THE BULLETIN OF SYMBOLIC LOGIC
Volume 8, Number 2, June 2002

TYPES IN LOGIC AND MATHEMATICS BEFORE 1940

FAIROUZ KAMAREDDINE, TWAN LAAN, AND ROB NEDERPELT

Abstract. In this article, we study the prehistory of type theory up to 1910 and its development between Russell and Whitehead's *Principia Mathematica* ([71], 1910–1912) and Church's simply typed λ-calculus of 1940. We first argue that the concept of types has always been present in mathematics, though nobody was incorporating them explicitly as such, before the end of the 19th century. Then we proceed by describing how the logical paradoxes entered the formal systems of Frege, Cantor and Peano concentrating on Frege's *Grundgesetze der Arithmetik* for which Russell applied his famous paradox[1] and this led him to introduce the first theory of types, the Ramified Type Theory (RTT). We present RTT formally using the modern notation for type theory and we discuss how Ramsey, Hilbert and Ackermann removed the orders from RTT leading to the simple theory of types STT. We present STT and Church's own simply typed λ-calculus ($\lambda{\to}_C$[2]) and we finish by comparing RTT, STT and $\lambda{\to}_C$.

§1. Introduction. Nowadays, type theory has many applications and is used in many different disciplines. Even within logic and mathematics, there are many different type systems. They serve several purposes, and are formulated in various ways. But, before 1903 when Russell first introduced a type theory (see appendix of [62]), there were no formulations of any type theory. It is only since the second half of the twentieth century that we see explosions of type theories. In this article, we follow the evolution of type theory up to 1940. We give a historical account as to why formulations of type theory came into being and we describe the very first formulation of type theory: Russell and Whitehead's ramified theory of types (based on Russell's work [63] of 1908 which succeeded [62]). In addition, we

Received October 13, 2000; revised December 17, 2001.

We are grateful for the useful feedback from and discussions with Henk Barendregt, Andreas Blass, Ivor Grattan-Guinness, Roger Hindley and Joe Wells. Randall Holmes read our work in thorough detail, provided extremely useful comments and implemented an impressive proof checker for the system of Principia Mathematica which helps one to see the extent to which Russell and Whitehead were successful in their early attempt at the logical formalization of mathematics.

[1]Russell discovered his paradox when he read Cantor's work.

[2]We write $\lambda{\to}_C$ for the original calculus of Church as presented in [14]. Note that this is different from the calculus $\lambda \to$ used in frameworks like the Barendregt cube and the pure type systems found in [3].

1079-8986/02/0802-0001/$7.10

describe the simple theory of types that resulted from simplifications made by Hilbert and Ackermann and Ramsey to the ramified theory of types. Finally, we give the simply typed λ-calculus based on Church's seminal paper of 1940. It is important to stress that we do not give an extensive history of the subject.[3] Other developments deserve attention and we refer the reader especially to Ivor Grattan-Guinness's book [35] for an excellent historical account of many of the concepts discussed in this paper. We also refer the reader to Cocchiarella's work in [16, 17] and Landini's book [49].

Following the historical line from Frege (1879) and urged by the threat of the paradoxes, Russell and Whitehead developed the ramified theory of types [62, 63, 71] which was later simplified (or deramified) by Ramsey [59] and Hilbert and Ackermann [37] into the simple theory of types. The simple theory of types existed before the λ-calculus was invented by Church in 1932. Nevertheless, nowadays, when one refers to simple type theory, one usually means Church's simply typed λ-calculus of 1940. It should be noted furthermore, that Russell's type structure was different from that of Church. The former was set-based with linear sequences of types. The latter was function-based.

Our article is not only intended to introduce the prehistory of type theory (up to 1910) and its development between *Principia Mathematica* ([71], 1910–1912) and Church's simply typed λ-calculus of 1940, but also we will present Russell's ramified theory of types and the simple theory of types due to Ramsey, in a modern setting.[4] The presentation of the ramified theory of types in a modern setting was already given in [48] but there, neither the deramification nor the connection of the ramified type theory with that of Church's simple theory of types was given. Similarly, although these theories have already been described in a modern framework, the relation between the modern description and the original system has not always been made clear. This is particularly the case because the original systems are quite far from the modern framework with respect to notation, level of formality and/or purpose. We will describe the ramified and simple theories of types within the modern framework in such a way that:

- We respect the ideas and the philosophy underlying the original system;

[3] Curry, in his work on combinatory logic, introduced before 1940 an influential notion of typing that is still used nowadays when one refers to typing à la Curry as opposed to typing à la Church. Similarly, in 1937, Quine in [55] introduced his New Foundations which retained typing axioms, but abandoned the idea of representing types formally as Russell did. Quine's NF presupposes the very simple linear type theory with types 0 for individuals, 1 for sets of individuals, 2 for sets of sets of individuals, etc.

[4] Note that there was a large unsatisfaction with ramified types and the reducibility axiom and this led to various calls for deramification. We already mentioned Hilbert and Ackermann, there is also the work of Leon Chwistek amongst others. In this paper, we concentrate on the simple theory of types as envisaged by Ramsey.

- We meet contemporary requirements on formality and accuracy.

The explicit and formal use of types (and thus an early form of what is presently called "type theory") was originally intended to prevent the paradoxes that occurred in logic and mathematics at the end of the 19th and the beginning of the 20th century. But it was not the only method developed for this purpose. Another tool was the fine-tuning of Cantor's Set Theory [9, 10] by Zermelo [73], and the iterative conception of set (see [7]) that resulted from the foundation axiom of Zermelo-Fraenkel's set theory ZF. Although it was clear that in ZF, the foundation axiom does not help in avoiding the paradoxes, it was added as a technical refinement. The separation axiom which replaced the unrestricted *comprehension axioms* is the one responsible for avoiding the paradoxes. This axiom goes as follows [23, 2]:

(*Comprehension*)
For each open well-formed formula Φ, $\exists y\, \forall x\, [(x \in y) \iff \Phi(x)]$ where y is not free in $\Phi(x)$.

This unrestricted comprehension leads to a paradox by taking $\Phi(x)$ to be $\neg(x \in x)$:

$$\exists y\, \forall x\, [(x \in y) \iff \neg(x \in x)] \Longrightarrow \exists y\, [(y \in y) \iff \neg(y \in y)].$$

Such a comprehension axiom assumes that each open well-formed expression determines a concept whose extension exists and is the set of all those elements which satisfy the concept. *Iterative sets* were proposed to avoid the paradox and came to being by altering not the language, but the axioms of the theory. The most straightforward such theory is ZF (Zermelo-Fraenkel) where the axioms are made to fit the limitation of size doctrine. As an example, the above comprehension principle is altered to the following:

(*Separation*)
For each open well formed formula Φ, $\exists y\, \forall x\, [(x \in y) \Longrightarrow (x \in z) \wedge \Phi]$ where y does not occur in Φ.

It is this new axiom which is responsible for the elimination of the paradox: to prove the existence of $\{\, x : \neg(x \in x) \,\}$ we need a z big enough so that $\{\, x : \neg(x \in x) \,\}$ is included in z. But we cannot show the existence of such a z. More precisely the paradox is restricted in ZF as follows:
Take $\Phi(z)$ to be $\neg(z \in z)$, and take $y = \{\, x : (x \in z) \wedge \neg(x \in x) \,\}$.

- If $(y \in y) \Longrightarrow (y \in z)$ and $\neg(y \in y)$ contradiction,
- If $\neg(y \in y) \Longrightarrow$
  - if $(y \in z) \Longrightarrow (y \in y)$ contradiction,
  - if $\neg(y \in z)$ then we are fine.

Note however that we still have the syntactical ability to consider whether a set belongs to itself or not, but we are not committed to any set actually belonging to itself. For example, if we take $\Phi(x)$ to be $x \in x$ then

$$\exists z\ \forall x\ [(x \in z) \iff (x \in \alpha \wedge x \in x)];$$

in this case, although $x \in x$ is well-formed, it is likely to be false in the intended interpretation for any value of x. In the middle period of the development of ZF, it was felt that the following foundation axiom (which is independent of and consistent with all other axioms of ZF) has to be added:[5]

(FA) $$(\exists x)(x \in a) \Longrightarrow (\exists x \in a)(\forall y \in x)\neg(y \in a).$$

As a corollary of (FA), we have that there is no set a which has itself as its only element, for if there was then take $x = a$ in the antecedant of (FA) above and you get $(\exists x \in a)(\forall y \in x)\neg(y \in a)$, which is absurd.

It is worth pointing out that although very different conceptually, both the simple theory of types and ZF (which includes (FA)), give rise to an iterative concept of set. That is, both require the elements of a set be present before a new set can be constructed [7]. We cannot however stop our discussion of set theory here. Quine's stratification in NF [55], and ML [56], (two non-iterative set theories) are sufficiently type-like to merit some discussion. Quine restricted the axiom of comprehension, to obtain the following:

(SCP) $$\exists x\ \forall y\ [(y \in x) \iff \Phi(y)]$$

where x is not free in $\Phi(y)$ and $\Phi(y)$ is stratified.[6]

Quine's NF has attracted a lot of research. Specker [67] refuted the axiom of Choice in NF, Jensen [41] established that NF with Urelements is consistent (even when augmented with Choice, Infinity and unrestricted mathematical induction). However, consistency of NF remains an open problem. Moreover, NF is weak for mathematical induction (for propositions not expressible in type theory). Also, NF is said to lack motivation because its axiom of comprehension is justified only on technical grounds and one's mental image of set theory does not lead to such an axiom. To overcome some of the difficulties, Quine replaced (SCP) by two axioms, one for class existence and one for elementhood. The rule of class existence provides for the existence of the classes of all elements satisfying any condition Φ, stratified or not. The rule of elementhood is such as to provide the elementhood of just those classes which exist for NF. Therefore, the two axioms of comprehension of ML:

---

[5]This changed in the 1980s when Peter Aczel introduced his non-well founded set theory which relied on the Anti-Foundation axiom.

[6]Assume a first-order theory where for each primitive predicate $F(x_1, \dots, x_n)$, we have integer constants $F_1, \dots, F_n$. A formula Φ in the language of that theory is said to be *stratified* if there is an integer-valued function σ with domain the set of variables appearing in ϕ with the property that in each atomic formula $F(x_{i_1}, \dots, x_{i_n})$ which appears in ϕ, and each integer $1 \leq j \leq n$, we have $\sigma(x_{i_j}) - \sigma(x_{i_1}) = F_j - F_1$.

(*Comprehension by a set*)

$\exists y\ \forall x\ (x \in y \iff \Phi(x))$, where x and y range over sets, $\Phi(x)$ is stratified with set variables only in which y does not occur free.

(*Impredicative comprehension by a class*)

$\exists y\ \forall x\ (x \in y \iff \Phi(x))$, where x ranges over sets, $\Phi(x)$ is any formula in which y does not occur free.

ML was liked both for the manipulative convenience we regain in it and the symmetrical universe it furnishes. The earlier version of the first edition of [56] was subject to the Burali-Forti paradox—the well ordered set Ω of all ordinals has an ordinal which is greater than any member of Ω and hence is greater than Ω. In the second edition of [56], Quine corrected the axiomatization of *ML* (following a suggestion of Hao Wang) so that it is demonstrably consistent if *NF* is consistent. This latter version does not face the Burali-Forti paradox.

The approach of type theory however, is completely different from the set-theoretical approach. First, in the type theoretical approach, it is the language that is altered in order to avoid the paradox, and not the axioms.[7] Moreover, since Church's λ-calculus being extended with simple types in 1940, type theory has continued to focus on the notion of *function* in logic and mathematics. Since 1940, functions have remained one of the main objects of study for type theorists.

The historical remarks in this article have been taken from various resources. The most important ones are [6, 19, 68, 44, 53, 69, 72].

In Section 2 we discuss the prehistory of type theory. We first argue that the concept of types has always been present in mathematics, though nobody

[7]The first two accounts of avoiding the paradox by restricting the language were due to Russell and Poincaré. They both disallowed impredicative specification: only predicative specification (as will be defined below) was to be permitted. Russell's own solution (in [63]) was to adopt the vicious circle principle which can be roughly stated as follows: "*No entity determined by a condition that refers to a certain totality should belong to this totality*". Poincaré (in [54]) took refuge in banning "*les définitions non prédicatives*" which were taken by him to be: *Definitions by a relation between the object to be defined and all individuals of a kind of which either the object itself to be defined is supposed to be a part or other things that cannot be themselves defined except by the object to be defined.* So both Russell and Poincaré required only predicative sets to be considered, where $A = \{\, x : \Phi(x) \,\}$ is predicative if and only if Φ contains no variable which can take A as a value. This helps because it is otherwise very easy to get a vicious circle fallacy if we let the arguments of a certain propositional function (or the elements of a set) presuppose the function (or the set) itself. Russell's and Poincaré's solution was to use predicative comprehension, instances of which start with individuals, then generate sets, then new sets and so on as in the following example: Take 0 at level 0, $\{0\}$ at level 1, $\{0, \{0\}\}$ at level 2, and so on. Russell's ramified theory of types in Principia Mathematica applied the vicious circle principle, assuming all the elements of the set before constructing it. This theory obviously overcomes the paradox for the sentence Φ denoting $\neg(y \in y)$ is not predicative.

was incorporating them explicitly as such, before the end of the 19th century (Section 2.1). We study the way in which types implicitly occurred in logic and mathematics before there was an explicit theory of types. Then, we proceed by describing how the logical paradoxes entered the formal systems of Frege, Cantor and Peano in Section 2.2. We pay special attention to the formalisation of logic that is made in Frege's *Begriffsschrift* [25] and *Grundgesetze der Arithmetik* [28, 31], as in this system many basic ideas are presented that are later used in type theory. Moreover, the system of *Grundgesetze der Arithmetik* is the one for which Russell derives his famous paradox, and this paradox was the reason for Russell to introduce the first theory of types.

This first type theory is the subject of Section 3. Whitehead and Russell present their theory, the Ramified Type Theory (RTT), in an informal way. Several rough descriptions of this theory have been given in the literature (see for instance [14, 15, 37, 59]) but we present a formalisation of RTT given in [48] that is directly based on the presentation of RTT in Whitehead and Russell's *Principia Mathematica* ([71], 1910–12). The construction of this formalisation is not a simple task. Whitehead and Russell do not present a clear syntax for their so-called *propositional functions* in [71], neither do they make a clear difference between syntax and semantics. We constantly explain/defend our formalisation by using actual text from *Principia Mathematica*. We explain how the formal definition of propositional function is faithful to the original ideas exposed in *Principia Mathematica* and how the formalisation of the notion of propositional function makes it possible to express the notion of substitution of *Principia Mathematica* in terms of λ-calculus.

In 1926, Ramsey [59] proposes an important simplification of RTT, the *simple* theory of types. This simple type theory has become the basis for many modern type systems, and for the simply typed λ-calculus of Church [14]. The simplification consisted of the removal of one of the two hierarchies from the RTT. The hierarchy of types is maintained, while the hierarchy of orders is removed. In Section 4 we discuss this process of so-called *deramification*.[8]

In Section 5 we follow this process of deramification to present Hilbert and Ackermann's [37] and Ramsey's [59] simple theory of types STT. We also present the well-known simple theory of types of Church $\lambda{\rightarrow}_C$ [14]. We compare RTT, STT and $\lambda{\rightarrow}_C$.

We conclude in Section 6.

[8]Note that though the orders do not occur in the mainstream of type theories, they still provide an important intuition for logicians and play an important role in "categorising" logical theories: first-order, second-order, higher-order. For a discussion of the use of orders in modern systems, and its relation to Russell's notion of orders, the reader is referred to [43].

§2. Prehistory of types. In this section, we discuss the development of type theory before it was actually baptised. This may sound like a contradiction. But types have played an important (though not very apparent) role in mathematics even before the theory of types was explicitly introduced by Russell in 1908 [63]. Moreover, knowledge of the development of logic and mathematics before 1908, and especially of the occurrence of the logical paradoxes at the turn of the 20th century, provides insight in the way in which Russell and others formulated their theories of types.

When the first formalisations of parts of mathematics and logic appeared, the types were left implicit. Cantor's Set Theory [9, 10], Peano's formalisation of the theory of natural numbers in [51], and Frege's *Begriffsschrift* [25] and *Grundgesetze der Arithmetik* [28, 31] did not have a formal type system. The type of an object is indicated by means of natural language ("Let *a* be a proposition") or is taken for granted. Types were informally present in the background of these theories, but a formal representation of the types was not incorporated: one could say that they were separated from logic and mathematics.

However, even without a formalisation of the notion of types, the introduction of formal language had considerable advantages in the description of mathematical notions. The formalisation made it easier to give a precise definition of important abstract concepts, like the concept of function. The precise formulation allowed for a generalisation of the notion of function to include not only functions that take numbers as an argument, and return a number, but also functions that can take and return other sorts of arguments (like propositions, but also functions). Unfortunately, this also allowed logical paradoxes to enter the formal theory, without the (informal) type mechanism being able to prevent that.

2.1. Paradox threats. The most fundamental idea behind type theory is being able to distinguish between different classes of objects (*types*).[9] Until the end of the 19th century it had hardly ever been necessary to make this ability explicit. The mathematical language itself was predominantly informal, and so was the use of classes of objects.

It is, however, difficult to argue that there were no types before Russell "invented" them in 1903. Already around 325 B.C., Euclid began his *Elements* (page 153 of [22]) with the following primitive definitions:

1. A *point* is that which has no part;
2. A *line* is breadthless length.

From these two basic notions of "point" and "line", Euclid defined more complex notions, like the notion of "circle":

[9]Note that it is controversial whether types should be literally taken to be classes of objects. We do not use this correspondence literally.

15. A *circle* is a plane figure contained by one line such that all the straight lines falling upon it from one point among those lying within the figure are equal to one another.

At first sight, these three observations are mere definitions. But these three pieces of text do not only *define* the notions of point, line and circle, they also show that Euclid *distinguished* between points, lines and circles. Throughout the *Elements*, Euclid always mentioned to which class an object belonged (the class of points, the class of lines, etc.). In doing so, he prevented undesired results, like the intersection of two points (instead of two lines).

Undesired results? Euclid himself would probably have said: *impossible* results. When talking of an intersection, intuition implicitly forced him to think about what we would nowadays call the *type* of the objects of which he wanted to construct the intersection. As the intersection of two points is not supported by intuition, he did not even *try* to undertake such a construction.

Euclid's attitude to, and implicit use of type theory was maintained by the mathematicians and logicians of the next twenty-one centuries. From the 19th century on, mathematical systems became less intuitive, for several reasons:

1. The system itself was complex, or abstract. An example was the theory of convergence in real analysis;
2. The system is a formal system, for example, the formalisation of logic in Frege's *Begriffsschrift*;
3. (In the second half of the 20th century:) It is not a human being working with the system, but something with less intuition, in particular: a computer.

We will call these three situations *paradox threats*. In all these cases, there is not enough intuition to activate the (implicitly present) type theory to warn against an impossible situation. One proceeds to reason within the impossible situation and then obtains a result that may be wrong or paradoxical: an *undesired* situation. We mention examples related to the three situations above:

S 1. The controversial results on convergence of series in analysis obtained in the 17th and 18th century, due to lack of knowledge on what real numbers actually are;

S 2. The logical paradoxes that arose from self-application of functions. Self-application *may lead* to intuitively impossible situations, but this is easily forgotten when working in a formal system in which such self-application can be expressed. The result is undesirable: a logical paradox;[10]

S 3. An untyped computer program may receive instructions from a not too watchful user to add the number 3 to the string `four`. The computer, unaware of the fact that `four` is not a number, starts his calculation. It is not programmed to handle the calculation of $3 + \texttt{four}$. The result of this calculation is unpredictable. The computer may: (a) give an answer that is clearly wrong; (b) give no answer at all; (c) give an answer that is not so clearly wrong (for example, 6); or (d) accidentally give the right answer. Especially situation (c) is highly undesirable.

The example S 2 is the main subject of the next section (Section 2.2).

2.2. Paradox threats in formal systems. In the 19th century, the need for a more precise style in mathematics arose. Controversial results had appeared in analysis. Many of these controversies were solved by the work of Cauchy. For instance, he introduced a precise definition of convergence in his *Cours d'Analyse* [11]. Due to the more exact definition of real numbers given by Dedekind [21], the rules for reasoning with real numbers became even more precise.

In 1879, Frege published his *Begriffsschrift* [25], in which he presented the first formalisation of logic that was uncommonly precise for those days. Until then, it had been possible to make mathematical and logical concepts more clear by textual refinement in the natural language in which they were described. Frege was not satisfied with this:

> " ... I found the inadequacy of language to be an obstacle; no matter how unwieldy the expressions I was ready to accept, I was less and less able, as the relations became more and more complex, to attain the precision that my purpose required."
>
> (*Begriffsschrift*, Preface)

Frege therefore presented a completely formal system, whose

> "first purpose 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."
>
> (*Begriffsschrift*, Preface)

[10]Note that there are logical systems in which self-application is both consistent and useful. Consider untyped λ-calculus or second-order polymorphic λ-calculus, not to mention systems related to Quine's "New Foundations". For a modern intuition, trained in systems that arose during or after the developments up to 1940, self-application is much more obviously problematic than it was then (though certainly it was problematic for Russell!) For example, self-application appears not to have been essentially problematic for Curry, who was developing combinatory logic during that period.

2.2.1. *Functions and their course of values.* The introduction of a very general definition of function was the key to the formalisation of logic. Frege defined what we will call the *Abstraction Principle*:

ABSTRACTION PRINCIPLE 1.

> *"If in an expression,* [...] *a simple or a compound sign has one or more occurrences and if we regard that sign as replaceable in all or some of these occurrences by something else* (*but everywhere by the same thing*), *then we call the part that remains invariant in the expression a function, and the replaceable part the argument of the function."*
>
> (*Begriffsschrift*, *Section 9*)

Up to this section in Begriffsschrift, Frege put no restrictions on what could play the role of an argument. An argument could be a number (as was the situation in analysis), but also a proposition, or a function. Similarly, the result of applying a function to an argument did not necessarily have to be a number. (In Section 11 of his Begriffsschrift, Frege makes restrictions as we will see below.) Functions of more than one argument were constructed by a method that is very close to the method presented by Schönfinkel [64] in 1924:

ABSTRACTION PRINCIPLE 2.

> *"If, given a function, we think of a sign*[11] *that was hitherto regarded as not replaceable as being replaceable at some or all of its occurrences, then by adopting this conception we obtain a function that has a new argument in addition to those it had before."*
>
> (*Begriffsschrift*, *Section 9*)

With this definition of function, two of the three possible paradox threats mentioned on p. 192 occurred:

1. The generalisation of the concept of function made the system more abstract and less intuitive. The fact that functions could have different types of arguments is at the basis of the Russell Paradox;
2. Frege introduced a *formal* system instead of the informal systems that were used up till then. Type theory, that would be helpful in distinguishing between the different types of arguments that a function might take, was left informal.

