GENERALITY OF LOGICAL TYPES

Brice Halimi
Philosophy / U. Paris Ouest (ireph) & sphere
92001 Nanterre, France
bhalimi@u-paris10.fr

My aim is to examine logical types in *Principia Mathematica* from two (partly independent) perspectives. The first one pertains to the ambiguity of the notion of logical type as introduced in the Introduction (to the first edition). I claim that a distinction has to be made between types as called for in the context of paradoxes, and types as logical prototypes. The second perspective bears on typical ambiguity as described in Russell and Whitehead's "Prefatory Statement of Symbolic Conventions", inasmuch as it lends itself to a comparison with specific systems of modern typed lambda-calculus. In particular, a recent paper shows that the theory of logical types can be formalized in the way of a lambda-calculus. This opens the way to an interesting reconciliation between type theories in the Russellian sense of the word, and type theories in the modern sense. But typical ambiguity is not taken into account in the paper. I would like to take up that question of typical ambiguity, by extending the typed lambda-calculus to be used.

Two kinds of generality can be attributed to logical types in *Principia Mathematica*, and they ought to be clearly distinguished. The first one, which I will call "external generality", pertains to the formality of types as introduced in the Introduction (to the first edition) of *Principia*. I claim indeed that the formal system of ramified type theory lends itself to different "epistemic realizations", in which each type is assigned a particular interpretation, namely the set of all concrete propositional functions of that type in the epistemic realization that is considered. For example, the concrete propositional function "x is green" will correspond to a universal if it turns out to be an object of acquaintance in the epistemic perspective at stake, and otherwise to a definite description involving higher-order quantification. In the first case only, it will be part of the interpretation of the type of predicative first-order

russell: the Journal of Bertrand Russell Studies
The Bertrand Russell Research Centre, McMaster U.
n.s. 31 (summer 2011): 85–107
issn 0036-01631; online 1913-8032

propositional functions. That variety of possible epistemic counterparts of each type is what substantiates and explains its formality.

The second kind of generality that can be attributed to logical types, and which I will call "internal generality", bears on typical ambiguity, as described in Russell and Whitehead's "Prefatory Statement of Symbolic Conventions" at the beginning of *Principia*, Volume II. My claim is that it can be formalized within specific systems of modern typed lambda-calculus. In particular, a recent paper (Kamareddine *et al.* 2002) shows that the theory of logical types can be formalized in the way of a lambda-calculus. This opens the way to an interesting reconciliation between type theories in the Russellian sense of the word, and type theories in the modern sense. But typical ambiguity is not taken into account in the paper. I would like to take up that question of typical ambiguity, by extending the typed lambda-calculus to be used.

1. FORMAL THEORY AND EPISTEMIC REALIZATIONS OF LOGICAL TYPES

Referring to Quine and Sommerville, Nicholas Griffin makes the following remark:

> … Russell's type theory is to some extent context sensitive: for example, an item which, in one context, may be taken to be simple may, in another, turn out to be complex; and thus terms like "individual" or "first-truth" are not stable across contexts…. An example occurs in connection with the word "Socrates" which when used by Socrates himself denotes a simple individual of Socrates' acquaintance; whereas, when used by someone who has never met Socrates, it is a complex hidden description to be analyzed by Russell's theory of descriptions. Thus Socrates is a possible value of /$\hat{x}$ is an individual/ only for Socrates himself, for others, with no acquaintance with Socrates, Socrates is not a possible value for that function…. In general, since different people are acquainted with different items, the range of total variation for functions like /$\hat{x}$ is an individual/ will be different for different people. Thus it is intolerable to treat such functions as propositional functions of logic. (Griffin 1980, p. 138)

This is in fact, I think, a general and very important point, which is worth developing. Indeed, the ramified type theory set out in the Introduction of the first edition of *Principia Mathematica* constitutes a formal system that gives rise to a multiplicity of epistemic analyses, each analysis depending on the stock of individuals and universals with which the sub-

ject under consideration is acquainted. Let's consider, for instance, the proposition expressing that a certain individual *a* is green and that another individual *b* is a great general. If I am in acquaintance with both the individual *a* and the universal *Green*, I shall say, using the formal language of *Principia*:

(1) Ga (elementary truth, first-order proposition)

On the other hand, having only access to *a* as to "the *F*", I shall say:

(2) $(\exists x) : (y) . Fy . \equiv . y = x . \,\&\, . Gx$
(first-level truth, first-order proposition)

Having only access to green as in "the colour of grass" ("CoG"), I shall say:

(3) $(\exists \phi) :. (\psi) : \mathrm{CoG}(\psi) . \equiv . \psi = \phi : \,\&\, \phi a$
(second-level truth, second-order proposition)

Eventually, by combining the two descriptions, I shall say:

(4) $(\exists x) . (\exists \phi) :. (\psi) : \mathrm{CoG}(\psi) . \equiv . \psi = \phi : \,\&$
$: (y) . \phi y . \equiv . y = x : \,\&\, : \phi ! x$
(second-level truth, second-order proposition)

In the third case, "green" is nothing but an incomplete symbol. Nevertheless, its occurrence in any context will generate a second-order quantification. In view of this fact, green may be identified with a second-order propositional function with one individual argument. Therefore, in that case, supposing furthermore that *Being a great general* is a universal to me, my *epistemic diagram* will be the following:

- Individuals: *a*, *Being a great general*
- Predicative propositional functions of individuals: *x is a great general*
- Second-order propositional functions of individuals: "*x* is green", i.e., *x has the colour of grass*,

whereas in the first case, supposing furthermore that I understand the

property of being a great general only as the property of having all predicates common to great generals in history, it will be:

- Individuals: *a*, *Green*
- Predicative propositional functions of individuals: *x is green*
- Second-order propositional functions of individuals: "*x* is a great general", i.e., *x has all the predicates that make a great general.*

And so forth for the other cases. The quotation marks are meant to express the fact that the first epistemic subject does not speak for herself, but only uses a predicate that she borrows from another subject's language. The contextual definition of "green" in one case, or of "being a great general" in the other, works as an interpretation rule from an epistemic universe to another.

