On Building Abstract Terms in Type Systems*

Giuseppe Primiero

1. Abstraction and predication

This paper offers some historical and conceptual remarks on the philosophical and logical procedures of abstraction, based on an account of the notions of concept and function. In order to provide a complete analyis, one should start by considering Plato's theory of Ideas, which provides the first interpretation of "abstract terms" in the history of philosophy[1]. The nature of the most general Forms, the related problem of the knowledge thereof, their connection to existing (concrete) objects, are the essential features of the Platonic theory of knowledge and of his metaphysics. The Platonic approach is grounded on the principle of conceptual priority of Ideas over their partecipations, the Forms existing separeted from all the particulars: the former are interpreted as *standard* particulars to which other particulars conform. Nonetheless, my investigation will start rather by Aristotle, who held first the relation of predication to be the basis for defining abstraction: from this I will try to consider some important ideas for the notion of abstraction in Type Systems.

By rejecting the Platonic understanding of general Forms, Aristotle maintains that the logical relation of predication is the starting point for any account of the Categories: these are intended as the forms of both what there is, and of what can be said[2]. Consequently, to speak about the existence of a certain category means properly to make a predication about a certain substance (as the first category, to which the others refer). To analyse predicative expressions (in order to explain the related predicables) will thus amount to a proper *abstraction procedure*. The Aristotelian understanding of abstract terms is in turn given by two related aspects, expressing the underlying distinction between abstract methods and abstract terms:

---

* This research has been made possible thanks to a post-doc research fellowship founded by the University of Palermo (Italy) and spent at the Faculty of Philosophy at the Universiteit Leiden (the Netherlands).

[1] Different passages in the dialogues can be referred to for his theory of abstraction, among them: *Phaedo*, 100a–101e; *Theaetetus* 201d–202c; *Republic*, 476a, 596a; and the entire discussion in the *Sophist*.

[2] Aristotle, *Categories*, par.2.

- the notion of *universal* (καθὸλυ);
- the notion of object produced by (a method of) abstraction (τὰ εξ αφαίρεσεος λεγόμενα, εν αφαίρεσει, δί αφαίρεσεος, αφαίρειν).

Starting by the classical form of judgement "P belongs to S" (with P and S used as schematic letters respectively for predicate and subject), one says that P *belongs universally* to S if:

- the predicate P belongs to every element S;
- that P belongs to S is due to S itself in such that it is an S (in virtue of S and qua S), i.e. not by accidens.

A universal is therefore identified with a predicable satisfying the two previous conditions[3]. On the other hand, the idea of (a method of) abstraction (by which a certain abstract term is produced) corresponds to a *removal operation*[4]. This idea is formulated in the general logical context of the *Analytics* in terms of the operation of removing particular predicates, namely those not falling under the previously given description of the universal. The removal operation preserves only the definitional predicates for the subject (its defining categories), and this corresponds to proceeding from the particular to the more general of the categories[5]. This procedure amounts to concepts formation by abstraction, in terms of the classification of the properties belonging to objects or entities, thus providing their hierarchy of universality. The determination of universals in terms of the particulars defines moreover the peculiar aspect of demonstration[6].

The distinction between formal procedures of abstraction and abstract entities (concepts) seems thus to be already developed in the Aristotelian logic and metaphysics, where "concept" must be intended as the abstract entity resulting from progressive universalization of predications. In the following I will consider further the relation between abstraction and concepts, to show the development of the logical notions involved.

2. Universality and meaning

The relation between general names and related particulars, first questioned by Plato at length for example in the *Parmenides*, is one of the greatest heredities the Middle Age received from antiquity. Porphyry in his *Isagoge*, or introduction

[3] For this explanation cf. e.g. *Metaphysics B*, 4, 1000a1; Γ, 9, 1017b 35; *Z*, 13, 1038b11–13; *Posterior Analytics* I, 4, 73b 26–74a 3.

[4] The standard example is given in the *Physics*, in terms of perceptibles defined as physical magnitudes, something which by nature can be added or removed. Cf. Aristotle, *Physics*, Book 3.

[5] See e.g. *Posterior Analytics*, I, 18, 81b 3–7 and *Metaphysics*, M 2, 1077b 10.

