P(\exists)$ instead of $\exists xP(x)$ (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 $\exists(P)$). If we now allow for binary predicate

supplementation that is needed; it enables one to recognize the places where the sign for the argument must go in.“

[4] 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.

constants, the situation changes: we cannot write $R(\exists,\forall)$, for this would be ambiguous between $\exists x\forall yR(x,y)$ and $\forall y\exists xR(x,y)$. Another problem arises when quantifiers are attached to non-atomic formulas (even when these contain only unary predicate constants): $P(\exists)\wedge Q(\exists)$ would be ambiguous between $\exists x(P(x)\wedge Q(x))$ and $(\exists xP(x))\wedge(\exists xQ(x))$. However, the needed disambiguation can be carried in ways which *avoid* variables: we can, e.g., state the linear order of the quantifiers using superscripts, so that $\exists x\forall yR(x,y)$ would be $R(\exists^1,\forall^2)$, whereas $\forall y\exists x$ $R(x,y)$ would be $R(\exists^2,\forall^1)$; and similarly $\exists x(P(x)\wedge Q(x))$ would be $P(\exists^1)\wedge Q(\exists^1)$, whereas $(\exists xP(x))\wedge(\exists xQ(x))$ would be $P(\exists^1)\wedge Q(\exists^2)$.

An obvious objection is that such superscript notation would not lead to a syntax which 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; we omit functors for simplicity) give a formula.

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

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

(iv) $\exists$ ($\forall$) plus a variable plus a formula give 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) by (iv’) and to add (v) and (vi):

(iv’) $\exists$ ($\forall$) plus a predicate give a formula.

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

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

Then, writing predicates as arguments of quantifiers[5], 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 $\exists(P\&Q)$ and $\exists(P)\wedge\exists(Q)$. The situation complicates itself, of course, when we consider full (i.e. nonmonadic) first-order predicate calculus; for the presence of predicate constants of arities greater than one necessitate further rules for predicates. (Thus in this case we have, e.g., formulas such as $\exists xR(x,x)$, which must be treated as $\exists$ applied to something as the ‘reflexivization’ of the binary predicate R; so we need a rule which takes R and yields a unary predicate which applies to an individual just in case the individual stands in the relation of R to itself.). However, that all of this can be handled in an analogous vein is precisely what Quine has demonstrated.

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

[5] 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.

1.3 Free variables

Now what about free variables and open formulas? For them, the most straightforward approach appears to be to take them as mere steping stones on the way to closed formulas. Seen thus, a variable x and a formula F containing x free are only intermediaries to formulas such as ∃xF and ∀xF (or, more generally, λxF[6]); and they can be 'kicked away' when the destination is achieved. Their usefulness consists in the fact that as the steping stones they are extremely simple and elegant; and they provide for the usual seminally simple grammar of the predicate calculus. Seen thus, free variables are simply to-be-bound variables. Nevertheless what we have stated in the previous section means that bound variables are dispensable, and hence that they are better seen not as real constituents of the calculus; so if free variables are only their pre-bound stages, they are surely dispensable too.

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 logical analysis of natural language; we should ask whether open formulas are in some sense necessary for the analysis of language.

Hence the question facing us now is the following: are there natural language sentences which can be reasonably seen as counterparts of (corresponding to, adequately analyzable by means of, having as their logical forms) open logical formulas? To answer this, we must first investigate precisely in which sense a natural language sentence can be seen as a counterpart to a formula of a logical calculus, in which the former's 'logical form' can be expressed by the latter.

2. Logic and Natural Language

2.1 Origins of Logical Analysis

At the beginning of this paper we said that logical calculi can be used to regiment natural language; but more is true: logical calculi *resulted* from regimentation of natural language. Going back to Frege once more, we see that his *concept script*, a predecessor of our predicate calculus, emerged directly from his desire to bring natural language to the form where we are rid of everything which is not important from the point of view of consequence.[7]

[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'.

[7] Thus, Frege (1879, p. IV) writes: „[Die vorliegende Begriffsschrift] soll also zunächst dazu dienen, die Bündigkeit einer Schlußkette auf die sicherste Weise zu prüfen und jede Voraussetzung, die sich unbemerkt einschleichen will, anzuzeigen, damit letztere auf ihren Ursprung untersucht werden

When we look at the syntax of predicate calculus, we can see that it is based on three kinds of rules:

(i) rules of *predication*, connecting predicates with the appropriate number of terms;

(ii) rules of *logical operators*, constructing more complex statements out of simpler statements with the help of logical operators (negation, conjunction, disjunction, implication); and

(iii) rules of *quantification*, prefixing quantifiers to statements.

It is clear that the rules of the first two kinds straightforwardly reflect basic syntactic operations of natural language: these are, respectively, that we can associate a verb phrase with noun phrases to form a sentence, and that we can negate a sentence or put two sentences together by means of a connective, respectively. The rules of the third kind, however, are different, they do *not* reflect a basic syntactic structure of natural language; in fact, they stem, as we have seen, from certain *metalinguistic* considerations. We have already seen how Frege employed variables to allow him to say such things as 'when we saturate a gappy expression in certain way(s), we get certain value(s)'.

Thus, the structure of a quantificational formula (i.e. of a formula of the form *quantifier* + *variable* + *formula*) does *not* reflect the structure of a common natural language sentence, it is rather a shortcut for saying 'if we fill that gap, which in the *formula* is marked by the *variable*, with various things, we get a true sentence (at least) so many times as claimed by the *quantifier*'. Hence to say that, e.g., the formula ∃xPx is true is to say that if we keep filling the gap x in Px with various things, we gain a statement which is true at least once.

The adoption of this kind of quantificational rules has had far-reaching consequences. On one hand, it has fostered the impressive development of modern logic and mathematics which we have been witnessing; on the other hand it has caused much confusion concerning the logical analysis of natural language. Frege's choosing of this way to regiment our way of talking has caused many people to feel that quantifiers and variables are entities somehow covertly contained within natural language. But we should keep in mind that, as we have seen, it is perfectly possible (although perhaps less effective) to do Fregean logic *without* variables and variable-binding quantifiers.