1.1 *Types as translation patterns*

Principia's ramified type theory allows us to distinguish among the statements from (1) to (4), even between (3) and (4)—hence the utility of fine-grained types. Indeed, these statements express propositions involving propositional functions *of different types*, even if these propositions correspond, in some way, to the "same" state of affairs. The logical structure of the situation will thus vary from subject to subject, and so will more generally the realization of the whole theory of logical types.

Principia mentions only variables of propositional functions, because otherwise a particular way of analyzing reality would be wrongly privileged. On the contrary, the theory of logical types remains a neutral scaffolding that every epistemic subject implements in a specific way. In other words, the epistemic counterpart of ramified type theory is an open multiplicity of realizations, each of which selects which terms will inhabit such and such type.

All terms turn out to be predicative, because they all get interpreted by predicates—which ones depends on the epistemic universe that happens to be considered. To get back to Russell's example, "Napoleon was a great general", as uttered by some subject S_1, is understood by some other subject S_2 as: "$(\psi) : f!(\psi!\hat{z}) . \supset . \psi!(\text{Napoleon})$". (We suppose that "being a predicate required in a great general" is a predicative second-order propositional function $f!(\psi!\hat{z})$ common to both S_1 and S_2.) Types of ramified type theory keep track of properties that some

subject cannot identify with any predicate that would be available to her. Here the translation rule

$$[Gx]_{S_1} = [(\psi) : f!(\psi!\hat{z}) . \supset . \psi!x]_{S_2}$$

adds no non-predicative terms to S_2's universe: "$G\hat{x}$" is only emulated from S_2's point of view, that is, introduced in S_2's language as a mere symbol in order to account for some of S_1's sentences. Still, S_2 needs to be able to quantify over such symbols, which means using variables ranging in fact (in S_2's perspective) over a domain of non-predicative terms, but only in a substitutional way. In other words, non-predicative functional terms are only the nominal equivalent, within one epistemic perspective, of what is accessible, within another epistemic perspective, as a predicative function. Accordingly, non-predicative variables are introduced, but only in view of a substitutional reading of quantification, as a way to render predicative quantification in some other epistemic universe. Here the complex type of S_2's counterpart of $G!\hat{x}$ underlies the translation of terms such as "Gx" by providing it with a pattern common to all the instances of the same structure (see the rules for type assignment below).

Viewed as translation patterns between epistemic universes, types correspond to all possible particular functional forms that one can specify without overstepping the sphere of the schematic. Types are assigned to variables so it is usually thought that they basically consist in ranges.[1] But types are primarily types of propositional functions, and types of variables only in a derivative way. Types are not domains, but forms, whose fine-grainedness ought to be maximal.

This means at least that, in the realm of ramified type theory, the type of any propositional function $\phi\hat{x}$ can be conceived of as a diagram displaying not only the simple types of the apparent variables occurring in ϕ, but the number of such apparent variables, as well as their respective types; and not only that, but also the number and the respective types of all the real variables, so that the diagram of anything involved in the arguments, as well as (possibly) in the arguments of these arguments (if the arguments at issue are themselves functions), in the arguments of the arguments of the arguments, and so forth, in an inductive manner. So

[1] See Chihara 1973, p. 43, for an example.

the general form of a functional type ought to be: $\langle t^a_1, \ldots, t^a_m; t^r_1, \ldots, t^r_n \rangle$, where the t^a_j's are the respective types of apparent variables, and the t^r_i's are the respective types of real variables, ordered according to their occurrences. The type of individuals then becomes $\langle - ; - \rangle = o$. The type of $(x) \,.\, \psi(x, \hat{y})$ is $\langle o ; o \rangle$, the type of $(\psi) \,.\, f!(\phi!\hat{z}, \psi!\hat{z}, x)$ is $\langle \langle - ; o \rangle; \langle - ; o \rangle, o \rangle$, that is, $\langle (o); (o), o \rangle$, where $(o) = \langle -, o \rangle$. The order is not mentioned, because it can always be determined directly from the type: the order of an individual is 0, and the order $|\phi|$ of a function ϕ of type $\langle t^a_1, \ldots, t^a_m; t^r_1, \ldots, t^r_n \rangle$ is: $\max\left(|t^a_i|, |t^r_j|\right)_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} + 1$. Predicative propositional functions can then be defined as usual, as the functions whose order is the least possible with respect to their type. By extension, I shall call a type predicative when it is the type of predicative functions, and a variable predicative when all its instances have a predicative type.

1.2 *A Russellian concept of model*

The reference in *Principia* to an epistemic perspective on logical types, and the need to sharply distinguish between the formal setting of ramified type theory, on the one hand, and such-and-such particular epistemic realization of it, on the other hand, is shown in particular by the following well-known passage:

> ... "$\phi!x$" is a function which contains no apparent variables, but contains the two real variables $\phi!\hat{z}$ and x. (It should be observed that when ϕ is assigned, we may obtain a function whose values do involve individuals as apparent variables, for example if $\phi!x$ is $(y) \,.\, \psi(x, y)$. But so long as ϕ is variable, $\phi!x$ contains no apparent variables.) (*PM* 1: 52)