[6] See Aristotle, *Posterior Analytics*, I, 11, 77a 5–9.

to Aristotle's *Categories*, considers the problem of universals explicitely. The nature of Platonic Forms (or of Aristotelian categories) became then a ground problem for later philosophers in the Middle Age, among them Boethius in his *Commentaries on Porphyry's Introduction* discussed it, and Abelard maintained that Ideas preexist the creation as patterns which determine divine Providence in creating the best of the possible worlds (a thesis known as exemplarism).

The theory of universals, which is clearly connected to the nature of abstract entities and therefore to abstraction, was later deeply influenced by the semantic theory of *suppositio*. One of the most relevant interpretations was the thereon based notion of *generality* developed by Ockham, and the derived theory of abstract entities. The theory of supposition furnishes notoriously a semantic treatment for the properties of terms in a sentence: it actually consists in determining the context of semantic validity of a categorematic term in a proposition. Ockham uses the powerful theory of supposition in connection with the theory of universals, by considering the relation between categorematic and syncategorematic terms (i.e. respectively the counterparts of Aristotelian categories and of modern logical constants), in order to answer the question whether universal terms have proper signification. He maintains that a term which supposits generally in a proposition (i.e. under the specification of the syncategorematic "all"), supposits for every term contained in the appellative domain determined by the general noun. In paragraphs 6–8 of his *Summa Logicae*[7], he states that each of two names which are respectively the abstract and the concrete of the same concept (e.g. humanity-man, animality-animal, hotness-heat) are not synonymous, i.e. they do not stay in relation of supposition for the same thing, and what is predicated of one of them cannot be also predicated of the other one. This amounts, in turn, to explicate the problem of predication holding for general abstract and concrete terms in relation to meaning, and identity of meaning is established in terms of their definitions, therefore appealing to their *supposita simplex*. Thus according to Ockham *"every universal is one particular thing and [...] is not a universal except in its signification, in its signifying many things"*[8]. According to this quotation (which expresses Ockham's nominalism), the referents of terms render the distinction between universal and particulars, whereas it exists a common *suppositum*, a meaning determined as an affection of the soul. This is also confirmed by the thesis that a concrete term, being a predicate in a proposition, supposits for a form, as "white" for "whiteness" in "Socrates is white"[9]. The interesting thesis held by Ockham is therefore that the term in the universal form introduces the context of semantic validity of any

[7] Ockham (1349), these paragraphs are titled respectively: "On concrete and abstract names that are synonyms", "The correct account of abstract and concrete names" and "On the third mode of concrete and abstract names".

[8] Ockham (1349, par.14, p.78)

[9] Ockham (1349, par.63, p.189).

case of predication in which the same name is used in the concrete form: this means also that universals are applicable to concrete things insofar the latter resemble each other and the concept resembles each of them.

According to Aristotle the nature of abstract terms was obtained by a procedure of removal from the concrete predications, a position which finds many variants in Averroes, Aquinas, Scotus. By introducing explicitely the semantic relation of supposition in this context, Ockham describes an abstract term as allowing those concrete predications to be formulated, by displaying the context of semantic validity in which they can be performed, and thus working as their logical presupposition. The two approaches differ in the understanding of the conceptual priority of concepts and the hierarchy of predications.

3. From functions to types

After this debate, here exemplified by the theories of Aristotle and Ockham, an essential change is provided by the new Fregean paradigm: on the one hand, it preserves the essential role of predication in the definition of concepts; on the other hand, it provides a completely new (logical) form for such a notion, notoriously in terms of functions. The most relevant consequence of this theoretical shift is the connection to the problem of impredicativity and the development of the hierarchy of predications, which will lead to the invention of logical types.

The Fregean approach is a direct critique of the Aristotelian understanding of abstraction as the determination of a unity among many separeted enitites. Frege presents abstraction in relation to the notion of function in the *Begriffschrift*, in terms of the so-called *Abstraction Principle*: if in an expression a simple or compound sign has one or more occurrences and that sign is recognised as replaceable in all of its occurrences by some other sign, the invariant part is then a function and the replaceable part is its argument[10]. On the basis of this general principle, Frege develops the related notion of concept: a concept is not the result of a removal operation in the Aristotelian sense, rather it is the reference (*Bedeutung*) of a predicative expression[11]. This is obtained by explaining a predicate as an unsaturated object, whereas an argument for it is the instantiation of a concept saturating the former[12]. The classical judgemental form "S is P/P belongs to S" is thus changed to a new functional structure $F(x)$, where F plays the role of the predicate, to be evaluated for a certain object (the variable

[10] Frege (1879, par.9).