So, Frege had to proceed with caution. And so he did, at this stage. He remarked that

> "if the [...] letter [sign] occurs as a function sign, this circumstance [should] be taken into account."
>
> (*Begriffsschrift*, Section 11)

[11] We can now regard a sign that previously was considered replaceable as replaceable also in those places in which up to this point it was considered fixed. [footnote by Frege]

This could be interpreted as if Frege was aware of some typing rule that does not allow to substitute functions for object variables or objects for function variables. In his paper *Function and Concept* [27], Frege more explicitly stated:

> " Now just as functions are fundamentally different from objects, so also functions whose arguments are and must be functions are fundamentally different from functions whose arguments are objects and cannot be anything else. I call the latter first-level, the former second-level."
>
> (*Function and Concept*, pp. 26–27)

A few pages later he proceeded:

> "In regard to second-level functions with one argument, we must make a distinction, according as the role of this argument can be played by a function of one or of two arguments."
>
> (*Function and Concept*, p. 29)

Therefore, we may safely conclude that Frege avoided the two paradox threats in the *Begriffsschrift*. In *Function and Concept* we even see that he was aware of the fact that making a difference between first-level and second-level objects is essential in preventing certain paradoxes:

> "The ontological proof of God's existence suffers from the fallacy of treating existence as a first-level concept."
>
> (*Function and Concept*, p. 27, footnote)

The *Begriffsschrift*, however, was only a prelude to Frege's writings. In *Grundlagen der Arithmetik* [26] he argued that mathematics can be seen as a branch of logic. In *Grundgesetze der Arithmetik* [28, 31] he actually described the elementary parts of arithmetics within an extension of the logical framework that was presented in the *Begriffsschrift*.

Frege approached the paradox threats for a second time at the end of Section 2 of his *Grundgesetze*. There he defined the expression "the function $\Phi(x)$ has the same course-of-values as the function $\Psi(x)$" by

> "the functions $\Phi(x)$ and $\Psi(x)$ always have the same value for the same argument."
>
> (*Grundgesetze*, p. 7)

Note that functions $\Phi(x)$ and $\Psi(x)$ may have equal courses-of-values even if they have different definitions. For instance, let $\Phi(x)$ be $x \wedge \neg x$, and $\Psi(x)$ be $x \leftrightarrow \neg x$, for all propositions x. Then $\Phi(x) = \Psi(x)$ for all x. So $\Phi(x)$ and $\Psi(x)$ are different functions, but have the same course-of-values.

Frege denoted the course-of-values of a function $\Phi(x)$ by $\grave{\varepsilon}\Phi(\varepsilon)$.[12] The definition of equal courses-of-values could therefore be expressed as

$$\grave{\varepsilon}f(\varepsilon) = \grave{\varepsilon}g(\varepsilon) \longleftrightarrow \forall a[f(a) = g(a)]. \tag{1}$$

In modern terminology, we could say that the functions $\Phi(x)$ and $\Psi(x)$ have the same course-of-values if they have the same *graph*.

Frege did not provide a satisfying intuition for the formal notion of course-of-values of a function. He treated courses-of-values as ordinary objects. As a consequence, a function that takes objects as arguments could have its own course-of-values as an argument. In modern terminology: a function that takes objects as arguments can have its own graph as an argument. All essential information of a function is contained in its graph. So intuitively, a system in which a function can be applied to its own graph should have similar possibilities as a system in which a function can be applied to itself. Frege excluded the paradox threats from his system by forbidding self-application, but due to his treatment of courses-of-values these threats were able to enter his system through a back door.

2.2.2. *The Russell Paradox in the Grundgesetze.* In 1902, Russell wrote a letter to Frege [61], in which he informed Frege that he had discovered a paradox in Frege's *Begriffsschrift*. Russell gave his well-known argument, defining the propositional function $f(x)$ by $\neg x(x)$ (in Russell's words: "to be a predicate that cannot be predicated of itself"). He assumed $f(f)$. Then by definition of f, $\neg f(f)$, a contradiction. Therefore: $\neg f(f)$ holds. But then (again by definition of f), $f(f)$ holds. Russell concluded that both $f(f)$ and $\neg f(f)$ hold, a contradiction.

Only six days later, Frege answered Russell that Russell's derivation of the paradox was incorrect [24]. He explained that the self-application $f(f)$ is not possible in the *Begriffsschrift*. $f(x)$ is a function, which requires an *object* as an argument, and a function cannot be an object in the *Begriffsschrift* (see 2.2.1).

In the same letter, however, Frege explained that Russell's argument could be amended to a paradox in the system of his *Grundgesetze*, using the course-of-values of functions. Frege's amendment was shortly explained in that letter, but he added an appendix of eleven pages to the second volume of his

[12] This may be the origin of Russell's notation $\hat{x}\Phi(x)$ for the class of objects that have the property Φ. According to a paper by J. B. Rosser [60], the notation $\hat{x}\Phi(x)$ has been at the basis of the current notation $\lambda x.\Phi$ in λ-calculus. Church is supposed to have written $\wedge x\Phi(x)$ for the function $x \mapsto \Phi(x)$, writing the hat in front of the x in order to distinguish this function from the class $\hat{x}\Phi(x)$. For typographical reasons, the $\wedge$ is supposed to have changed into a λ. On the other hand, J. P. Seldin informed us [66] that he had asked Church about it in 1982, and that Church had answered that there was no particular reason for choosing λ, that some letter was needed and λ happened to have been chosen. Moreover, Curry had told him that Church had a manuscript in which there were many occurrences of λ already in 1929, so three years before the paper [12] appeared.

Grundgesetze in which he provided a very detailed and correct description of the paradox.

The derivation goes as follows (using the same argument as Frege, though replacing Frege's two-dimensional notation by the nowadays more usual one-dimensional notation). First, define the function $f(x)$ by:

$$\neg\forall\varphi\ [(\grave{\alpha}\varphi(\alpha) = x) \longrightarrow \varphi(x)]$$

and write $K = \grave{\varepsilon} f(\varepsilon)$. By (1) we have, for any function $g(x)$, that $\grave{\varepsilon} g(\varepsilon) = \grave{\varepsilon} f(\varepsilon) \longrightarrow g(K) = f(K)$ and this implies

$$f(K) \longrightarrow ((\grave{\varepsilon} g(\varepsilon) = K) \longrightarrow g(K)). \tag{2}$$

As this holds for any function $g(x)$, we have

$$f(K) \longrightarrow \forall\varphi\ [(\grave{\varepsilon}\varphi(\varepsilon) = K) \to \varphi(K)]. \tag{3}$$

On the other hand, for any function g,

$$\forall\varphi\ [(\grave{\varepsilon}\varphi(\varepsilon) = K) \to \varphi(K)] \longrightarrow ((\grave{\varepsilon} g(\varepsilon) = K) \to g(K)).$$

Substituting $f(x)$ for $g(x)$ results in:

$$\forall\varphi\ [(\grave{\varepsilon}\varphi(\varepsilon) = K) \to \varphi(K)] \longrightarrow ((\grave{\varepsilon} f(\varepsilon) = K) \to f(K))$$

and as $\grave{\varepsilon} f(\varepsilon) = K$ by definition of K,

$$\forall\varphi\ [(\grave{\varepsilon}\varphi(\varepsilon) = K) \to \varphi(K)] \longrightarrow f(K).$$

Using the definition of f, we obtain

$$\forall\varphi\ [(\grave{\varepsilon}\varphi(\varepsilon) = K) \to \varphi(K)] \longrightarrow \neg\forall\varphi[(\grave{\varepsilon}\varphi(\varepsilon) = K) \to \varphi(K)],$$

hence by reductio ad absurdum, $\neg\forall\varphi\ [(\grave{\alpha}\varphi(\alpha) = K) \to \varphi(K)]$, or shorthand:

$$f(K). \tag{4}$$

Applying (3) results in $\forall\varphi\ [(\grave{\alpha}\varphi(\alpha) = K) \to \varphi(K)]$, which implies

$$\neg\neg\forall\varphi\ [(\grave{\alpha}\varphi(\alpha) = K) \to \varphi(K)],$$

or shorthand:

$$\neg f(K). \tag{5}$$

(4) and (5) contradict each other.

2.2.3. *How wrong was Frege?* In the history of the Russell Paradox, Frege is often depicted as the pitiful person whose system was inconsistent. This suggests that Frege's system was the only one that was inconsistent, and that Frege was very inaccurate in his writings. On these points, history does Frege an injustice.

In fact, Frege's system was much more accurate than other systems of those days. Peano's work, for instance, was less precise on several points:

- Peano hardly paid any attention to logic, especially not to quantification theory;
- Peano did not make a strict distinction between his symbolism and the objects underlying this symbolism. Frege was much more accurate on this point (see also his paper *Über Sinn und Bedeutung* [29]);
- Frege made a strict distinction between a proposition (as an object of interest or discussion) and the *assertion* of a proposition. Frege denoted a proposition, in general, by $-A$, and the assertion of the proposition by $\vdash A$. The symbol $\vdash$ is still widely used in logic and type theory. Peano did not make this distinction and simply wrote A.

Nevertheless, Peano's work was very popular, for several reasons:

- Peano had able collaborators, and in general had a better eye for presentation and publicity. For instance, he bought his own press, so that he could supervise the printing of his journal *Rivista di Matematica* and *Formulaire* [52]
- Peano used a symbolism much more familiar to the notations that were used in those days by mathematicians (and many of his notations, like $\in$ for "is an element of", and $\supset$ for logical implication, are also used in Russell's *Principia Mathematica*, and are actually still in use).

Frege's work did not have these advantages and was hardly read before 1902.[13] In the last paragraph of [30], Frege concluded:

> ... I observe merely that the Peano notation is unquestionably more convenient for the typesetter, and in many cases takes up less room than mine, but that these advantages seem to me, due

[13]When Peano published his formalisation of mathematics in 1889 [51] he clearly did not know Frege's *Begriffsschrift*, as he did not mention the work, and was not aware of Frege's formalisation of quantification theory. Peano considered quantification theory to be "abstruse" in [52], on which Frege proudly reacted:

> "In this respect my conceptual notion of 1879 is superior to the Peano one. Already, at that time, I specified all the laws necessary for my designation of generality, so that nothing fundamental remains to be examined. These laws are few in number, and I do not know why they should be said to be abstruse. If it is otherwise with the Peano conceptual notation, then this is due to the unsuitable notation."
>
> ([30], p. 376)

> to the inferior perspicuity and logical defectiveness, to have been paid for too dearly—at any rate for the purposes I want to pursue.
>
> (*Ueber die Begriffschrift des Herrn Peano und meine Eigene*, p. 378)

Frege's system was not the only paradoxical one. The Russell Paradox can be derived in Peano's system as well, by defining the class

$$K \stackrel{\text{def}}{=} \{\, x \mid x \notin x \,\}$$

and deriving $K \in K \longleftrightarrow K \notin K$.

2.2.4. *The importance of Russell's Paradox.* Russell's paradox was certainly not the first or only paradox in history. Paradoxes were already widely known in antiquity. The first known paradox is the Achilles paradox of Zeno of Elea. It is a purely mathematical paradox. Due to a precise formulation of mathematics and especially the concept of real numbers, the paradox can now be satisfactorily solved.

The oldest logical paradox is probably the Liar's Paradox, also known as the Paradox of Epimenides. It can be very shortly formulated by the sentence "This sentence is not true". The paradox was widely known in antiquity. For instance, it is referred to in the Bible (Titus 1:12). It is based on the confusion between language and meta-language.

The Burali-Forti paradox ([8], 1897) is the first of the modern paradoxes. It is a paradox within Cantor's theory on ordinal numbers.[14] Cantor's paradox on the largest cardinal number occurs in the same field. It must have been discovered by Cantor around 1895, but was not published before 1932.

The logicians considered these paradoxes to be out of the scope of logic: the paradoxes based on the Liar's Paradox could be regarded as a problem of linguistics, and the paradoxes of Cantor and Burali-Forti occurred in what was considered in those days a highly questionable part of mathematics: Cantor's Set Theory.

The Russell Paradox, however, was a paradox that could be formulated in all the systems that were presented at the end of the 19th century (except for Frege's *Begriffsschrift*). It was at the very basics of logic. It could not be disregarded, and a solution to it had to be found. Russell's solution to the paradoxes was to use type theory.

§3. Type theory in Principia Mathematica. When Russell proved Frege's *Grundgesetze* to be inconsistent, Frege was not the only person in trouble. In Russell's letter to Frege (1902), we read:

> "I am on the point of finishing a book on the principles of mathematics"
>
> (*Letter to Frege*, [61])

[14]It is instructive to note that Cantor was aware of the Burali-Forti paradox and did not think that it rendered his system incoherent.

Therefore, Russell had to find a solution to the paradoxes, before he could finish his book.

His paper *Mathematical logic as based on the theory of types* [63] (1908), in which a first step is made towards the Ramified Theory of Types, started with a description of the most important contradictions that were known up till then, including Russell's own paradox. He then concluded:

> "In all the above contradictions there is a common characteristic, which we may describe as self-reference or reflexiveness. [...] In each contradiction something is said about *all* cases of some kind, and from what is said a new case seems to be generated, which both is and is not of the same kind as the cases of which *all* were concerned in what was said."
>
> (*Ibid.*)

Russell's plan was, therefore, to avoid the paradoxes by avoiding all possible self-references. He postulated the "vicious circle principle":

VICIOUS CIRCLE PRINCIPLE 3.

> *"Whatever involves all of a collection must not be one of the collection."*
>
> ([47], *p.* 200)

Russell applies this principle very strictly. He implemented it using types, in particular the so-called *ramified* types. The theory presented in *Mathematical logic as based on the theory of types* was elaborated in Chapter II of the Introduction to the famous *Principia Mathematica* [71] (1910–1912). In the *Principia*, Whitehead and Russell founded mathematics on logic, as far as possible. The result was a very formal and accurate build-up of mathematics, avoiding the logical paradoxes.

The logical part of the *Principia* was based on the works of Frege. This was acknowledged by Whitehead and Russell in the preface, and can also be seen throughout the description of Type Theory. The notion of function is based on Frege's Abstraction Principles 1 and 2, and the *Principia* notation $\hat{x} f(x)$ for a class looks very similar to Frege's $\grave{\varepsilon} f(\varepsilon)$ for course-of-values.

An important difference is that Whitehead and Russell treated functions as first-class citizens. Frege used courses-of-values as a way of speaking about functions (and was confronted with a paradox); in the *Principia* a direct approach was possible. Equality, for instance, was defined for objects as well as for functions by means of Leibniz equality ($x = y$ if and only if $f(x) \leftrightarrow f(y)$ for all propositional functions f—see [71], $*13{\cdot}11$).

The description of the Ramified Theory of Types (RTT) in the *Principia* was, though extensive, still informal. It is clear that Type Theory had not yet become an independent subject. The theory

> "only recommended itself to us in the first instance by its ability to solve certain contradictions"
>
> (*Principia Mathematica*, p. 37)

And though

> "it has also a certain consonance with common sense which makes it inherently credible"
>
> (*Principia Mathematica*, p. 37)

(probably, Whitehead and Russell refer to the implicit, intuitive use of types by mathematicians as explained in Section 2.1), Type Theory was not introduced because it was interesting on its own, but because it had to serve as a tool for logic and mathematics. A formalisation of Type Theory, therefore, was not considered in those days.

Though the description of the ramified type theory in the *Principia* was still informal, it was clearly present throughout the work. It was not mentioned very often, but when necessary, Russell made a remark on the ramified type theory. This is an important difference with the earlier writings of Frege, Peano and Cantor.

If we want to compare RTT with contemporary type systems, we have to make a formalisation of RTT. Though there are many descriptions of RTT available in the literature (like [14, 15, 37, 59] and Section 27 of [65]), none of these descriptions presents a formalisation that is both accurate and as close as possible to the ideas of the *Principia*. [47, 48] fill up this gap in the literature. In this section, we review the formalisation of [47, 48] explaining how it faithfully represents the intentions of Russell and Whitehead.

Formalisation of the ramified type theory is by no means easy:

- Important formal notions, especially the notion of substitution, remained completely unexplained in the *Principia*;
- The accuracy of Frege's work was not present in Russell's. This was already observed by Gödel, who said that the precision of Frege was lost in the writings of Russell, and who, due to the informality of some basic notions of the *Principia*, had to give his paper [33] the title *Über formal unentscheidbare Sätze der Principia Mathematica und* verwandter *Systeme*.

In Section 2.2.1 we saw that Frege generalised the notion of function from analysis. For Russell's formalisation of mathematics within logic, a special kind of these functions was needed: the so-called *propositional* functions. A propositional function (pf) always returns a *proposition* when it is applied to suitable arguments. In Section 3.1, we introduce a formalised version of these pfs. This makes it possible to compare pfs with other formal systems, like λ-calculus (which we do in Section 3.2), and to give a precise definition of substitution (Section 3.4).

In Section 3.5 we give a formalisation of Russell's notion of ramified type, followed by a formal definition of the notion *the pf f is of type t*. We motivate this definition by referring to passages in the *Principia*. As the formalisation of pf is precise enough to be translated to λ-calculus, we can make a comparison between RTT and current type systems.

Thanks to our formal notation and its relation to λ-calculus, we are able to prove properties of RTT in an easy way, using properties of modern type systems. This will be done in Section 3.8. Due to the new notation it is relatively easy to see that we have proved variants of well-known theorems from Type Theory, like Strong Normalisation, Free Variable Lemma, Strengthening Lemma, Unicity of Types and Subterm Lemma.

In Section 3.9 we answer in full detail the question which pfs are typable. We also make a comparison between our notion of typable pf, and the corresponding notion in the *Principia*, and conclude that these two notions of typable pf coincide.

3.1. Principia's propositional functions. In this section we present a formalisation of the propositional functions (pfs) of the *Principia* by introducing a syntax that is as close as possible to the ideas of the *Principia*. Intuition about this syntax is provided by translating pfs into λ-terms in Section 3.2. A special attention is devoted to the notion of substitution. This notion is clearly present in the *Principia*, but not formally defined. Due to our translation of pfs to λ-calculus, we are able to give a precise definition.

3.1.1. *Definition.* The definition of propositional function in the *Principia* is as follows:

> "By a "propositional function" we mean something which contains a variable x, and expresses a *proposition* as soon as a value is assigned to x."
>
> (*Principia Mathematica*, p. 38)

Pfs are, however, *constructed* from propositions with the use of the Abstraction Principles: they arise when in a proposition one or more occurrences of a sign are replaced by a variable. Therefore we have to begin our formalisation with certain basic propositions, certain basic signs, and signs that indicate a replaceable object. For this purpose we use

- A set $\mathcal{A}$ of *individual symbols* (the basic signs);
- A set $\mathcal{V}$ of *variables* (the signs that indicate replaceable objects);
- A set $\mathcal{R}$ of *relation symbols* together with a map $\mathfrak{a}\colon \mathcal{R} \to \mathbb{N}^+$ indicating the *arity* of each relation-symbol (these are used to form the basic propositions).

We want to have a sufficient supply of individual symbols, variables and relation symbols and therefore assume that $\mathcal{A}$ and $\mathcal{V}$ are infinite (but countable), and that $\{ R \in \mathcal{R} \mid \mathfrak{a}(R) = n \}$ is infinite (but countable) for each $n \in \mathbb{N}^+$. We assume that $\{\mathtt{a}_1, \mathtt{a}_2, \dots\} \subseteq \mathcal{A}$, $\{\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{x}_1, \dots\} \subseteq \mathcal{V}$ and $\{\mathtt{R}, \mathtt{S}, \dots\} \subseteq \mathcal{R}$.

We use $a_1, a_2, \ldots$ as metavariables over $\mathcal{A}$; $x, y, z, x_1, \ldots$ as metavariables over $\mathcal{V}$ and $R, S, \ldots$ as metavariables over $\mathcal{R}$. For technical reasons we assume that there is an *order* (e.g., alphabetical) on $\mathcal{V}$. We write $x < y$ if x is ordered before y, and not equal to y (so: $<$ is *strict*). In particular, we assume that $\mathtt{x} < \mathtt{x}_1 < \ldots \mathtt{y} < \mathtt{y}_1 < \ldots \mathtt{z} < \mathtt{z}_1 \ldots$ and: for each x there is a y with $x < y$.

Definition 4 (Atomic propositions). A list of symbols of the form

$$R(a_1, \ldots, a_{\mathfrak{a}(R)})$$

is called an *atomic proposition*.

Other names used for these atomic propositions in the *Principia* are *elementary judgements* and *elementary propositions* (cf. [71], pp. xv, 43–45, and 91).

Propositional functions in *Principia Mathematica* are generated from atomic propositions by two means:

- The use of logical connectives and quantifiers;
- Abstraction from (earlier generated) propositional functions, using the abstraction principles.

This leads to the following formal definition of propositional function.

Definition 5 (Propositional functions). We define a collection $\mathcal{P}$ of *propositional functions* (pfs), and for each element f of $\mathcal{P}$ we simultaneously define the collection $\mathrm{FV}(f)$ of *free variables* of f:

1. If $i_1, \ldots, i_{\mathfrak{a}(R)} \in \mathcal{A} \cup \mathcal{V}$ then $R(i_1, \ldots, i_{\mathfrak{a}(R)}) \in \mathcal{P}$.
   $\mathrm{FV}(R(i_1, \ldots, i_{\mathfrak{a}(R)})) \stackrel{\text{def}}{=} \{i_1, \ldots, i_{\mathfrak{a}(R)}\} \cap \mathcal{V}$;
2. If $f, g \in \mathcal{P}$ then $f \vee g \in \mathcal{P}$ and $\neg f \in \mathcal{P}$.
   $\mathrm{FV}(f \vee g) \stackrel{\text{def}}{=} \mathrm{FV}(f) \cup \mathrm{FV}(g)$; $\mathrm{FV}(\neg f) \stackrel{\text{def}}{=} \mathrm{FV}(f)$;
3. If $f \in \mathcal{P}$ and $x \in \mathrm{FV}(f)$ then $\forall x[f] \in \mathcal{P}$.
   $\mathrm{FV}(\forall x[f]) \stackrel{\text{def}}{=} \mathrm{FV}(f) \setminus \{x\}$;
4. If $n \in \mathbb{N}$ and $k_1, \ldots, k_n \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{P}$, then $z(k_1, \ldots, k_n) \in \mathcal{P}$.
   $\mathrm{FV}(z(k_1, \ldots, k_n)) \stackrel{\text{def}}{=} \{z, k_1, \ldots, k_n\} \cap \mathcal{V}$.
   If $n = 0$ then we write $z()$ in order to distinguish the pf $z()$ from the variable z[15];
5. All pfs can be constructed by using the construction-rules 1, 2, 3 and 4 above.

We use the letters f, g, h as meta-variables over $\mathcal{P}$.

Note that in clause 4. of the above definition, the variable binding in pf arguments of terms $z(k_1, \ldots, k_n)$ may be quite unexpected. We explain this feature in detail in Section 3.3 and especially in Remark 12.

[15]It is important to note that a variable is not a pf. See for instance [62], Chapter VIII: "The variable", p. 94 of the 7th impression.

DEFINITION 6 (Propositions). A propositional function f is a *proposition* if $\mathrm{FV}(f) = \varnothing$.

EXAMPLE 7. We give some examples of (higher-order) pfs of the form $z(k_1, \dots, k_n)$ in ordinary mathematics. To keep the link with mathematics clear, we use some extra logical connectives like $\leftrightarrow$ and $\rightarrow$.