2.2 Logical Form?

We have just seen that Frege's original view of the 'logical form' of an expression was something like 'that which remains of the expression if we strip off everything which is irrelevant from the viewpoint of consequence'. However, soon Frege himself (and then especially his followers, notably Russell) concluded that the logical form is something which may differ wildly from the surface form and that logical analysis is thus a far more intricate enterprise than simple 'stripping off'. (Russell's famous paper 'On Denoting' is particularly instructive from this viewpoint: there the author tries to demonstrate that the logical form of

könne. Deshalb ist auf den Ausdruck alles dessen verzichtet worden, was für die Schlußfolge ohne Bedeutung ist." ("That is why I decided to forego expressing anything that is without significance for the inferential sequence. Its [of the present ideography] first purpose, therefore, is to provide us with the most reliable test of the validity of a chain of inferences and to point out every presupposition that tries to sneak in unnoticed, so that its origin can be investigated.")

sentences containing descriptions differs radically from what the surface of the sentences would suggest.[8])

This has lead to the view that logical form is something buried deep inside each expression and waiting for a 'logical analysis' to dig it out and bring it to light. Accepting this picture seems to offer a suitable basis for saying that 'natural language contains variables': if the logical form of an expression contains variables, then (as the form seems to be something really present 'inside' the expression) the expression itself can be said to contain variables. However, we are going to indicate that this whole picture of logical form as a definite thing extractable from inside an expression is dubious. (Moreover, it follows from the above considerations that even if we accepted this picture, or some weakened version of it, expressions could be seen as containing variables only if they had *open* formulas as their logical forms.)

To see the dubiousness of the whole picture we must consider the general question of what counts as a criterion for being a logical form of a given sentence. What justifies us in saying that the logical form of a sentence S is given by a formula F? A necessary condition seems to be that F must capture the truth conditions of S. But if this were also a sufficient condition, it would mean that an expression has infinitely many logical forms: for if F captures the truth conditions of S, then so do all formulas logically equivalent to it, notably $F \wedge F$, $F \wedge F \wedge F$ etc. This means that if we want to have logical form as a unique thing extractable from the expression, we must add some other necessary condition.

An obvious candidate seems to be the condition that the logical form of S is rendered by the 'simplest' or 'least redundant' of those formulas which capture the truth conditions of S. However, even if these vague notions could be made precise enough within a given logical calculus, it is hard to see how they could make sense *across* logical calculi: is the Russellian analysis of *A man walks*, namely $\exists x(\mathbf{man}(x) \wedge \mathbf{walks}(x))$, 'simpler' than that which is offered by, say, the theory of generalized quantifiers, namely $(\mathbf{a}(\mathbf{man}))(\mathbf{walk})$[9]?

An improvement may be (and I think indeed is) to speak about 'closeness to surface form' instead of about 'simplicity' or 'nonredundancy': to say that the logical form of S is that of the formulas capturing the truth conditions of S whose structure is the closest to the (surface) structure of S. However, it seems to be clear that if we employ a logical calculus with a sufficiently rich syntax, we can make the analysis as close to the surface as we desire, and this would mean that there is no nontrivial concept of logical form (as opposed to surface form) after all. (And as it would render every two sentences differing on the surface as differing in logical form, it would apparently contradict the basic intuition behind the concept of logical form.)

The moral of these considerations seems to be the following: we may regiment *A man walks* as $\exists x(\mathbf{man}(x) \wedge \mathbf{walks}(x))$ (which has the advantage of utilizing only the simple and perspicuous apparatus of first-order logic), or we may regiment it as $(\mathbf{a}(\mathbf{man}))(\mathbf{walk})$ (which is closer to the surface, but logically more involved), or we may regiment it, say, within dynamic logic as $\exists_\mathbf{D} x(\mathbf{man}(x) \wedge_\mathbf{D} \mathbf{walks}(x))$[10] (which captures some semantic aspects of the analyzed sentence, which are ignored by the 'static' analysis, which are nevertheless not obviously a matter of truth conditions), or as $(\mathbf{walk})(\mathbf{a_D}(\mathbf{man}))$[11] (which does the same as the

[8] See Russell (1905).
[9] See Barwise & Cooper (1981).
[10] See Groenendijk & Stokhof (1991); the indices indicate that the indexed operators are not the ones of standard logic, but rather their dynamic counterparts.
[11] See Peregrin & von Heusinger (1995).

previous one while being again closer to the surface). Each of these analyses may be useful for some purposes, but none of them is *the* analysis (there also seems to be a certain trade-off between exhaustiveness of analysis and simplicity and prosperousness of the means employed). Hence there seems to be nothing as *the* logical form, but rather only various ways to envisage truth conditions (or perhaps some more general semantic features)[12].

However, what if we accept that logical form may be nothing over and above surface form, and claim that *this* structure sometimes corresponds to an open formula? What if we argue that there are some expressions, e.g. pronouns, which behave expressly like free variables? Two kinds of objections to such a view can again be raised. First and foremost, it is not entirely clear what is meant, in this context, by 'behave like a variable'. We have seen that the original role of a variable was to mark a place within a gappy expression - and this is a *metalinguistic role*, a role within a certain enterprise of decomposing expressions and reasoning *about* them (especially examining what happens when we fill their gaps in various ways). And this does not seem to be a role meaningfully ascribable to an expression of natural language.

So what about the more general characterization of variable, like: that which can stand for any element of a domain? Does natural language not comprise expressions which behave in this way? Does natural language not contain expressions which are said to have *variable* or *distributive reference*? Here the trouble is that to classify a natural language expression as something 'which can stand for any element of a domain', we would require the fixed relation of *standing for* or *reference* - and elsewhere I have argued at length that the relation of reference is too tricky to be taken as an 'unexplained explainer' (see Peregrin, 1995, Chapter 8; and especially Peregrin, to appear).

The second trouble is that even if we grant that some elements of natural language, e.g. pronouns, stand variably for different things, it is hard to deny that in other respects they are often very much *unlike* variables. Take the sentence *He walks*. In which respects is it like the open formula **walk**(x) (or taking into account the gender of *he*, **walk**(x)∧**man**(x))? I am afraid that only in a single one: in that its subject is not a name (an individual constant). For when one says *he walks*, he surely does *not* invite the hearers to substitute every (or any) conceivable individual for *he*. *He walks* is always synonymous with a sentence containing a name (or at least a description) in place of *he* (if I say *he walks* pointing at Salman Rushdie, it is synonymous with *Salman Rushdie walks*), albeit it is synonymous with different sentences on different occasions. In contrast, **walk**(x) is clearly equivalent with no formula of the shape **walk**(*c*) with constant *c*; and the context in which it occurs play no role with respect to its being equivalent to other formulas.