[11] Frege (1892). Notoriously, he explains moreover the reference of singular terms as the objects they stand for, and that of indicative sentences as their courses of values. See also his (1892b, p.193). Cf. Primiero (2004). .

[12] Frege (1891, p.6).

representing a place-holder for it). To build up a judgement means therefore essentially to *evaluate* a concept for an argument, and the distinction between concept and object determines the former as an abstract term. According to Frege therefore, a concept cannot play the role of the reference of the grammatical subject: it has to be converted into (or represented by) an object, which allows for its evaluation. According to this theory of *concepts as predicates* it is not even necessary for the predicate to be logically possible; the existence of an object instantiating a self-contradictory predicate is obviously a completely different matter.

The formal definition by abstraction of the old-fashioned notion of concept, establishing a class of given objects which satisfy an equivalence relation *R* such that reflexivity, identity and transitivity hold, can be provided also for functional expressions, for which one has to specify their course of values only by means of terms for which they can be evaluated: this makes the notion of function corresponding to its extension, namely the correlation of its arguments and its values[13]. A concept intended as an abstract term (represented by a function) has a range corresponding to the usual logical extension, i.e. the set of all objects falling under it. In *Funktion und Objekt* Frege avoids the essential paradox of the *Grundgesetze* by considering a first-level and a second-level form of abstraction, producing different kinds of functions[14]. Frege had essentially obtained the same result of the later Russellian Ramified Type Theory (RTT)[15], but treating *Wertverläufe* as ordinary objects, he allows a function to be applied on its very same course-of-values (corresponding to a function applied to its own graph), thus he cannot avoid the possibility of impredicative definitions. In the development of the notion of function for RTT, based on the distinction between the hierarchy of types of propositions and that of propositional functions, an important conceptual change occurs: the connection between predication and concepts as abstract entities is forgotten, and it is only partially recovered in terms of the notion of function as the stable part of the abstraction procedure. Correspondingly, the abstraction procedure in the formation of propositional functions is the basis of RTT, where it holds the double hierarchy of simple types and of orders, which also allows to abstract by universal quantification on all propositional functions of a certain order. The thesis that abstraction has now a rather different meaning is confirmed by the understanding of generality for functions as interpreted by Russell. In RTT, this property either means the assertion of any value of a propositional function, or the assertion that the

[13] The latter is then an ordered pair, corresponding to the notion of function as a graph; conceptually different is the simple dependent object considered before. In the next section, the notion of function involved in the analysis of type systems will be yet a different one. The clarification of the distinction of these three notions is due to B.G. Sundholm.

[14] Frege (1891, pp.26–27).

[15] Russell (1908).

function is always true: in the first case, one refers to real variables (any value of the function is asserted); in the second, to apparent variables (the function is always true). The notion of abstraction which leads to the function as independent object (and in turn to contradiction whenever the appropriate hierarchy of predications is not considered) acts on the values (real variables), what Russell called a proper *propositional function*[16]. Thus the type of functions does not only depend on the type of arguments, rather also on the type of apparent variables (place-holders)[17].

Typing procedures introduced by Russell are not just the solution to the problem of impredicativity: they present a new interpretation of the notion of function and a different approach to abstraction. In particular, abstract terms require to be explained on the basis of the logical notion of type, as objects of an higher level.

4. Abstraction from type-free to typed λ-calculus

The traditional interpretation of the notion of function due to Leibniz, Bernoulli and Euler, is that of an analytic expression in one or more variables. Frege and Russell formulated the logical notion based on the relation of predication, which refers to substitution and evaluation as its defining operations. In the Fregean interpretation a function stands also by itself as an independent object of individual type, defined by the correlation to its course of values. By the Russellian theory of types, functions as formal structures of predication are admissible on the basis of the order of their objects. This evolution led to a different model of function and to the notion of type: it restores the old-fashioned notion of function as rule rather than as graph, i.e. it consists of an operation from an argument to a value.

The formalization of abstraction in terms of functional expressions à la Russell substituting the notion of Fregean concepts, where real variables take the place of the abstraction procedure, is further exemplified in the simple typed λ-calculus developed by Church (1940). It combines the Russellian calculus and the operation of deramification, which removes all the orders on types. In a type-free structure the objects of study are both functions and arguments; the alphabet of such a calculus is formed by λ-terms, which are formal expressions for functions and for applications of functions. In this kind of calculus, actually

[16] Russell (1908, p.157).