1. The pfs $\mathtt{z}(\mathtt{x})$ and $\mathtt{z}(\mathtt{y})$ in the definition of equality according to Leibniz: By definition $\mathtt{x} = \mathtt{y}$ if and only if $\forall \mathtt{z}[\mathtt{z}(\mathtt{x}) \leftrightarrow \mathtt{z}(\mathtt{y})]$;
2. The pfs $\mathtt{z}(0)$, $\mathtt{z}(\mathtt{x})$ and $\mathtt{z}(\mathtt{y})$ in the formulation of the principle of mathematical induction:

$$\forall \mathtt{z}\, [\mathtt{z}(0) \rightarrow (\forall \mathtt{x}\, \forall \mathtt{y}\, [\mathtt{z}(\mathtt{x}) \rightarrow (\mathtt{S}(\mathtt{x}, \mathtt{y}) \rightarrow \mathtt{z}(\mathtt{y}))]) \rightarrow \forall \mathtt{x}[\mathtt{z}(\mathtt{x})]].$$

   (we suppose that the relation symbol $\mathtt{S}$ represents the successor function: $\mathtt{S}(x, y)$ holds if and only if y is the successor of x);
3. $\mathtt{z}(\,)$ in the formulation of the law of the excluded middle: $\forall \mathtt{z}\, [\mathtt{z}(\,) \vee \neg \mathtt{z}(\,)]$.

3.2. Principia's propositional functions as λ-terms. The binding structure and the notion of free variable of pfs become more clear if we translate pfs to λ-terms. Moreover, such a translation will be useful at several places in this article, for instance when we give a definition of substitution.

We first translate one of the examples of Example 7. Then we give a formal definition of the translation that we have in mind. After that we provide additional remarks and intuition on pfs.

EXAMPLE 8. Consider the pf $f \equiv \forall \mathtt{z}\, [\mathtt{z}(\mathtt{x}) \leftrightarrow \mathtt{z}(\mathtt{y})]$ of Example 7.1. Two objects x and y are Leibniz-equal if and only if they share the same properties. These objects are represented by the variables $\mathtt{x}$ and $\mathtt{y}$. The variable $\mathtt{z}$ is a variable for properties of objects, in other words: predicates over objects. Such a predicate is a function that takes the object as argument, and returns a truth value. The expression $\mathtt{z}(\mathtt{x})$ indicates that the predicate that is taken for $\mathtt{z}$ must be applied to the object that is taken for $\mathtt{x}$. Therefore, we translate $\mathtt{z}(\mathtt{x})$ by an application of $\mathtt{z}$ to $\mathtt{x}$ in λ-calculus: $\mathtt{zx}$. Similarly we translate the expression $\mathtt{z}(\mathtt{y})$ by $\mathtt{zy}$.

Just as in [14], we can interpret logical connectives as functions. Therefore we can translate $\mathtt{z}(\mathtt{x}) \leftrightarrow \mathtt{z}(\mathtt{y})$ by the λ-term $\leftrightarrow(\mathtt{zx})(\mathtt{zy})$. We handle the translation of universal quantification also as in [14], hence $\forall \mathtt{z}[\dots]$ translates to $\forall(\lambda \mathtt{z}.\dots)$. As an effect we get a λ-term $\forall(\lambda \mathtt{z}.{\leftrightarrow}(\mathtt{zx})(\mathtt{zy}))$ with two free variables, $\mathtt{x}$ and $\mathtt{y}$. But we want to have a *function* taking two *arguments*. This can be solved by a double λ-abstraction. The final result is $\lambda \mathtt{x}.\lambda \mathtt{y}.\forall(\lambda \mathtt{z}.(\leftrightarrow(\mathtt{zx})(\mathtt{zy})))$.

We remark that the pf f has two free variables, $\mathtt{x}$ and $\mathtt{y}$. These two free variables correspond to the two arguments that the propositional function takes, and therefore to the two λ-abstractions that are at the front of the translation of f.

In the following definition, we translate the propositional functions to λ-terms in a similar way as we did in Example 8. Let $f \in \mathcal{P}$ and let $x_1 < \cdots < x_m$ be the free variables of f. We define a λ-term $\overline{f}$. We do this in such a way that $\overline{f} \equiv \lambda x_1.\cdots\lambda x_m.F$, where F is a λ-term that is not of the form $\lambda x.F'$. To keep notations uniform, we also give translations $\overline{a}$ for $a \in \mathcal{A}$ and $\overline{x}$ for $x \in \mathcal{V}$. To keep notations short, we use $\lambda_{i=1}^{m} x_i.F$ as shorthand for $\lambda x_1.\cdots\lambda x_m.F$.

Definition 9 (Translating propositional functions to λ-terms).

- $\overline{a} \stackrel{\text{def}}{=} a$ for $a \in \mathcal{A}$;
- $\overline{x} \stackrel{\text{def}}{=} x$ for $x \in \mathcal{V}$;
- Now assume $f \in \mathcal{P}$ has free variables $x_1 < \cdots < x_m$. Use induction on the structure of f:
  - $f \equiv R(i_1, \ldots, i_{\mathfrak{a}(R)})$. Then $\overline{f} \stackrel{\text{def}}{=} \lambda_{i=1}^{m} x_i.Ri_1 \cdots i_{\mathfrak{a}(R)}$;
  - $f \equiv f_1 \vee f_2$. We can assume that for $j = 1, 2$, $\overline{f_j} \equiv \lambda_{i=1}^{m_j} y_i^j.F_j$, where $y_1^j < \cdots < y_{m_j}^j$ are the free variables of f_j.
    Then $\overline{f} \stackrel{\text{def}}{=} \lambda_{i=1}^{m} x_i.\vee F_1 F_2$.
    If $f \equiv \neg f'$ then we can assume that $\overline{f'} \equiv \lambda_{i=1}^{m} x_i.F$, because $x_1 < \cdots < x_m$ are the free variables of f'. Let $\overline{f} \stackrel{\text{def}}{=} \lambda_{i=1}^{m} x_i.\neg F$;
  - $f \equiv z(k_1, \ldots, k_n)$. Let $\overline{f} \equiv \lambda_{i=1}^{m} x_i.z\overline{k_1}\cdots\overline{k_n}$;
  - $f \equiv \forall x[f']$. We can assume that $\overline{f} \equiv \lambda_{i=1}^{j-1} x_i.\lambda x.\lambda_{i=j}^{m} x_i.F$, because $x_1, \ldots, x_m, x$ are the free variables of f'.
    Define $\overline{f} \equiv \lambda_{i=1}^{m} x_i.\forall(\lambda x.F)$.

Example 10.

f	$\overline{f}$
$\mathtt{R(x)}$	$\lambda\mathtt{x.Rx}$
$\mathtt{z(R(x), S(a))}$	$\lambda\mathtt{z.z}(\lambda\mathtt{x.Rx})\mathtt{(Sa)}$
$\mathtt{z_1(a)} \vee \mathtt{z_2()}$	$\lambda\mathtt{z_1}.\lambda\mathtt{z_2}.\vee\mathtt{(z_1a)z_2}$
$\mathtt{z(y(R(x)))}$	$\lambda\mathtt{z.z}(\lambda\mathtt{y.y}(\lambda\mathtt{x.Rx}))$
$\forall\mathtt{x}\ [\mathtt{R(x)}]$	$\forall(\lambda\mathtt{x.Rx})$

Lemma 11 (Properties of $\bar{\ }$). *Let $f \in \mathcal{P}$.*

1. $\text{FV}(\overline{f}) = \varnothing$;
2. $\overline{f}$ *is in β-normal form*;
3. $\overline{f}$ *is a λI-term*;
4. *If $x_1 < \cdots < x_m$ are the free variables of f, then $\overline{f} \equiv \lambda_{i=1}^{m} x_i.F$, where F is not of the form $\lambda x.F'$.*

Proof. By induction on the structure of f. ⊣

Observe that we use FV for indicating both the free variables of a pf and the free variables of a λ-term. We take care that it will always be clear in which meaning we use FV. In the above definition we also assume familiarity with the notion of λI-term (see [4]): in λI, terms of the form $\lambda x.F$ are only allowed if x appears as a free variable in F.

3.3. Remarks on Principia's pfs and their translation in λ-calculus.

REMARK 12. We show that the propositional functions of Definition 5 are indeed objects that exist in the theory of Russell.

1. In Rule 1 we describe the atomic propositions, and the atomic propositions in which one or more individuals have been replaced by variables due to one or more applications of the abstraction principles. The abstraction principles are not only present in the works of Frege, but also in the *Principia* (cf. for instance $*9{\cdot}14$ and $*9{\cdot}15$);
2. Rule 2 describes the use of the logical connectives $\vee$ and $\neg$. These logical connectives are also used in the *Principia*. Implication[16], conjunction[17] and logical equivalence[18] are defined in terms of negation and disjunction. In examples, we sometimes use symbols for implication, conjunction and logical equivalence as abbreviations;
3. Rule 3 describes the use of the universal quantifier. It is explicitly stated in the *Principia* (cf. pp. 14–16) that the pf $\forall x[f]$ can only be constructed if f is a pf that contains x as a variable. Existential quantification[19] is defined in terms of negation and universal quantification;[20]
4. Rule 4 is also an instantiation of the abstraction principle. The pfs that can be constructed by using the construction-rules 1–3 only are exactly the pfs of what in these days would be called first-order predicate logic. With rule 4, higher-order pfs can be constructed. This is based on the following idea. Let f be a (fixed) pf in which $k_1, \ldots, k_n$ occur. We can interpret f as an *instantiation* of a function that has taken arguments $k_1, \ldots, k_n$. We now generalise this to $z(k_1, \ldots, k_n)$, representing *any* function z taking these arguments. Such a construction is also explicitly present in the *Principia*:

   > "the first matrices[21] that occur are those whose values are of the forms φx, $\psi(x, y)$, $\chi(x, y, z, \ldots)$, i.e., where the arguments, however many there may be, are all individuals. Such [propositional] functions we will call '*first-order* functions.' We may now introduce a notation to express 'any first-order function.'"
   >
   > (*Principia Mathematica*, p. 51)

---

[16] cf. *Principia*, $*1{\cdot}01$, p. 94

[17] cf. *Principia*, $*3{\cdot}01$, p. 109

[18] cf. *Principia*, $*4{\cdot}01$, p. 117

[19] cf. *Principia*, $*10{\cdot}01$, p. 140

[20] In the original system of *Principia*, existential quantification is not defined in terms of negation and universal quantification, but negation of universally quantified statements is defined in terms of existential quantification. But in *Principia*, $*10{\cdot}01$, p. 140, we see that Russell does present the definition of the existential quantifier as an alternative approach.

[21] See Remark 14 [footnote of the authors].

REMARK 13. The definition of free variable needs some special attention. We must notice that, for instance, $\text{FV}(\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{S}(\mathtt{a}))) = \{\mathtt{z}\}$ and not $\{\mathtt{x},\mathtt{z}\}$. Similarly, $\text{FV}(\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{y})) = \{\mathtt{y},\mathtt{z}\}$ and not $\{\mathtt{x},\mathtt{y},\mathtt{z}\}$. The reason for this is that the notion of free variable should harmonise with the intuitive notion of "argument place" of Frege and Russell. As was indicated in Remark 12.4, in the first example above, $\mathtt{z}$ represents an arbitrary function that takes $\mathtt{R}(\mathtt{x})$ and $\mathtt{S}(\mathtt{a})$ as arguments and returns a proposition. This means that we do not have to supply an argument for $\mathtt{x}$ "by hand". As soon as we feed a suitable[22] argument f to $\mathtt{z}$ in $\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{S}(\mathtt{a}))$, f will take the arguments $\mathtt{R}(\mathtt{x})$ and $\mathtt{S}(\mathtt{a})$, and return a proposition.

This idea is also clearly reflected in the translation of $\mathtt{z}(\mathtt{R}(\mathtt{x}),\mathtt{S}(\mathtt{a}))$ to the λ-term $\lambda\mathtt{z}.\mathtt{z}(\lambda\mathtt{x}.\mathtt{Rx})(\mathtt{Sa})$. The variable $\mathtt{x}$ is bound in a subterm $\lambda\mathtt{x}.\mathtt{Rx}$ that is an argument to the variable $\mathtt{z}$. The full λ-term is a function of $\mathtt{z}$ only. See example 24.

REMARK 14. It appears that there is also an alternative way of constructing pfs in the *Principia*. Whitehead and Russell distinguish between quantifier-free pfs (so-called *matrices*, i.e., the pfs that can be constructed using construction-rules 1, 2 and 4). Then they form pfs by defining that

- Any matrix is a pf;
- If f is a pf and $x \in \text{FV}(f)$ then $\forall x\ [f]$ is a pf with free variables $\text{FV}(f) \setminus \{x\}$.

This definition is a little different from our Definition 5, as a pf of the form $z(\forall x\ [f])$ is not a matrix and therefore not a pf according to this alternative definition. Nevertheless we feel that Whitehead and Russell intended to give our Definition 5. In the *Principia* ([71], $*54$) they define the natural number 0 as the propositional function $\forall x\ [\neg z(x)]$.[23] In defining the principle of induction on natural numbers, one needs to express the property "0 has the property y", or: $y(0)$. But $y(0)$ is not a pf according to this alternative definition, as 0 contains quantifiers.

Therefore we feel that our Definition 5, which is also based on the definition of function by Frege and on the definition of propositional function on p. 38 of the *Principia*, is the definition that was meant by Whitehead and Russell.

REMARK 15. Note that pfs as such do not yet obey to the vicious circle principle 3 (on page 200). For example, $\neg\mathtt{z}(\mathtt{z})$ (the pf that is at the basis of the Russell paradox) is a pf. In Section 3.5 we will assign types to some

[22]At this stage, we cannot provide a formalisation of "suitable". This can only be done after we have introduced types, and formalised the notion "the pf f is of type t".

[23]This definition is based on Frege's definition in *Grundlagen der Arithmetik* [26] (1884). See [71], vol. II, p. 4. In [26], the natural number n is defined as the class of predicates f for which there are exactly n objects a for which $f(a)$ holds. Hence 0 is the class of predicates f for which $f(a)$ does not hold for any object a. So 0 can be described by the pf $\forall\mathtt{x}\ [\neg\mathtt{z}(\mathtt{x})]$. We prefer this notation to the notation $\iota`\emptyset$ (which also lends itself), because the logical operators $\neg$ and $\forall$ are more freely available in type theory.

pfs, and it will be shown (Remark 85) that no type can be assigned to the pf $\neg\mathtt{z}(\mathtt{z})$.

REMARK 16. Before we make further developments of the theory based on pfs, we must decide which of the two syntaxes introduced above shall be used in the sequel. It looks attractive to use the syntax of λ-calculus:

- This syntax is well-known;
- It is used for many other type systems, so it makes the comparison of ramified type theory with modern type systems easier;
- There is a lot of meta-theory on typed and untyped λ-calculus. This can be useful when proving certain properties of the formalisation of the ramified theory of types that is to be introduced in the next sections;
- The syntax of λ-calculus gives a better look on the notion of free variable than the syntax of pfs.

Nevertheless, we shall only indirectly use λ-calculus for our further study of the ramified type theory in this article. We have several reasons for that:

- There are much more λ-terms than there are pfs. More precisely, the mapping $\bar{}$ is not surjective. As we want to study the theory of *Principia Mathematica* as precise as possible, we only want to study the propositional functions, which are directly related to the syntax used by Russell and Whitehead. Not using pf-syntax may result in a system in which it is not clear which term belongs to the original ramified type theory and which term does not;
- The syntax of λ-calculus is strongly curried. This would give problems in the definition of substitution. In a pf $\mathtt{R}(\mathtt{x},\mathtt{y})$ we may want to substitute some object a for $\mathtt{y}$ without substituting anything for $\mathtt{x}$. In λ-calculus, substitution should be translated to application followed by β-reduction to β-normal form. If we want to substitute something for $\mathtt{y}$ in the translation $\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{Rxy}$ of $\mathtt{R}(\mathtt{x},\mathtt{y})$, we have to substitute something for $\mathtt{x}$ first. Choosing a different representation of propositional functions does not help: the representation $\lambda\mathtt{y}.\lambda\mathtt{x}.\mathtt{Ryx}$ would have given problems if we wanted to substitute something for $\mathtt{x}$ without substituting something for $\mathtt{y}$;
- The translation of pfs to λ-calculus makes it possible to use the meta-theory and the intuition of λ-calculus when we need it without losing control over the original system.

3.4. Principia's substitution and related notions. We proceed our discussion of pfs by defining a number of related notions. If a pf $z(k_1,\dots,k_n)$ takes an argument f for the variable z, the list $k_1, \dots, k_n$ indicates what should be substituted for the free variables of f (cf. also Remark 12.4). We therefore call this list the list of *parameters* of $z(k_1,\dots,k_n)$.

Another important notion is the notion of α-equality.[24] We want the pfs $\mathtt{R}(\mathtt{x})$ and $\mathtt{R}(\mathtt{y})$ to be the same. However, we want the pfs $\mathtt{S}(\mathtt{x},\mathtt{y})$ and $\mathtt{S}(\mathtt{y},\mathtt{x})$ to be different. The reason for this is the alphabetical order of the variables $\mathtt{x},\mathtt{y}$. As $\mathtt{x}<\mathtt{y}$, we will consider $\mathtt{x}$ to be the "first" variable of the pfs $\mathtt{S}(\mathtt{x},\mathtt{y})$ and $\mathtt{S}(\mathtt{y},\mathtt{x})$, and $\mathtt{y}$ the "second" variable . The place of the "first" variable in $\mathtt{S}(\mathtt{x},\mathtt{y})$, however, is different from the place of the "first" variable in $\mathtt{S}(\mathtt{y},\mathtt{x})$.[25] We therefore present the following definition of α-equality:

Definition 17 (α-equality). Let f and g be pfs. We say that f and g are *α-equal*, notation $f =_\alpha g$, if there is a bijection $\varphi\colon \mathcal{V}\to\mathcal{V}$ such that

- g can be obtained from f by replacing each variable that occurs in f by its φ-image;
- $x<y$ if and only if $\varphi(x)<\varphi(y)$.

This definition corresponds to the definition of α-equality in λ-calculus in the following way:

Lemma 18. *Let $f, g\in\mathcal{P}$. $f =_\alpha g$ if and only if $\overline{f} =_\alpha \overline{g}$.*

Sometimes, we are not that precise, and want the pfs $\mathtt{S}(\mathtt{x},\mathtt{y})$ and $\mathtt{S}(\mathtt{y},\mathtt{x})$ to be α-equal. This can be a consideration especially if we are not interested in which free variable is "first" and which is "second". We call this weakened notion of α-equality: α_P-equality (α-equality modulo permutation):

Definition 19 (α-equality modulo permutation). Let f and g be pfs. We say that f and g are *α_P-equal*, notation $f =_{\alpha_P} g$, if there is a bijection $\varphi\colon\mathcal{V}\to\mathcal{V}$ such that g can be obtained from f by replacing each variable that occurs in f by its φ-image.

As function construction in *Principia Mathematica* can be compared to β-expansion plus removing an argument in λ-calculus, this suggests that instantiation in the *Principia* must be comparable to application plus β-reduction in λ-calculus. In [46] we showed that this is indeed the case. There, we gave a definition of instantiation using the syntax of and the intuition behind pfs. We showed that this definition is faithful to the original ideas of the *Principia* and that it can be imitated in λ-calculus using a translation similar to the one in Definition 9. This allows us to give a definition of

[24] Historically, it is not correct to use this terminology when discussing Type Theory of the *Principia* which dates from the first decade of the 20th century. The term α-equality originates from Curry and Feys' book *Combinatory Logic* [20], which appeared only in 1958. In that book, conversion rules for the λ-calculus are numbered with Greek letters α, β, which led to the names α- and β-conversion. In earlier papers of Church, Rosser and Kleene, these rules were numbered with Roman capitals I, II, and the terminology α-conversion, β-conversion, was not used.

[25] Compare this with their equivalents in λ-calculus $\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{Sxy}$ and $\lambda\mathtt{x}.\lambda\mathtt{y}.\mathtt{Syx}$, which are not α-equal, either. We do not want to use the λ-notation for determining which variable is "first" and which is "second", for reasons to be explained in Remark 31. See also Remark 16.

substitution for pfs that is based on that imitation in λ-calculus, as we do below.

As was argued in Remark 16, the mapping $f \mapsto \overline{f}$ is not perfectly suited for a definition of substitution. This was due to the currying of the λ-abstractions that are at the front of the term $\overline{f}$. We therefore take a slightly different notation and remove these front abstractions from $\overline{f}$:

Definition 20. Let $f \in \mathcal{P}$ with free variables $x_1 < \cdots < x_m$. Then $\overline{f} \equiv \lambda_{i=1}^{m} x_i.F$ for some λ-term F by Lemma 11.4. Let $\widetilde{f} \stackrel{\text{def}}{=} F$.

Example 21.

f	$\widetilde{f}$
$\mathtt{R}(\mathtt{x})$	$\mathtt{Rx}$
$\mathtt{z}(\mathtt{R}(\mathtt{x}), \mathtt{S}(\mathtt{a}))$	$\mathtt{z}(\lambda\mathtt{x}.\mathtt{Rx})(\mathtt{Sa})$
$\mathtt{z}_1(\mathtt{a}) \vee \mathtt{z}_2(\,)$	$\vee(\mathtt{z}_1\mathtt{a})\mathtt{z}_2$
$\mathtt{z}(\mathtt{y}(\mathtt{R}(\mathtt{x})))$	$\mathtt{z}(\lambda\mathtt{y}.\mathtt{y}(\lambda\mathtt{x}.\mathtt{Rx}))$
$\forall\mathtt{x}\,[\mathtt{R}(\mathtt{x})]$	$\forall(\lambda\mathtt{x}.\mathtt{Rx})$

The mapping $f \mapsto \widetilde{f}$ has similar properties as $f \mapsto \overline{f}$ (cf. Lemma 11):

Lemma 22 (Properties of $\widetilde{\ }$).

1. $\text{FV}(f) = \text{FV}(\widetilde{f})$;
2. *$\widetilde{f}$ is in β-normal form for all f;*
3. *$\widetilde{f}$ is a λI-term for all f;*
4. *$\overline{f}$ is a closure of $\widetilde{f}$;*
5. *If $\widetilde{f} =_\alpha \widetilde{g}$, then $f =_\alpha g$.*

With the λ-notation we can rely on the notions of β-reduction and β-normal form to give the following definition of substitution:

Definition 23 (Substitution). Let $f \in \mathcal{P}$, assume $x_1, \ldots, x_n$ are distinct variables, and $g_1, \ldots, g_n \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{P}$. Assume that the λ-term $(\lambda x_1 \cdots x_n.\widetilde{f})\overline{g_1} \cdots \overline{g_n}$ has a β-normal form H. Assume $h \in \mathcal{P}$ such that $\widetilde{h} \equiv H$. (If such an h exists, it is unique due to Lemma 22.5.) We define the simultaneous substitution of $g_1, \ldots, g_n$ for $x_1, \ldots, x_n$ in f by: $f[x_1, \ldots, x_n := g_1, \ldots, g_n] \stackrel{\text{def}}{=} h$.

We sometimes abbreviate $f[x_1, \ldots, x_n := g_1, \ldots, g_n]$ to $f[x_i := g_i]_{i=1}^{n}$ or $f[\overrightarrow{x} := \overrightarrow{g}]$.

So substitution in RTT can be seen as application plus β-reduction to β-normal form in λ-calculus. Definition 23 is schematically reflected in Figure 1.

