Frege's theory of types

Bruno Bentzen

Frege never explicitly advocated a doctrine of types like Russell, but a naive type theory can be found in his concept-script in his 1893 *Grundgesetze der Arithmetik*. Frege did not endorse a semantic account of typing, unlike most type theorists who use different types for booleans, natural numbers, products, etc. His concept-script is confined to a syntactic approach where object terms are identified with closed terms but the behavior of function terms alternates between that of closed terms of a function type and open terms. My aim in this paper is to rehabilitate Frege as a key figure in the history of type theory with an exegetical account of the developments anticipated in his concept-script from the perspective of modern type theory.

1 Introduction

It is often said that the study of type theory as we know it today only officially begins with Russell's efforts to resolve the contradictions arising from the naive notion of class especially from 1903 until the consolidation of his ramified theory of types in 1908. It is only passingly mentioned that Frege already had envisioned many of the essential ingredients for a type theory in the formal system presented in his 1893 *Grundgesetze der Arithmetik*.[1] Unlike Russell, never once did Frege explicitly advocate a doctrine of types. As pointed out by Vanrie (2020), he even forcefully rejected an early version of the theory of types presented to him by Russell in their correspondence. Indeed, in contrast to the spirit of modern type theory, Frege did not assign every term to a different type depending on what sort of entity is intended as its meaning. There is no type distinction between his terms for truth values, ordered pairs, natural numbers, and value-ranges. Nevertheless, there seems to be a relevant sense in which a simple type system is left implicit in the concept-script, as previously mentioned in Kamareddine et al. (2002) and Klev (2017). However, as far as I can see, Frege's naive type theory still has not received the comprehensive study it deserves, despite representing an important milestone in the history of type theory.

The aim of this paper is thus to rehabilitate Frege as a key figure in the development of type theory by providing an exegetical account of the type-theoretic views anticipated in his concept-script from a modern perspective. In Section 2 I introduce the basic concepts from type theory that will be utilized in my analysis and in Section 3 I explain how Frege's distinction between object and function terms gives rise to an implicit type system. In Section 4, I claim that Frege's conventional usage of Roman and Greek letters allows him to distinguish between open and closed terms in a way that resembles the type-theoretic distinction between general hypothetical and categorical judgments. In Section 5, I examine some consequences of his treatment of equality for functions and their value-ranges and his distinction between unary and binary functions. In Section 6, I investigate Frege's anticipation of the lambda calculus and his understanding of functions, which is based on a confusion between open terms and closed terms of the function type. Section 7 provides some concluding remarks.

2 Basic concepts from type theory

To begin with, we must first say a few things about the basic concepts of type theory we will be using in our analysis, namely, judgments, terms, types. In modern type theory, which derives from Martin-Löf (1984), it is common to write the judgment that states that a is a term of type σ as $a : \sigma$. We say that a judgment of this form is categorical when it is asserted without any hypotheses. As a general rule, however, a judgment of this form can also be said to be general and hypothetical when it depends on one or more typing assumptions. In this case it is asserted as

$$\Gamma \vdash a : \sigma,$$

where Γ is a finite list of typed variables $x_i : \sigma_i$ called the context of the general hypothetical judgment. When the context is empty the general hypothetical judgment just boils down to a categorical one. Moreover, we also say that a typed term is open when it is asserted by means of a general hypothetical judgment, regardless of whether or not the variables declared in the context occur in them. Otherwise, a typed term asserted by means of a categorical judgment is said to be closed, since it can only have bound variables. Finally, in the intended interpretation of open terms given by Martin-Löf (1982) in his meaning explanations, an open term is said to have type σ when it always results in closed terms of type σ after all the typed variables occurring in the context of their respective general hypothetical judgment are replaced with closed terms of the same type.

I will establish most of my analysis of the type system underlying the concept-script in the setting of the simplified fragment of type theory known as simple type theory. I will be particularly interested in the introduction, elimination, and computation rules proposed by Church (1940) for the characterization of the function type $\sigma \rightarrow \tau$, the type of functions that assign terms of type σ to values of type τ. It should be noted that its introduction rule regulates the construction of functions given by lambda terms by means of a type-preserving abstraction operation that acts on a determined free variable of an open term:

$$\frac{\Gamma, x : \sigma \vdash f : \tau}{\Gamma \vdash \lambda x.f : \sigma \rightarrow \tau.} \tag{1}$$

The elimination rule stipulates a function application operation where types are used to restrict the domain and codomain of the given function:

$$\frac{\Gamma \vdash f : \sigma \rightarrow \tau \quad \Gamma \vdash x : \sigma}{\Gamma \vdash \mathsf{app}(f, x) : \tau.} \tag{2}$$

The computation rule is the so-called β-equality. It tells us that when a lambda term is applied to a function argument what we obtain as the result is the replacement of a bound variable in the body of the function term with the given argument:

$$\frac{\Gamma, x : \sigma \vdash f : \tau \qquad \Gamma \vdash a : \sigma}{\Gamma \vdash \mathsf{app}(\lambda x.f, a) = f[a/x] : \tau.} \tag{3}$$

Through the lens of a correspondence famously discovered by Howard (1980) between propositions and types, which certainly owes very little to Frege's work, the terms of the function type can be regarded as proofs of implications.