A different way to salvage the absolutist notion of logical form may appeal to the analyses of the human 'language faculty' as provided by Noam Chomsky. Chomsky's theories seem to imply that logical form *is* something 'tangible', which in some sense really exists and can be *depicted*. However, a little reflection reveals that the Chomskian notion of logical form has little to do with logic and hence with the realm where we have concluded the nature of the concept of variable should be sought. Chomsky's *logical form* is one of the 'levels of representation' which constitute his reconstruction of 'human language faculty', a level which constitutes, as he puts is, the interface between language and other cognitive systems (see, e.g., Chomsky, 1986). Chomsky claims that his logical form „does in fact have many of the notational properties of familiar logical form, including the use of quantifier-

[12] As Quine (1972, p.453) puts it: „Logical analysis ... may go one way or another depending on one's specific logical purpose."

variable notation", that this is, however, a *contingent* fact. (ibid., 156)[13]. Moreover, Chomsky defines *variable* as a certain species of what he calls the empty category, namely an „$\overline{\mathrm{A}}$-bound r-expression that must have Case by the visibility condition" (ibid., 164). I think it is quite clear that these concepts of logical form and of variable have nothing to do with the Fregean, and hence with the principal logical, ones.[14]

3. Variables as Means of 'Functional Classification'

3.1 Variables and Rules of Composition

The moral of the previous chapters is that from the viewpoint of analysis of natural language, variables are, in principle, dispensable. However, if we look at the way logical analysis is usually being carried out, what we see is that they are, in fact, far from dispensed with; and this indicates that despite of being dispensable, variables must be remarkably useful. In this chapter we are going to examine what this usefulness consists in, and we are going to try to indicate how the role of variables can be easily misunderstood.

Elsewhere I claimed that variables are best seen as „auxiliaries helping us to codify complicated rules" (see Peregrin, 1995, p. 103). Here I want to elaborate on this point: variables, I am going to show, are useful tools to characterize complex rules of composition (especially grammatical rules of linguistic systems), and also to specify the behavior (functioning) of some elements of these systems with respect to the rules. I claim that this is what creates the illusion of variables being 'contained' within natural language expressions: we use variables to articulate the linguistic, and especially semantic, functions of expressions, and mistake the articulations for expositions of the expressions' 'insides'. However, variables are no more inside expressions than holes for screws are inside screwdrivers.

To justify this thesis, we must first articulate a general framework to talk about objects and their components. So let us imagine a realm of objects some of which are parts of others. The realm can be clearly seen as a set, call it R, and the part-whole relationships can be articulated in terms of a family $<O_i>_{i\in I}$ of operators over R which represent the 'ways of composing' elements of R into other elements of R. This means that $e=O_i(e_1,...,e_n)$ renders the fact that e is composed of $e_1,...,e_n$ (in the way O_i). Seen thus, the part-whole system can be considered as the ordered pair $<R, <O_i>_{i\in I}>$ and hence as the (partial) algebra whose carrier is R and whose operators are O_i. Alternatively we can imagine the elements of R as categorized into *sorts* $<R_j>_{j\in J}$ according to their behavior w.r.t. the ways of composition, and also imagine the ways of composition $<O_i>_{i\in I}$ refined into $<O_i'>_{i\in I'}$ so that each O_i' is from the Cartesian product of some sorts into a sort; and we can thus see the system as the *many-sorted* algebra $<<R_j>_{j\in J}, <O_i'>_{i\in I'}>$ (where for each O_i' there exist $j_1,...,j_m,j_{m+1}$ so that the domain $D(O_i')$ of O_i' equals the Cartesian product $R_{j1}\times...\times R_{jm}$ and the range $R(O_i')$ of O_i' is included

[13] Personally, I find it hard to see the positive contents of this claim: for what does it mean, in this context, to have 'notational properties'?

[14] Unfortunatelly many people do *not* realize that the *logical form* of logicians and that of Chomsky are two essentially different things, and that if we use the same word to refer to both of them, it is nothing more than a pure homonymy. I think that not seeing this and merging logical analysis with Chomskian linguistics is tantamount to aiming at theories akin to such as would result from merging metallurgy and biology into a unified theory of 'nails'.

in R_{jm+1})[15]. If O is an operator of a many-sorted algebra A and R is a sort of A, then we shall say that *O takes (elements of) R (as its ith arguments)* iff *D*(O) = $R_1 \times ... \times R_{i-1} \times R \times R_{i+1} \times ... \times R_n$ for some sorts $R_1,..., R_{i-1},R_{i+1},...,R_n$ of A; and if O takes elements of R as its ith arguments and does not take elements of R as its jth arguments for any j≠i, then we shall say that O takes (elements of) R *uniquely*.

Language is one of the important things which can be seen as part-whole systems and hence as partial or many-sorted algebras: the carrier of the algebra is constituted by the expressions, and the operators are the grammatical ways of putting expressions together to yield more complex expressions (in the case of language, we shall thus sometimes speak about *rules* instead of about operators). For the sake of illustration, let us consider a very simple language L_0 containing names, unary predicates, and sentences each of which is the concatenation of a name and a predicate. If N is the set of names, P the set of predicates, S the set of sentences, and PRED the operation of concatenating a name with a predicate into a sentence (thus PRED(*n*,*p*)= $n^\frown p$, where $n^\frown p$ symbolizes the concatenation of *n* and *p*[16]), then we can see L_0 as the partial algebra <N∪P∪S,<PRED>>; or, better, as the many-sorted algebra <<N,P,S>,<PRED>>, where PRED is from N×P into S. According to our definitions, PRED takes names as first arguments, and it takes predicates as second arguments; it takes both names and predicates uniquely.