[17] A condition which notoriously Russell restricted by formulating the Axiom of Reducibility (AR): for each formula f there is a formula g with a predicative type such that f and g are logically equivalent, where a type is predicative if none of its objects is of a higher order than the order of the elements of the class to which this object should belong.

everything is or is meant to represent a function, based on a composed process of *abstraction* and *application*[18]. Their combination is essential to the formulation of functions, and application is in fact the main operation, whereas abstraction is complementary. The system of operations is completed by *reduction*, consisting in the process of computing from a λ-abstracted term to its value. The model of abstraction here at hand is of a different kind than the Fregean notion of function: the result of abstraction is performed by an operator, and it produces a function rather than a predicate or a concept. Consider the numerical expression $2 + 3$, and its transformation into a function, by which one takes into account first the λ-term $(\lambda x.x + 3)2$ which is the β-expansion of the given numerical expression: in this transformation we have a certain argument (2) which is substituted by the argument-variable (x) via the *lambda*-abstractor. What is peculiar in this operation is that one has already the abstracted term (which in turn performs the role of an abstractor operator on values) without having necessarily the starting term from which one abstracts. The β-expansion in the λ-calculus is thus joined to the operation of removing an argument and it corresponds to the function construction. Respectively, instantiation of the latter corresponds to application plus β-reduction of the former[19]. The conceptual identity between the predicative part and the argument is relevant to the understanding of the notion of abstraction involved: the very same expression can perform the role of the operator and of its object. In this sense no term represents the result of an abstraction procedure, nor the context-determining term of predication (as in the case of the relation of supposition). The abstractor operators are such that they can be applied to functions without considering the order of progressively higher types: the abstraction is a pure operation, not a complete process and functions are first-class citizens. The resulting notion of function for these calculi is therefore considered in terms of evalutation (function values), rather than in terms of objects (abstract function).

The typed version of λ-calculi affects then the simple version in an essential way: every term of the calculus has now a normal form, i.e. every possible β-and η-reductions terminate, which makes the set of typable λ-terms entirely recursive. The introduction of types in order to describe the functional behaviour of the terms is relevant in two ways: first, it transforms the idea of abstraction connected to these calculi; second, it provides a bridge between the notion of function and the one of type. On the basis of the Brouwer-Heyting-Kolmogorov interpretation of propositions-as-proofs, in the typed version of this calculus a proof of an implication is a construction, and accordingly a construction of an implication is a function (from the proof of the antecedent to the proof of the consequent). The intrepretation of all operations in terms of their types, due to

[18] See e.g. Laan (1997, pp.4–5).
[19] Laan (1997, p.43).

the Curry-Howard isomorphism, leads to a different consideration of abstraction procedures. A useful way to explain this, which holds for all kind of typed systems, is to consider the *information*-content of the expressions: the explicit formulation in the syntax of constructions and bound variables in the environments represents the formal structure for which operations of abstraction can be defined[20]. Typed systems can in fact be intended as such that types for all variables and terms are fixed, and expressions contain *full type information* (whereas a so-called type-assignment system would not have such a full information in the basic syntax). The procedure which allow to transform a fully built term into a "core one", i.e. one which provide only the necessary type information, can be seen as an operation of erasing the domains of abstraction; conversely, the typability of terms consists in filling in a proof-trace with the missing elements. According to this relation between informational content of terms and procedures of abstraction, one needs now to distinguish between two different uses of "abstraction": abstraction as $\lambda a.M$ consists in a proof by generalization; an abstract type, will be instead the term obtained by means of an existential type. This distinction is of the greatest importance to understand the nature of abstraction for type systems: it is clearly based on the logical nature of types and on the essential connection which seems to hold between abstraction and information.

5. Other examples of typed systems

Other examples of abstraction procedures in typed systems show interesting properties connected to the explanations provided in the previous sections.

A *prototype proof*, whose notion was first formulated by Herbrand[21], is the proof of a universally quantified statement, whose verification is applicable to each specific instance of the quantified variable. It is executed by assuming a certain generic element of the set the quantification ranges over, in this way making the proof independent from that specific element. This shows the formulation of a *model* to be applied in different cases[22]. A second case in which the same notion of model is clearly involved in connection to abstraction is that of *abstract data types* (ADT). ADT are defined as a set of data values (*abstract data structure*) and associated operations (*interface*) that are considered inde-

[20] This results clearly in the different formulations known as Curry-style and Church-style typed λ-calculi. See e.g. Sørensen, Urzyczyn (2006, ch. 3).