Just as simple type theory can be viewed as a generalized form of propositional logic, there is, as it is to be expected, a more sophisticated version of type theory that represents a generalization of predicate calculus. This version of type theory was developed by Martin-Löf (1975) to serve as a formal basis for constructive mathematics and, although to a much lesser but still appreciable extent, some of Frege's insights are reflected in it. It is known as dependent type theory because it features dependent types, a constructive extension of Frege's conception of predicates as functions that assign objects to reified truth values. Dependent types allow for an elegant treatment of quantifiers and identity. I will not pursue those matters here, since, for the purposes of this paper, it will be enough to observe a unifying treatment of quantification and function abstraction in dependent type theory that sheds light on Frege's confusions between function terms and open terms in the concept-script.

3 Object and function types

It has long been observed that some of Frege's insights are accepted in the deramified theory of types proposed by Church (1940), a simple type theory that restricts the operations of the untyped lambda calculus developed in the 1930s with the adoption of a type system composed of a type of individuals, functions, and truth values. More specifically, we already find Quine (1940) mentioning that Frege's distinctive approach to function abstraction and predication is revived with Church's innovative technique of lambda abstraction and his treatment of predicates as functions to the type of booleans. Later, Quine (1955) even comes to recognize Frege's theory of function levels as an anticipation, to some degree, of a theory of types. The reason is because for Frege functions are categorized into first, second, and third levels, depending on whether their domain of arguments is strictly restricted to that of objects, first-level functions, or second-level functions, respectively. There is no need for higher functions in the concept-script because his treatment of value-ranges allows Frege to represent functions at ground level as objects.

Taking a cue from Quine (1955), let me emphasize that the sharp distinction between names of objects and names of functions drawn by Frege in *Grundgesetze* suggests that the concept-script always implicitly operates on typed terms. Every well-formed term in the theory is supposed to refer to either an object or a function, which, in turn, are two disjoint categories of things in Frege's ontology. For the purposes at hand, it suffices to recall that the title of function is reserved to things that have an unsaturated essence and that are always in need of completion, as already stressed in *Grundgesetze* §1. In contrast, objects are assumed to be saturated and complete entities. Given that numbers, truth values, and value-ranges do not exhibit saturation and are not in need of completion, they cannot be viewed as functions; thus, according to this dichotomy, the only alternative left is to regard them as objects:

> *Objects* stand opposed to functions. Accordingly, I count as an *object* everything that is not a function, e.g., numbers, truth-values and the value-ranges introduced below. Thus, names of objects, the *proper names*, do not in themselves carry argument places; like the objects themselves, they are saturated. (*Grundgesetze* §2)

It is helpful to think of the syntactic differentiation between object and function terms as reflected in the defining criterion for whether an expression may count as a function term of the language, namely, the presence of argument-places in the body of the expression. Since the distinction between objects and functions is exclusive, it follows that only expressions that lack argument-places may count as object terms of the language of the concept-script.

For a term to be well-formed in the concept-script it must first consist only of signs introduced as primitive or by definition and its Gothic and small Greek letters must be used only in accordance to their purpose. More concretely, a well-formed term should be either a primitive term, a compound term formed out of a well-formed function term from the filling of its argument places with well-formed terms that are fitting for them, or an atomic term defined to be a well-formed term in the two ways described above.[2] Every well-formed term is supposed to refer to something. In total there are only eight primitive terms in the language, all of which are function terms. If we ignore for a moment the distinction between functions of one or two arguments and focus on their levels we can describe them as follows. The primitive first-level functions terms of the concept-script are the horizontal, negation, implication, and definite article operator, which I respectively write as

$$-\!\!\!-\!\!\!-\xi, \quad \neg\xi, \quad \xi \supset \zeta, \quad \xi = \zeta, \quad \backslash\xi,$$

where their object argument places are marked using lowercase Greek letters such as ξ or ζ. The primitive second-level function terms are the first-order universal quantifier and value-range operator, respectively written as

$$\forall(\mathfrak{a})\phi(\mathfrak{a}), \quad \grave{\epsilon}\phi(\epsilon),$$

where their first-level argument places are indicated with the lowercase Greek letter ϕ. Finally, the only primitive third-level function term considered is the second-order quantifier

$$\forall(\mathfrak{f})\mu_\beta\mathfrak{f}(\beta),$$

whose second-level argument-place is, confusingly enough, marked with the the lowercase Greek letter μ, where the letter β is used to bind the first-level argument-places of the second-level function term used as argument. Since the concept-script has no primitive object terms, it is only possible to form compound or defined object terms. This can be done by filling the argument places of second- and third-level function terms in the appropriate way.

We find not only such operational type constraints in the concept-script but also type annotations, which are implicitly present as metavariables ranging over types in the expository metalanguage of the book. In fact, implicit type conventions are extensively used by Frege to determine whether a variable is intended to indicate an object or a function using specific letter conventions, which are all different for Roman, Greek, and Gothic letters. Ultimately, the type inference is left to the reader and depends on the context, but here are some examples of his conventions:

- Roman object letter: a–e, i–l, n–z;
- Greek object letter: α–ε, Γ–P;
- Gothic object letter: $\mathfrak{a}$–$\mathfrak{e}$;
- first-level Roman function letter: f–h, F–H;
- first-level Greek function letter: Φ–Ψ;
- first-level Gothic function letter: $\mathfrak{f}$–$\mathfrak{h}$;
- second-level Roman function letter: M;
- second-level Greek function letter: μ, Ω.