Let us also consider the extension L_1 of L_0: L_1 contains names, predicates and sentences like L_0, but in addition to L_0 it contains binary sentential connectives and such sentences which consist of two sentences linked by a connective. This means that L_1 can be seen as the many-sorted algebra <<N,P,C,S>,<PRED,CON>>, where CON is from S×C×S into S and CON(s_1,c,s_2)= $s_1{}^\frown c^\frown s_2$. CON takes sentences as its first arguments and also as its third arguments; so it does *not* take sentences uniquely. Let us further assume that CON contains the expression ‘and’.

Now given the rules of L_1, we can take a name *n* and two predicates p_1 and p_2 and form a sentence consisting of two subsentences, $n^\frown p_1$ and $n^\frown p_2$, connected by ‘and’. We can do this by first applying PRED to p_1 and *n*, then PRED to p_2 and *n*, and then CON to ‘and’ and the results of the previous two operations. Thus, there is a (‘complex’) operator which takes a name *n* and two predicates p_1 and p_2 into the sentence $n^\frown p_1{}^\frown$‘and’$^\frown n^\frown p_2$. The existence of this complex rule is in a sense implicit to the existence of PRED and CON (and ‘and’). Now to characterize this complex operator, we have to say something like: *first, PRED is applied to the first predicate and the name, then PRED is applied to the second predicate and the name, and then CON is applied to ‘and’ and the results of the previous two operations*; or better to say, as we did, *first PRED is applied to p_1 and n, then PRED is applied to p_2 and n, and then CON is applied to ‘and’ and the results of the previous two operations*. When we

[15] For the concept of many-sorted algebra and for its applications to natural language see Janssen (1983). Of course not every algebra could be reasonably seen as amounting to a part-whole system: the relation of *being a proper part* is essentially *acyclic* (i.e. its transitive closure is antisymmetric); and hence if an algebra is to be seen as (amounting to) a part-whole system, it has to satisfy this restriction. Thus, the operators of, e.g., an algebra in which, for some elements *a*, *b*, *c*, *d* and some operators O and O’, *a*=O(*b*,*c*) and *b*=O’(*a*,*d*) cannot be reasonably seen as ‘rules of composition’ - for this would mean that we would have to see *b* as a proper part of *a* and in the same time *a* as the proper part of *b*.

[16] As we want L_0 to be English-like, we in fact assume that PRED is more than mere concatenation, that it modifies the predicate in the appropriate way (in the simplest case by appending the suffix ‘-s’). However, for the sake of simplicity we shall speak simply about concatenation.

choose the latter way of articulation, we need some symbols to regiment the expressions ’the first predicate’, ‘the second predicate’ and ‘the name’ of the former articulation. If we employ such *variables*, we can directly designate the complex operator by some self-explicating schema like CON(PRED(n,p_1),’and’,PRED(n,p_2)). This means that in this context we employ variables to derive designators of complex operators from those of basic operators (plus, as the case may be, those of elements).

Complex operators which are in this way implicit to the operators of an algebra A are sometimes called *polynomials over A*. Polynomials arise out of composing and iterating the operators of A (and some trivial operators, namely projections, i.e. functions mapping n-tuples on their ith constituents). The operator described in the previous paragraph, CON(PRED(n,p_1),’and’,PRED(n,p_2)), is a polynomial over L_1. To say that operators of an algebra are closed under forming polynomials is to say that the operators include projections and that they can be composed and iterated (and it seems that the operations of forming wholes from parts should be closed in this sense).

Now if we look at a book where polynomials are defined (Grätzer, 1979, Janssen, 1983), what we see is that the employment of some kind of variables is quite essential to the whole enterprise. However, in this context, variables are simply tools employed to form self-explicating designators for complex operators or rules. In general, if we accept composibility of operators (and the existence of projections) as a general principle, we accept that the existence of any family of operators involves the existence of all polynomials based on the family; and variables are indispensable tools of articulating canonical names for the polynomials. Notwithstanding this though, we must realize that here we are using variables on the *meta*level, we use them to talk *about* language, more precisely about rules of language. There is thus no question of these variables being ‘inside’ expressions.

3.2 Variables and Expressions’ Functioning

Let us now examine a part-whole system from the viewpoint of the ‘behavior’ of its elements. The behavior clearly consists in the ways in which the elements, together with other elements, constitute compounds.

To specify the behavior of an object it is necessary to summarize all cases of composition into which it can enter. Let us articulate this idea formally: let A be a many-sorted algebra, a an element of a sort R of A, and O an operator of A which takes R as its ith argument, i.e. such that there are some sorts $R_1,...,R_{i-1},R_{i+1},...,R_n,R_{n+1}$ of A such that D(O) = $R_1\times...\times R_{i-1}\times R\times R_{i+1}\times...\times R_n$ and R(O) = R_{n+1}. The *(O,i)-trace* of a will be the function $f^{O,i,a}$ from $R_1\times...\times R_{i-1}\times R_{i+1}\times...\times R_n$ into R_{n+1} such that if $<a_1,...,a_{i-1},a,a_{i+1},...,a_n>\in D(O)$ for some $a_1,...,a_{i-1},a_{i+1},...,a_n$, then $a_1,...,a_{i-1},a_{i+1},...,a_n\in D(f^{O,i,a})$ and $f^{O,i,a}(a_1,...,a_{i-1},a_{i+1},...,a_n)$ = $O(a_1,...,a_{i-1},a,a_{i+1},...,a_n)$. If O takes a uniquely, then we shall also speak about the O-trace of a instead of about its (O,i)-trace. The O-trace of a thus in a sense characterizes ‘the behavior of a with respect to O’; and all the traces of a w.r.t. all operators which take a characterize the behavior of a w.r.t. the whole system.

Now the only aspect of an item which is relevant *from the viewpoint of a system* is clearly its behavior w.r.t. the system; thus, from this viewpoint, if we are able to capture the behavior as an object (function), we could well deal directly with the behavior instead of with the element itself. Let us assume that an element is uniquely determined by its behavior, i.e. that there are no two elements with exactly the same behavior. A particularly instructive case obtains when items of a sort are taken by a single rule and are taken by it uniquely; or if, more generally, the behavior of the items w.r.t. all the rules which take them is uniquely determined by their behavior w.r.t. the single rule. This means: Let O be an operator and R a sort taken uniquely by O and such that any two elements of R sharing the same O-trace are identical (and hence share the same O'-trace for every O' which takes them). In this case, an element of R is uniquely determined by its O-trace; and then we can justifiably identify it with its O-trace.