Notice that $f[x_1, \ldots, x_n := g_1, \ldots, g_n]$ should be seen as a *simultaneous* substitution of $g_1, \ldots, g_n$ for $x_1, \ldots, x_n$. As the $\overline{g_i}$s are either closed λ-terms, or individuals, or variables, it is no problem to define this simultaneous

$$\begin{tikzcd}
f[x_1,\dots,x_n := g_1,\dots,g_n] \arrow[d] & = h \\
(\lambda x_1\cdots x_n.\widetilde{f})\overline{g_1}\cdots\overline{g_n} \arrow[r, two heads, "\mathrm{nf}"', "\beta"] & \widetilde{h} \arrow[u]
\end{tikzcd}$$

FIGURE 1. Substitution via β-reduction

substitution via a list of applications that results in a list of *consecutive* substitutions.

EXAMPLE 24.

1. $\mathtt{S}(\mathtt{x}_1)[\mathtt{x}_1 := \mathtt{a}_1] \equiv \mathtt{S}(\mathtt{a}_1)$, as $(\lambda\mathtt{x}_1.\mathtt{S}\mathtt{x}_1)\mathtt{a}_1 \rightarrow^{\mathrm{nf}}_{\beta} \mathtt{S}\mathtt{a}_1$;
2. $\mathtt{S}(\mathtt{x}_1)[\mathtt{x}_2 := \mathtt{a}_2] \equiv \mathtt{S}(\mathtt{x}_1)$, as $(\lambda\mathtt{x}_2.\mathtt{S}\mathtt{x}_1)\mathtt{a}_2 \rightarrow^{\mathrm{nf}}_{\beta} \mathtt{S}\mathtt{x}_1$;
3. $\mathtt{z}(\mathtt{S}(\mathtt{x}_1),\mathtt{x}_2,\mathtt{a}_2)[\mathtt{x}_1 := \mathtt{a}_1] \equiv \mathtt{z}(\mathtt{S}(\mathtt{x}_1),\mathtt{x}_2,\mathtt{a}_2)$ as $(\lambda\mathtt{x}_1.\mathtt{z}(\lambda\mathtt{x}_1.\mathtt{S}\mathtt{x}_1)\mathtt{x}_2\mathtt{a}_2)\mathtt{a}_1 \rightarrow^{\mathrm{nf}}_{\beta} \mathtt{z}(\lambda\mathtt{x}_1.\mathtt{S}\mathtt{x}_1)\mathtt{x}_2\mathtt{a}_2$. This illustrates that the λ-notation is more precise and convenient with respect to free variables. In $\mathtt{z}(\mathtt{S}(\mathtt{x}_1),\mathtt{x}_2,\mathtt{a}_2)$, it is not immediately clear whether $\mathtt{x}_1$ is a free variable or not and one might tend to write:
$$\mathtt{z}(\mathtt{S}(\mathtt{x}_1),\mathtt{x}_2,\mathtt{a}_2)[\mathtt{x}_1 := \mathtt{a}_1] \equiv \mathtt{z}(\mathtt{S}(\mathtt{a}_1),\mathtt{x}_2,\mathtt{a}_2).$$
The λ-notation is more explicit in showing that $\mathtt{x}_1 \notin \mathrm{FV}(\mathtt{z}(\mathtt{S}(\mathtt{x}_1),\mathtt{x}_2,\mathtt{a}_2))$;
4. $\mathtt{z}(\mathtt{R}(\mathtt{a}),\mathtt{S}(\mathtt{a}))[\mathtt{z} := \mathtt{z}_1(\,) \vee \mathtt{z}_2(\,)] \equiv \mathtt{R}(\mathtt{a}) \vee \mathtt{S}(\mathtt{a})$, as
$$\begin{aligned}&(\lambda\mathtt{z}.\mathtt{z}(\mathtt{Ra})(\mathtt{Sa}))(\lambda\mathtt{z}_1\mathtt{z}_2.\vee\mathtt{z}_1\mathtt{z}_2)\\ &\qquad\qquad \rightarrow_{\beta} (\lambda\mathtt{z}_1\mathtt{z}_2.\vee\mathtt{z}_1\mathtt{z}_2)(\mathtt{Ra})(\mathtt{Sa}) \twoheadrightarrow^{\mathrm{nf}}_{\beta} \vee(\mathtt{Ra})(\mathtt{Sa});\end{aligned}$$
5. $\mathtt{x}_2(\mathtt{x}_1,\mathtt{R}(\mathtt{x}_1))[\mathtt{x}_2 := \mathtt{x}_4(\mathtt{x}_3)] \equiv \mathtt{R}(\mathtt{x}_1)$ as $(\lambda\mathtt{x}_2.\mathtt{x}_2\mathtt{x}_1(\lambda\mathtt{x}_1.\mathtt{R}\mathtt{x}_1))(\lambda\mathtt{x}_3\mathtt{x}_4.\mathtt{x}_4\mathtt{x}_3)$ $\twoheadrightarrow_{\beta} (\lambda\mathtt{x}_3\mathtt{x}_4.\mathtt{x}_4\mathtt{x}_3)\mathtt{x}_1(\lambda\mathtt{x}_1.\mathtt{R}x_1) \twoheadrightarrow_{\beta} (\lambda\mathtt{x}_1.\mathtt{R}\mathtt{x}_1)\mathtt{x}_1 \rightarrow^{\mathrm{nf}}_{\beta} \mathtt{R}\mathtt{x}_1$.

REMARK 25. $f[x_1,\dots,x_n := g_1,\dots,g_n]$ is not always defined. For its existence we need:

- The existence of the normal form H in Definition 23. For instance, this normal form does not exist if we choose $n = 1$, $f \equiv \mathtt{x}_1(\mathtt{x}_1)$ and $g_1 \equiv \mathtt{x}_1(\mathtt{x}_1)$: then we obtain for the calculation of $f[\mathtt{x}_1 := g_1]$ the famous λ-term $(\lambda\mathtt{x}_1.\mathtt{x}_1\mathtt{x}_1)(\lambda\mathtt{x}_1.\mathtt{x}_1\mathtt{x}_1)$;
- The existence of a (unique) h such that $\widetilde{h} \equiv H$. For instance, if we take $n = 1$, $f \equiv \mathtt{z}(\mathtt{a})$ (with $\mathtt{z} \in \mathcal{V}$ and $\mathtt{a} \in \mathcal{A}$) and $g_1 \equiv \mathtt{a}$, then $H \equiv \mathtt{aa}$ and there is no $h \in \mathcal{P}$ such that $\widetilde{h} \equiv \mathtt{aa}$.

In Corollary 61, we will prove that, as long as we are within the type system RTT (to be introduced in Section 3.5), both H and h always exist uniquely. Until then, we implicitly assume that the substitution *exists* in the notation $f[x_1,\dots,x_n := g_1,\dots,g_n] = h$.

REMARK 26. If we compute a substitution $f[x_1, \ldots, x_n := g_1, \ldots, g_n]$, we have to reduce the λ-term $(\lambda x_1 \cdots x_n.\widetilde{f})\bar{g_1} \cdots \bar{g_n}$ to its β-normal form (if there is any). One might wonder whether this is too restrictive: In a reduction path to this normal form, there may be an intermediate result H that could be interpreted as the final result of the substitution $f[\overrightarrow{x}:=\overrightarrow{g}]$. However, this never happens, as any term that can be interpreted as such a result is always of the form $\widetilde{h}$, and is therefore always in β-normal form (Lemma 22.2).

REMARK 27. The alphabetical order of the variables plays a crucial role in the substitution process, as it determines in which order the free variables of a pf f are curried in the translation $\overline{f}$. For example, look at the substitutions $\mathtt{z}(\mathtt{a},\mathtt{b})[\mathtt{z} := \mathtt{R}(\mathtt{x},\mathtt{y})]$ and $\mathtt{z}(\mathtt{a},\mathtt{b})[\mathtt{z} := \mathtt{R}(\mathtt{y},\mathtt{x})]$. The result of the first one is obtained via the normal form of $(\lambda\mathtt{z.zab})(\lambda\mathtt{xy.Rxy})$, which is equal to $\mathtt{Rab}$, translated: $\mathtt{R}(\mathtt{a},\mathtt{b})$. The second one is calculated via $(\lambda\mathtt{z.zab})(\lambda\mathtt{xy.Ryx})$, resulting in $\mathtt{Rba}$ and $\mathtt{R}(\mathtt{b},\mathtt{a})$.

REMARK 28. Now that substitution has been properly defined, we could define that f is an *abstraction* of g if there are $x_1, \ldots, x_n \in \mathrm{FV}(f)$ and $h_1, \ldots, h_n \in \mathcal{A} \cup \mathcal{P}$ such that $f[\overrightarrow{x} := \overrightarrow{h}] \equiv g$, or, in λ-calculus notation: $(\lambda x_1 \cdots x_n.\widetilde{f})\overline{h_1} \cdots \overline{h_n} \twoheadrightarrow_\beta \widetilde{g}$. The set of abstractions of a pf g is therefore comparable with the set of β-expansions of the λ-term $\widetilde{g}$.

Some elementary calculation with substitutions can be done using the following lemma:

LEMMA 29.

1. *Assume* $(f_1 \vee f_2)[\overrightarrow{x} := \overrightarrow{h}]$ *exists. Then* $f_j[\overrightarrow{x} := \overrightarrow{h}]$ *exists for* $j = 1, 2$, *and*
$$(f_1 \vee f_2)[\overrightarrow{x} := \overrightarrow{h}] \equiv (f_1[\overrightarrow{x} := \overrightarrow{h}]) \vee (f_2[\overrightarrow{x} := \overrightarrow{h}]);$$
2. *Assume* $(\neg f)[\overrightarrow{x} := \overrightarrow{h}]$ *exists. Then* $f[\overrightarrow{x} := \overrightarrow{h}]$ *exists, and*
$$(\neg f)[\overrightarrow{x} := \overrightarrow{h}] \equiv \neg(f[\overrightarrow{x} := \overrightarrow{h}]);$$
3. *Assume* $(\forall x\colon t^a[f])[\overrightarrow{x} := \overrightarrow{h}]$ *exists, and* $x \notin \vec{x}$. *Then* $f[\overrightarrow{x} := \overrightarrow{h}]$ *exists, and*
$$(\forall x\colon t^a[f])[\overrightarrow{x} := \overrightarrow{h}] \equiv \forall x\colon t^a[f[\overrightarrow{x} := \overrightarrow{h}]];$$
4. *Assume* $z(k_1, \ldots, k_n)[z := f]$ *exists, and* $x_1 < \cdots < x_n$ *are the free variables of* f. *Then* $f[\overrightarrow{x}:=\overrightarrow{k}]$ *exists, and*
$$z(k_1, \ldots, k_n)[z := f] \equiv f[\overrightarrow{x} := \overrightarrow{k}];$$

5. *Assume* $z(k_1, \ldots, k_n)[\overrightarrow{x} := \overrightarrow{h}]$ *exists,* $z \equiv x_p$, *and* $y_1 < \cdots < y_n$ *are the free variables of* $k_p \in \mathcal{P}$. *Define* $k'_i \equiv h_j$ *if* $k_i \equiv x_j$, *and* $k'_i \equiv k_i$ *otherwise. Then* $k_p[\overrightarrow{y}:=\overrightarrow{k'}]$ *exists, and*

$$z(k_1, \ldots, k_n)[\overrightarrow{x} := \overrightarrow{h'}] \equiv k_p[y_j := \overrightarrow{k'}].$$

PROOF. Directly from the definition of substitution. ⊣

3.5. Principia's Ramified Theory of Types. In order to give a precise description of the type theory underlying the *Principia*, we need to explicitly introduce types (there is no such introduction in *Principia*), and to formalise the notion "the propositional function f has type t".

3.5.1. *Types.* Types in the *Principia* have a double hierarchy: one of (simple) types and one of orders. First, we introduce the first hierarchy, then we extend this hierarchy with orders, resulting in the ramified types of the *Principia*.

Simple types. As we saw in Section 2.2, Frege already distinguished between objects, functions that take objects as arguments, and functions that take functions as arguments. He also made a distinction between functions that take one and functions that take two arguments (see the quotations from *Function and Concept* on p. 195). In the *Principia*, Whitehead and Russell use a similar principle. Whilst Frege's argument for this distinction was only that functions are fundamentally different from objects, and that functions taking objects as arguments are fundamentally different from functions taking functions as arguments, Whitehead and Russell are more precise:

> "[The difference between objects and propositional functions] arises from the fact that a [propositional] function is essentially an ambiguity, and that, if it is to occur in a definite proposition, it must occur in such a way that the ambiguity has disappeared, and a wholly unambiguous statement has resulted."
>
> (*Principia Mathematica*, p. 47)

There is no definition of "type" in the *Principia*, only a definition of "being of the same type":

> "*Definition of "being of the same type."* The following is a step-by-step definition, the definition for higher types presupposing that for lower types. We say that u and v "are of the same type" if
> 1. both are individuals,
> 2. both are elementary [propositional] functions[26] taking arguments of the same type,

[26]See Definition 4 for the notion of elementary proposition. In the *Principia*, the term *elementary functions* refers to a pf that has only elementary propositions as value, when it takes suitable (well-typed) arguments. See *Principia*, p. 92.

3. u is a pf and v is its negation,
4. u is $\varphi\hat{x}$[27] or $\psi\hat{x}$, and v is $\varphi\hat{x} \vee \psi\hat{x}$, where $\varphi\hat{x}$ and $\psi\hat{x}$ are elementary pfs,
5. u is $(y).\varphi(\hat{x}, y)$[28] and v is $(z).\psi(\hat{x}, z)$, where $\varphi(\hat{x}, \hat{y})$, $\psi(\hat{x}, \hat{y})$ are of the same type,
6. both are elementary propositions,
7. u is a proposition and v is $\sim u$[29], or
8. u is $(x).\varphi x$ and v is $(y).\psi y$, where $\varphi\hat{x}$ and $\psi\hat{x}$ are of the same type."

(*Principia Mathematica*, $*9{\cdot}131$, p. 133)

The definition has to be seen as the definition of an equivalence relation. For instance, assume that $\varphi\hat{x}$, $\psi\hat{x}$ and $\chi\hat{x}$ are elementary pfs. By rule 4, $\varphi\hat{x}$ and $\varphi\hat{x} \vee \psi\hat{x}$ are of the same type, and so are $\varphi\hat{x}$ and $\varphi\hat{x} \vee \chi\hat{x}$. By (implicit) transitivity, $\varphi\hat{x} \vee \psi\hat{x}$ and $\varphi\hat{x} \vee \chi\hat{x}$ are of the same type.

The definition seems rather precise at first sight. But there are several remarks to be made:

- The notion "being of the same type" seems to be defined for pfs taking one argument only. On the other hand, rules 2 and 5 suggest that such a definition should be extended to pfs taking two arguments. How this should be done is not made explicit;
- According to this definition, $\mathtt{z}_1(\,)\vee\neg\mathtt{z}_1(\,)$ is not of the same type as $\mathtt{z}_1(\,)$. The only rules by which could be derived that $\mathtt{z}_1(\,)$ and $\mathtt{z}_1(\,) \vee \neg\mathtt{z}_1(\,)$ are of the same type, are rules 2 and 4. But if we want to use these rules, $\mathtt{z}_1(\,)$ must be an elementary pf, which it is not: It can take the argument $\forall\mathtt{x}\,[\mathtt{R}(\mathtt{x})]$, which has as result the proposition $\forall\mathtt{x}\,[\mathtt{R}(\mathtt{x})]$. This is not an elementary proposition and therefore $\mathtt{z}_1(\,)$ is not an elementary pf.

So there are significant omissions in this definition. However, the intention of the definition is clear: pfs that take a different number of arguments, or that take arguments of different types, cannot be of the same type.

In order to make precise what is meant by "being of the same type", it is easier to explain what these types "are". The notion "being of the same type" can then be replaced by "having the same type". The notion of simple type as defined below is due to Ramsey [59] (1926). Historically, it is incorrect to give Ramsey's definition of simple type before Russell's definition of ramified type, as Russell's definition is of an earlier date, and Ramsey's definition is in fact based on Russell's ideas and not the other way around. On the other hand, the ideas behind simple types were already explained by Frege (see the quotes from *Function and Concept* on page 195). Moreover, knowledge

[27] Whitehead and Russell use $\varphi\hat{x}$ to denote that φ is a pf that has, amongst others, x as a free variable. Similarly, they use $\varphi(\hat{x}, \hat{y})$ to indicate that φ has x, y amongst its free variables.

[28] Whitehead and Russell write $(x).\varphi(x)$ where we would write $\forall x\,[\varphi]$.

[29] $\sim u$ is *Principia* notation for $\neg u$.

of the intuition behind simple types will make it easier to understand the ramified ones.[30] Therefore we present Ramsey's definition first.

DEFINITION 30 (Ramsey's simple types).

1. $\mathtt{0}$ is a simple type;
2. If $t_1, \ldots, t_n$ are simple types, then also $(t_1, \ldots, t_n)$ is a simple type. $n = 0$ is allowed: then we obtain the simple type $()$;
3. All simple types can be constructed using the rules 1 and 2.

We use t, u, $t_1, \ldots$ as metavariables over simple types.

Here, $(t_1, \ldots, t_n)$ is the type of pfs that should take n arguments (have n free variables), the ith argument having type t_i. The type $()$ stands for the type of the propositions, and the type $\mathtt{0}$ stands for the type of the individuals.

REMARK 31. To formalise the notion of ith argument that a pf takes, we use the alphabetical order on variables that was introduced in Section 3.1. The ith argument taken by a pf will be substituted for the ith free variable of that pf, according to the alphabetical order.

Now it becomes clear why we considered the alphabetical order of variables in the definition of α-equality 17: we want α-equal pfs to have the same type. However, if f has type (t_1, t_2) and two free variables $x < y$, and g is the same as f except that the roles of x and y have been switched, then g will have type (t_2, t_1). Therefore we demand that the renaming of variables must maintain the alphabetical order. See also Remark 43.

EXAMPLE 32. The propositional function $\mathtt{R}(\mathtt{x})$ should have type $(\mathtt{0})$, as it takes one individual as argument.

The propositional function $\mathtt{z}(\mathtt{R}(\mathtt{x}), \mathtt{S}(\mathtt{a}))$ (see Remark 12.4) takes one argument. This argument must be a pf that can take $\mathtt{R}(\mathtt{x})$ as its first argument (so this first argument must be of type $(\mathtt{0})$), and a proposition (of type $()$) as its second argument. We conclude that in $\mathtt{z}(\mathtt{R}(\mathtt{x}), \mathtt{S}(\mathtt{a}))$, we must substitute pfs of type $((\mathtt{0}), ())$ for $\mathtt{z}$. Therefore, $\mathtt{z}(\mathtt{R}(\mathtt{x}), \mathtt{S}(\mathtt{a}))$ has type $(((\mathtt{0}), ()))$.

The intuition presented in Remark 31 and Example 32 will be formalised in Definition 40. Theorem 54 shows that this formalisation follows the intuition.

NOTATION 33. From now on we will use a slightly different notation for quantification in pfs. Instead of $\forall x\, [f]$ we now explicitly mention the type (say: t) over which x is quantified: $\forall x{:}\, t[f]$. We do the same with the translations of pfs to λ-calculus: instead of $\lambda x.F$ we write $\lambda x{:}\, T(t).F$.

[30] See [38, 39] for a further discussion of the difference between simple and ramified type theory, especially in connection with Quine's new foundations for which there is a consistency result for its predicative version (and hence one can get models of predicative type theory in which very strong versions of "systematic ambiguity" hold). In particular, [39] contains a discussion of the relationship between a predicative *linear* type scheme (with types indexed by the natural numbers) and the full ramified type scheme of *Principia*.

Ramified types. Up to now, the type of a pf only depends on the types of the arguments that it can take. In the *Principia*, a second hierarchy is introduced by regarding also the types of the variables that are bound by a quantifier (see *Principia*, pp. 51–55). Whitehead and Russell consider, for instance, the propositions $\mathtt{R}(\mathtt{a})$ and $\forall z\colon (\,)[\mathtt{z}() \vee \neg\mathtt{z}(\,)]$ to be of a different level. The first is an atomic proposition, while the latter is based on the pf $\mathtt{z}(\,) \vee \neg\mathtt{z}(\,)$. The pf $\mathtt{z}(\,) \vee \neg\mathtt{z}(\,)$ involves an arbitrary proposition $\mathtt{z}$, therefore $\forall\mathtt{z}\colon (\,)[\mathtt{z}(\,) \vee \neg\mathtt{z}(\,)]$ quantifies over all propositions $\mathtt{z}$. According to the vicious circle principle 3, $\forall\mathtt{z}\colon (\,)[\mathtt{z}(\,) \vee \neg\mathtt{z}(\,)]$ cannot belong to this collection of propositions.

This problem is solved by dividing types into *orders* (not to be confused with the *alphabetical order* on the variables). An order is simply a natural number. Basic propositions are of order 0, and in $\forall\mathtt{z}\colon (\,)[\mathtt{z}(\,) \vee \neg\mathtt{z}(\,)]$ we must mention the order of the propositions over which is quantified. The pf $\forall\mathtt{z}\colon (\,)^n[\mathtt{z}(\,) \vee \neg\mathtt{z}(\,)]$ quantifies over all propositions of order n, and has order $n + 1$.

The division of types into orders gives *ramified* types.

Definition 34 (Ramified types).

1. 0^0 is a ramified type;
2. If $t_1^{a_1}, \ldots, t_n^{a_n}$ are ramified types, and $a \in \mathbb{N}$, $a > \max(a_1, \ldots, a_n)$, then $(t_1^{a_1}, \ldots, t_n^{a_n})^a$ is a ramified type (if $n = 0$ then take $a \geq 0$);
3. All ramified types can be constructed using the rules 1 and 2.

If t^a is a ramified type, then a is called the *order* of t^a.

Remark 35. In $(t_1^{a_1}, \ldots, t_n^{a_n})^a$, we demand that $a > a_i$ for all i. This is because a pf of this type presupposes all the elements of type ${t_i}^{a_i}$, and therefore must be of an order that is higher than a_i.

Example 36. 0^0; $(0^0)^1$; $((0^0)^1, (0^0)^4)^5$; and $(0^0, (\,)^2, (0^0, (0^0)^1)^2)^7$ are all ramified types. However, $(0^0, (0^0, (0^0)^2)^2)^7$ is not a ramified type.

In the rest of Section 3 we simply speak of *types* when we mean *ramified* types, as long as no confusion arises.[31]

In the type $(0^0)^1$, all orders are "minimal", i.e., not higher than strictly necessary. This is, for instance, not the case in the type $(0^0)^2$. Types in which all orders are minimal are called *predicative* and play a special role in the Ramified Theory of Types. A formal definition:[32]

[31]Russell seems not to like the idea that propositions make up a type (see page 48 of the *Principia*). We do however use the type (or types) of propositions because at various places in *Principia*, Russell talks as if there are types of propositions, uses quantifiers over propositions and discusses orders of propositions. For example, in 9*131 Russell refers to elementary propositions as "being of the same type", in spite of the things he says elsewhere.

[32]This definition comes straight from Whitehead and Russell. It should be noted that ramified types which are not predicative are certainly not "impredicative" in the usual sense of that word.

DEFINITION 37 (Predicative types).