The distinction between Roman, Greek, and Gothic letters is equally essential to Frege's naive type theory since, as I will describe in more detail in the following sections, each has its own purpose in the concept-script. I have already explained how different lowercase Greek letters are used to distinguish the type of the argument place of a function term.

There is hardly any doubt all those elements suggest that the concept-script can be at least partially understood in the setting of a simple type theory with a ground type of individuals ι and a function type $\sigma \to \tau$. The requirement that every closed term must have a unique type thus implies the principle in the concept-script that every well-formed term either refer to an object or a function. One might be tempted to propose a tentative interpretation of Frege's theory of function levels as a hierarchy of types:

- Types of level 0: ι;
- Types of level 1: $\iota \to \iota$;
- Types of level 2: $(\iota \to \iota) \to \iota$;
- Types of level 3: $((\iota \to \iota) \to \iota) \to \iota$.

In other words, object terms are closed terms of type ι, first-level function terms are closed terms of type $\iota \to \iota$ and so on for second- and third-function level terms. But I am convinced that this proposal ignores the presence of binary functions of first level in the concept-script, which, as I suggest later in Section 5.3, generally have a curried type $\iota \to (\iota \to \iota)$. Crucially, it also neglects the fact that function terms are not consistently regarded as closed terms by Frege and are sometimes treated as open terms instead.

Just how seriously Frege takes the type constraints of the concept-script can be most readily seen from his initial response to the discovery of Russell's paradox. In his first letter to Russell sent on 22 June 1902, he emphasizes that "a predicate is predicated of itself" does not represent an acceptable term in his system because a predicate is as a rule a function and, according to his theory of function levels, can never have itself as argument.[3] More generally, it is easy to see that for any function term $f : \sigma \to \tau$, regardless of what types σ and τ stands for, the function application $f(f)$ will always be ill-typed. Unfortunately, given that value-ranges are taken to be objects in the type system, self-application is able to enter the concept-script through the back door as $f(\grave{\epsilon}f(\epsilon)) : \tau$, for $f : \iota \rightarrow \tau$, compromising the adequacy of type constraints of the concept-script.

4 Categorical and general hypothetical judgments

It is remarkable that the two forms of categorical judgment that have become standard in modern logic both can be traced back to Frege's concept-script. First, we have the usual judgment form that states that a proposition is true, which he writes using a turnstile notation:

$$\vdash\!\!\!- A.$$

Second, we have the accompanying judgment form that states that a sentence expresses a proposition, which is implicitly present in the 1879 *Begriffsschrift* in the early version of the concept-script through the content stroke

$$-\!\!\!- A$$

which then characterized a proposition as the content of a turnstile judgment, that is, a "judgeable content". The notion of content was originally the main semantic unit of the concept-script, and it was not until after the distinction between sense and reference in 1892 that a proposition came to be explained as the sense or thought of a sentence and a truth value as its reference. When the theory of sense and reference is formally incorporated in *Grundgesetze*, the turnstile judgment gets to be explained as the assertion that a sentence refers to the true and the content stroke ends up being treated as a function that refers to the true for the true as argument and the false otherwise, the horizontal.

As a result, it is no longer possible to assert the fact that a sentence expresses a proposition, since the concept-script does not have a counterpart for the content stroke anymore. In practice, that is achieved through the manipulation of functions into truth values such as the horizontal, negation, identity, conditional, and universal quantifier, which, when fully saturated, are assumed to have a truth value as reference, therefore expressing a proposition. Since sentences are handled as terms that refer to objects, and a sentence that refers to exactly one truth value cannot be both true and false, a consistency proof for the concept-script can be given by showing that every well-formed term in the system has a unique reference. That is precisely what Frege sets out to do in *Grundgesetze* §§29-31, an important part of the book that I shall return to later in this section.

4.1 Unsaturation and general hypothetical judgments

The assertion of a turnstile judgment is always supported by one or more implicit typing annotations, because the fact that the proposition expressed by a particular sentence is true will depend on assumptions that a letter occurring in the sentence refers to an object or function. In order to ensure full transparency with respect to the assumptions needed in the assertion of a turnstile judgment such as:

$$\vdash\!\!\!\!- A \supset (B \supset A).$$

I shall write all typing assumptions as explicit hypotheses on the left-hand side of the turnstile in a modern notation, where a true indicates that a refers to the true. For the sake of accuracy, I shall also write o for the subtype of ι consisting of two objects, the true and false:

$$A : o, B : o \vdash A \supset (B \supset A)\ \mathsf{true}.$$

Although it would not be wrong to say that Frege makes extensive use of categorical judgments, it would seem that, when all the typing annotations of a claim are made explicit in the way I just suggested, fewer categorical judgments are left throughout the book. That several judgments used in the exposition of the concept-script are in effect general hypothetical ones can be seen through Frege's adoption of Roman letters, which, as I shall now argue, gives rise to the notion of general hypothetical judgment:

> I shall call *names* only those signs or combinations of signs that refer to something. Roman letters, and combinations of signs in which those occur, are thus not *names* as they merely *indicate*. A combination of signs which contains Roman letters, and which always results in a proper name when every Roman letter is replaced by a name, I will call a *Roman object-marker*. In addition, a combination of signs which contains Roman letters and which always results in a function-name when every Roman letter is replaced by a name, I will call a *Roman function-marker* or *Roman marker* of a function. (*Grundgesetze* §17)

According to Frege, Roman markers only serve as indicators of things and cannot be accepted as referential terms. They are so called because they can only result in a referential term when all Roman letters occurring in them are replaced with referential terms. This means that Roman markers behave just like open terms in type theory. In particular, Roman object-markers are open terms of type ι and Roman function-markers are open terms of type $\sigma \to \tau$. This can be confirmed by seeing that, although Roman markers are not treated as well-formed terms of the concept-script, we can demand that they yield a well-formed term when all their Roman letters are instantiated with well-formed terms in the appropriate, type-respecting way. The only essential difference between Roman markers and open terms is that in the intended interpretation of type theory open terms can be ascribed meaning while in the informal semantics of the concept-script Roman markers do not have meaning.

Strangely enough, although function terms seem to be unsaturated in the same way as Roman markers, it is argued in *Grundgesetze* §29 that it is possible to show that a function term is referential if it always results in a referential term when applied with a referential term of suitable type. This same strategy reappears in Martin-Löf's (1982) meaning explanations of open terms, although it is hard to know if it was directly inspired by Frege's approach.[4] Either way, this puts more weight to my suspicion that Frege occasionally mistreats function terms as open terms. There is no reason why the above criterion of referentiality is applicable to function terms but not to Roman markers. Both are unsaturated expressions in the same manner, it seems, if we ignore the syntactic fact that the argument places of a function term are marked with lowercase Greek letters while the holes of a Roman marker with Roman letters.

In practice, however, Roman letters are specifically intended to express generality, just as the variables in a general hypothetical judgment do. In *Grundgesetze* §17, Frege states that a Roman marker indicating a truth value can always be transformed in a quantified sentence, which is the only purpose of Roman markers in the concept-script. In the case of first-level unary functions, this inference can be illustrated in the following way at this point:

$$\frac{\Gamma, x : \iota \vdash f(x) : o}{\Gamma \vdash \forall(\mathfrak{a}) f(\mathfrak{a}) : o}. \tag{4}$$

I will return to this point later. Every function term is unsaturated in the concept-script, as Frege repeatedly tells us, but not all unsaturated expressions of the language are accepted as a function term, since Roman markers are not function terms. In particular, every theorem stated with a Roman letter in the concept-script is actually a theorem schema formed by open sentences with free occurrences of sentential variables. By distinguishing function terms and Roman markers, Frege seems to create a distinction between two syntactically identical classes of unsaturated expressions. I will come back to this discussion in Section 6 after examining Frege's conception of value-ranges and his view of abstracted functions as saturated entities.

4.2 Saturation and categorical judgments

Although uppercase Greek letters are not officially part of the language of the concept-script either, they are in sharp contrast with Roman letters in that they serve as a stand-in for actual referential terms of the language. Heck (1997) views them as auxiliary names that are provisionally added to the language subject to the condition that they must refer to some object in the domain. Instead of relying on an assignment of values to free variables, as it is common in modern logic since Tarski, Frege makes use of auxiliary names that are assumed only to refer to some object.

The supporting role of auxiliary names is most evident in §31 of *Grundgesetze*, an intriguing section where Frege attempts to show that every well-formed term in the concept-script is referential by arguing by induction on their structure that $f(\xi)$ is referential if $f(\Delta)$ always has a reference for every object term Δ and that a is referential if $\Phi(a)$ always has a reference for every function term $\Phi(\xi)$.[5] Since this is an inductive proof of referentiality, I believe the application of uppercase Greek letters leaves no doubt that

Frege thinks of them as arbitrary terms that are assumed to have a reference. But contrary to what Heck suggests, they can refer not only to objects, but to functions as well, since the uppercase Greek letter Φ also plays the role of an auxiliary term. In fact, considering that every closed term is supposed to be referential, it seems that the claim that Δ is an auxiliary name amounts to the hypothesis that it is a closed term. Therefore, in a turnstile judgment such as

$$\vdash\!\!\!-\!\!-\; \Delta \supset (\Gamma \supset \Delta)$$

consisting of uppercase Greek letters such as Δ and Γ, what we actually have is a categorical judgment where every auxiliary name is assumed to be closed, which is to say that, by hypothesis, they are assumed to be derivable under no typing assumptions. The presuppositions involved in the reasoning above can be expressed more accurately as an inference where the notion of a closed term is made explicit:

$$\frac{\vdash A : o \qquad \vdash B : o}{\vdash A \supset (B \supset A)\ \mathsf{true}.}$$

In words, the inference above states that we are able to show that the sentence $A \supset (B \supset A)$ is true provided that we can assert that A and B are closed sentences. When judgments are represented in this type-theoretic way, as I shall do, we no longer need to follow Frege's conventional practice of writing auxiliary names as uppercase Greek letters.