Now suppose that we do this for all elements of R; that is, for every $a \in R$, we identify a with $f^{O,a}$. Then obviously for every $<a_1,...,a_n> \in D(O)$, $O(a_1,...,a_n) = a(a_1,...,a_{i-1},a_{i+1},...,a_n)$. Hence we are, in a sense, 'delegating' the working of O to the objects of the sort R; the work of the new O is merely to bring the capacities of these objects to bear on their fellow-arguments. Such 'functionalization' of items of a sort thus deprives the rule involved of all its substantial content rendering it a purely formal 'applicator' - all content is localized in the objectual form. Note also, that the function $f^{O,a}$ coincides with (the course of values of) the polynomial $O(x_1,...,x_{i-1},a,x_{i+1},...,x_n)$: the domain of both of them is $R_1 \times ... \times R_{i-1} \times R_{i+1} \times ... \times R_n$, their common range is R_{n+1}, and for every $a_1,...,a_{i-1},a_{i+1},...,a_n$ from the common domain, the value of both is $O(a_1,...,a_{i-1},a,a_{i+1},...,a_n)$. So by identifying a with its behavior, we are identifying it with a certain polynomial operator; and since we have found variables indispensable to articulate polynomials, we now see that they may be useful to articulate objects' behavior.

Let us return to our example part-whole system L_0. The behavior of the name n is characterized by its PRED-trace, i.e. by the function $f^{PRED,n}$ such that for every predicate p, $f^{PRED,n}(p) = PRED(n,p)$ ($= n^\frown p$). Similarly, the behavior of a predicate p is characterized by the function $f^{PRED,p}$ such that for every name n, $f^{PRED,p}(n) = PRED(n,p)$. Suppose now that we identify names with their 'behaviors': i.e. that we identify each n with its trace $f^{PRED,n}$. The operator PRED then becomes the application of a name to a predicate - since for every such new name n and every predicate p, $PRED(n,p) = n(p)$. Alternatively, we could identify predicates (instead of names) with their behaviors, i.e. each p with its trace $f^{PRED,p}$, which would result in PRED becoming the application of a predicate to a name.

Now imagine that we replaced *both* names and predicates with their behaviors: then, clearly, $PRED(n,p)$ would be neither $n(p)$, nor $p(n)$ (for then names would be functions from P into S, predicates would be functions from N into S, leaving none of them within the domain of the other). This indicates that optimally we might 'functionalize' only *one* of the sorts. The reason for this, informally speaking, is that the working of PRED can be delegated either to names, or to predicates[17], but to delegate it to both at once is to duplicate it to a harmful effect. Thus, if our aim is to shift as much of functioning as possible from rules to elements, our best course is to find for each rule its own sort to whose elements the working of the rule is delegated. (Whether we can thereby manage to 'empty' *all* rules depends on the structure of the system[18]).

[17] In fact, traditional logic has it in the latter way, while, e.g. Montague Grammar exploits the former one.

[18] The necessary condition is that there exists an injective mapping M of operators on sorts such that for every operator O, O takes M(O) and takes it uniquely. However, this will not suffice: in order to avoid circularity in the definition of functions, there has to exist a well-ordering of sorts such that for every operator O, the sort M(O) is strictly greater than all other sorts taken by O. The existence of such an ordering is tantamount to the existence of a 'categorial indexing', i.e. to the possibility of

Now if an element of the system is uniquely determined by its behavior, we can say that it is *the* element which behaves *thus and so*. We can, e.g., say that a predicate, say *walk*, is *the* element of L_0 which takes ‘Peter’ into ‘Peter walks’, ‘Mary’ into ‘Mary walks’ etc.; i.e. that it behaves in the way $f^{PRED,\text{'walk'}}$. Moreover, as every predicate’s behavior coincides with the workings of a certain polynomial rule, we can specify its behavior by pointing out the rule. Thus, if we want to specify the behavior of ‘walk’, we can point out the polynomial PRED(‘walk’,*x*) (or, expanding the definition of PRED, $x^\frown$’walk’). To indicate that what is now referred to is an *object*, rather than a rule, we usually make use of something like the lambda notation: we say that ‘walk’ is the item λ*x*.PRED(‘walk’,*x*), or that it is $\lambda x.x^\frown$’walk’[19]. However, to say this is *not* to say that the word ‘walk’ contains a variable, it is rather to envisage its behavior.

Hence, the conclusion is that as we can profitably employ variables to codify complex rules (polynomials), and as the behavior of some expressions is identifiable with the working of some such rules, *variables are helpful tools for specifying expressions’ behavior, tools of the ‘functional classification’ of expressions*[20].

3.3 Semantic Interpretation

Language is not simply a system of expressions composible into more complex expressions; its crucial feature is that its expressions *have meaning*. And indeed, the relevant functional classification of expressions, which invokes variables, is usually a matter of *semantic* functioning; i.e. not only a matter of the ways in which expressions take part in constituting more complex expressions, but rather a matter of the ways in which they contribute to the meaning of the wholes in which they occur.

Within our algebraic setting, this means that we should not reconstruct language as simply an algebra of expressions, but that we should see each element of the algebra associated with a meaning (whatever it may be[21]). And as meanings are assumed to be compositional, we can see mappings of expressions on their meanings, *semantic*

indexing sorts in such a way that each operator comes to combine an element of a sort $A/B_1...B_n$ with elements of the respective sorts $B_1,...,B_n$ into an element of the sort A. See also Peregrin (1992).

[19] We have been speaking rather loosely abour ‘identifying an element with a function’, about ‘an operator becoming an application’, etc. Each such identification involves the replacement of the original algebra by *another* algebra, which is nevertheless isomorphic with the original one. What makes it possible to pass freely from the one algebra to the other is that we are investigating *structural* properties. To algebraically reconstruct a factual range of items (e.g. the range of expressions of a factual language) is to define an algebra embodying the relevant structure of the range - what is important is the structure alone, the factual nature of the elements of the algebra is no more important than the nature of the marks chosen to site towns on a map. So we can also employ such items which, so to say, wear their functioning on their sleeves - and it is precisely this kind of reconstruction which often appears most useful.