1. $\mathtt{0}^0$ is a predicative type;
2. If $t_1{}^{a_1}, \ldots, t_n{}^{a_n}$ are predicative types, and $a = 1 + \max(a_1, \ldots, a_n)$ (take $a = 0$ if $n = 0$), then $(t_1^{a_1}, \ldots, t_n^{a_n})^a$ is a predicative type;
3. All predicative types can be constructed using the rules 1 and 2 above.

3.6. A formalisation RTT of the Ramified Theory of Types. In this section we formalise the intuition on types presented in Example 32 and (in Church's notation) in Definition 80 together with the intuition on orders. Before we can do this we must introduce some additional terminology.

In the pf $\mathtt{R}(\mathtt{x})$ we implicitly assume that $\mathtt{x}$ is a variable for which objects of type $\mathtt{0}$ must be substituted. For our formalisation we want to make the information on the type of a variable explicit. We do this by storing this information in so-called *contexts*. Contexts, common in modern type systems, are not used in the *Principia*.

DEFINITION 38 (Contexts). Let $x_1, \ldots, x_n \in \mathcal{V}$ be distinct variables, and assume $t_1^{a_1}, \ldots, t_n^{a_n}$ are ramified types. Then $\{x_1 : t_1^{a_1}, \ldots, x_n : t_n^{a_n}\}$ is a *context*. The set $\{x_1, \ldots, x_n\}$ is called the *domain* of the context and is denoted by $\mathrm{dom}(\{x_1 : t_1^{a_1}, \ldots, x_n : t_n^{a_n}\})$. We will use Greek capitals Γ, Δ as meta-variables over contexts.

The pfs $\mathtt{z}_1(\mathtt{y}_1)$ and $\mathtt{z}_2(\mathtt{y}_2)$ are α-equal, according to Definition 17. But in a context $\Gamma \equiv \{\, \mathtt{y}_1 : \mathtt{0}^0, \mathtt{z}_1 : (\mathtt{0}^0)^1, \mathtt{y}_2 : (\mathtt{0}^0)^1, \mathtt{z}_2 : ((\mathtt{0}^0)^1)^2\}$ one does not want to see $\mathtt{z}_1(\mathtt{y}_1)$ and $\mathtt{z}_2(\mathtt{y}_2)$ as equal, as the types of $\mathtt{y}_1$ and $\mathtt{y}_2$ differ, and the types of $\mathtt{z}_1$ and $\mathtt{z}_2$ differ as well. Therefore, we introduce a more restricted version of α-equality:

DEFINITION 39. Let Γ be a context and f and g pfs. We say that f and g are α_Γ-*equal*, notation $f =_{\alpha,\Gamma} g$, if there is a bijection $\varphi : \mathcal{V} \to \mathcal{V}$ such that

- g can be obtained from f by replacing each variable that occurs in f by its φ-image;
- $x < y$ if and only if $\varphi(x) < \varphi(y)$;
- $x : t \in \Gamma$ if and only if $\varphi(x) : t \in \Gamma$.

We will now define what we mean by $\Gamma \vdash f : t^a$, or, in words: f is of type t^a in the context Γ.[33] If $\Gamma \equiv \varnothing$ then we will write $\vdash f : t^a$. In this definition we will try to follow the line of the Principia as much as possible. For remarks, discussion and examples, see Section 3.7 below.

DEFINITION 40 (Ramified Theory of Types: RTT). The *judgement*

$$\Gamma \vdash f : t^a$$

[33]The symbol $\vdash$ in $\Gamma \vdash f : t^a$ is the same symbol that Frege used to assert a proposition. It enters Type Theory in 1934 [18], via Curry's combinatory logic. Curry defines a functionality combinator F in such a way that $\mathsf{F}XYf$ holds, exactly if f is a function from X to Y. To denote the assertion of $\mathsf{F}XYf$, Curry uses Frege's symbol $\vdash$.

is inductively defined as follows:

1. (start) For all a we have: $\vdash a : 0^0$.
   For all atomic pfs f we have: $\vdash f : (\,)^0$;
2. (connectives) Assume $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a$, $\Delta \vdash g : (u_1^{b_1}, \ldots, u_m^{b_m})^b$, and $x < y$ for all $x \in \mathrm{dom}(\Gamma)$ and $y \in \mathrm{dom}(\Delta)$. Then
$$\Gamma \cup \Delta \vdash f \vee g : (t_1^{a_1}, \ldots, t_n^{a_n}, u_1^{b_1}, \ldots, u_m^{b_m})^{\max(a,b)};$$
   and
$$\Gamma \vdash \neg f : (t_1^{a_1}, \ldots, t_n^{a_n})^a;$$
3. (abstraction from parameters) If $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_m^{a_m})^a$, $t_{m+1}^{a_{m+1}}$ is a *predicative* type[34], $g \in \mathcal{A} \cup \mathcal{P}$ is a parameter of f, $\Gamma \vdash g : t_{m+1}^{a_{m+1}}$, and $x < y$ for all $x \in \mathrm{dom}(\Gamma)$, then
$$\Gamma' \vdash h : (t_1^{a_1}, \ldots, t_{m+1}^{a_{m+1}})^{\max(a, a_{m+1}+1)}.$$
   Here, h is a pf obtained by replacing all parameters g' of f which are α_Γ-equal to g by y. Moreover, Γ' is the subset of the context $\Gamma \cup \{y : t_{m+1}^{a_{m+1}}\}$ such that $\mathrm{dom}(\Gamma')$ contains all and only those variables occurring in h;[35]
4. (abstraction from pfs) If $(t_1^{a_1}, \ldots, t_m^{a_m})^a$ is a *predicative* type, $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_m^{a_m})^a$, $x < z$ for all $x \in \mathrm{dom}(\Gamma)$, and $y_1 < \cdots < y_n$ are the free variables of f, then
$$\Gamma' \vdash z(y_1, \ldots, y_n) : (t_1^{a_1}, \ldots, t_m^{a_m}, (t_1^{a_1}, \ldots, t_m^{a_m})^a)^{a+1},$$
   where Γ' is the subset of $\Gamma \cup \{z : (t_1^{a_1}, \ldots, t_m^{a_m})^a\}$ such that $\mathrm{dom}(\Gamma') = \{y_1, \ldots, y_n, z\}$;
5. (weakening) If Γ, Δ are contexts, $\Gamma \subseteq \Delta$, and $\Gamma \vdash f : t^a$, then also $\Delta \vdash f : t^a$;
6. (substitution) If y is the ith free variable in f (according to the order on variables), and $\Gamma \cup \{y : t_i^{a_i}\} \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a$, and $\Gamma \vdash k : t_i^{a_i}$ then
$$\Gamma' \vdash f[y := k] : (t_1^{a_1}, \ldots, t_{i-1}^{a_{i-1}}, t_{i+1}^{a_{i+1}}, \ldots, t_n^{a_n})^b.$$
   Here, $b = 1 + \max(a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n, c)$ and
$$c = \max\{\, j \mid \forall x : t^j \text{occurs in } f[y := k]\,\}$$
   (if $n = 1$ and $\{\, j \mid \forall x : t^j$ occurs in $f[y := k]\,\} = \varnothing$ then take $b = 0$) and once more, Γ' is the subset of $\Gamma \cup \{\, y : t_i^{a_i}\,\}$ such that $\mathrm{dom}(\Gamma')$ contains all and only those variables occurring in $f[y := k]$;

[34] The restriction to predicative types only is based on *Principia*, pp. 53–54.
[35] In Lemma 52 we prove that this context always exists.

7. (permutation) If y is the ith free variable in f (according to the order on variables), and $\Gamma \cup \{ y : t_i^{a_i} \} \vdash f : (t_1^{a_1}, \dots, t_n^{a_n})^a$, and $x < y'$ for all $x \in \mathrm{dom}(\Gamma)$, then

$$\Gamma' \vdash f[y := y'] : (t_1^{a_1}, \dots, t_{i-1}^{a_{i-1}}, t_{i+1}^{a_{i+1}}, \dots, t_n^{a_n}, t_i^{a_i})^a.$$

Γ' is the subset of $\Gamma \cup \{y : t_i^{a_i}, y' : t_i^{a_i}\}$ such that dom Γ' contains all and only those variables occurring in $f[y := y']$;
8. (quantification) If y is the ith free variable in f (according to the order on variables), and $\Gamma \cup \{ y : t_i^{a_i} \} \vdash f : (t_1^{a_1}, \dots, t_n^{a_n})^a$, then

$$\Gamma \vdash \forall y : t_i^{a_i}[f] : (t_1^{a_1}, \dots, t_{i-1}^{a_{i-1}}, t_{i+1}^{a_{i+1}}, \dots, t_n^{a_n})^a.$$

Definition 41 (Legal propositional functions). A pf f is called *legal*, if there is a context Γ and a ramified type t^a such that $\Gamma \vdash f : t^a$.

Remark 42. In our attempt to faithfully implement Russell's ramified theory of types in the above definition, we face a limitation in the terms typable by our system. For example, it is not possible to type $x_1(x_2(x_1))$ or the pf $x(\forall y.(y() \vee \neg y()))$.[36] In fact, Russell intended (cf. page 165 of *Principia*) that non-predicative orders in his hierarchy are always obtained from predicative ones by quantification. Rule 8 of the above definition is the only one which creates non-predicative types but the increase of order is only at the top level of the type. This means that we cannot type terms $z(k_1, \dots, k_n)$ where one of the k_is happens to be of non-predicative type. In fact, Theorem 69 will prove that terms $z(k_1, \dots, k_n)$ are typable only if the k_is can be assigned predicative types. This may be considered as a serious restriction but our aim is to faithfully represent Russell's ramified theory of types.

A drawback to our system is that, without the ability to assign non-predicative types to variables, one cannot even *state* the axiom of reducibility. Russell himself may have noted the need for variables with non-predicative types when he introduced on page 165 of *Principia* a convention for variable functions without assigned order which he used in the formal statement of the axiom of reducibility. However, Russell did not allow quantification over such variables. In our paper we ignore the representation of the axiom of reducibility. In Section 4.2, we discuss the controversial nature of this axiom leading to the deramification and we leave the extension of our formalisation of Russell's ramified type theory to include the reducibility axiom as future work.

Finally, based on our above discussion, note that the third and fourth types given in Example 36 cannot be assigned as types to a legal pf in the sense of Definition 40. Future extensions must also address these examples.

[36]We are grateful for Randall Holmes for drawing our attention to this point.

3.7. Discussion and examples. We will make some remarks on Definition 40. First of all, we motivate the eight rules of Definition 40 by referring to passages in the *Principia*. Then we make some technical remarks, and give some examples of how the rules work. It will be made clear that the substitution rule is problematic, because substitution is not clearly defined in the *Principia*.

REMARK 43. We will motivate RTT (Definition 40) by referring to the Principia:

1. Individuals and elementary judgements (atomic propositions) are, also in the *Principia*, the basic ingredients for creating legal pfs;[37]
2. We can see rule 2 "at work" in $*12$, p. 163 of the *Principia*[38]:

   > "We can build up a number of new formulas, such as [...] $\varphi!x \vee \varphi!y$, $\varphi!x \vee \psi!x$, $\varphi!x \vee \psi!y$, [...] and so on."
   >
   > (*Principia Mathematica*, $*12$, p. 163))

   The restriction about contexts that we make in rule 2 has technical reasons and is not made in the *Principia*. It will be discussed in Remark 45;
3. Rule 3 is justified by $*9{\cdot}14$ and $*9{\cdot}15$ in the *Principia*. It is an instantiation of the abstraction principles 1 and 2 for functions that was already proposed by Frege (see Section 2.2). In Frege's definition one does not have to replace all parameters g' that are α_Γ-equal to g, but one can also take some of these parameters. In Section 3.9 we show that this is not a serious restriction.

   The restriction to *predicative* types is in line with the *Principia* (cf. *Principia*, pp. 53–54);
4. Rule 4 is based on the Introduction of the *Principia*. There, pfs are constructed, and

   > "the first matrices that occur are those whose values are of the forms φx, $\psi(x, y)$, $\chi(x, y, z, \dots)$, i.e., where the arguments, however many there may be, are all individuals. Such [propositional] functions we will call '*first-order* functions.' We may now introduce a notation to express 'any first-order function.'"
   >
   > (*Principia Mathematica*, p. 51)

   This quote from the *Principia* is, again, an instance of Frege's abstraction principles, and so is rule 4 of our formalisation. It results in second order pfs, and the process can be iterated to obtain pfs of higher orders.

   Rule 4 makes it possible to introduce variables of higher order. In fact, leaving out rule 4 would lead to first-order predicate logic, as

[37] As for individuals: see *Principia*, $*9$, p. 132, where "Individual" is presented as a primitive idea. As for elementary judgements: See *Principia*, Introduction, pp. 43–45.

[38] In the *Principia*, Whitehead and Russell write $\varphi!x$ instead of φx to indicate that φx is not only (what we would call) a pf, but even a *legal* pf.

without rule 4 it is impossible to introduce variables of types that differ from 0^0.

The use of predicative types only is inspired by the *Principia*, again;

5. The weakening rule cannot be found in the *Principia*, because no formal contexts are used there. It is implicitly present, however: the addition of an extra variable to the set of variables does not affect the well-typedness of pfs that were already constructed;
6. The rule of substitution is based on $*9{\cdot}14$ and $*9{\cdot}15$ of the *Principia*, and can be seen as an inverse of the abstraction operators in rules 3 and 4. Notice that we do not know yet whether the substitution $f[y := k]$ exists or not. Therefore, we limit the use of rule 6 to the cases in which the substitution exists. In Section 3.8 we show that it always exists if the premises of rule 6 are fulfilled;
7. In the system above, the (sequential) order of the t_is is related to the alphabetic order of the free variables of the pf f that has type $(t_1, \ldots, t_n)$ (see the remark before Definition 17, Remark 31, and Theorem 54). This alphabetic order plays a role in the clear presentation of results like Theorem 54, and in the definition of substitution (see Remark 27).

   With rule 7 we want to express that the order of the t_is in $(t_1, \ldots, t_n)$ and the alphabetic order of the variables are not characteristics of the *Principia*, but are only introduced for the technical reasons explained in this remark. This is worked out in Corollary 55;
8. Notice that in the quantification rule, both f and $\forall x : t_i^{a_i}.f$ have order a. The intuition is that the order of a propositional function f equals one plus the maximum of the orders of *all* the variables (either free or bound by a quantifier) in f. This is in line with the *Principia*: see [71], page 53. See also the introduction to Definition 34, and the proof of Lemma 56 below.

REMARK 44. Rules 3 and 4 are a restricted version of the abstraction principles of Frege, with less power. It is, for instance, not possible to imitate all the abstractions of Remark 12 by using rules 3 and 4 only. But in combination with the other rules, rule 3 and 4 are sufficient (see Example 50 for the cases of Remark 12, and Section 3.9, especially Theorem 69).

REMARK 45. In rule 2 of RTT, we make the assumption that the variables of Γ must all come before the variables of Δ. The reason for this is that we want to prevent undesired results like

$$\mathtt{x}_1 : 0^0 \vdash \mathtt{R}_1(\mathtt{x}_1) \vee \mathtt{R}_2(\mathtt{x}_1) : (0^0, 0^0)^1.$$

In fact, $\mathtt{R}_1(\mathtt{x}_1) \vee \mathtt{R}_2(\mathtt{x}_1)$ has *only one* free variable, so its type should be $(0^0)^1$ and not $(0^0, 0^0)^1$ (see Example 49, second part). For technical reasons (the order of the $t_i^{a_i}$s; see also Theorem 54) we strengthen the assumption such that for $x \in \mathrm{dom}(\Gamma)$ and $y \in \mathrm{dom}(\Delta)$, $x < y$ must hold.

As Whitehead and Russell do not have a formal notation for types, they do not forbid this kind of construction in the *Principia*.

In Lemma 67 we show that our limitation to contexts with disjoint domains as made in rule 2 is not a real limitation: all the desired judgements can still be derived for contexts with non-disjoint domains.

REMARK 46. In both rules 3 and 4 we see that it is necessary to introduce at least one new variable. It is, for instance, not possible to interpret the proposition $\mathtt{R}(\mathtt{a})$ as a (constant) pf of type $(0^0)^1$. This is in line with the abstraction principles of Frege and Russell. In Frege's definition 1, for example, it is explicitly mentioned that the object that is to be replaced occurs at least once in the expression.

Translated to λ-calculus this means that the *Principia* have λI-terms, only. See also Lemma 11.3 and Lemma 22.3.

REMARK 47. Contexts as used in RTT contain, in a sense, too much information: not only information on all free variables, but also information on non-free variables (cf. rules 3, 6 and 7). The set of non-free variables contains more than only the variables that are bound by a quantifier. For example, in the pf $\mathtt{z}(\mathtt{R}(\mathtt{x}))$, $\mathtt{x}$ is neither free, nor bound by a quantifier).

REMARK 48. The system is based on the abstraction principles of Frege. In a context Γ, one cannot introduce a variable of a certain type t unless one has a pf (or an individual) f that has type t in Γ. This is different from modern, λ-calculus based systems, where one can introduce a variable of a type u without knowing whether or not there are terms of this type u.

We give some examples, in order to illustrate how our system works. Example 49 shows applications of the rules. Example 50 makes a link between the intuitive notion of abstraction that was explained in Remark 12 and the abstraction rules 3 and 4 of our system.

We will use a notation of the form

$$\frac{X_1 \cdots X_n}{Y} N,$$

indicating that from the judgements $X_1, \ldots, X_n$, we can infer the judgement Y by using the RTT-rule of Definition 40 with number N. As usual, this is called a *derivation step*. Subsequent derivation steps give a *derivation*. A derivation *of a judgement* Y is a derivation tree with Y as root (the final conclusion). The types in the examples below are all predicative (as a pf of impredicative type must have a quantifier, and the examples below are quantifier-free). To avoid too much notation, we omit the orders.

EXAMPLE 49. The following type derivations are valid in RTT.

1. $\vdash \mathtt{S}(\mathtt{a}_1, \mathtt{a}_2) : ()$ by rule 1 of Definition 40;
2. $\dfrac{\vdash \mathtt{R}_1(\mathtt{a}_1) : () \qquad \vdash \mathtt{R}_2(\mathtt{a}_1) : ()}{\vdash \mathtt{R}_1(\mathtt{a}_1) \vee \mathtt{R}_2(\mathtt{a}_1) : ()}$ rule 2

but *not*:

$$\frac{\mathtt{x}_1\colon 0 \vdash \mathtt{R}_1(\mathtt{x}_1)\colon (0) \qquad \mathtt{x}_1\colon 0 \vdash \mathtt{R}_2(\mathtt{x}_1)\colon (0)}{\mathtt{x}_1\colon 0 \vdash \mathtt{R}_1(\mathtt{x}_1) \vee \mathtt{R}_2(\mathtt{x}_1)\colon (0,0)}\ \text{rule 2}$$

($\mathtt{x}_1 \not< \mathtt{x}_1$ because $<$ is strict). To obtain $\mathtt{R}_1(\mathtt{x}_1) \vee \mathtt{R}_2(\mathtt{x}_1)$ we must make a different start:

$$\frac{\dfrac{\vdash \mathtt{R}_1(\mathtt{a}_1)\colon () \qquad \vdash \mathtt{R}_2(\mathtt{a}_1)\colon ()}{\vdash \mathtt{R}_1(\mathtt{a}_1) \vee \mathtt{R}_2(\mathtt{a}_1)\colon ()}\ \text{rule 2} \qquad \vdash \mathtt{a}_1\colon 0}{\mathtt{x}_1\colon 0 \vdash \mathtt{R}_1(\mathtt{x}_1) \vee \mathtt{R}_2(\mathtt{x}_1)\colon (0)}\ \text{rule 3;}$$

3. $$\frac{\begin{array}{lcl}\mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0, \mathtt{z}_1\colon ((0),(0)) & \vdash & \mathtt{z}_1(\mathtt{R}(\mathtt{x}_1), \mathtt{R}(\mathtt{x}_2))\colon (((0),(0))) \\ \mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0, \mathtt{z}_1\colon ((0),(0)) & \vdash & \mathtt{R}(\mathtt{x}_1)\colon (0)\end{array}}{\mathtt{z}_1\colon ((0),(0)), \mathtt{z}_2\colon (0) \vdash \mathtt{z}_1(\mathtt{z}_2, \mathtt{z}_2)\colon (((0),(0)),(0))}\ \text{rule 3}$$

As $\mathtt{R}(\mathtt{x}_1)$ is α-equal to $\mathtt{R}(\mathtt{x}_2)$ in the context, both $\mathtt{R}(\mathtt{x}_1)$ and $\mathtt{R}(\mathtt{x}_2)$ are replaced by the newly introduced variable $\mathtt{z}_2$;

4. $$\frac{\mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0 \vdash \mathtt{S}(\mathtt{x}_1, \mathtt{x}_2)\colon (0,0)}{\mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0, \mathtt{z}\colon (0,0) \vdash \mathtt{z}(\mathtt{x}_1, \mathtt{x}_2)\colon (0,0,(0,0))}\ \text{rule 4;}$$

5. $$\frac{\dfrac{\mathtt{x}_1\colon 0 \vdash \mathtt{R}_1(\mathtt{x}_1) \vee \mathtt{R}_2(\mathtt{x}_1)\colon (0) \qquad \vdash \mathtt{a}_1\colon 0}{\vdash \mathtt{R}_1(\mathtt{a}_1) \vee \mathtt{R}_2(\mathtt{a}_1)\colon ()}\ \text{rule 6;}}{\mathtt{x}_1\colon 0 \vdash \mathtt{R}_1(\mathtt{a}_1) \vee \mathtt{R}_2(\mathtt{a}_1)\colon ()}\ \text{rule 5}$$

6. $$\frac{\begin{array}{lcl}\mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0, \mathtt{x}_3\colon (0,0) & \vdash & \mathtt{R}(\mathtt{x}_1) \vee \neg \mathtt{x}_3(\mathtt{x}_1, \mathtt{x}_2)\colon (0,0,(0,0)) \\ \mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0 & \vdash & \mathtt{T}(\mathtt{x}_1, \mathtt{x}_1, \mathtt{x}_2)\colon (0,0)\end{array}}{\mathtt{x}_1\colon 0, \mathtt{x}_2\colon 0 \vdash \mathtt{R}(\mathtt{x}_1) \vee \neg \mathtt{T}(\mathtt{x}_1, \mathtt{x}_1, \mathtt{x}_2)\colon (0,0)}\ \text{rule 6}$$

$\mathtt{T}(\mathtt{x}_1, \mathtt{x}_1, \mathtt{x}_2)$ is substituted for $\mathtt{x}_3$.

EXAMPLE 50. We give a formal derivation of the examples of the abstraction rules that were given in Remark 12. Again, we omit the orders.