5 Value-ranges and functions

As the doctrine of value-ranges as objects is the main reason why the type restrictions of the concept-script turned out to be ineffective, it is worth a detailed examination. In the type system of the concept-script, this doctrine is expressed through the tacit requirement that given a function $f(\xi)$ its value-range term $\grave{\epsilon} f(\epsilon)$ refer to an object. In our notation, the idea can be more easily articulated as a rule of inference:

$$\frac{\Gamma \vdash f(\xi) : \iota \to \iota}{\Gamma \vdash \grave{\epsilon} f(\epsilon) : \iota.} \qquad (5)$$

More precisely, value-ranges are regarded as first-class objects obtained by function abstraction, which, in turn, is incorporated as a function of second level which has the value-range of a first-level unary function as value. Informally, it is common to think of the value-range $\grave{\epsilon} f(\epsilon)$ as the graph of the function $f(\xi)$ for an argument ξ.

5.1 Gothic letters and Basic law V

The notion of value-range is governed by Basic Law V, the infamous axiom that states that the value-range of the function f is the same as the value-range of the function g just in case the functions f and g always have the same values for the same arguments:

$$\vdash\!\!\!\!\longrightarrow \grave{\epsilon} f(\epsilon) = \grave{\alpha} g(\alpha) \leftrightarrow \forall(\mathfrak{a}) f(\mathfrak{a}) = g(\mathfrak{a}).$$

As I noted in the previous section, this is in fact an axiom schema since the function letters f and g are open terms. So, to begin with, we actually have a hypothetical judgment:

$$f, g : \iota \to \iota \vdash \grave{\epsilon} f(\epsilon) = \grave{\alpha} g(\alpha) \leftrightarrow \forall(\mathfrak{a}) f(\mathfrak{a}) = g(\mathfrak{a}) \ \mathsf{true}.$$

Not only Roman letters have implicit typing assumptions. Here we find another example with the Gothic letters designated to mark the variables bound by a quantifier. This convention is patently clear in *Grundgesetze* §20, for example, where the theorem given as

$$\vdash\!\!\!\!\text{——}\ \forall(\mathfrak{f})(\forall(\mathfrak{a})\mathfrak{f}(\mathfrak{a})) \supset \mathfrak{f}(x)$$

actually stands for the fully explicit theorem schema, which, for convenience, I shall write without Gothic letters as:

$$x : \iota \vdash \forall(f : \iota \to \iota)(\forall(a : \iota)f(a)) \supset f(x) \text{ true.}$$

That is because $x : \iota$ and $f(x) : \iota$ can only mean that $f : \iota \to \iota$. Now, if we apply the same strategy to Basic Law V as well, we are able to bring it to its definite form:

$$f, g : \iota \to \iota \vdash \grave{\epsilon}f(\epsilon) = \grave{\alpha}g(\alpha) \leftrightarrow \forall(x : \iota)f(x) = g(x) \text{ true.}$$

Notice, however, that without the additional condition (5), which makes self-application possible, there is in principle nothing inherently contradictory about Basic Law V per se. The problem is that the requirement (5) that the value-range of any function be an object is implicit in the premise that $\grave{\epsilon}f(\epsilon) = \grave{\alpha}g(\alpha)$ is a well-formed identity statement, considering that for Frege identity is a first-level relation. I will now turn to a more detailed analysis of this point.

5.2 Identity is a first-level relation

Frege is known for conceiving identity as an all-inclusive relation in the domain of all objects. But, given his strict distinction between objects and functions, it immediately follows that for him one should never be allowed to speak of two functions as being the same. Thus, when a mathematician expresses the view that two functions are identical they are incurring a type mismatch error. Actually, what they should have in mind is the idea of two functions being coextensional, as noted in Ruffino (2003), that is, the first-level requirement that f and g always have the same value for the same arguments. If we decide to take seriously Frege's suggestion that the two halves of Basic Law V express the same sense but in a different way, which appears in *Funktion und Begriff* as a passing remark (Frege, 1891, p.27), we can even express coextensionality more directly in first-order terms via function abstraction as an identity statement between the value-ranges of f and g.

Even when Frege is explicitly speaking of an identity criterion for functions he recognizes it as a relation of second level that must be distinguished from the usual identity relation for objects. In his posthumous comments on sense and reference, written between 1892 and 1895, he makes this point very clear, adding that such a second-level relation expresses the same sense as function coextensionality:

> For every argument the function $x^2 = 1$ has the same (truth-) value as the function $(x + 1)^2 = 2(x + 1)$ i.e. every object falling under the concept *less by 1 than a number whose square is equal to its double* falls under the concept *square root of 1* and conversely. If we expressed this thought in the way that we gave above, we should have
>
> $$\alpha^2 = 1 \overset{\alpha}{\asymp} (\alpha + 1)^2 = 2(\alpha + 1)$$
>
> What we would have here is that second level relation which corresponds to, but should not be confused with, equality (complete coincidence) between objects. If we write it $\forall(\mathfrak{a})(\mathfrak{a}^2 = 1) = (\mathfrak{a}+1)^2 = 2(\mathfrak{a}+1)$, we have expressed what is essentially the same thought, construed as an equation between values of