[20] The term *functional classification* is purposefully chosen to allude to the same term employed by Wilfrid Sellars (1974).

[21] I have repeatedly stressed (see esp. Peregrin, 1995) that language cannot be seen as a nomenclature, as a set of labels stuck on pre-existing real-world objects. However, once we drop the assumption that meanings are ‘real’, ‘pre-linguistic’ objects, then the picture of meanings as associated with expressions becomes harmless and is often useful.

interpretations, as homomorphisms from the algebra of expressions into an algebra of denotations[22].

Now such a *semantic interpretation* can be seen as having two parts, corresponding to the two different ways of defining the denotations, and hence explicating the semantic functions, of individual sorts of expressions. (i) The semantic functioning of expressions of some sorts, of the 'basic' ones, are 'explicated' only in an utterly trivial way - it is simply stipulated that each of these expressions has *some* denotation. This means that for the basic sorts of expressions it is simply assumed that there exist corresponding domains of denotations and that the elements of the former are mapped on those of the latter. Nothing nontrivial is said about the nature of the domains and their elements. (ii) The semantic functioning of expressions of the other, 'nonbasic' sorts is then explicated relatively to the functioning of those of the 'basic' ones. This is to say, the denotations of expressions of the nonbasic sorts are identified with their behviors, i.e. with the ways in which they constitute, together with the denotations of expressions composible with their ones, denotations of the resulting compounds.

To see this working, let us consider L_0 once more. The usual way to define semantics for a language of this kind is to assume a domain U of *individuals* and a domain B of *propositions* (which might be, in the simplest case, the two truth values), to map each name on an element of U, each predicate on a function from U to B (we can call such functions *properties*), and each sentence consisting of a name *n* and a predicate *p* on that element of B which arises out of the application of the function denoted by *p* on the denotation of *n*. This is to say that the semantic interpretation of L_0 is a homomorphism from L_0 into the algebra D_0=<<U,[U→B],B>,<APPL>>, where [U→B] is the set of functions from U to B and APPL takes an element from U and an element from [U→B] into the value of the application of the latter to the former (APPL(x,y)=y(x)). Thus, N and S are treated as semantically 'basic' categories of L_0: their expressions are simply taken to denote some elements of the corresponding domains U and B. In contrast to them, the semantic function of the elements of P is really explicated - albeit only relatively to the semantic functions of the elements of the basic categories.

Passing from L_0 to L_1, we add another non-basic category: its elements are semantically interpreted by being mapped on functions from B×B to B (call such functions *propositional junctors*) so that the denotation of a sentence consisting of two sentences connected by the connective results from the application of the function denoted by the connective to the denotations of the two contained sentences. The denotation-algebra is now D_1=<<U,[U→B],B,[B×B→B]>,<$APPL_1$,$APPL_2$>>, where $APPL_1$ is from U×[U→B] into B such that $APPL_1(x,y)=y(x)$, and $APPL_2$ is from B×[B×B→B]×B into B such that $APPL_2(x,y,z)=y(x,z)$.

Now let us consider the operation taking an individual *i* and two properties r_1 and r_2 into a proposition that the individual has both the properties. This operation consists in first applying $APPL_1$ to *i* and r_1, then applying $APPL_1$ to *i* and r_2, and finally applying $APPL_2$ to the conjunction function and the result of the previous two operations. If we denote the conjunction function by **&**, the operator is the polynomial $APPL_2(APPL_1(i,r_1),\mathbf{\&},APPL_1(i,r_2))$. Allowing the explicit names of the application operators to give way to the usual bracket notation gives us $\mathbf{\&}(r_1(i), r_2(i))$; and by writing **&** in the usual infix way, we have $r_1(i)$ **&** $r_2(i)$. Now if we consider the semantic interpretation of L_1 in D_1 and assume that 'and' denotes **&**, we see that this polynomial operator over D_1 is the semantic

[22] See Janssen (1983, Chapter I).

counterpart of the polynomial CON(PRED(n,p_1),'and',PRED(n, p_2)) over $L_{1;}$ we can say that the former embodies the semantic function of the latter (or that the latter expresses, or represents, or stands for, the former).

The same applies to the elements of the algebras. That kind of function of an element of L_1 which was discussed in the previous section can be called its *syntactic* function, whereas its *semantic* function is actually the function of its denotation within the algebra D_1; and since the denotation has been devised precisely so as to directly coincide with its function, it is this denotation itself. We have seen that the syntactic function of the predicate 'walk' is the function λx.PRED(x,'walk') (= $\lambda x.x^\frown$'walk'). The denotation of 'walk' is now the function which takes the individual denoted by the name 'Peter' into the proposition denoted by the sentence 'Peter walks', the individual denoted by the name 'Mary' into the proposition denoted by the sentence 'Mary walks', etc. If we designate that element of D_1 on which an expression of L_1 is mapped by the italicized form of the expression (so that the individual denoted by the name 'Peter' is *Peter*, the proposition denoted by the sentence 'Peter walks' is *Peter walks* etc.), we can say that the denotation of 'walk' is the item *walk* such that *walk*(*Peter*) = *Peter walks*, *walk*(*Mary*) = *Mary walks*, etc.; hence, in other words (or rather signs), the denotation of 'walk' is $\lambda x.walk(x)$. Thus, while the *syntactic* function of 'walk' (= its behavior w.r.t. the algebra L_1) is $\lambda x.x^\frown$'walk', its *semantic* function (= the behavior of its denotation w.r.t. the algebra D_1) is $\lambda x.walk(x)$.