• Constructing $\mathtt{z}(\mathtt{a}) \vee \mathtt{S}(\mathtt{a})$ from $\mathtt{R}(\mathtt{a}) \vee \mathtt{S}(\mathtt{a})$ cannot be done with the use of rule 4 only. The following derivation is correct (to save space, we use "r*x*" to denote "rule *x*"):

$$\frac{\dfrac{\dfrac{\vdash \mathtt{a}\colon 0}{\mathtt{z}\colon (0) \vdash \mathtt{a}\colon 0}\ \text{r5} \qquad \dfrac{\dfrac{\vdash \mathtt{a}\colon 0 \qquad \vdash \mathtt{R}(\mathtt{a})\colon ()}{\mathtt{x}\colon 0 \vdash \mathtt{R}(\mathtt{x})\colon (0)}\ \text{r3}}{\mathtt{x}\colon 0, \mathtt{z}\colon (0) \vdash \mathtt{z}(\mathtt{x})\colon (0,(0))}\ \text{r4}}{\mathtt{z}\colon (0) \vdash \mathtt{z}(\mathtt{a})\colon ((0))}\ \text{r6} \qquad \vdash \mathtt{S}(\mathtt{a})\colon ()}{\mathtt{z}\colon (0) \vdash \mathtt{z}(\mathtt{a}) \vee \mathtt{S}(\mathtt{a})\colon ((0))}\ \text{r2.}$$

To obtain $\mathtt{z}(\mathtt{a})$ instead of $\mathtt{z}()$, we must transform $\mathtt{R}(\mathtt{a})$ into a pf $\mathtt{R}(\mathtt{x})$ by abstracting from $\mathtt{a}$. Then we can construct $\mathtt{z}(\mathtt{x})$ by abstraction from pfs (rule 4). In this way, the "frame" for $\mathtt{z}(\mathtt{a})$ is of the right form. Substituting $\mathtt{a}$ for $\mathtt{x}$ gives $\mathtt{z}(\mathtt{a})$ (and "neutralises" the application of rule 3 at the top of the derivation). Simply applying rule 4 on the judgement $\vdash \mathtt{R}(\mathtt{a})\colon ()$ does not work: it results in $\mathtt{z}() \vdash \mathtt{z}()\colon (())$;

• Constructing $z_1()\vee z_2()$ is easier: $z_1()$ can be obtained by abstracting from $R(a)$, and $z_2()$ similarly from $S(a)$. Result:

$$\dfrac{\dfrac{\vdash R(a)\colon ()}{z_1\colon ()\vdash z_1()\colon (())}\text{ rule 4}\qquad \dfrac{\vdash S(a)\colon ()}{z_2\colon ()\vdash z_2()\colon (())}\text{ rule 4}}{z_1\colon (), z_2\colon ()\vdash z_1()\vee z_2()\colon ((),())}\text{ rule 2.}$$

We see that in fact two abstractions are needed to construct this pf: we must abstract from $R(a)$ as an instance of the pf $z_1()$, and from $S(a)$ as an instance of the pf $z_2()$. As rule 4 does not work on parts of pfs, these abstractions have to be made before we use rule 2. Applying rule 4 on $\vdash R(a)\vee S(a)\colon ()^0$ would result in $z\colon ()\vdash z()\colon (())$;

• We can extend the derivation of $z_1\colon (), z_2\colon ()\vdash z_1()\vee z_2()\colon ((),())$ to obtain a type for $z(R(a),S(a))$:

$$\dfrac{\dfrac{\dfrac{x_1\colon (), x_2\colon ()\vdash x_1()\vee x_2()\colon ((),())}{x_1\colon (), x_2\colon (), z\colon ((),())\vdash z(x_1,x_2)\colon ((),(),((),()))}\text{ rule 4}}{x_2\colon (), z\colon ((),())\vdash z(R(a),x_2)\colon ((),((),()))}\text{ rule 6}}{z\colon ((),())\vdash z(R(a),S(a))\colon (((),()))}\text{ rule 6}$$

(for reasons of space, we omitted the premises $z\colon ((),()), x_2\colon ()\vdash R(a)\colon ()$ and $z\colon ((),())\vdash S(a)\colon ()$ of the first and second application of the substitution rule);

• For the derivation of the type of $z(R(x),S(a))$ we first make a derivation of the "frame" $z(y_1,y_2)$ of this pf (to save space, we use "rx" to denote "rule x"):

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash a\colon 0}{y_1\colon (0)\vdash a\colon 0}\text{ r5}\quad \dfrac{\dfrac{\vdash a\colon 0\quad \vdash R(a)\colon ()}{x\colon 0\vdash R(x)\colon (0)}\text{ r3}}{x\colon 0, y_1\colon (0)\vdash y_1(x)\colon (0,(0))}\text{ r4}}{y_1\colon (0)\vdash y_1(a)\colon ((0))}\text{ r6}\qquad \dfrac{\vdash R(a)\colon ()}{y_2\colon ()\vdash y_2()\colon (())}\text{ r4}}{y_1\colon (0), y_2\colon ()\vdash y_1(a)\vee y_2()\colon ((0),())}\text{ r2}}{y_1\colon (0), y_2\colon (), z\colon ((0),())\vdash z(y_1,y_2)\colon ((0),(),((0),()))}\text{ r4.}$$

Then we derive $x\colon 0\vdash R(x)\colon (0)$ and $\vdash S(a)\colon ()$, and after applying the weakening rule, we can substitute $R(x)$ for y_1 and $S(a)$ for y_2. As a result, we get

$$z\colon ((0),()), x\colon 0\vdash z(R(x),S(a))\colon (((0),())).$$

EXAMPLE 51. In the example below, the orders are important:

$$\dfrac{\dfrac{\dfrac{\vdash R(a)\colon ()^0\qquad \dfrac{\vdash R(a)\colon ()^0}{\vdash \neg R(a)\colon ()^0}\text{ rule 2}}{\vdash R(a)\vee\neg R(a)\colon ()^0}\text{ rule 2}}{z\colon ()^0\vdash z()\vee\neg z\colon (()^0)^1}\text{ rule 4}}{\vdash \forall z\colon ()^0\,[z()\vee\neg z()]\colon ()^1}\text{ rule 8.}$$

We see that $\forall z\colon ()^0\,[z()^0\vee\neg z()]$ does not have a predicative type. This is the case because this pf has a bound variable z that is of a higher order than the order of any free variable (as there are no free variables here). Therefore, the order of this pf is determined by the order of the bound variable z.

We still need to prove that the contexts in the conclusions of rules 3, 4 and 6 exist. This follows from the following Lemma:

LEMMA 52. *Assume* $\Gamma \vdash f : t^a$. *Then*

1. (Free variable lemma) *All variables of* f *that are not bound by a quantifier are in* $\mathrm{dom}(\Gamma)$;
2. (Strengthening lemma) *If* Δ *is the* (*unique*) *subset of* Γ *such that* $\mathrm{dom}(\Delta)$ *contains all and only those variables of* f *that are not bound by a quantifier, then* $\Delta \vdash f : t^a$.

PROOF. An easy induction on the definition of $\Gamma \vdash f : t^a$. ⊣

3.8. Properties of RTT.

3.8.1. *Types and free variables.* In this section we treat some meta-properties of RTT. Using the λ-notation for pfs, we can often refer to known results in typed λ-calculus.[39] For proofs and further details, see [48, 47].

THEOREM 53 (First Free Variable Theorem). *Let* $f \in \mathcal{P}$; $k_1, \ldots, k_n \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{P}$.

$$\mathrm{FV}(f[x_1, \ldots, x_n := k_1, \ldots, k_n]) \\ = (\mathrm{FV}(f) \setminus \{x_1, \ldots, x_n\}) \cup \{\, k_i \in \mathcal{V} \mid x_i \in \mathrm{FV}(f)\,\}.$$

THEOREM 54 (Second Free Variable Theorem). *Assume that we can derive* $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a$, *and* $x_1 < \cdots < x_m$ *are the free variables of* f. *Then* $m = n$ *and* $x_i : t_i^{a_i} \in \Gamma$ *for all* $i \leq n$.

PROOF. An easy induction on $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a$. For rules 6 and 7, use Theorem 53. ⊣

We can now prove a corollary that we promised in Remark 43.7:

COROLLARY 55. *If* $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a$ *and* φ *is a bijection* $\{1, \ldots, n\} \to \{1, \ldots, n\}$ *then there is a context* Γ' *and a pf* f' *which is* α_P*-equal to* f *such that* $\Gamma' \vdash f' : \left(t_{\varphi(1)}^{a_{\varphi(1)}}, \ldots, t_{\varphi(n)}^{a_{\varphi(n)}}\right)^a$.

We can also prove unicity of types and unicity of orders. Orders are unique in the following sense:

LEMMA 56. *Assume* $\Gamma \vdash f : t^a$. *If* x *occurs in* f *and* $x : u^b \in \Gamma$, *then* u^b *is predicative. Moreover, if also* $\Gamma \vdash f : t'^{a'}$, *then* $a = a'$.

PROOF. By induction on the derivation of $\Gamma \vdash f : t^a$ one shows that a variable x that occurs in f always has a predicative type in Γ, and that both a and a' equal one plus the maximum of the orders of all the (free and non-free) variables that occur in f. ⊣

[39] The meta-properties can also be proved directly, without λ-calculus: see [46].

COROLLARY 57 (Unicity of types for pfs). *Assume* Γ *is a context*, f *is a pf*, $\Gamma \vdash f : t^a$ *and* $\Gamma \vdash f : u^b$. *Then* $t^a \equiv u^b$.

PROOF. $t \equiv u$ follows from Theorem 54; $a = b$ from Lemma 56. ⊣

REMARK 58. We cannot omit the context Γ in Corollary 57. For example, the pf $\mathtt{z}(\mathtt{x})$ can have different types in different contexts, as is illustrated by the following derivations (we have omitted the orders as they can be calculated via Lemma 56):

$$\dfrac{\dfrac{\vdash \mathtt{R}(\mathtt{a}_1) \colon () \qquad \vdash \mathtt{a}_1 \colon 0}{\mathtt{x} \colon 0 \vdash \mathtt{R}(\mathtt{x}) \colon (0)}\ \text{rule 3}}{\mathtt{x} \colon 0, \mathtt{z} \colon (0) \vdash \mathtt{z}(\mathtt{x}) \colon (0, (0))}\ \text{rule 4}$$

versus

$$\dfrac{\dfrac{\vdash \mathtt{R}(\mathtt{a}_1) \colon ()}{\mathtt{x} \colon () \vdash \mathtt{x}() \colon (())}\ \text{rule 4}}{\mathtt{x} \colon (), \mathtt{z} \colon (()) \vdash \mathtt{z}(\mathtt{x}) \colon ((), (()))}\ \text{rule 4.}$$

Theorem 54 and Corollary 57 show that our system RTT makes sense, in a certain way: The type of a pf only depends on the context and does not depend on the way in which we derived the type of that pf.

As a corollary of Corollary 57 we find:

COROLLARY 59. *If*

$$\Gamma \vdash f : t^a, \quad \Gamma \vdash k : u^b, \quad x : u^b \in \Gamma \quad \textit{and} \quad \Gamma \vdash f[x := k] : t'^{a'}$$

then $a \geq a'$.

PROOF. If $x \notin \text{FV}(f)$ then $f \equiv f[x := k]$ and the corollary follows from Unicity of Types (Corollary 57). If $x \in \text{FV}(f)$ then the variables that occur in $f[x := k]$, occur either in f or in k, and as the order of k is smaller than the order of f $(x \in \text{FV}(f)$, so $b < a)$, the corollary follows from the proof of Lemma 56. ⊣

We conclude this section with mending the two loose ends discussed in Remark 25 which play a role in RTT-Definition 40:

First, using the strong normalisation of Church's $\lambda{\to}_C$, it is easy to see that:

THEOREM 60 (Existence of normal forms). *Take* $i \leq n$. *Assume*

$$\Gamma \cup \{y \colon t_i^{a_i}\} \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a \qquad \textit{and} \qquad \Gamma \vdash k \colon t_i^{a_i}$$

(*so the preconditions of rule* 6 *of* RTT *are fulfilled*). *Then* $(\lambda y \colon T(t_i^{a_i}).\widetilde{f})\overline{k}$ *is strongly normalising.*

Substitution always exists in the case of RTT-rule 6 of Definition 40.

THEOREM 61 (Existence of substitution). *If* $f \in \mathcal{P}$, y *is the* i*th free variable in* f, $\Gamma \cup \{y \colon t_i^{a_i}\} \vdash f : (t_1^{a_1}, \ldots, t_n^{a_n})^a$, *and* $\Gamma \vdash k \colon t_i^{a_i}$, *then* $f[y := k]$ *exists.*

3.9. Legal propositional functions in RTT. We recall Definition 41: a pf f is called *legal* if $\Gamma \vdash f : t^a$ for some Γ and t^a. We will check whether this definition of legal pf coincides with the definition of formula that was given in the *Principia*. For this purpose we prove a number of lemmas concerning the relation between legal pfs and predicative types.

We do not distinguish between pfs that are α_P-equal, nor between types $(t_1, \ldots, t_n)$ and $(t_{\varphi(1)}, \ldots, t_{\varphi(n)})$ for a bijection φ. This is justified by Corollary 55 and by the fact, that pfs that are α_P-equal are supposed to be the same in the *Principia* too.

We define the notion "up to α_P-equality" formally:

Definition 62. Let $f \in \mathcal{P}$, Γ a context, t^a a type. *f is of type t^a in the context Γ up to α_P-equality*, notation $\Gamma \vdash f : t^a \pmod{\alpha_P}$, if there is $f' \in \mathcal{P}$, a context Γ' and a bijection $\varphi\colon \mathcal{V} \to \mathcal{V}$ such that

- $\Gamma' \vdash f' : t^a$;
- f' and f are α_P-equal via the bijection φ;
- $\Gamma' = \{\, \varphi(x) : u^b \mid x : u^b \in \Gamma \,\}$.

We say that f is *legal* in the context Γ *up to α_P-equality* if there is a type u^b such that $\Gamma \vdash f : u^b \pmod{\alpha_P}$. We say that f is *legal up to α_P-equality* if there is a context Γ such that f is legal in Γ up to α_P-equality.

The following lemma states that all predicative types are "inhabited":

Lemma 63. *If t^a is predicative then there are f, Γ such that $\Gamma \vdash f : t^a$.*

Proof. We use induction on predicative types. $\dashv$

Remark 64. From a modern point of view, this is a remarkable lemma. Many modern type systems are based on the principle of propositions-as-types. In such systems types represent propositions, and terms inhabiting such a type represent proofs of that proposition. In a propositions-as-types based system in which all types are inhabited, all propositions are provable. Such a system would be (logically) inconsistent. RTT is not based on propositions-as-types, and there is nothing paradoxical or inconsistent in the fact that all RTT-types are inhabited.

This lemma can be generalised to some non-predicative types:

Corollary 65. *If $(t_1^{a_1}, \ldots, t_m^{a_m})^a$ is a type such that the $t_i^{a_i}$ are all predicative, then there are f and Γ such that $\Gamma \vdash f : (t_1^{a_1}, \ldots, t_m^{a_m})^a$.*

We can also show that $z(k_1, \ldots, k_m)$ is legal if $k_1, \ldots, k_m$ are either legal pfs or variables, and z is "fresh".

Lemma 66. *If $k_1, \ldots, k_n \in \mathcal{A} \cup \mathcal{V} \cup \mathcal{P}$, $t^a = (t_1^{a_1}, \ldots, t_n^{a_n})^a$ is a predicative type, $\Gamma \vdash k_i : t_i^{a_i}$ for all $k_i \in \mathcal{A} \cup \mathcal{P}$ and $k_i : t_i^{a_i} \in \Gamma$ for all $k_i \in \mathcal{V}$, and $z \in \mathcal{V} \setminus \mathrm{dom}(\Gamma)$, then $z(k_1, \ldots, k_n)$ is legal in the context $\Gamma \cup \{z : t^a\}$ (up to α_P-equality).*

It is also not hard to show that $f \vee g$ is legal if f and g are (see also Remark 45):

LEMMA 67. *If f and g are legal in contexts Γ_1 and Γ_2, respectively, and $\Gamma_1 \cup \Gamma_2$ is a context, then $f \vee g$ is legal in the context $\Gamma_1 \cup \Gamma_2$ (up to α_P-equality).*

The following lemma is easy to prove and will be used in the proof of the main result of this section.

LEMMA 68. *If $R(i_1, \dots, i_{\mathfrak{a}(R)})$ is a pf with free variables $x_1 < \cdots < x_m$, then it is legal in the context $\{ x_j : 0 \mid 1 \leq j \leq m \}$.*

PROOF. Write $f = R(i_1, \dots, i_{\mathfrak{a}(R)})$. Let $a_1, \dots, a_m \in \mathcal{A}$ be m different individuals that do not occur in f, and replace each variable x_j in f by a_j, calling the result f'. By the first rule of RTT, f' is legal in the empty context. Re-introducing the variables $x_1, \dots, x_m$ (by applying rule 3 of RTT m times) for the individuals $a_1, \dots, a_m$, respectively, we obtain that f is legal in the context $\{ x_j : 0 \mid 1 \leq j \leq m \}$. ⊣

Finally, we can give a characterisation of the legal pfs:

THEOREM 69 (Legal pfs in RTT). *Let $f \in \mathcal{P}$. f is legal (mod α_P) if and only if:*

- *$f \equiv R(i_1, \dots, i_{\mathfrak{a}(R)})$, or*
- *$f \equiv z(k_1, \dots, k_n)$, $z \neq k_j$ for all $k_j \in \mathcal{V}$ and z does not occur in any $k_j \in \mathcal{P}$, and there is Γ with $\text{FV}(f) \subseteq \text{DOM}(\Gamma)$ and for all $k_j \in \mathcal{P}$, $\Gamma \vdash k_j{:}t_j^{a_j}$ for some predicative type $t_j^{a_j}$, or*
- *$f \equiv \neg f'$ and f' is legal (mod α_P) or*
- *$f \equiv f_1 \vee f_2$ and there are Γ_i and $t_i^{a_i}$ such that $\Gamma_i \vdash f_i : t_i^{a_i}$ (mod α_P) for $i = 1, 2$ and $\Gamma_1 \cup \Gamma_2$ is a context, or*
- *$f \equiv \forall x : t^a.f'$ and f' is legal.*

PROOF. Use induction on the structure of f. ⊣

We can now answer the question whether our legal pfs (as given in Definition 41) are the same as the formulas of the *Principia*.

First of all, we must notice that all the legal pfs from Definition 41 are also formulas of the *Principia*: This was motivated in Remark 43.

Moreover, we proved (in Theorem 69) that if f is a pf, then the only reasons why f cannot be legal (according to Definition 41) are:

- There is a constituent $z(k_1, \dots, k_m)$ of f in which z occurs in one of the k_is;
- There is a constituent $z(k_1, \dots, k_m)$ of f and a $j \in \{1, \dots, m\}$ such that k_j is a pf, but not a legal pf;
- f contains two non-overlapping constituents f_1, f_2 that cannot be typed in one and the same context;
- There is a legal constituent $z(k_1, \dots, k_m)$ of f which is not of predicative type.

Pfs of the first type cannot be legal in the *Principia*, because of the vicious circle principle. The same holds for pfs of the second type, because also in the *Principia*, parameters cannot be untyped. The third problem is a non-issue in the *Principia*. Formal contexts are not present in the *Principia*, but have been introduced in this article to make a precise analysis of RTT possible. Propositional functions of the *Principia* are always constructed in one, implicitly defined, context.[40] A formula, therefore, cannot contain two non-overlapping constituents that cannot be typed in the same context. This excludes pfs of the third type. As to the fourth type, it represents Russell's assumption that non-predicative orders in his hierarchy are always obtained from predicative ones by generalization (i.e., by quantification). Of course Russell's assumption is not true of terms $z(k_1, \ldots, k_n)$ where one of the k_is happens to be of non-predicative type. This means that both our system and Russell's intended system are not able to type such terms.

We conclude that we have described the legal pfs of the *Principia Mathematica* with the formal system RTT.

We present some refinements of Theorem 69:

THEOREM 70. *Assume* $\Gamma \vdash f : t^a$.

- *If* $f \equiv R(i_1, \ldots, i_{\mathfrak{a}(R)})$ *and* $x \in \text{FV}(f)$ *then* $x : 0^0 \in \Gamma$;
- *If* $f \equiv z(k_1, \ldots, k_m)$ *then there are* $u_1^{b_1}, \ldots, u_m^{b_m}, b$ *such that*
  - $z : (u_1^{b_1}, \ldots, u_m^{b_m})^b \in \Gamma$;
  - $\Gamma \vdash k_i : u_i^{b_i}$ *for* $k_i \in \mathcal{A} \cup \mathcal{P}$;
  - $k_i : u_i^{b_i} \in \Gamma$ *for* $k_i \in \mathcal{V}$.

In this section we gave a formalisation of the Ramified Theory of Types. Some of the main ideas underlying this theory were already present in Frege's Abstraction Principles 1 and 2.

RTT not only prevents the paradoxes of Frege's *Grundgesetze der Arithmetik*, but also guarantees the well-definedness of substitution (Theorem 61).

[40] It is worth remarking that it is possible to formalize *Principia* without resorting to explicit contexts at all. For example, Randall Holmes has an implementation which constructs contexts from the structures of the terms analysed. Following Randall Holmes, the price of this is that the types deduced for terms by his checker are polymorphic: for STT this isn't a problem at all (it's an advantage); Holmes expresses that in RTT, the handling of polymorphic types was quite difficult—he had to allow orders defined in terms of the unknown orders of polymorphic types. Further, in RTT, the type checker had to be much smarter than the STT checker, because it had to be able to deduce identity between polymorphic types in order to successfully infer types for quite simple terms (such as the "definition of equality" $(\forall x.(x(y) \leftrightarrow x(z)))$, where the two variables y and z are both polymorphic, and one has to be careful to determine that they have the same type (because they are in the same argument of the same unknown pf x) before attempting the final type-checking of the term: if one is not careful about the order in which things are done, two incompatible types for x will be deduced depending on unknown and possibly different orders for y and z).

This second problem was not realized in the *Principia*, where substitution did not even have a proper definition.

There is a close relation between substitution in *Principia* and β-reduction in λ-calculus (Definition 23). RTT has characteristics that are also the basic properties of modern type systems for λ-calculus.

As there is no real reduction in RTT, we don't have an equivalent of the Subject Reduction theorem. However, the fact that the Free Variable property (Theorem 54) is maintained under substitution can be seen as a (very weak) form of Subject Reduction.

Expressing Russell's propositional functions in λ-calculus has made it possible to compare these pfs with λ-terms. We found that pfs can be seen as λ-terms, but in a rather simple way:

- A pf is always a λI-term, i.e., if $\lambda x : A.B$ is a subterm of the translation $\widetilde{f}$ of a pf f, then $x \in \mathrm{FV}(B)$;
- Substitution in the *Principia* can be seen as application plus β-reduction to normal form.

Although the description of the Ramified Theory of Types in the *Principia* is very informal, it is remarkable that an accurate formalisation of this system can be made (see Theorem 69 and the discussion that follows it). The formalisation shows that Russell and Whitehead's ideas on the notion of types, though very informal to modern standards, must have been very thorough and to the point.

A characteristic of RTT that is maintained in many modern type systems is the syntactic nature of the system: type and order of a pf are determined on purely syntactical grounds. No attention is paid to the interpretation of such a pf. This is remarkable, as the propositions $\forall \mathtt{x}\colon 0^0\,[\mathtt{R}(\mathtt{x})]$ and $\forall \mathtt{x}\colon 0^0[\mathtt{R}(\mathtt{x})] \vee \forall \mathtt{z}\colon (\,)^9\,[\mathtt{z}(\,) \wedge \neg \mathtt{z}(\,)]$ are logically equivalent in most logics,[41] though they are of different type (the former pf has type $(\,)^1$ and the latter has type $(\,)^{10}$). In [42], it is shown that other viewpoints are possible besides this concentration on syntax.