> functions that holds generally. (Frege, 1979, p.121)[6]

By linking all these suggestions together, that is, the idea that both sides of Basic Law V and both sides of the second-level relation of function correspondence have the same sense, it can be argued that $f(\alpha) \overset{\alpha}{\asymp} g(\alpha)$ expresses the same sense as the first-level identity statement between value-range terms $\grave{\epsilon}f(\epsilon) = \grave{\alpha}g(\alpha)$. Thus, it appears that Frege relies on his doctrine of value-ranges as objects to escape the restriction that one is not allowed to speak of identical functions.

Before we close our discussion on identity, it should be noted that in dependent type theory Frege's second-level relation of function correspondence takes the form of the principle of function extensionality, which says that two functions are equal just in case they are coextensional:

$$f, g : \iota \to \iota \vdash f =_{\iota\to\iota} g \leftrightarrow \forall(x : \iota) f(x) =_{\iota} g(x) \ \mathsf{true}.$$

In dependent type theory, an identity proposition is conceived as a sortal relation $a =_{\sigma} b$ that is limited to two terms a and b of a same type σ. It thus takes the shape of a ternary relation, as emphasized by Klev (2017). Before we can ask whether two terms are equal we must first ask whether they have the same type. Comparing incomparables is never allowed, for we cannot form a well-formed identity statement between two terms of a different type. Just as envisaged by Frege, identity of objects and identity of functions are regarded as different relations. The difference is that in dependent type theory no identity relation is taken to be more fundamental than the other.[7]

5.3 Currying and double value-ranges

On an equal footing with Frege's theory of function levels is his distinction between unary and binary functions. What is the type of a binary function? The matter is not so simple, because, although they act on a pair of arguments, binary functions can also be partially saturated with a single term in the concept-script, resulting in a unary function:

> So far only functions with a single argument have been talked about; but we can easily pass on to functions with two arguments. These stand in need of double completion insofar as a function with one argument is obtained after their completion by one argument has been effected. Only after yet another completion do we arrive at an object, and this object is then called the value of the function for the two arguments. (*Grundgesetze* §4)

Partial saturation creates ambiguity in the interpretation of function application and suggests that Frege actually thinks of binary functions as unary functions from objects to unary functions, an idea rediscovered by Schönfinkel (1924) and later further developed in Curry and Feys (1958). Thus, as often observed, Frege was perhaps the first to make use of the concept of currying, the technique of dispensing with functions of multiple arguments by allowing functions to have other functions as values. Partial saturation means that a first-level binary function has the type $\iota \to (\iota \to \iota)$. This is supposedly the type of implication and identity.

However, there is no particular order for the partial saturation of a binary function in the concept-script, so Frege uses different lowercase Greek letters to make each argument place explicit. Given a function term $g(\xi, \zeta)$

and a term a of suitable type, we always have two possible choices: first apply a to the ξ-argument place and obtain a well-formed unary function $g(a, \zeta)$ or to the ζ-argument place and obtain $g(\xi, a)$ instead. Thus, I am inclined to disagree with Potts (1979) that, while in the lambda calculus functions are allowed to be values, in the concept-script the output of a function is always restricted to the domain of objects. Frege's binary functions are essentially curried functions that have unary functions as values.

In the lambda calculus, the use of explicit argument-marks is unnecessary because function abstraction leaves no room for ambiguity in the determination of the argument order of a function. In particular, the two ways of partially saturating a binary function given above may be respectively represented as $\mathsf{app}(g, a)$ and $\lambda x.\mathsf{app}(\mathsf{app}(g, x), a)$. Recently, Simons (2019) called attention to the fact that the concept-script has a very similar feature for handling application of curried functions with the use of double value-ranges from *Grundgesetze* §36. Since, for instance, the term $g(a, \zeta)$ refers to a unary function, we can obtain its corresponding value-range via $\grave{\epsilon}g(a, \epsilon)$. Now, by removing a from this value-range term, a legitimate term formation method explained in §30, we can form a new unary function term $\grave{\epsilon}g(\zeta, \epsilon)$, whose value-range is given by $\grave{\alpha}\grave{\epsilon}g(\alpha, \epsilon)$, the double value-range of the binary function $g(\xi, \zeta)$.