Saying that the semantic function of 'walk' is $\lambda x.walk(x)$, however, has only little informative value: $\lambda x.walk(x)$ is nothing else than *walk*, and saying that the denotation of 'walk' is *walk* is nothing more than saying that the denotation of 'walk' is that element of the denotation algebra which is denoted by 'walk'. To see how a semantic function can be pointed out in a less trivial way, let us consider the extension L_2 of L_1. L_2 has the same carrier as L_1, but in addition to the operators of L_1 it contains the operator PCON taking a predicate, a connective, and a predicate into a predicate: thus if p_1 and p_2 are predicates and c a connective, then PCON(p_1,c,p_2) is a predicate; and thus if n is a name, then PRED(n,PCON(p_1,c,p_2)) is a sentence. Hence, PCON takes, e.g., the expressions 'walk', 'and' and 'whistle' into the complex predicate 'walk and whistle', which can then be combined with 'Peter' into 'Peter walks and whistles'. In order to extend a semantic interpretation of L_1 in D_1 to the interpretation of L_2, we must extend D_1 with an operator which would be the semantic counterpart of PCON - let us call it PCON*. It seems to be natural to require that 'Peter walks and whistles' denote the same proposition as 'Peter walks and Peter whistles'[23]; that is, to require that for every n, p_1, c and p_2, the denotation of PRED(n,PCON(p_1,c,p_2)) coincides with that of CON(PRED(n,p_1),c,PRED(n,p_2)). If this is so, then for every individual i, every properties r_1 and r_2, and every propositional junctor o, PCON*(r_1,o,r_2)(i) = $r_1(i)\ o\ r_2(i)$. That is, the predicate PCON*(r_1,o,r_2) takes an individual i into the proposition $r_1(i)\ o\ r_2(i)$; hence PCON*(r_1,o,r_2) = $\lambda i.\ r_1(i)\ o\ r_2(i)$. This means that, e.g., PCON*(*walk*,**&**,*whistle*) (= the denotation of PCON('walk','and','whistle'), i.e. the denotation of 'walk and whistle') is $\lambda i.\ walk(i)$ **&** $whistle(i)$[24]. In this way we describe certain elements of the denotation algebra, and thereby the semantic functions of certain elements of the expression algebra, by means of variables. We take for granted the denotations of the

[23] It certainly is natural if we assume that 'denote the same proposition' is to explicate 'have the same truth conditions'.

[24] Remember that '**&**' was introduced earlier as a name for what we now see as the denotation of 'and', i.e. for *and*.

basic predicates and connectives (‘walk’, ‘whistle’, ‘and’), and we explicate the denotations of, e.g., complex predicates (‘walk and whistle’) in their terms.

Therefore we are using variables to designate the functioning of the elements of the denotation algebra, and as we also take the elements of the expression algebra to be endowed with meanings via being mapped on the elements of the denotation algebra, we thus use variables to explicate their meaning. However, by saying that ‘walk and whistles’ means (or denotes, or stands for, or expresses, or whatever) $\lambda i.\ \mathit{walk}(i)\ \mathbf{\&}\ \mathit{whistle}(i)$ we are not saying that ‘walk and whistles’ *contains* the variable i: what we say is that the semantic behavior (= meaning) of the expression is describable in terms of the result of its application to individuals.

3.4 Composing and Decomposing Constructions

Now let us take an even more abstract vantage point: let us think about constructing entities in general. We construct complex entities by putting simpler entities ‘together’: within the physical world (as when we construct a chair out of logs and nails), or within the world of abstract entities (as when we construct the sum out of two numbers). The ways of ‘putting together’ are ‘domain-specific’: when we construct a chair, then they consist in certain fastenings of objects one to another with the help of other objects; when we construct a number, they consist in adding, multiplying and the like.

However, there seem to be some quite general principles governing *all* kinds of constructings. It seems that it is in the nature of the concept of *constructing* that constructions can be composed and iterated, and that the trivial constructions, like choice, are generally available. This is reflected by the assumption that by explicitly accepting a set of basic constructions we implicitly accept all the polynomials based on them. The idea is that if we are able to construct x out of y and y out of z, then we are surely able to construct x out of z; more generally, if we are able to construct x out of $x_1,...,x_n$ and if we are able to construct x_i out of $x^i_1,..., x^i_{mi}$ (for every i), then we are able to construct x out of $x^1_1,...,x^1_{m1},..., x^n_1,...,x^n_{mn}$. In short, it seems to be in the nature of constructing that constructions are composible (and hence closed under forming polynomials). Thus, addition of polynomials only makes explicit what is already implicit. In algebraic terms, this means that an algebra A^+ which is the polynomial extension of a given algebra A (i.e. which differs from A in that its family of operators contains some operators which, while not operators of A, are nevertheless polynomial over A) amounts to *the same* constructional system as A.

Janssen (1983) has proven the following theorem: if A and B are many-sorted algebras, h a homomorphism from A to B, and A^+ a polynomial extension of A, then there is a polynomial extension B^+ of B, and a unique extension h^+ to a homomorphism from A^+ to B^+. If we assume that semantic interpretation is by its nature a homomorphism, then the theorem implies that addition of polynomial symbols is trivial also in the sense of not being able to tamper with semantic interpretation: every semantic interpretation of a language can be extended to a semantic interpretation of a polynomial extension of the language.

Now besides the fact that the space of constructions is closed under composition, we may consider its being closed under the inverse operation, *de*composition. Whereas the principle of compositibility of constructions says that we can always merge two subsequent constructional steps into one, the inverse principle of decomposition says that we can always divide a nontrivial constructional step into two subsequent simpler steps. The idea is that if we can use *x*, *y* and *v* to construct *z*, then it is also possible to, first, use *x* and *y* to construct an intermediary entity *w*, and then use *w* and *v* to construct *z*. In general, the principle of decomposition says that if we can construct *x* out of $x_1,...,x_n$ and if $\{i_1,...,i_m\}$ and $\{j_1,...j_k\}$ are disjoint sets whose union is {1,...,n}, then there is a *y* which can be constructed out of $x_{i1},...,x_{im}$ and such that we can construct *x* out of $y,x_{j1},...,x_{jk}$.

It is, of course, dubious, whether we can assume such a principle to hold generally: when we are constructing physical things, then, as experience teaches us, we cannot decompose each step as we please: if we, e.g., fasten two logs together by a string, we cannot combine one of the logs with the string into a ‘half-way construct’ ready for subsequent combination with the other log to yield the ultimate construct (the two logs tied together). However, the situation is different within the realm of abstract entities: there seems to be no reason not to accept decomposibility as a general principle. (Indeed, this fact might perhaps be one of the characterizing differences between the two realms.)