At first sight, it seems difficult to make sense of the ambiguity of $\phi!x$ that Russell suggests. Russell does not mean that, as long as $\phi!x$ is not analyzed, it cannot contain apparent variables: that would amount to saying that $\phi!x$'s internal structure does not exist as long as we do not want to see it. Russell could mean that $\phi!x$ is only a schematic matrix for genuine propositional functions such as $(y) \,.\, \psi(x, y)$. But then the internal structure of $\phi!x$ should mirror that of its possible instances, and so $\phi!x$ could be non-predicative, which Russell seems to exclude. In fact, I think that $\phi!x$ and $(y) \,.\, \psi(x, y)$ belong to two totally different environments, and that the ambiguity that Russell brings out relies upon the gap between those two environments. Indeed, as a formal term of the theory of logical types, "$\phi!x$" is but a first-order predicative variable. Its

assigned value in a given epistemic universe will be a specific predicate, but the counterpart of that predicate in another epistemic universe may be a propositional function whose scheme is $(y) \,.\, \psi(x, y)$—where "$\psi(x, y)$" is not another variable, but a schematic letter standing for some actual binary relation. The two cognitive subjects whose epistemic universes are considered here will use the same term, but analyze it in two different ways.

The "realizations" of the theory of types are all possible epistemic diagrams as above, together with translation rules between them. Unlike the model-theoretic perspective, the Russellian notion of interpretation takes elementary equivalence (or the corresponding feature) for granted: as a matter of principle, two "corresponding" private statements have the same truth value (let us not clarify the "correspondence" which is at stake here). There is no real comparison between the different Tarskian interpretations of a logical language and the different realizations of ramified type theory, because, in the latter case, the resources of the language (among which are the available proper names) are precisely the changing parameters. Thus, on the one hand, Russell considers that the variable ϕ is "assigned" concrete propositional functions, which is quite analogous to the way in which values are assigned to variables in Tarskian semantics. But, on the other hand, two different realizations differ in their logical analysis of reality, and never in the truth values they give to sentences, which is at complete odds with Tarskian semantics. Two different models of ramified type theory are like two sections of the same universe, and types correspond, in each section, to reference marks for the logical representation of that section: predicative types correspond to the frontal sides, and non-predicative types to the dotted lines that occur in a drawing in perspective.

If a subject S_1 knows a only as "the G", she will assert something like "F(the G)", i.e.:

(5) $\quad (\exists x) : (y) \,.\, Gy \,.\equiv.\, y = x : \,\&\,.\, Fx$

without being able to indicate any possible instance for x. Any other subject S_2 in acquaintance with a, on the contrary, would be able to instantiate x. No subject is acquainted with all the possible instances of x, that is, with everything. The admission of a, as a quasi-value for an entity variable, into S_1's epistemic domain, hinges on S_1's believing that

S_2's identification of (what S_1 conceives of as) the G with some actual entity a is reliable. Granted that the G actually exists, "the G" becomes what Russell calls a construction, with the same status as other logical constructions. This means in particular that this definite description then "has (speaking formally) all the logical properties of symbols which directly represent objects"[2]: it can, by extension, instantiate any individual variable.

Individual variables do not refer to actual objects of acquaintance only, but also, for each subject S, to the virtual objects of acquaintance that S receives from any other subject whom she deems to be trustworthy. This is how S_1 gets the missing quasi-value of x in (5)—to the extent that she trusts S_2. The all-inclusive range of "x" becomes in that way the asymptotic result of the information flow shared by all truthful epistemic subjects, or the symmetric equilibrium hopefully reached by all epistemic players. It does not consist in a single domain of objects of acquaintance, but is comprised, from the point of view of each subject, of proper values *and* of quasi-values conveyed by others.

1.3 *Epistemic model theory*

Epistemic realizations are a natural thing to bring up as soon as *Principia*'s ramified type theory is understood as a formal scheme to be applied in a natural-language environment. The reducibility axiom lends itself to an epistemological interpretation:

> Although our *access* to a function ϕ is mediated by quantification over other functions, this in no way precludes the *existence*, within the hierarchy, of extensionally equivalent predicative functions or functions of order 1.... The axioms of acquaintance and reducibility postulate (respectively) the possibility of knowing individuals and classes in terms of functions that possess a certain epistemic transparency, a transparency embodied by acquaintance in the one case and the absence of complex forms of quantification in the other. Classes occur in the hierarchy only under the guise of predicative functions, which are the means by which they are known. Reducibility thus postulates a concordance between mathematical reality and our knowledge of it that the ramified theory is otherwise unable to demonstrate. (Demopoulos and Clark 2005, p. 158)

[2] *PM* ∗14.18, 1: 180. See ∗20.71 about classes. The formal properties that Russell alludes to here are all the laws of the "theory of apparent variables", such as ∗9.2 (the rule of universal instantiation).

Once the epistemological background of *Principia*'s types is given some attention, the reducibility schemata can be understood as meaning that, for any propositional function, there is always an epistemic realization in which that function, as a symbol, is interpreted by a genuine term. In the case of individuals, any object introduced by a definite description and assumed to exist (as opposed to "the actual king of France"), must correspond, up to a shift in epistemic context, to an actual individual. This is a kind of quasi-acquaintance principle.

In the case of propositional functions, the axiom schema says that any non-empty functional term corresponds, in some epistemic universe, to an actual predicate, going in fact with a whole list of conditions. Let's consider, for example, a translation pattern such as

$$[(\phi) \,.\, f!(\phi!\hat{z}, x)]_{S_1} \equiv [Fx]_{S_2} \qquad (\star).$$

This equivalence holds even though, among all the predicates ϕ at stake in ($\star$) that S_1 has access to, there are some for which S_2 does not have any predicative counterpart. Suppose now that $f!(\hat{\phi}, \hat{z})$ itself belongs to S_2's epistemic universe. Then, for any particular G also belonging to that universe, $Fx \,.\, \supset_x \,.\, f(G!\hat{z}, x)$ has to be valid in S_2's realization. The validity of $Fx \,.\, \supset_x \,.\, f(G!\hat{z}, x)$ is part of the list of conditions that goes along with ($\star$).