In the concept-script only functions can take things as arguments, never objects. Because value-ranges are objects, it is not possible to apply them to other objects using the primitive function application operation. Instead, Frege has a special purpose application function, introduced in §34, which is not a built-in primitive operation on a par with the ordinary function application, which from $f(\xi) : \sigma \to \tau$ and $a : \sigma$ results in $f(a) : \tau$. Instead, it is a definable binary function of the concept-script, derived using the definite article function of §11 and Basic Law V. I shall write this application function as an open term

$$x, y : \iota \vdash x \cap y : o$$

in order to avoid the implicit convention of adopting lowercase Greek letters for argument places. It is stipulated that $x \cap y$ refers to $f(x)$ if y is a value-range $\grave{\epsilon} f(\epsilon)$ and to the false otherwise.[8] In fact, we have an explicit equality that holds for value-ranges, which under our interpretation of closed terms, can be represented as an equality rule:

$$\frac{\vdash f : \iota \to \iota \qquad \vdash a : \iota}{\vdash a \cap \grave{\epsilon} f(\epsilon) = f(a) \ \mathsf{true.}} \tag{6}$$

Now, because a double value-range is just a value-range with a doubly-iterated function abstraction, if we are given a binary function $g(\xi, \zeta)$ and suitable terms a and b, it can be observed the following equality holds:

$$\begin{aligned} b \cap a \cap \grave{\alpha}\grave{\epsilon} g(\alpha, \epsilon) &= b \cap \grave{\epsilon} g(a, \epsilon) \\ &= g(a, b). \end{aligned}$$

This technique eliminates the ambiguity in the order of application of a binary function, allowing Frege to explicitly distinguish the terms $\grave{\alpha}\grave{\epsilon} g(\alpha, \epsilon)$ and $\grave{\alpha}\grave{\epsilon} g(\epsilon, \alpha)$, which generally refer to distinct objects when g is not commutative. Without the use of double value-ranges, it is not possible to determine the order of application of a binary function.

There has been one instance, as noted in Simons (2019), where Frege articulates a notion of simultaneous application of double value-ranges with ordered pairs. The proposal is only put forward much later in *Grundgesetze*, in §144, together with the definition of a pairing function $x; y$ as the value-range of a double iteration $\grave{\epsilon}(x \cap y \cap \epsilon)$. Interestingly, what Frege proposes is not strictly speaking the application of a double value-range to an ordered pair but a reverse-order application of an ordered pair to a double value-range:

$$\begin{aligned}\grave{\alpha}\grave{\epsilon}g(\alpha,\epsilon)\cap a;b &= \grave{\alpha}\grave{\epsilon}g(\alpha,\epsilon)\cap\grave{\epsilon}(a\cap b\cap\epsilon)\\ &= b\cap a\cap\grave{\alpha}\grave{\epsilon}g(\alpha,\epsilon)\\ &= b\cap\grave{\epsilon}g(a,\epsilon)\\ &= g(a,b).\end{aligned}$$

In type theories with a product type $\sigma \times \tau$, whose terms are ordered pairs $(a, b) : \sigma \times \tau$ for $a : \sigma$ and $b : \tau$, currying is commonly expressed as a logical equivalence between the types $\sigma \times \tau \rightarrow \rho$ and $\sigma \rightarrow (\tau \rightarrow \rho)$. It states the existence of a function that transforms a binary function into its curried form and one function that takes a curried function and transforms it into a binary one. Frege's conception of currying could not possibly be better stated with his identification of simultaneous and iterated applications, except that since $x; y$ is defined to be a value-range, ordered pairs are not terms of a product type. They are objects like any others. Since the value of a function term has to be defined for all object terms, any function can take an ordered pair as argument and still be unary!

6 Church's revival of function abstraction

If the most prudent way out of the contradiction is to view value-ranges not as objects but as functions, such a stipulation would make the distinction between functions and value-ranges lose its purpose, given that value-ranges are themselves determined by functions:

$$\frac{\Gamma \vdash f(\xi) : \iota \rightarrow \iota}{\Gamma \vdash \grave{\epsilon}f(\epsilon) : \iota \rightarrow \iota.}$$

It was Church (1940) who first realized with his lambda calculus how to adequately dissolve the conflict between functions and value-ranges and how to capture Frege's intuition that we form value-ranges by abstraction. The idea simply involves a functional abstraction on open terms, which are then regarded as closed function terms via the introduction of an abstract binding operation that ranges over all their free occurrences of variables. That is, instead of abstracting functions to form saturated objects, we abstract unsaturated objects to form functions. But if we were to express this insight in a Fregean style, it would *not* be as

$$\frac{\Gamma, x : \iota \vdash f(x) : \iota}{\Gamma \vdash \grave{\epsilon} f(\epsilon) : \iota \to \iota}$$

because the assumption that $f(x)$ is an open term already presupposes the conception of function application and therefore that of function. Given that a general hypothetical judgment already determines what variables an open term may depend on, we can correct the bad rule above by considering the most general form of an open term:

$$\frac{\Gamma, x : \iota \vdash f : \iota}{\Gamma \vdash \grave{x} f : \iota \to \iota.}$$

This is just a particular case of the previously mentioned introduction rule for the function type (1) restricted to unary functions in the domain and codomain of objects. Instead of Frege's smooth-breathing, which, along with Russell's circumflex notation, is recognized by Church (1942) himself as one of the precursors of his lambda-notation, we could have adopted Church's $\lambda x.f$ to denote such functions.