Now assume that decomposition not only always exists, but also is unique. This means that if C is a unary construction and $\{i_1,...,i_m\}$ and $\{j_1,...j_k\}$ are disjoint sets whose union is {1,...,n}, then there is a uniquely determined pair of constructions C_1 and C_2 so that $C(x_1,...,x_n) = C_2(C_1(x_{i1},...,x_{im}),x_{j1},...,x_{jk})$; and hence for any concrete objects $a_1,...,a_n$ from the domain of C there is an intermediary, ‘half-way’ construct $C_1(a_{i1},...,a_{im})$ which is constructed out of $a_{i1},...,a_{im}$ and is capable of yielding the ultimate construct $C(a_1,...,a_n)$ together with the rest of the arguments $a_{j1},...,a_{jk}$. The object $C_1(a_{i1},...,a_{im})$ can be characterized by its disposition to take its part in constructing $C(a_1,...,a_n)$ - and the plausible way to designate it seems to be something like the lambda notation.

When we consider composing operators, we need to designate complex, composed operators (polynomials), and we employ variables to help us build complex designators for these operators from the simple names of basic operators. When we consider decomposing operators, certain objects become known as the ‘half-way constructs’ and they become illuminatingly designatable as such. Thus an object may become viewable as, e.g., a ‘half-way construct’ on the way from the objects $x_1,...,x_n$ to the object $C(x_1,...,x_n)$: as that ‘half-way construct’ which arises out of exploiting $x_{i1},...,x_{im}$, but still not exploiting $x_{j1},...,x_{jk}$. Such an object is then plausibly designated by means of an expression such as ‘$\lambda x_{j1}...x_{jk}.C(x_1,...,x_n)$’.

As polynomial symbols characterize the range polynomial operators, the new symbols I proposed earlier (see Peregrin, 1992) to call *abstractive*, characterize a more general range of operators, *abstractive* operators. What I then claimed was that the addition of such abstractive operators to an algebra is still trivial in a sense analogous to that in which it is trivial to add polynomials. To see this, we must realize that *semantic interpretation*, as the term is usually understood, is not simply a homomorphism, but typically a homomorphism into an algebra of a specific kind. In the terminology of the present paper, it is a homomorphism into an algebra the operators of which are *formal*, i.e. devoid of content (which means, as explained in the previous chapter, that they only bring the capacities of some elements of the algebra to bear on some other elements). In Peregrin (1992) I called such algebras *applicative* and I have proven the following theorem: if A is an algebra, B an applicative algebra, h a homomorhpism from A to B, and A^+ an *abstractive* extension of A, then there is an extension B^+ of B, and a unique extension h^+ of h such that h^+ is a homomorphism from A^+ to B^+.

This means that by employing not only variables, but also the lambda operator, we still only make explicit the implicit semantic capacities. However, while polynomial symbols are bound to stand for operators, abstractive symbols (lambda-terms) may, in the limit case when all variables are ‘lambda-abstracted away’, stand for objects. In this way variables appear to be revealing us something about things (in particular about expressions); but what they really do is only revealing (making explicit, *explicating*) certain implicit capacities of the things' behavior.

4. Conclusion

Regimenting a natural language expression by a formula containing variables should be seen as revealing neither the variables' covert presence 'within' the expression, nor their presence 'within the expression's meaning', but rather as a *functional*, usually *semantic*, *characterization* of the expression. Variables have been introduced, and are still best seen as certain *meta*linguistic tools, as means of designating functions.

References

Barwise, J., Cooper, R.(1981): 'Generalized Quantifiers and Natural Language', *Linguistics and Philosophy* 4, 159-219.

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

Chomsky, N.(1986): *Knowledge of Language*, Praeger, New York.

Church, A. (1940): 'A Formulation of the Simple Theory of Types', *Journal of Symbolic Logic* 5, 56-68.

Došen, K. (1988): 'Second-order logic without variables', in *Categorial Grammar* (ed. W.Buszkowski, W.Marcizsewski & J. van Benthem), Benjamins, Amsterdam.

Frege, G. (1879): *Begriffsschrift*, Nebert, Halle; translated as *Begriffsschrift*, in van Heijenoort (1971), 1-82.

Frege, G.(1891): 'Function und Begriff', ein Vortrag, gehlaten in der Sitzung vom 9.1.1891 der Jenaischen Gesellschaft für Medizin und Naturwissenschaft, Jena; translated as *Function and Concept* in Geach and Black (1952), 21-41.

Geach P. and M. Black (1952): *Translations from the Philosophical Writings of Gottlob Frege*, Blackwell, Oxford.

Grätzer, G.(1979): *Universal Algebra*, Springer, New York.

Groenendijk, J., M.Stokhof (1991): 'Dynamic Predicate Logic', *Linguistics and Philosophy* 14, 39-101.

Janssen, T.M.V. (1983): *Foundations and Applications of Montague Grammar*, dissertation, Mathematical Centre, Amsterdam.

Peregrin, J. (1992): 'The Role of Variables in the Formalization of Natural Language', in *Proceedings of Eight Amsterdam Colloquium*, Amsterdam, 463-481.

Peregrin, J. and von Heusinger, K. (1995): 'Dynamic Semantics with Choice Functions', in *Choice Functions in Natural Language Semantics* (ed. U. Egli & K. von Heusinger), Universität Konstanz, 43-67.

Peregrin, J. (1995): *Doing Worlds with Words*, Kluwer, Dordrecht.

Peregrin, J. (to appear): 'Reference and Inference', *Proceedings of the Workshop „Reference and Anaphorical Relations"*, to be published by Universität Konstanz

Quine, W.V. (1960): 'Variables explained away', Proceedings of the American Philosophical Society 104, 343-347.

Quine, W.V. (1972): 'Methodological Reflections on Current Linguistic Theory', in *Semantics of Natural Language* (ed. D.Davidson and G.Harman), Reidel, Dordrecht, 442-454.

Russell, B. (1905): 'On denoting', *Mind* 14, 479-493.

Sellars, W. (1974): ‘Meaning as Functional Classification’, *Synthèse* 27, 417-437.

van Heijenoort, J., ed. (1971): *From Frege to Gödel: A Source Book from Mathematical Logic*, Harvard University Press, Cambridge (Mass.)