Once again, I think that epistemic realizations are the semantic counterpart of the theory of logical types, and that distinguishing the latter from the former is the only way to understand the status of non-predicative terms—and in particular to understand their syntactic possibility despite the fact that one never finds variables of non-predicative propositional functions in *Principia*. It should be added that epistemic subjects mean nothing else here but complete epistemic diagrams, that is, ways in which reality could be carved up. Hence, they are not metaphysically loaded, and besides remain quite remote from actual concrete cognitive subjects. So, in fact, a formal semantics based on epistemic diagrams has only subsequent connections with epistemology properly speaking.[3]

The array of all possible interpretations of a single type, according to

[3] On that score, there is an issue about whether it is possible or impossible, in Russell's view, that objects do exist without belonging to any epistemic universe. I leave that question aside.

different epistemic perspectives, is the true content of its *formality*. Since it is based on actual applications of ramified type theory to the analysis of ordinary language and knowledge conditions, the generality thus brought up can be said to be external. On the contrary, the *ambiguity* of logical types, as justified in Russell and Whitehead's "Prefatory Statement", is a totally different kind of generality bestowed on types in *Principia*. Since it consists in the ability to use types as representatives of indefinitely many others, within the same hierarchy of purely logical types, typical ambiguity can be described as an internal generality, rather than as an external one. It is often conceived of as a useful device that allows the system of *Principia* to have a type of all types without laying itself open to the logical inconsistency of having such a universal type. Though powerful and innocuous it may be, it is commonly considered as an informal convention that is not really amenable to any intra-systematic treatment, and, to that extent, it involves a kind of shortcoming of the system.

On the contrary, I would like to defend the view that typical ambiguity is an intrinsic part of the notion of logical type, and not a feature merely tacked on to the conception of types as developed in the Introduction. Typical ambiguity has been deemed to be an *ad hoc* solution to discharge a generality that was at the same time precluded for impredicativity reasons. Russell was trying to "have his cake and eat it too".[4] Such a diagnosis can be corrected: typical ambiguity can indeed be understood as a feature that is on a par with the rest of the system. All that is needed is to provide for its formal regimentation. Such a formalization has been explored, in particular by Harper and Pollack (1991), but it does not consider *Principia* as anything other than a historical landmark, and

[4] See Feferman 2004. According to Feferman, "the use of typical ambiguity is a way of saving face" (p. 138) despite the fact that, as Russell himself acknowledges, "there are as many 0's as there are types." Feferman proposes a system of set theory with countably-many nested "reflective universes" U_n, each of which is an elementary substructure of the universe. The resulting system ZF / $U_{<\omega}$, which is proved to be a conservative extension of ZF, is intended primarily to provide a foundational framework for category theory, and in particular to account for cases of self-membership such as the category of all categories. Feferman considers that type theory, on the contrary, "doesn't lend itself to the flexible expression of mathematical properties" (p. 140). In what follows, I wish to vindicate both type theory, in the form of modern systems of lambda-calculus, and category theory, by showing that the former is rich enough to formalize Russellian typical ambiguity, and that the latter is the only possible semantical framework for those systems.

involves the difficult tools of the calculus of constructions. In the following, I hope to show that a more simple extension of typed lambda-calculus can be sufficient to formalize typical ambiguity in a neat way.

2. TYPICAL AMBIGUITY AND POLYMORPHISM OF TYPES

Typical ambiguity is actually a theme that shows up quite early in Russell's logical works. This is borne out by a manuscript dating back to 1907, and entitled "Types":

> We must write $f!\alpha$ for a function whose arguments are of any type other than individuals: then
>
> $$\alpha \in \dot{\beta}(f!\beta) . = . f!\alpha \qquad \text{Df}$$
>
> and so on. In this way, the theory of all types can be done at once.
>
> There is some obscurity about the *Primitive idea* $f!\alpha$. May this contain apparent variables of any type, however high? …
>
> … We can construct $f!\hat{\phi x}$ with any ϕ we like, and then substitute a more complicated ϕ. The fact that ϕ *may* contain an f as apparent variable does not matter at all. The way to explain things is as follows:
>
> (1). ϕx stands for anything containing x, and being a proposition for every value of x.
>
> (2). $f!\hat{\phi x}$ stands for anything containing values of ϕ, and containing these values with arguments which are constants or apparent variables, and containing these values only in propositional positions.
>
> (3). $F!f!\hat{\phi x}$ stands for anything containing values of f in a similar way.
>
> The point is that f and F stand for ways of construction, and do not presuppose any knowledge of what is to be put in as argument beyond what can be got from lower types alone.[5]

The "ways of construction" to which Russell refers in 1907 are construed a few years later, in Russell and Whitehead's "Prefatory Statement", as "symbolic forms". A symbolic form is nothing other than what different types have in common, to the extent that they arise from the same functional skeleton, for different typical determinations of the variables. Thus typical ambiguity confirms that types are not levels or domains, but themselves propositional schemes:

[5] Bertrand Russell Archives, manuscript 230.030890, fos. 32–3.

> When a proposition containing typically ambiguous symbols *can be proved* to be true in the lowest significant type, and we can "see" that symbolically the same proof holds in any other assigned type, we say that the proposition has "permanent truth." …
>
> It is convenient to call the symbolic form of a propositional function simply a "*symbolic form*". Thus, if a symbolic form contains symbols of ambiguous type it represents different propositional functions according as the types of its ambiguous symbols are differently adjusted.…
>
> To "assert a symbolic form" is to assert each of the propositional functions arising for the set of possible typical determinations which are somewhere enumerated. We have in fact enumerated a very limited number of types starting from that of individuals, and we "see" that this process can be indefinitely continued by analogy. (*PM* 2: xii)

Admittedly, Russell and Whitehead's resort to "seeing" betrays a kind of embarrassment. Type variables would be needed to express formally what Russell and Whitehead here allude to. But, for obvious reasons of circularity, it is impossible to quantify over types. That is why typical ambiguity has often been understood as a mere corrective. My aim is to qualify that predicament, drawing on modern type-theoretic frameworks. A recent paper (Kamareddine *et al.* 2002) shows that the theory of logical types can be formalized by means of a lambda-calculus. But typical ambiguity is left aside. I would like to extend the suggestion by showing that typical ambiguity can be captured, on one condition: shifting to stronger "polymorphic" type-theoretic calculi.