[21] See Longo (2000, p.2).

[22] Longo (2000) provides a nice interpretation of the notion of prototypic proof in type systems under the propositions-as-types interpretation, where one can consider this kind of proofs as λ-terms: this is done by considering a simple type called *generic*, i.e. such that it can be assumed as a variable, and whose proof is provided by a specific instance called "parametric", i.e. which can be uniformly substituted.

pendently from any specific implementation. These data represent the result of an operation of abstraction intended as the process of deleting unnecessary details from necessary characteristics (i.e. again to formulate a model by removing information) in order to solve a problem (i.e. to provide the correct operations on a certain set of data). Clearly, the process of obtaining the relevant data is exemplified by the *abstract predicates* entering into the solution of the problem; as in the case of the application function, this process of abstraction is never taken separately from the determination of the operations which are to be performed on the empty schema. To mention a last example: by *polytypic abstraction* one understands instead formalizations and verifications abstracted in respect to a large class of datatypes, which is especially relevant in functional programming. A simple and nice example is that of the function *map* in the Hindler-Miller type system (map: $\forall A, B.(A \rightarrow B) \rightarrow (list(A) \rightarrow list(B))$) which provides a structure of transformation of data lists in other kind of data, with an untouched schema or model, irrelevant to the kind of data instantiated as object of that function[23]. Also in this case we are treating with a procedure of abstraction by which an empty model is obtained, able to implement all the different data of a certain range of (distinct) equivalent types, and to be used in terms of application[24].

6. Types, abstraction and information

The connection between types and information in the light of the abstraction procedures can be reconsidered under the syntactic-semantic method of Martin-Löf's Type Theory. The relevant notion of type in such a system is related to abstraction both from the philosophical and the purely formal viewpoints: its interpretation provides also interesting comments on the analysis done up to now. The main argument is that the syntactic procedures (rule-based operations) are not the unique way to account for abstraction: on the one hand one has to consider the removal operation by which the notion of emtpy (polymorphic) model is obtained; on the other hand, the notion of abstract (meaning-determining and predicative-component) object is involved by the definition of types themselves. In both cases, the notion of information plays a key-role.

At the syntactic predicative level, the notion of abstraction is satisfied in terms of rules. Abstraction and application rules concern the informative con-

[23] See Pfeifer, Ruess (1998).

[24] This obviously requires that the basic distinction between monomorphic and polymorphic languages holds; its better application is given by the so called *parametric polymorphism*, according to which the same object or function can be used uniformly in different type contexts without changes (provided all data are represented). See Cardelli, Wegner (1985, p.477).

tent of expressions in terms of the specific constructions. The rule of Π-introduction defines an independent object of the lowest individual type

$$\frac{\begin{array}{c}[x : A]\\ b(x) : B(x)\end{array}}{\lambda((x)b(x)) : (\Pi x : A)B(x)}$$

with the related Π-elimination or Application rule; on the other hand, abstraction consists in functions formation of the higher type, i.e. if x is a variable of type A and b is a term of type B, then $(x)b$ is a term of type $A \to B$:

$$\frac{\begin{array}{c}[x : A]\\ b : B\end{array}}{(x)b : A \to B}$$

explained by the ordinary β-rule, expressing what does it mean to apply an abstraction to an object in A:

$$\frac{a : A \quad b : B[x : A]}{((x)b)(a) = b[x/a] : B[x/a]}$$

The distinction between universalization and abstraction on contents is therefore clearly stated in terms of syntactic operations. The object of abstraction is here the informative content of judgements, as stated by the *Forget-restore Principle*[25]: the principle says that to build up an abstract concept from a raw flow of data, one must disregard some information, and an abstraction is constructive when the information forgotten can be restored at will. Under this principle, abstraction corresponds to an operation of forgetting from irrelevant computational information, whereas instantiation is the restoring of such information. In particular, by a procedure of abstraction one obtains the transition from the monomorphic to the polymorphic versions of the theory.

Operational abstraction leads to consider higher types as abstract terms themselves. By insisting on the procedure of abstraction in terms of removing the informational content of the constructions, one formulates the judgement declaring the truth of the involved types[26]:

$$\frac{a : A}{A \ \mathit{true}}$$

[25] Valentini (1998).
[26] This formulation simply expresses the propositions-as-sets interpretation, see Nordström, Petersson, Smith (1990, p.37).