§4. History of the deramification.

4.1. The problematic character of RTT. The main part of the *Principia* is devoted to the development of logic and mathematics using the legal pfs of the ramified type theory. It appears that RTT is not easy to use. The main reason for this is the implementation of the so-called *ramification*: the division of simple types into orders. We illustrate this with two examples:

EXAMPLE 71 (Equality). One tends to define the notion of *equality* in the style of Leibniz ([32]):

$$\mathtt{x} =_L \mathtt{y} \overset{\text{def}}{\leftrightarrow} \forall \mathtt{z}\,[\mathtt{z}(\mathtt{x}) \leftrightarrow \mathtt{z}(\mathtt{y})],$$

[41] At least in all the logical systems that Russell had in mind when he wrote the *Principia*.

or in words: Two individuals are equal if and only if they have exactly the same properties.

Unfortunately, in order to express this general notion in our formal system, we have to incorporate *all* pfs $\forall \mathtt{z}\colon (0^0)^n\ [\mathtt{z}(\mathtt{x}) \leftrightarrow \mathtt{z}(\mathtt{y})]$ for $n > 1$, and this cannot be expressed in one pf.

The ramification does not only influence definitions in logic. Some important mathematical concepts cannot be defined any more:

Example 72 (Real numbers, least upper bounds). Dedekind constructed the real numbers from the rationals using so-called Dedekind cuts. In this construction, a real number is a set r of rationals such that

- $r \neq \varnothing$;
- $r \neq \mathbb{Q}$;
- If $x \in r$ and $y < x$ then $y \in r$;
- If $x \in r$ then there is $y \in r$ with $x < y$.

For instance, the real number $1/2$ is represented by the set $\{\, x \in \mathbb{Q} \mid 2x < 1 \,\}$, and the real number $\sqrt{2}$ is represented by the set $\{\, x \in \mathbb{Q} \mid x < 0 \text{ or } x^2 < 2 \,\}$.

If we take $\mathbb{Q}$ as the set of individuals $\mathcal{A}$, and assume that the binary relation $<$ on $\mathbb{Q}$ is an element of $\mathcal{R}$, the set of relations, we can see real numbers as unary predicates f over $\mathbb{Q}$ such that

$$(6)\quad \exists \mathtt{x}\colon 0^0\ [\mathtt{z}(\mathtt{x})] \wedge \exists \mathtt{x}\colon 0^0[\neg \mathtt{z}(\mathtt{x})] \\ \wedge \forall x\colon 0^0\ [\forall y\colon 0^0\ [\mathtt{z}(\mathtt{x}) \to \mathtt{y} < \mathtt{x} \to \mathtt{z}(\mathtt{y})]] \\ \wedge \forall x\colon 0^0\ [\mathtt{z}(\mathtt{x}) \to \exists \mathtt{y}\colon 0^0\ [\mathtt{z}(\mathtt{y}) \wedge \mathtt{x} < \mathtt{y}]]$$

holds if we substitute f for $\mathtt{z}$. We will abbreviate the predicate (6) (with the free variable $\mathtt{z}$) as $\mathbb{R}$. It has type $((0^0)^1)^2$, and real numbers can be seen as pfs of type $(0^0)^1$. We will, for shortness of notation, write $\mathbb{R}(f)$ for $\mathbb{R}[\mathtt{z} := f]$, so $\mathbb{R} \equiv \mathbb{R}(\mathtt{z})$. A real number r is smaller than or equal to another real number r' if for all x with $r(x)$, also $r'(x)$ holds. We write, shorthand, $r \leq r'$ if r is smaller than or equal to r'.

In traditional mathematics, the above would define a system that obeys the traditional axioms for real numbers. In particular, the theorem of the least upper bound holds for this system. This theorem states that each non-empty subset of $\mathbb{R}$ with an upper bound has a least upper bound. In our formalism:

$$\forall \mathtt{v} \subseteq \mathbb{R}\ \Big[\big(\exists \mathtt{z}_1 \in \mathbb{R}\ [\mathtt{v}(\mathtt{z}_1)] \wedge \exists \mathtt{z}_2 \in \mathbb{R}\ \forall \mathtt{z}_3 \in \mathbb{R}\ [\mathtt{v}(\mathtt{z}_3) \to \mathtt{z}_3 \leq \mathtt{z}_2]\big) \\ \to \exists \mathtt{z}_1 \in \mathbb{R}\ \big[\forall \mathtt{z}_2 \in \mathbb{R}\ [\mathtt{v}(\mathtt{z}_2) \to \mathtt{z}_2 \leq \mathtt{z}_1] \\ \wedge \forall \mathtt{z}_3 \in \mathbb{R}\ [\forall \mathtt{z}_4 \in \mathbb{R}\ [\mathtt{v}(\mathtt{z}_4) \to \mathtt{z}_4 \leq \mathtt{z}_3] \to \mathtt{z}_1 \leq \mathtt{z}_3]\big]\Big].$$

(We write, shorthand, $\forall v \subseteq \mathbb{R}\ [g]$ to denote $\forall v\colon ((0^0)^1)^2\ [\forall \mathtt{u}\colon (0^0)^1\ [v(\mathtt{u}) \to \mathbb{R}(\mathtt{u})] \to g]$, and $\forall z \in \mathbb{R}\ [g]$ to denote $\forall z\colon (0^0)^1\ [\mathbb{R}(z) \to g]$).

If we try to prove this theorem within the system of Dedekind as formulated in the *Principia*-language RTT, we have to specify a type t^a for the variable $\mathtt{z}_1$. As $\mathtt{z}_1$ must be a real number, its type must be $(0^0)^1$. If we give a proof of the theorem, and construct some object f that should be the least upper bound of a set of real numbers V, f will depend on V. Therefore, a general description of f will have a variable v for V in it. As v is of order 2, f must be of order 3 or more. Therefore, f cannot be a real number, since real numbers have order 1. This makes it impossible to give a constructive proof of the theorem of the least upper bound within a ramified type theory.

This is a consequence of the fact that it is not possible in RTT to give a definition of an object that refers to the class to which this object belongs (because of the Vicious Circle Principle). Such a definition is called an *impredicative* definition. The relation with the notion of impredicative *type* is immediate:[42] an object defined by an impredicative definition is of a higher order than the order of the elements of the class to which this object should belong. This means that the defined object f has an *impredicative* type.

Nowadays we would consider the use of the Vicious Circle Principle too strict. We consider the impredicative definition of f as a matter of *syntax*, whilst the existence of the object f has to do with *semantics*.[43] The fact that we are not able to give a predicative definition of f does not imply that such an object does not exist. Here we must remark that Russell and Whitehead did not make a distinction between syntax and semantics in the *Principia*.[44] Therefore they had to interpret the Vicious Circle Principle in the strict way above.

---

[42]This terminology is again the one assumed by *Principia* and not everyone agrees with it. There is actually no problem with the formulation of "impredicative" types from a predicative standpoint: objects of these types are predicatively respectable. An object of truly impredicative type would be defined using quantifiers over *its own* type or even higher types (as is allowed in simple type theory), and would not be typable in the ramified theory at all.

[43]This is obviously a point on which one might disagree. For example, Randall Holmes is unconvinced by the remarks about "syntax" and "semantics". He believes that the syntactical criteria of ramified type theory are a correct implementation of the Vicious Circle Principle and that the Vicious Circle Principle is best understood as a criterion appropriate for *definitions*. According to Randall Holmes, if instances of abstraction or comprehension principles are to be thought of as definitions, then impredicative abstraction or comprehension is indeed questionable and hence the conclusion to be drawn is that abstraction or comprehension axioms are not definitions, but assertions of matters of fact (so "semantic" rather than "syntactic"), and so are not subject to the Vicious Circle Principle (it is not that it should be applied in a more lenient way, but that it does not apply at all). But as long as the Vicious Circle Principle is to be applied, syntactical criteria are appropriate: what a correct definition is should be a matter of syntax.

[44]Though the basic ideas for this were already present in the works of Frege. See for instance *Über Sinn und Bedeutung* [29].

4.2. The Axiom of Reducibility. Russell and Whitehead tried to solve these problems explained in Section 4.1 with the so-called *axiom of reducibility*.

AXIOM 73 (Axiom of Reducibility). *For each formula f, there is a formula g with a predicative type such that f and g are (logically) equivalent.*

Accepting this axiom, one may define equality on formulas of order 1 only:

$$\mathtt{x} =_1 \mathtt{y} \stackrel{\text{def}}{=} \forall \mathtt{z}\colon (0^0)^1\, [\mathtt{z}(\mathtt{x}) \leftrightarrow \mathtt{z}(\mathtt{y})].$$

If f is a function of type $(0^0)^n$ for some $n > 1$, and $\mathtt{a}$ and $\mathtt{b}$ are individuals for which the Leibniz equality $\mathtt{a} =_L \mathtt{b}$ holds then $f(\mathtt{a}) \leftrightarrow f(\mathtt{b})$ holds: With the Axiom of Reducibility we can determine a predicative function g (so of type $(0^0)^1$), equivalent to f. As g has order 1, $g(\mathtt{a}) \leftrightarrow g(\mathtt{b})$ holds. And because f and g are equivalent, also $f(\mathtt{a}) \leftrightarrow f(\mathtt{b})$ holds. This solves the problem of Example 71. A similar solution gives, in Example 72, the proof of the theorem of the least upper bound.

The validity of the Axiom of Reducibility has been questioned from the moment it was introduced. In the introduction to the 2nd edition of the *Principia*, Whitehead and Russell admit:

> "This axiom has a purely pragmatic justification: it leads to the desired results, and to no others. But clearly it is not the sort of axiom with which we can rest content."
>
> (*Principia Mathematica*, p. xiv)

Though Weyl [70] made an effort to develop analysis within the Ramified Theory of Types (but without the Axiom of Reducibility), and various parts of mathematics can be developed within RTT and without the Axiom,[45] the general attitude towards RTT (without the axiom) was that the system was too restrictive, and that a better solution had to be found.

4.3. Deramification. The first impulse to such a solution was given by Ramsey in 1926 [59]. He recalls that the Vicious Circle Principle 3 was postulated in order to prevent the paradoxes. Though all the paradoxes were prevented by this Principle, Ramsey considers it essential to divide them into two parts:

1. One group of paradoxes is removed

> "by pointing out that a propositional function cannot significantly take itself as argument, and by dividing functions and classes into a hierarchy of types according to their possible arguments."
>
> (The Foundations of Mathematics, p. 356)

[45]See [40], where many algebraic notions are developed within the Nuprl Proof Development System, a proof checker based on the hierarchy of types and orders of RTT without the Axiom of Reducibility.

This means that a class can never be a member of itself. The paradoxes solved by introducing the hierarchy of types (but not orders), like the Russell paradox, and the Burali-Forti paradox, are *logical* or *syntactical* paradoxes;

2. The second group of paradoxes is excluded by the hierarchy of orders. These paradoxes (like the Liar's paradox, and the Richard Paradox) are based on the confusion of language and meta-language. These paradoxes are, therefore, not of a purely mathematical or logical nature. When a proper distinction between object language (the pfs of the system RTT, for example) and meta-language is made, these so-called *semantical* paradoxes disappear immediately.

Ramsey agrees with the part of the theory that eliminates the syntactic paradoxes. This part is in fact RTT without the orders of the types. The second part, the hierarchy of orders, does not gain Ramsey's support, for the reasons described above. Moreover, by accepting the hierarchy in its full extent one either has to accept the Axiom of Reducibility or reject ordinary real analysis. Ramsey is supported in his view by Hilbert and Ackermann [37]. They all suggest a *deramification* of the theory, i.e., leaving out the orders of the types. When making a proper distinction between language and meta-language, the deramification will not lead to a re-introduction of the (semantic) paradoxes.

The solution proposed by Ramsey, and Hilbert and Ackermann, looks better than the Axiom of Reducibility. Nevertheless, both deramification and the Axiom of Reducibility are violations of the Vicious Circle Principle, and reasons (of a more fundamental character than "they do not lead to a re-introduction of the semantic paradoxes" and "it leads to the desired results, and to no others") why these violations can be harmlessly made must be given. Gödel [34] fills in this gap. He points out that whether one accepts this second principle or not, depends on the philosophical point of view that one has with respect to logical and mathematical objects:

> "it seems that the vicious circle principle [...] applies only if the entities involved are constructed by ourselves. In this case there must clearly exist a definition (namely the description of the construction) which does not refer to a totality to which the object defined belongs, because the construction of a thing can certainly not be based on a totality of things to which the thing to be constructed itself belongs. If, however, it is a question of objects that exist independently of our constructions, there is nothing in the least absurd in the existence of totalities containing members, which can be described only by reference to this totality."
>
> (Russell's mathematical logic)

The remark puts the Vicious Circle Principle back from a proposition (a statement that is either true or false, without any doubt) to a philosophical principle that will be easily accepted by, for instance, intuitionists (for whom mathematics is a pure mental construction) or constructivists, but that will be rejected, at least in its full strength, by mathematicians with a more platonic point of view. It should be noted that intuitionistic mathematics is quite often impredicative although different "constructivist" mathematicians have different opinions about this.

Gödel is supported in his ideas by Quine [58], sections 34 and 35. Quine's criticism on impredicative definitions (for instance, the definition of the least upper bound of a nonempty subset of the real numbers with an upper bound) is not on the definition of a special symbol, but rather on the very assumption of the *existence* of such an object at all. Quine continues by stating that even for Poincaré, who was an opponent of impredicative definitions and deramification, one of the doctrines of classes is that they are there "from the beginning". So, even for Poincaré there should be no evident fallacy in impredicative definitions.

The deramification has played an important role in the development of type theory. In 1932 and 1933, Church presented his (untyped) λ-calculus [12, 13]. In 1940 he combined this theory with a deramified version of Russell's theory of types to the system that is known as the *simply typed λ-calculus*[46].

§5. The Simple Theory of Types.

5.1. Constructing the Simple Theory of Types STT from RTT. So far, we have seen the development of type theory since the appearance of *Principia Mathematica* (1910–1912) went through a process of deramification where Ramsey [59], and Hilbert and Ackermann [37], simplified the Ramified Theory of Types by removing the orders. The result is known as the *Simple Theory of Types* (STT).

Nowadays, STT is known via Church's formalisation in λ-calculus. However, STT already existed (1926) before λ-calculus did (1932), and is therefore not inextricably bound up with λ-calculus. In this section we show how we can obtain a formalisation of STT directly from the formalisation of RTT that was presented in Section 3 by simply removing the orders. Most of the properties that were proved for RTT hold for STT as well, including Unicity of Types and Strong Normalisation. The proofs are all similar to the proofs that were given for RTT. We also make a comparison between Church's formalisation in λ-calculus and the formalisation of STT that is obtained from

[46]Thus, the adjective *simple* is used to distinguish the theory from the more complicated—both in its construction with a double hierarchy and in its use—*ramified* theory. The classification "simple", therefore, has nothing to do with the fact that STT, formulated with λ-calculus as described in [14], is the simplest system of the Barendregt Cube (see [3]).

RTT. It appears that Church's system is much more than only a formalisation. Because of the λ-calculus, Church's system is more expressive.[47]

It is straightforward to carry out the deramification as it was originally proposed by Ramsey, Hilbert and Ackermann: We take the formalisation of RTT that was presented in Section 3, and leave out all the orders and the references to orders (including the notions of predicative and impredicative types). The system we obtain in this way will be denoted STT. The types used in the system are the simple types of Definition 30.

The following definitions, lemmas, theorems and corollaries, including their proofs, can be adapted to STT without any problems: Definitions 38, 39, 40, 41, Lemma 52, Theorems 53 (first free variable theorem), 54 (second free variable theorem), Corollaries 55, 57 (unicity of types), and Theorem 61 (existence of substitution).

The description of legal pfs for STT follows the same line as in Section 3.9, with straightforward adaptions of Definition 62, and Lemmas 63 (now, all simple types are inhabited), 66, 67, 68, and finally Theorem 69 (characterisation of legal pfs):

THEOREM 74 (Legal pfs in STT). *Let* $f \in \mathcal{P}$. f *is legal* (*mod* α) *if and only if*:

- $f \equiv R(i_1, \ldots, i_{\mathfrak{a}(R)})$, *or*
- $f \equiv z(k_1, \ldots, k_n)$, $z \neq k_j$ *for all* $k_j \in \mathcal{V}$ *and* z *does not occur in any* $k_j \in \mathcal{P}$, *and there is* Γ *with* $\text{FV}(f) \subseteq \text{DOM}(\Gamma)$ *and for all* $k_j \in \mathcal{P}$, $\Gamma \vdash k_j : t_j$, *or*
- $f \equiv \neg f'$ *and* f' *is legal* (*mod* α) *or* $f \equiv f_1 \vee f_2$, *there are* Γ_i *and* t_i *such that* $\Gamma_i \vdash f_i : t_i$ (*mod* α) *and* $\Gamma_1 \cup \Gamma_2$ *is a context*, *or*
- $f \equiv \forall x : t.f'$ *and* f' *is legal.*

A comparison between the formalisations of STT and RTT can easily be made using Theorems 74 and 69. We find that

- All RTT-legal pfs are (when the ramified types behind the quantifiers are replaced by their corresponding simple types) STT-legal;
- A STT-legal pf f is RTT-legal, except when f contains a subformula of the form $z(k_1, \ldots, k_n)$, where one or more of the k_js are not RTT-legal or can only be typed in RTT by an impredicative type.

[47] The removal of orders from type theory may suggest that orders are to be blamed for the restrictiveness of RTT, and that the concept of order is problematic. [42] shows that this is not necessarily the case by introducing a system KTT, based on Kripke's Hierarchy of Truths [45], that has an approach completely opposite to STT. Whilst STT is *order*-free, and *types* play the main role, Kripke's Hierarchy of Truths is *type*-free, and *orders* play an important, though not a restrictive, role. The main difference between Kripke's and Russell's notion of order is that Russell's classification is purely syntactical, whilst Kripke's is essentially semantical. [42] shows that RTT can be embedded in KTT and that there is a straightforward relation between the orders in RTT and the hierarchy of truths of KTT.

5.2. Church's simply typed λ-calculus $\lambda{\rightarrow}_C$. We give a definition of the simply typed λ-calculus as introduced by Church [14] in 1940.

The types and terms in the original presentation of $\lambda{\rightarrow}_C$ are a bit different from the presentation in [3]. We give some explanation after repeating the original definition:

DEFINITION 75 (Types of $\lambda{\rightarrow}_C$). The types of $\lambda{\rightarrow}_C$ are defined as follows:

- $\imath$ and o are types;
- If α and β are types, then so is $\alpha \rightarrow \beta$.

We denote the set of simple types by $\mathbb{T}$.

$\imath$ represents the type of individuals; o is the type of propositions. $\alpha \rightarrow \beta$ is the type of functions with domain α and range β. We use $\alpha, \beta, \ldots$ as meta-variables over types. $\rightarrow$ associates to the right: $\alpha \rightarrow \beta \rightarrow \gamma$ denotes $\alpha \rightarrow (\beta \rightarrow \gamma)$.

DEFINITION 76 (Terms of $\lambda{\rightarrow}_C$). The terms of $\lambda{\rightarrow}_C$ are the following:

- $\neg$, $\wedge$, $\forall_\alpha$ for each type α, and $\imath_\alpha$ for each type α, are terms;
- A variable is a term;
- If A, B are terms, then so is AB;
- If A is a term, and x a variable, then $\lambda x\colon \alpha.A$ is a term.

REMARK 77. We see that the constants $\neg$, $\wedge$, $\forall_\alpha$ and $\imath_\alpha$ are terms. This may need some explanation for the modern reader.

- Church considers $\neg$ and $\wedge$ to be functions. The function $\neg$ takes a proposition as argument, and returns a proposition; similarly $\wedge$ takes two propositions as arguments, and returns a proposition. In Definition 79, we see that $\neg$ and $\wedge$ are assigned the corresponding types $o \rightarrow o$ and $o \rightarrow o \rightarrow o$;
- More remarkable: $\forall_\alpha$ and $\imath_\alpha$ are just terms, and do not act as binding operators. The usual variable binding of $\forall_\alpha$ and $\imath_\alpha$ is obtained via λ-abstraction: instead of $\forall x\colon \alpha.f$, Church writes $\forall_\alpha(\lambda x\colon \alpha.f)$. In this way, $\forall_\alpha$ is a function that takes a propositional function of type $\alpha \rightarrow o$ as argument, and returns a proposition (a term of type o). In Definition 79, $\forall_\alpha$ obtains the corresponding type $(\alpha \rightarrow o) \rightarrow o$. Similarly, the unique choice operator $\imath_\alpha$ takes a propositional function of type $\alpha \rightarrow o$ as argument, and returns a term of type α. The term $\imath x\colon \alpha.f$, or in Church's notation: $\imath_\alpha(\lambda x\colon \alpha.f)$, has as interpretation: the (unique) object t of type α for which $f[x := t]$ holds. Correspondingly, the type of $\imath_\alpha$ is $(\alpha \rightarrow o) \rightarrow \alpha$.

DEFINITION 78 (Contexts of $\lambda{\rightarrow}_C$). A *context* in $\lambda{\rightarrow}_C$ is a set

$$\{x_1\colon \alpha_1, \ldots, x_n\colon \alpha_n\}$$

where the x_i are distinct variables and the α_i are types.

Some terms are typable (legal) in $\lambda{\rightarrow}_C$, according to the following derivation rules:

DEFINITION 79 (Typing rules of $\lambda{\to}_C$). The judgement $\Gamma \vdash A : \alpha$ holds if it can be derived using the following rules:

- $\Gamma \vdash \neg\colon o \to o$;
  $\Gamma \vdash \wedge\colon o \to o \to o$;
  $\Gamma \vdash \forall_\alpha\colon (\alpha \to o) \to o$;
  $\Gamma \vdash \imath_\alpha\colon (\alpha \to o) \to \alpha$;
- $\Gamma \vdash x\colon \alpha$ if $x\colon \alpha \in \Gamma$;
- If $\Gamma, x\colon \alpha \vdash A\colon \beta$ then $\Gamma \vdash (\lambda x\colon \alpha.A)\colon \alpha \to \beta$;
- If $\Gamma \vdash A\colon \alpha \to \beta$ and $\Gamma \vdash B\colon \alpha$ then $\Gamma \vdash (AB)\colon \beta$.

We use $\vdash_{\lambda\to_C}$ if we need to distinguish derivability in $\lambda{\to}_C$ from derivability in other type systems.

The simply typed λ-calculus can be seen as a pure type system, and therefore has the properties of pure type systems [3]. To adapt the simply typed λ-calculus to a pure type system, some amendments are made:

- The two basic types $\imath$, o are replaced by an infinite set of *type variables*;
- The constants $\neg$, $\wedge$, $\forall_\alpha$ and $\imath_\alpha$ are not introduced in the PTS-presentation.

These adaptions do not seriously affect the system and are only used to make $\lambda{\to}_C$ fit in the PTS-framework.