2.1 *Typical ambiguity as parametricity*

Polymorphism is a feature of programming languages in which "generic data types" are introduced that allow programmers to express the generality of *uniform* algorithms.[6] An example of that kind of behaviour is the programme which performs the swapping of the values of two variables ranging over integers:[7]

```
SWAP (var m, n: Int)
var t: Int
Begin
t: = m
```

[6] See Scedrov 1990.

[7] Crole 1993, p. 202.

$m = n$
$n := t$
End

It is clear that the programme works exactly the same for variables of any other type. It is, of course, much more efficient to be able to implement a general swapping programme, regardless of the type of the variables. This is made possible by the introduction of *type variables*:

GENERAL SWAP (type: X; var m, n: X)
var t: X
Begin
$t := m$
$m := n$
$n := t$
End

where the type variable X is assigned a value at each call of the procedure.

This is exactly the kind of situation that Russell had in mind. Type variables have been introduced in the realm of modern type theories to substantiate the idea that certain procedures have a generic value.

2.2 *Polymorphic second-order type theory*

Polymorphic second-order type theory (PSO) is actually a whole family of formal systems of typed lambda-calculus which provide a formal syntax for writing down functional terms and which include variables of types. The characteristic feature of PSO is that it allows explicit abstraction or quantification on type variables, both in types and terms, giving rise to "polymorphic types" and "polymorphic terms". That is why PSO is also called "impredicative type theory" (inasmuch as a type can be introduced through a quantification over the collection of all types).

In ordinary lambda-calculus each term codes a proof of the proposition coded by its type: this is the "propositions-as-types" interpretation. Consequently, any polymorphic term takes types as inputs (including its own type), giving terms as outputs, and can be viewed as a generic proof, that is, as a uniform procedure to prove propositions which differ only

with respect to the kind of things to which they pertain, and whose respective proofs, consequently, are "structurally" the same. A polymorphic term is nothing but the representation of the pattern common to analogous proofs. This holds for proofs as well as for operations such as the swapping of values.

Let me first set out the main lines of PSO. The basic type symbols are type variables X, Y, Z, ... Types are then constructed inductively through construction rules. In particular, if "F" is any type symbol of non-zero arity n, and if $\Phi_1, \ldots, \Phi_n$ are n types, then $F(\Phi_1, \ldots, \Phi_n)$ is a type (think here of propositional connectives as binary functional types). On top of that, if a type variable X occurs in a type $F(X)$, second-order abstraction $\forall X$ gives rise to a universal type $\forall X.\, F(X)$. The syntax is summed up in the following list:[8]

$$\Phi := X \mid F(\Phi, \ldots, \Phi) \text{ (for instance } \Phi \times \Phi \text{ and } \Phi \rightarrow \Phi) \mid \forall X.\, \Phi.$$

The rules to generate well-formed types refer to *type contexts*, that is, lists Δ of distinct type variables that are supposed to serve as elementary blocks for the construction of the well-formed (or "proved") types.

Proved types:

- Variables:

$$\frac{}{\Delta, X, \Delta' \vdash X}$$

- Functions:

$$\frac{\Delta \vdash \Phi_1 \quad \ldots \quad \Delta \vdash \Phi_n}{\Delta \vdash F(\Phi_1, \ldots \Phi_n)}$$

- Universal types:

$$\frac{\Delta, X \vdash \Phi}{\Delta \vdash \forall X.\, \Phi} \qquad (X \notin \Delta)$$

[8] I follow here Roy Crole's exposition of PSO given in Crole 1993.

Using the rules above, derivation trees can be built whose conclusion is a judgment $\Delta \vdash \Phi$, called a *proof* of the type Φ in the context Δ.

Terms are defined starting with variables of different types, and then using construction rules: function symbols α with specified arity and sorting $\sigma(\alpha) = [\Phi_1, \ldots, \Phi_n, \Phi]$, term abstraction and application, type abstraction and application. This gives the following term syntax:

$$f = x \mid \alpha^{\sigma(\alpha)}(f, \ldots, f) \mid \lambda x : \Phi \,.\, f \mid ff \mid \lambda X \,.\, f \mid f\,\Phi.$$

As one can see, types are not only used to type terms, but also, as variables, to build them. Terms having (possibly besides term variables) type variables among their constituents can then be considered as "parametrized" proofs or operations. To return to the former example of the swapping of values, $\lambda x : \mathrm{Int} \,.\, \lambda y : \mathrm{Int} \,.\, \langle y, x \rangle$ generalizes into $\Lambda X \,.\, \lambda x : X \,.\, \lambda y : X \,.\, \langle y, x \rangle$.

As is the case for types, rules for generating well-typed terms mention contexts. A term context is a sequence $\Gamma = [x_1 : \Phi_1, \ldots, x_m : \Phi_m]$ of distinct typed variables (the variables are said to make up the *domain* of the context, $\mathrm{dom}(\Gamma)$). One writes $\Delta \vdash \Gamma$ (Δ being a type context) to mean $\Delta \vdash \Phi_i$ for each Φ_i in Γ.