A multi-level typed λ-calculus can be provided for rigorous treatment of judgements of the form "*A* true", on the basis of canonical expressions of the form $a : A$. In this sense, abstraction procedures allow for type-expressions of the form *A type*, provided that abstraction applies as follows:

$$\frac{A \quad set(/prop)}{A \ type}$$

This procedure leads to the analysis of types as independent objects of predication; they are presuppositions for judgements in which those types are used[27]. The explanation of this abstraction procedure is given accordingly to the syntactic-semantic method of Martin-Löf's Type Theory, in which types come conceptually before their objects. This means that the conceptual order between types and their instantiation goes from the former to the latter (i.e. types are abstract terms in respect to their constructions), whereas in the order of knowledge one proceeds from existential predications to their types by means of an abstraction procedure. Higher types, i.e. of the monomorphic kind, should therefore be explained as abstract terms, in connection to predication and semantic context: they recover essential features of abstraction lost in the functional interpretation.

Giuseppe Primiero
Centre for Logic and Philosophy of Science
Universiteit Gent
Blandijnberg 2
B-9000 Gent, Belgium
Giuseppe.Primiero@UGent.be

References

Aristotle. *Works*. The Loeb Classical Library.

Cardelli, L., & Wegner, P. (1985). On Understanding Types, Data Abstraction and Polymorphism. *Computing Surveys, 17* (4), 471–522.

Church, A. (1940). A formulation of the simple theory of types. *The Journal of Symbolic Logic, 5*, 56–68.

[27] See Primiero (forth).

Frege, G. (1891). Funktion und Begriff. Lecture given on 1891, at Jenaischen Gesellschaft für Medizin und Naturwissenschaft. In M. Textor (Ed.) (2002). *Funktion – Begriff – Bedeutung* (pp. 2–22). Vandenhoeck & Ruprecht.

Frege, G. (1892). Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, NF 100, 25–50. In M. Textor (Ed.) (2002). *Funktion – Begriff – Bedeutung* (pp. 23–46). Vandenhoeck & Ruprecht.

Frege, G. (1892b). Über Begriff und Gegenstand. *Vrijschrift für wissenschaftliche Philosophie, 16*, 192–205. In M. Textor (Ed.) (2002). *Funktion – Begriff – Bedeutung* (pp. 47–60). Vandenhoeck & Ruprecht.

Laan, T. (1997). *The Evolution of Type Theory in Logic and Mathematics*. PhD Dissertation Thesis. Technische Universiteit Eindhoven. Enschede: Print Partners Ipskamp.

Longo, G. (2000). Prototype Proofs in Type Theory. *Mathematical Logic Quarterly, 46* (3).

Martin-Löf, P. (1993). *Philosophical Aspects of Intuitionistic Type Theory*. Unpublished notes of lectures given at the Faculteit Wijsbegeerte Leiden.

Nordström, B., & Petersson, K., & Smith, J. (1990). *Programming in Martin-Löf 's Type Theory – An Introduction*. Oxford: Clarendon Press.

Ockham, W. (1349). *Summa Logicae – Part One: Logic of Terms*. Translated and introduced by M. J. Loux. Notre Dame/London: University of Notre Dame Press.

Pfeifer, H., & Ruess, H. (1998). Polytipic Abstraction in Type Theory. *Informal Proceedings of Workshop on Generic Programming (WGP98)*. Marstrand, Sweden.

Primiero, G. (2004). The Determination of Reference in a Constructive Setting. *Giornale di Metafisica, 26* (3), 483–502.

Primiero, G. (forth). Presuppositions, Assumptions, Premises. Forthcoming.

Russell, B. (1908). Mathematical Logic as based on the Theory of Types. In J. van Heijenoort (Ed.) (1999). *From Frege to Gödel – A source book in mathematical logic, 1879–1931* (pp. 150–182). ToExcel Press.

Sørensen, M. H., & Urzyczyn, P. (2006). *Lectures on the Curry-Howard Isomorphism*. Studies in Logic and the Foundations of Mathematics, vol. 149. Amsterdam: Elsevier.

Valentini, S. (1998). The forget-restore principle: a paradigmatic example. In Sambin, Smith (Eds.), *Twenty-five years of Construcitve Type Theory* (pp. 275–283). Oxford: Clarendon Press.