5.3. Comparing RTT and $\lambda{\to}_C$. Apart from the orders, RTT is a subsystem of $\lambda{\to}_C$ via the embeddings $\bar{\ }$ of Section 3.2 and a mapping T that we define below. There are, however, important differences between the way in which the type of a pf is determined in RTT, and the way in which the type of a λ-term is determined in λ-Church. The rules of RTT, and the method of deriving the types of pfs that was presented in Section 3.9, have a bottom-up character: one can only introduce a variable of a certain type in a context Γ, if there is a pf that has that type in Γ. In $\lambda{\to}_C$, one can introduce variables of any type without wondering whether such a type is inhabited or not.

Church's $\lambda{\to}_C$ is more general than RTT in the sense that Church does not only describe (typable) *propositional* functions. In $\lambda{\to}_C$, also functions of type $\tau \to \imath$ (where $\imath$ is the type of individuals) can be described, and functions that take such functions as arguments, etc.

Just as propositional functions can be translated to λ-terms, simple types (see Definition 30) can be translated to types of the simply typed λ-calculus of Church.

DEFINITION 80 (Translating simple types to $\lambda{\to}_C$-types). We define a type $T(t)$ for each simple type t by induction:

1. $T(0) \stackrel{\text{def}}{=} \imath$;
2. $T((t_1,\dots,t_n)) \stackrel{\text{def}}{=} T(t_1) \to \cdots \to T(t_n) \to o$.

A simple type t of Definition 30 has the same interpretation as its translation $T(t)$. Moreover, T is injective:

LEMMA 81. *If t and u are simple types* (*Definition* 30), *then* $T(t) = T(u)$ *if and only if* $t = u$.

PROOF. Induction on the definition of simple type. ⊣

The mapping T is injective when restricted to predicative types:

LEMMA 82. *If* t^a *and* u^b *are predicative types, then* $T(t^a) = T(u^b)$ *if and only if* $t^a = u^b$.

PROOF. Induction on the definition of predicative type. ⊣

Ramified types can also be translated to types of the simply typed λ-calculus. However, we lose the orders if we do so.

DEFINITION 83 (Translating ramified types to $\lambda{\rightarrow}_C$-types). We define a type $T(t)$ for each ramified type t by induction:

1. $T(0^0) \stackrel{\text{def}}{=} \iota$;
2. $T((t_1^{a_1}, \ldots, t_n^{a_n})^a) \stackrel{\text{def}}{=} T(t_1) \to \cdots \to T(t_n) \to o$.

Now we can relate typing in RTT to that of Church's $\lambda{\rightarrow}_C$:

THEOREM 84 (Typability in RTT implies typability in $\lambda{\rightarrow}_C$). *If* $\Gamma \vdash f : t^a$ *in* RTT *then*

1. $T(\Gamma) \vdash_{\lambda\rightarrow_C} \widetilde{f} : o$;
2. $T(\varnothing) \vdash_{\lambda\rightarrow_C} \overline{f} : T(t^a)$.

PROOF. A straightforward induction on $\Gamma \vdash f : t^a$ with the use of Theorem 54 and the Subject Reduction property for λ-Church. ⊣

REMARK 85. Observe that the above theorem immediately excludes the pf that leads to the Russell Paradox from the well-typed pfs: If $\neg\mathtt{z}(\mathtt{z})$ were legal then the λ-term $\neg(\mathtt{zz})$ would be typable in λ-Church, which is not the case (see [3]).

5.4. Comparison of STT with Church's $\lambda{\rightarrow}_C$. The mappings T for types and $\bar{}$ for terms (see Definitions 83 and 9), adapted for STT, make it possible to compare STT with $\lambda{\rightarrow}_C$.

Regarding the types, we find that T gives an injective correspondence between types of STT and $\lambda{\rightarrow}_C$. T is clearly not surjective, as $T(t)$ is never of the form $\alpha \to \iota$ (this follows directly from Definition 80). This indicates an important difference between STT and $\lambda{\rightarrow}_C$. In RTT and STT, functions (other than propositional functions) have to be defined via relations (and this is the way it is done in *Principia Mathematica*). The value of such a function f, described via the relation R, for a certain value a is described using the ι-operator: $\iota y.R(a, y)$ (to be interpreted as: the unique y for which $R(a, y)$ holds). Things get even more complicated if one realizes that the ι-operator is not a part of the syntax used in *Principia Mathematica*, but an abbreviation with a not so straightforward translation (see [71], pp. 66–71). In $\lambda{\rightarrow}_C$, as everywhere in λ-calculus, functions (both propositional functions and other ones) are first-class citizens, which means that the construction with the ι-operator is not the first tool to be used when constructing a function. If one has an algorithm (a λ-term) that describes the function f, the value of f for the argument a can be easily described via the term fa. And even if such an algorithm is not at hand, one can use the ι-operator, which is part of the syntax of $\lambda{\rightarrow}_C$. This makes $\lambda{\rightarrow}_C$ much easier to use for the formalisation of logic and mathematics than RTT and STT.

Regarding the terms, $\bar{\ }$ provides an injective correspondence between terms of STT and $\lambda{\rightarrow}_C$. Again, this mapping is not surjective, for several reasons:

- T is not surjective. As there is no t with $T(t) = \iota \rightarrow \iota$, there cannot be a legal pf f such that $\overline{f} \equiv \lambda \mathtt{x}{:}\ \iota.\mathtt{x}$;
- We already observed that $\overline{f}$ is a λI-term for all $f \in \mathcal{P}$. $\lambda{\rightarrow}_C$ also allows terms like $\lambda \mathtt{x}{:}\ \alpha.\mathtt{y}$;
- If $\overline{f} \equiv zH_1 \cdots H_n$ for some $z \in \mathcal{V}$ and some terms $H_1, \ldots, H_n$, the H_is must be either *closed* λ-terms, or variables, or individuals. This means that there is no $f \in \mathcal{P}$ such that $\overline{f} \equiv \lambda \mathtt{z}{:}\ o{\rightarrow}o.\lambda \mathtt{x}{:}\ \iota.\mathtt{z}(\mathtt{Rx})$, since $\mathtt{Rx}$ contains the free variable $\mathtt{x}$ and is neither a variable nor an individual;
- We remark that $\overline{f}$ is always a closed λ-term, so there is no $f \in \mathcal{P}$ such that $\overline{f} \equiv \mathtt{x}$;
- It has already been remarked that the ι-operator is part of the syntax of $\lambda{\rightarrow}_C$, and this is not the case in STT and RTT.

The discussion above makes clear that $\lambda{\rightarrow}_C$ is a far more expressive system than RTT and STT. Type-theoretically, it generalises the idea of function types of Frege and Russell from propositional functions to more general functions.

Philosophically, there is another important difference between STT and $\lambda{\rightarrow}_C$. The systems STT and RTT have a strong bottom-up approach: To type a higher-order pf one has to start with propositions of order 0. Only by applying the abstraction principles, it is possible to obtain higher-order pfs. In $\lambda{\rightarrow}_C$, one can introduce a variable of a higher-order type at once, without having to refer to terms of lower order.

§6. Conclusion. In this article, we gave a history of type theory up to 1910 and presented in detail the first type theory RTT due to Russell which he used to prevent the paradoxes of Frege's *Grundgesetze der Arithmetik*. Then we discussed the deramification of RTT (i.e., the removal of orders) leading to the simple theory of types STT. We also presented Church's simply typed λ-calculus $\lambda{\rightarrow}_C$ and compared the three type systems RTT, STT and $\lambda{\rightarrow}_C$.

Some of the main ideas underlying RTT were already present in Frege's Abstraction Principles 1 and 2.

RTT not only prevents the paradoxes of Frege's *Grundgesetze der Arithmetik*, but also guarantees the well-definedness of substitution, as we have shown in Corollary 61. This second problem was not realized in the *Principia*, where substitution did not even have a proper definition.

There is a close relation between substitution in *Principia* and β-reduction in λ-calculus (Definition 23). RTT has characteristics that are also the basic properties of modern type systems for λ-calculus. As there is no real reduction in RTT, we don't have an equivalent of the Subject Reduction theorem. However, the fact that the Free Variable property 54 is maintained under substitution can be seen as a (very weak) form of Subject Reduction.

Although the description of the Ramified Theory of Types in the *Principia* is very informal, it is remarkable that an accurate formalisation of this system can be made (see Theorem 69 and the discussion that follows it). The formalisation shows that Russell and Whitehead's ideas on the notion of types, though very informal to modern standards, must have been very thorough and to the point.

Apart from the orders, RTT is a subsystem of Church's $\lambda{\rightarrow}_C$ of [14] via the embeddings $\bar{}$ of Section 3.2 and T of Section 5.3. There are, however, important differences between the way in which the type of a pf is determined in RTT, and the way in which the type of a λ-term is determined in λ-Church. The rules of RTT, and the method of deriving the types of pfs that was presented in Section 3.9, have a bottom-up character: one can only introduce a variable of a certain type in a context Γ, if there is a pf that has that type in Γ. In $\lambda{\rightarrow}_C$, one can introduce variables of any type without wondering whether such a type is inhabited or not.

Church's $\lambda{\rightarrow}_C$ is more general than RTT in the sense that Church does not only describe (typable) *propositional* functions. In $\lambda{\rightarrow}_C$, also functions of type $\tau \rightarrow \iota$ (where ι is the type of individuals) can be described, and functions that take such functions as arguments, etc.

A characteristic of RTT that is maintained in many modern type systems is the syntactic nature of the system: type and order of a pf are determined on purely syntactical grounds. No attention is paid to the interpretation of such a pf. This is remarkable, as the propositions $\forall \mathtt{x}{:}\ 0^0\ [\mathtt{R}(\mathtt{x})]$ and $\forall \mathtt{x}{:}\ 0^0\ [\mathtt{R}(\mathtt{x})] \vee \forall \mathtt{z}{:}\ (\,)^9\ [\mathtt{z}(\,) \wedge \neg \mathtt{z}(\,)]$ are logically equivalent in most logics,[48] though they are of different type (the former pf has type $(\,)^1$ and the latter has type $(\,)^{10}$).

We saw in Section 4.1 that the Ramified Theory of Types is very restrictive for the description of mathematics within logic, because it is not possible to formulate impredicative definitions in RTT.

This was already realised by Russell and Whitehead, who tried to solve this by postulating the Axiom of Reducibility (Axiom 73). This axiom has been criticised from the moment it was written down, both by Russell and Whitehead themselves and by others. Ramsey, Hilbert and Ackermann were not satisfied by RTT's orders and therefore deramified RTT: They removed the orders. They observed that this does not lead to known paradoxes as long as a proper distinction between language and metalanguage is made.

[48] At least in all the logical systems that Russell had in mind when he wrote the *Principia*.

Gödel and Quine observed that the deramification does not violate the Vicious Circle Principle, as long as one accepts that objects and pfs exist independently of our constructions.

The main line in the history continues with non-ramified theories. For example, Church's combination of λ-calculus with simple type theory, the basis for most modern type systems, has no orders. Similarly to RTT however, $\lambda{\rightarrow}_C$ is very restrictive. The hierarchy of the simple theory of types used by $\lambda{\rightarrow}_C$ leads to a duplication of work. For example, numbers, booleans, the identity function have to be defined at every level. This led to the development of type theories that are *polymorphic* and hence avoid this unsatisfactory and inefficient duplication of work.[49] In [3], the reader may find a review of some of these type theories.

REFERENCES

[1] S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum (editors), ***Handbook of logic in computer science, Volume 2: Background: Computational structures***, Oxford University Press, 1992.

[2] Y. Bar-Hillel, A. Fraenkel, and A. Levy, ***Foundations of set theory***, North-Holland, 1973.

[3] H. P. Barendregt, *Lambda calculi with types*, in [1], pp. 117–309, Oxford University Press, 1992.

[4] ———, ***The lambda calculus: its syntax and semantics***, revised ed., Studies in Logic and the Foundations of Mathematics, vol. 103, North-Holland, Amsterdam, 1984.

[5] P. Benacerraf and H. Putnam (editors), ***Philosophy of mathematics***, second ed., Cambridge University Press, 1983.

[6] E. W. Beth, ***The foundations of mathematics***, Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1959.

[7] G. Boolos, *The iterative conception of set*, ***Philosophy***, vol. LXVIII (1971), pp. 215–231.

[8] C. Burali-Forti, *Una questione sui numeri transfiniti*, ***Rendiconti del Circolo Matematico di Palermo***, vol. 11 (1897), pp. 154–164, English translation in [68], pp. 104–112.

[9] G. Cantor, *Beiträge zur Begründung der transfiniten Mengenlehre (Erster Artikel)*, ***Mathematische Annalen***, vol. 46 (1895), pp. 481–512.

[10] ———, *Beiträge zur Begründung der transfiniten Mengenlehre (Zweiter Artikel)*, ***Mathematische Annalen***, vol. 49 (1897), pp. 207–246.

[11] A.-L. Cauchy, ***Cours d'Analyse de l'Ecole Royale Polytechnique***, Debure, Paris, 1821, also as *Œuvres Complètes* (2), vol. III, Gauthier-Villars, Paris, 1897.

[12] A. Church, *A set of postulates for the foundation of logic (1)*, ***Annals of Mathematics***, vol. 33 (1932), pp. 346–366.

[13] ———, *A set of postulates for the foundation of logic (2)*, ***Annals of Mathematics***, vol. 34 (1933), pp. 839–864.

[14] ———, *A formulation of the simple theory of types*, ***The Journal of Symbolic Logic***, vol. 5 (1940), pp. 56–68.

[49] Note that polymorphism was already recognized by Russell as *typical ambiguity* (cf. pages 161 and 162 of *Principia*). Moreover, Quine's NF and ML are polymorphic systems.

[15] ———, *Comparison of Russell's resolution of the semantic antinomies with that of Tarski*, ***The Journal of Symbolic Logic***, vol. 41 (1976), pp. 747–760.

[16] N. B. Cocchiarella, *Frege's double correlation thesis and Quine's set theories NF and ML*, ***Philosophical Logic***, vol. 13 (1984).

[17] ———, *Philosophical perspectives on formal theories of predication*, ***Handbook of Philosophical Logic***, vol. 4 (1986).

[18] H. B. Curry, *Functionality in combinatory logic*, ***Proceedings of the National Academy of Science of the USA***, vol. 20 (1934), pp. 584–590.

[19] ———, ***Foundations of mathematical logic***, McGraw-Hill Series in Higher Mathematics, McGraw-Hill Book Company, Inc., 1963.

[20] H. B. Curry and R. Feys, ***Combinatory logic I***, Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1958.

[21] R. Dedekind, ***Stetigkeit und irrationale Zahlen***, Vieweg & Sohn, Braunschweig, 1872.

[22] Euclid, ***The Elements***, 325 B.C., English translation in [36].

[23] S. Feferman, *Towards useful type-free theories I*, ***Symbolic Logic***, vol. 49 (1984), pp. 75–111.

[24] G. Frege, *Letter to Russell*, English translation in [68], pp. 127–128, 1902.

[25] ———, ***Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens***, Nebert, Halle, 1879, also in [68], pp. 1–82.

[26] ———, ***Grundlagen der Arithmetik, eine logisch-mathematische Untersuchung über den Begriff der Zahl***, Breslau, 1884.

[27] ———, ***Funktion und Begriff, Vortrag gehalten in der Sitzung vom 9. Januar der Jenaischen Gesellschaft für Medicin und Naturwissenschaft***, Hermann Pohle, Jena, 1891, English translation in [50], pp. 137–156.

[28] ———, ***Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet***, vol. I, Pohle, Jena, 1892, reprinted 1962 (Olms, Hildesheim).

[29] ———, *Über Sinn und Bedeutung*, ***Zeitschrift für Philosophie und philosophische Kritikf* (new series)**, vol. 100 (1892), pp. 25–50, English translation in [50], pp. 157–177.

[30] ———, *Ueber die Begriffschrift des Herrn Peano und meine eigene*, ***Berichte über die Verhandlungen der Königlich Sächsischen Gesellschaft der Wissenschaften zu Leipzig, Mathematisch-physikalische Klasse 48***, 1896, English translation in [50], pp. 234–248, pp. 361–378.

[31] ———, ***Grundgesetze der Arithmetik, begriffsschriftlich abgeleitet***, vol. II, Pohle, Jena, 1903, reprinted 1962 (Olms, Hildesheim).

[32] C. I. Gerhardt (editor), ***Die Philosophischen Schriften von Gottfried Wilhelm Leibniz***, Berlin, 1890.

[33] K. Gödel, *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I*, ***Monatshefte für Mathematik und Physik***, vol. 38 (1931), pp. 173–198, in German; English translation in [68], pp. 592–618.

[34] ———, *Russell's mathematical logic*, ***The philosophy of Bertrand Russell*** (P. A. Schlipp, editor), Northwestern University, Evanston & Chicago, 1944, also in [5], pp. 447–469.

[35] I. Grattan-Guinness, ***The search for mathematical roots, 1870–1930***, Princeton University Press, 2001.

[36] T. L. Heath, ***The thirteen books of Euclid's Elements***, Dover Publications, Inc., New York, 1956.

[37] D. Hilbert and W. Ackermann, ***Grundzüge der Theoretischen Logik***, first ed., Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band XXVII, Springer-Verlag, Berlin, 1928.

[38] R. Holmes, *Systems of combinatory logic related to predicative and "mildly impredicative" fragments of quine's "new foundations"*, ***Annals of Pure and Applied Logic***, vol. 59 (1993), pp. 45–53.

[39] ———, *Subsystems of Quine's "New Foundations" with predicativity restrictions*, ***Notre Dame Journal of Formal Logic***, vol. 40 (1999), no. 2, pp. 183–196.

[40] P. B. JACKSON, *Enhancing the nuprl proof development system and applying it to computational abstract algebra*, ***Ph.D. thesis***, Cornell University, Ithaca, New York, 1995.

[41] R. B. JENSEN, *On the consistency of a slight modification of Quine's NF*, ***Synthese***, vol. 19 (1969), pp. 250–263.

[42] F. KAMAREDDINE and T. LAAN, *A reflection on Russell's ramified types and Kripke's hierarchy of truths*, ***Journal of the Interest Group in Pure and Applied Logic***, vol. 4 (1996), no. 2, pp. 195–213.

[43] ———, *A correspondence between Martin-Löf type theory, the ramified theory of types and pure type systems*, ***Logic, Language and Information***, vol. 10 (2001), no. 3, pp. 375–402.

[44] G. T. KNEEBONE, ***Mathematical logic and the foundations of mathematics***, D. Van Nostrand Comp., London, New York, Toronto, 1963.

[45] S. KRIPKE, *Outline of a theory of truth*, ***Journal of Philosophy***, vol. 72 (1975), pp. 690–716.

[46] T. LAAN, *A formalization of the Ramified Type Theory*, ***Technical Report 94-33***, TUE Computing Science Reports, Eindhoven University of Technology, 1994.

[47] ———, *The evolution of type theory in logic and mathematics*, ***Ph.D. thesis***, Eindhoven University of Technology, 1997.

[48] T. LAAN and R. P. NEDERPELT, *A modern elaboration of the Ramified Theory of Types*, ***Studia Logica***, vol. 57 (1996), no. 2/3, pp. 243–278.

[49] G. LANDINI, ***Russell's hidden substitutional theory***, Oxford University Press, 1998.

[50] B. McGuinness (editor), ***Gottlob Frege: Collected papers on mathematics, logic, and philosophy***, Basil Blackwell, Oxford, 1984.

[51] G. PEANO, ***Arithmetices principia, nova methodo exposita***, Bocca, Turin, 1889, English translation in [68], pp. 83–97.

[52] ———, ***Formulaire de Mathématique***, Bocca, Turin, 1894–1908, 5 successive versions; the final edition issued as *Formulario Mathematico*.

[53] W. PEREMANS, *Ups and downs of type theory*, ***Technical Report 94-14***, TUE Computing Science Notes, Eindhoven University of Technology, 1994.

[54] H. POINCARÉ, *Du rôle de l'intuition et de la logique en mathematiques*, ***C. R. du IIme Cong. Intern. des Math., Paris 1900***, (1902), pp. 200–202.

[55] W. VAN ORMAN QUINE, *New foundations for mathematical logic*, ***American Mathematical Monthly***, vol. 44 (1937), pp. 70–80, also in [57], pp. 80–101.

[56] ———, ***Mathematical logic***, Norton, New York, 1940, revised edition Harvard University Press, Cambridge, 1951.

[57] ———, ***From a logical point of view: 9 logico-philosophical essays***, second ed., Harvard University Press, Cambridge, Massachusetts, 1961.

[58] ———, ***Set theory and its logic***, Harvard University Press, Cambridge, Massachusetts, 1963.

[59] F. P. RAMSEY, *The foundations of mathematics*, ***Proceedings of the London Mathematical Society* (second series)**, vol. 25 (1926), pp. 338–384.

[60] J. B. ROSSER, *Highlights of the history of the lambda-calculus*, ***Annals of the History of Computing***, vol. 6 (1984), no. 4, pp. 337–349.

[61] B. RUSSELL, *Letter to Frege*, English translation in [68], pp. 124–125, 1902.

[62] ———, ***The principles of mathematics***, Allen & Unwin, London, 1903.

[63] ———, *Mathematical logic as based on the theory of types*, ***American Journal of Mathematics***, vol. 30 (1908), pp. 222–262, also in [68], pp. 150–182.

[64] M. SCHÖNFINKEL, *Über die Bausteine der mathematischen Logik*, ***Mathematische Annalen***, vol. 92 (1924), pp. 305–316, also in [68], pp. 355–366.

[65] K. SCHÜTTE, ***Beweistheorie***, Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen, Band 103, Springer-Verlag, Berlin, 1960.

[66] J. P. SELDIN, *Personal communication*, 1996.

[67] E. P. SPECKER, *The axiom of choice in Quine's New Foundations for mathematical logic*, ***Proceedings of the National Academy of Sciences of the USA***, vol. 39 (1953), pp. 972–975.

[68] J. van Heijenoort (editor), ***From Frege to Gödel: A source book in mathematical logic, 1879–1931***, Harvard University Press, Cambridge, Massachusetts, 1967.

[69] A. C. M. VAN ROOIJ, ***Analyse voor Beginners***, Epsilon Uitgaven, Utrecht, 1986.

[70] H. WEYL, ***Das Kontinuum***, Veit, Leipzig, 1918, in German; also in: ***Das Kontinuum und andere Monographien***, Chelsea Pub. Comp., New York, 1960.

[71] A. N. WHITEHEAD and B. RUSSELL, ***Principia Mathematica***, vol. I, II, III, Cambridge University Press, 1910, 1912, 1913[1], 1925, 1925, 1927[2], all references are to the first volume unless otherwise stated.

[72] R. L. WILDER, ***The foundations of mathematics***, second ed., Robert E. Krieger Publishing Company, Inc., New York, 1965.

[73] E. ZERMELO, *Untersuchungen über die Grundlagen der Mengenlehre*, ***Mathematische Annalen***, vol. 65 (1908), pp. 261–281.

COMPUTING AND ELECTRICAL ENGINEERING
HERIOT-WATT UNIVERSITY
RICCARTON, EDINBURGH EH14 4AS, SCOTLAND
E-mail: fairouz@cee.hw.ac.uk

WEERDSTEDE 45
3431 LS NIEUWEGEIN, THE NETHERLANDS
E-mail: twan.laan@wxs.nl

MATHEMATICS AND COMPUTING SCIENCE
EINDHOVEN UNIVERSITY OF TECHNOLOGY
P. O. BOX 513
5600 MB EINDHOVEN, THE NETHERLANDS
E-mail: r.p.nederpelt@tue.nl