Proved terms (typing rules):

- Term Variables:

$$\frac{\Delta \vdash \Gamma \qquad \Delta \vdash \Phi \qquad \Delta \vdash \Gamma'}{\Delta \parallel \Gamma, x : \Phi, \Gamma' \vdash x : \Phi}$$

- Binary Products:

$$\frac{\Delta \parallel \Gamma \vdash f : \Phi \qquad \Delta \parallel \Gamma \vdash f' : \Psi}{\Delta \parallel \Gamma \vdash \langle f, f' \rangle : \Phi \times \Psi}$$

- Functional abstraction:

$$\frac{\Delta \parallel \Gamma, x : \Phi \vdash f : \Psi}{\Delta \parallel \Gamma \vdash \lambda x : \Phi \,.\, f : \Phi \to \Psi}$$

- Application to a term:

$$\frac{\Delta \| \Gamma \vdash f : \Phi \to \Psi \qquad \Delta \| \Gamma \vdash g : \Phi}{\Delta \| \Gamma \vdash fg : \Psi}$$

- Type abstraction:

$$\frac{\Delta, X \| \Gamma \vdash f : \Phi}{\Delta \| \Gamma \vdash \Lambda X . f : \forall X . \Phi} \qquad (X \notin \Gamma)$$

- Application to a type:

$$\frac{\Delta \| \Gamma \vdash f : \forall X . \Phi \qquad \Delta \vdash \Psi}{\Delta \| \Gamma \vdash f\, \Psi : \Phi\, [\Psi/X]}$$

The above-mentioned rules allow one to construct derivation trees for judgments $\Delta \| \Gamma \vdash t : \Phi$, called a *proof* of the term $t : \Phi$ in the double context Δ, Γ.

It remains to set the conversion rules—that is, the rules of the lambda-calculus properly speaking, which consist in equalities between proved terms:

- Functional equalities:

$$\frac{\Delta \| \Gamma, x : \Phi \vdash F : \Psi \qquad \Delta \| \Gamma \vdash f : \Phi}{\Delta \| \Gamma \vdash (\lambda x : \Phi . F) f = F[f/x] : \Psi}$$

- Polymorphic equalities:

$$\frac{\Delta, X \| \Gamma \vdash f : \Phi \qquad \Delta \vdash \Psi}{\Delta \| \Gamma \vdash (\Lambda X . f)\, \Psi = f[\Psi/X] : \Phi\, [\Psi/X]} \qquad (X \notin \Gamma)$$

As we can finally see, PSO allows a type-theoretic formalization of vicious circularity. Indeed, a polymorphic type such as $\Lambda X . T(X)$ can be applied to any type, including itself. Besides, a polymorphic term takes as input any type, including its own. This clearly violates the vicious-circle principle. How could it be possible, on that basis, to catch up with *Principia*'s theory of logical types?

2.3 *Typical ambiguity in polymorphic clothing*

Following Kamareddine *et al.* (2002), one can think of *Principia*'s theory of logical types as a type theory with propositional functions instead of lambda-calculus terms, and types in the sense of *Principia* instead of propositions-as-types. But it remains to implement the rules for polymorphic lambda-calculus so as to formalize typical ambiguity.

In compliance with the fine-grained presentation of types I have advocated earlier, the rules for types are:

(i) $o = \langle - \,;\, - \rangle^o$ is a ramified type (the type of individuals);
(ii) if $t_1^{\alpha_1}, \ldots, t_m^{\alpha_m}, u_1^{\beta_1}, \ldots, u_n^{\beta_n}$ are ramified types, then $\langle t_1^{\alpha_1}, \ldots, t_m^{\alpha_m}; u_1^{\beta_1}, \ldots, u_n^{\beta_n} \rangle^\gamma$ is a ramified type, with $\gamma = \max(\alpha_i, \beta_j) + 1$.

(As already stated, the mention of orders is in fact redundant.)

Let's move on now to the typing rules. All propositional functions here are considered up to "α-equivalence", that is, up to a change of variables (in keeping with their respective types). The main rules are:

- Individuals:

$$\frac{}{\Delta \| \Gamma \vdash a : o}$$

- Functional variables:

$$\frac{\Delta \vdash \Gamma \qquad \Delta \vdash \Phi \qquad \Delta \vdash \Gamma'}{\Delta \| \Gamma, f : \Phi, \Gamma' \vdash f : \Phi}$$

- Connectives:

$$\frac{\Delta \| \Gamma \vdash f : t}{\Delta \| \Gamma \vdash \sim f : t}$$

$$\frac{\Delta \| \Gamma \vdash f : (t_1, \ldots, t_m; u_1, \ldots, u_n) \quad \Delta \| \Gamma \vdash g : (t'_1, \ldots, t'_k; u'_1, \ldots, u'_l)}{\Delta \| \Gamma \vdash f \wedge g : (t_1, \ldots, t_m, t'_1, \ldots, t'_k; u_1, \ldots, u_n, u'_1, \ldots, u'_l)}$$

(merging the types corresponding to identical variables or constants)

- Abstraction from parameters:

$$\frac{\Delta \| \Gamma \vdash g : t \qquad \Delta \| \Gamma \vdash f : (t_1, \ldots, t_m; u_1, \ldots, u_n)}{\Delta \| \Gamma \vdash f\,[z^t/g] : (t_1, \ldots, t_m; u_1, \ldots, u_n, t)}$$

(where g is a parameter in f)

- Quantification:

$$\frac{\Delta \| \Gamma, x_j : u_j \vdash f : (t_1, \ldots, t_m; u_1, \ldots, u_j, \ldots, u_n)}{\Delta \| \Gamma \vdash (x_j)\,.\,f : (t_1, \ldots, t_m, u_j; u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_n)}$$

(x_j is the j-th free variable in f)

- Polymorphism:

$$\frac{\Delta, X \| \Gamma \vdash f : t \qquad f \text{ is predicative}}{\Delta \| \Gamma \vdash \Lambda X\,.\,f : \forall X\,.\,t} \qquad (X \notin \Gamma)$$

- Type Application:

$$\frac{\Delta \| \Gamma \vdash \Lambda X\,.\,f : \forall X\,.\,t \qquad \Delta \vdash t' : \text{Type}}{\Delta \| \Gamma \vdash (f)t' : t[t'/X]}$$

The resulting system PRTT (polymorphic ramified type theory) aims at formalizing *Principia*'s theory of logical types when typical ambiguity is built into the theory.

In Kamareddine *et al.* (2002) a rule for abstraction from propositional functions is needed:

$$\frac{\Gamma, y : t_i^{a_i} \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a \qquad f \text{ is predicative} \qquad \mathrm{FV}(f) \subseteq \mathrm{dom}(\Gamma)}{\Gamma \cup \{z : (t_1^{a_1}, \ldots, t_n^{a_n})^a\} \vdash z(y_1, \ldots, y_k) : (t_1^{a_1}, \ldots, t_n^{a_n}, (t_1^{a_1}, \ldots, t_n^{a_n})^a)^{a+1}}$$

($z \notin \mathrm{dom}(\Gamma)$)

Such a rule is not needed in PRTT: variables of any type can be introduced at the onset, because all bound variables are recorded within each type, as in the Term Variables rule of PSO.

2.4 *Typical ambiguity and type-theoretic genericity*

There is a way to extend type-theoretically the idea of typical ambiguity while sticking more closely to Russell's conception. Indeed, typical ambiguity in the "Prefatory Statement" is an example of parametric generality, as opposed to quantificational generality. Picking up on a suggestion due to Carnap (1983, p. 51), Giuseppe Longo and Thomas Fruchart have suggested accounting for the "generic" value of types in PSO by adding the following:

"Axiom C": *If* $M : \forall X . \sigma$ *and* X *is not among the free variables of* σ, *then* $M\tau = M\tau'$ *for all types* τ, τ'.

> Axiom C intuitively means that an input (τ), which is not used to establish the type (as σ does not depend on X) of the corresponding output value ($M\tau$), bears no information as input. So if M has the type $\forall X . \sigma$ and X is not free in the type σ (i.e. σ is not a function of X), then it does not matter whether one applies M to τ or τ' and one may consider both results to be equal.
>
> (Fruchart and Longo 1999, p. 46)

Axiom C is consistent with Girard's system F, and the system $Fc = F +$ Axiom C can prove the following "Genericity Theorem":

Let M *and* N *have type* $\forall X . \sigma$. *If* $M\tau =_{Fc} N\tau$ *for one type* τ, *then* $M =_{Fc} N$.

"In other words, the behaviour of polymorphic terms is so 'uniform' that one can reduce *Fc* equality on *every* possible types [*sic*] to *Fc* equality on one *single* type τ (no matter which one!)" (*ibid.*). That means that the proof or computation coded by a polymorphic term does not depend on the input type: in some cases, a particular instance $\sigma[\Phi/X]$, being parametric in Φ, can be described as being obtained by the uniform substitution of Φ for a type variable X, and thus may suffice to determine the fully general proof, i.e., to get to the universal $\forall X . \sigma$. When this happens, the type Φ is said to be *generic*. This is not always the case, otherwise any term $\sigma : \Phi$ could lead to a proof of the absurdum $\forall X . X$.

But Russell precisely does not claim that every proposition containing typically ambiguous symbols can be proved to be true in any assigned type or, in other words, that it has "permanent truth". Typical ambiguity is brought into play only in the specific realm of "formal arithmetic".

Type-theoretic genericity can be a way to describe accurately the fact that arithmetical truths, particularly, have a "stable truth-value".

2.5 *Semantical issues*

Let's now look at a natural set-theoretic semantics for PRTT. The idea is to interpret each type θ as a set, namely as the set $[\theta]$ of all the terms whose type is θ. Because of the quantification binding type variables, types are considered themselves as members of a set $\mathcal{U}$ of sets, so that functions between the sets in $\mathcal{U}$ can serve as interpretations of terms:

- $[o] = I \in \mathcal{U}$;
- If $[f(x)] \in A = [(t_1, \ldots, t_m; u_1, \ldots, u_n)] \in \mathcal{U}$ for any $x : t$ with $B = [t] \in \mathcal{U}$, then $[f] \in B^A \in \mathcal{U}$;
- If $[f] \in A = [(t_1, \ldots, t_m; u_1, \ldots, u_n)] \in \mathcal{U}$ and $A_j = [u_j] \in \mathcal{U}$, then $[(x_j) . f] \in \prod_{y \in [A_j]} [f(y)]$;
- If $\Lambda X . f : \forall X . \Phi$ and $F = [X \mapsto \Phi(X)] \in \mathcal{U}^{\mathcal{U}}$, then $[\Lambda X . f] \in \prod_{X \in \mathcal{U}} F(X) \in \mathcal{U}$.

To take stock: $\mathcal{U}$ contains a (presumably) non-empty set I of individuals and is closed under exponents as well as under dependent products over $\mathcal{U}$ (for any $F \in \mathcal{U}^{\mathcal{U}}$, $\prod_{X \in \mathcal{U}} F(X) \in \mathcal{U}$). It turns out that there cannot be any such set-theoretic model of PRRT. Indeed, John C. Reynolds proved that there is not set of sets $\mathcal{U}$ that is closed under exponents and dependent products over itself, except for a set $\mathcal{U}$ whose every member $X \in \mathcal{U}$ has at most one element.[9] This would imply in particular that there is only one individual (one element only in I). This is, of course, not practicable as an option.

Actually, one of the main available interpretations of polymorphic type theory is not set-theoretic, but pertains to "fibred category theory". More specifically, proved types $X_1, X_2, \ldots X_n \vdash \Phi$ are interpreted as maps $[X_1, X_2, \ldots X_n \vdash \Phi] : U^n \to U$ in a category C consisting of all the finite products of some distinguished object U. A proved type is thus a

[9] See Reynolds 1984 and Jacobs 1999, 8.3.3.

member of some $C(U^n, U)$. For example, the proved type $X_1, X_2 \vdash X_1$ is interpreted as the projection $U^2 \to U$. Then, the rules of derivation of proved types become maps between such $C(U^n, U)$'s. For example, binary product consists in a family of operations $B_n : C(U^n, U) \times C(U^n, U) \to C(U^n, U)$, and abstraction on types in a family $\forall_n : C(U \times U^n, U) \to C(U^n, U)$. Finally, for each proved term $X_1, \ldots X_n \parallel \Phi_1, \ldots, \Phi_m \vdash f : \Phi$, suppose that each proved type Φ_j has been interpreted as $\phi_j = [X_1, \ldots, X_n \vdash \Phi_j] \in C(U^n, U)$, and that Φ has been interpreted as $\phi = [X_1, \ldots, X_n \vdash \Phi] \in C(U^n, U)$. It is then natural to interpret f as a morphism $[f] : \phi_1 \times \ldots \times \phi_m \to \phi$ in the category $C(U^n, U)$. It is, of course, necessary to throw in further constraints in order to account for the typing rules, as well as for the functional and polymorphic equalities.

Still, the underlying idea is clear: it is to consider a base category C consisting of all the finite products of some object U, and to assign to each object U^n in C—thought of as a type context—the "fibre" $C(U^n, U)$, that is, a category whose objects are proved types in that context, and whose morphisms are corresponding proved terms. The judgment $\Delta \parallel \Gamma \vdash t : \Phi$ becomes a logical relation in the fibre over the type context $\Delta \vdash \Gamma$. Hence, each such type context becomes an index for a logic describing what happens in that context,[10] and substitution of a type for a type variable amounts to reindexing.

Since categorical semantics is called for as a way to handle impredicativity, it would be interesting to see how to adapt it to the case of PRTT. I leave it for further examination.

2.6 *Conclusion*

To sum up briefly, there are two kinds of generality of logical types in *Principia*: *formality*, which is an external generality peculiar to types, and *ambiguity*, which is an internal one. Even though typical ambiguity is described by Russell himself as a form of context relativity (*PM* 1: 65), these two kinds of generality are independent and should not be confused. The first one has to do with the fact that type theory gives rise to a variety of realizations, so that each type will be inhabited by different terms according to the epistemic perspective that is considered. The second one has to do with typical ambiguity properly speaking, not as a shortcoming or a stopgap of *Principia*'s system, nor as a slackening in the original

[10] See Jacobs 1999, p. 173.

logical seriousness, but as a positive feature that can be accounted for in a logical way.

Each of the two threads that I have just set out gives rise to a semantical perspective. In the first case, I have argued that each epistemic realization of the formal system of ramified type theory constitutes in its own right a "model" of that theory. In the second case, I have brought up the possibility of construing typical ambiguity polymorphically, and to provide the resulting polymorphic ramified type theory with a model, along the lines of the categorical interpretation of second-order lambda-calculus. These two perspectives are enough to suggest substantial connections between *Principia*'s logic and modern semantics.

REFERENCES

Carnap, R. 1983. "The Logicist Foundations of Mathematics". In P. Benacerraf and H. Putnam, eds. *Philosophy of Mathematics: Selected Readings*. 2nd edn. Cambridge: Cambridge U. P.

Chihara, C. S. 1973. *Ontology and the Vicious-Circle Principle*. Ithaca and London: Cornell U. P.

Crole, R. L. 1993. *Categories for Types*. Cambridge: Cambridge U. P.

Demopoulos, W., and P. Clark. 2005. "The Logicism of Frege, Dedekind and Russell". In S. Shapiro, ed. *The Oxford Handbook of Philosophy of Mathematics and Logic*. Oxford: Oxford U. P.

Feferman, S. 2004. "Typical Ambiguity: Trying to Have Your Cake and Eat It Too". In G. Link, ed. *One Hundred Years of Russell's Paradox*. Berlin: Walter de Gruyter.

Fruchart, T., and G. Longo. 1999. "Carnap's Remarks on Impredicative Definitions and the Genericity Theorem". In A. Cantini, E. Casari, and P. Minari, eds. *Logic and Foundations of Mathematics; Selected Contributed Papers of the Tenth International Congress of Logic, Methodology and Philosophy of Science, Florence, August 1995*. Dordrecht: Kluwer.

Griffin, N. 1980. "Russell on the Nature of Logic (1903–1913)". *Synthese* 45: 117–88.

Harper, R., and R. Pollack. 1991. "Type Checking with Universes". *Theoretical Computer Science* 89: 107–36.

Jacobs, B. 1999. *Categorical Logic and Type Theory*. Amsterdam: Elsevier.

Kamareddine, F., T. Laan, and R. Nederpelt. 2002. "Types in Logic and Mathematics before 1940". *The Bulletin of Symbolic Logic* 8: 185–245.

Reynolds, J. C. 1984. "Polymorphism Is Not Set-Theoretic". In G. Kahn, D.

MacQueen, and G. Plotkin, eds. *Semantics of Data Types*. Berlin: Springer.
Scedrov, A. 1990. "A Guide to Polymorphic Types". In P. Odifreddi, ed. *Logic and Computer Science*. London: Academic Press.
Whitehead, A. N., and B. Russell. 1925–27. *Principia Mathematica*. 2nd edn. 3 vols. Cambridge: Cambridge U. P. (1st edn., 1910–13.)