With the identification of value-ranges with functions in the above sense, the concept-script no longer would need two distinct forms of function application, the primitive function application operation $f(x)$ and the derived value-range application function $x \cap f$. Instead, a single notion of application would be needed, a unifying, primitive operation $\mathsf{app}(f, x)$ that comes with the typing structure of function application $f(x)$ but somehow inherits the computation rule of $a \cap \grave{\epsilon} f(\epsilon)$ which in §34 is stipulated to denote the same as $f(a)$, as stated in (6). Such an operation is provided by the elimination rule for the function type (2) in simple type theory, whose computation rule is stated in the β-equality rule (3) in terms of substitution of free variables of open terms. Even another main rule of the lambda calculus originally proposed by Church in the 1930s, namely, α-equality, the stipulation that two lambda terms that use different variable names are still the same, was already informally envisioned by Frege (1891), who declares that a function given by '$x^2 - 4x$' can be written as '$y^2 - 4y$' without altering its sense.[9] In this sense, it seems to me one could go as far as to say that all the essential ingredients for the simply typed lambda calculus can be found in the concept-script, except for the crucial idea that value-range or lambda terms are function terms.

6.1 Closed terms and object terms

I believe the turning point in the success of Church's revival of function abstraction lies in his decision to keep closed terms and object terms apart, which seems to go directly against Frege's convictions. I suggested in Section 4 that the concept-script has no clear criterion for distinguishing between function terms and open terms or Roman markers. Every closed term ends up being treated as an object term regardless of its meaning just because it does not have any argument places or Roman letters. That is precisely what is implied by Frege in a letter to Russell of 13 November 1904, where he entertains the idea of employing a rough-breathing notation $\acute{\epsilon}(\epsilon^2 = 1)$ for the function term that he would otherwise write with explicit argument places as $\xi^2 = 1$:

> But this notation would lead to the same difficulties as my value-range notation and in addition to a new one. For a range of values is supposedly an object and its name a proper name; but '$\acute{\epsilon}(\epsilon^2 = 1)$' would supposedly be a function name which would require completion by a sign following it. '$\acute{\epsilon}(\epsilon^2 = 1)1$' would have the same meaning as '$1^2 = 1$', and accordingly, '$\acute{\epsilon}(\epsilon^2 = 1) \supset$' would have to have the same meaning as '$\supset^2 = 1$', which, however, would be meaningless. '$\acute{\epsilon}(\epsilon^2 = 1)$' would be defined only in connection with an argument sign following it, and it would nevertheless be used without one; it would be defined as a function sign and used as a proper name, which will not do. (Frege, 1980, p.161–162).

Frege considers the missing element for his anticipation of the lambda calculus in this passage, the idea that function abstraction results in closed terms of the function type. But he quickly abandons the proposal because he feels it would be incompatible with his resolve that the nature of function consists in its unsaturatedness and need of completion.

It is important to stress that Frege's objection here is not completely clear and the way he attempts to sustain his claim leaves much to be desired. The reason why the term $\acute{\epsilon}(\epsilon^2 = 1) \supset$ would lack a reference is not because of an unfortunate choice of notation, as Frege suggests, but because the expression $\supset$ is not referential. The only substantial complaint he makes seems to be that a function term should not incorporate application as juxtaposition because then the function term would not have argument-places occurring in the body of the expression and, therefore, be fully saturated and complete. In other words, according to Frege's syntactic distinction, it would be an object term.

For him, it is possible to use $\acute{\epsilon}(\epsilon^2 = 1)$ in isolation as an object term because it has no occurrences of argument places while $\xi^2 = 1$ is a proper function term because the expression itself is unsaturated and requires completion.

Put differently, in the quotation above, Frege takes for granted in the concept-script a misleading identification between closed and object terms, which perhaps explains his reasons for maintaining that truth value, value-range, and number terms all belong to the same type of individuals. Unlike in modern type theory, where the type of a term depends on what it is supposed to mean, in the concept-script typing is a purely syntactic property. If a well-formed term has no argument places or Roman letters it will be an object term, regardless of whether it is intended to refer to a truth value, value-range, or number. This is a view that suits well his logicist ambitions, because all objects of arithmetic are supposed to be logical objects, and, according to Frege, value-ranges of logical functions have a special status as the most fundamental kind of logical object.[10]

> I myself was long reluctant to recognize value-ranges and hence classes; but I saw no other possibility of placing arithmetic on a logical foundation. But the question is, how do we apprehend logical objects? And I have found no other answer to it than this, we apprehend them as extensions of concepts, or more generally, as value-ranges of functions. (Frege, 1980, pp. 140–141)

Frege's transsortal identification of truth values with value-ranges proposed in §10 of *Grundgesetze*, one of the most extensively studied sections of the book, makes it clear that in his view value-ranges are the most basic logical objects that populate the universe of arithmetic. If there were different types for truth values, value-ranges, and numbers, it would not be possible for Frege to stipulate that the true and the false are value ranges of certain non-coextensional functions as he ultimately does in §10. Moreover, it would be impossible to define, later in §42, the concept of number in terms of value-ranges. If value-ranges were accepted as functions Frege would have no choice but to abandon the thesis that there exists an infinity of logical objects. The domain of logical objects of the concept-script would be severely restricted to only the true and the false. Crucial steps for the vindication of his logicism would be lost.

The puzzling recognition of closed terms as object terms in Frege's writings can also be observed in his conviction that one should expect a violation of the law of excluded middle if closed terms were to refer to typed objects. Vanrie (2020) has recently emphasized that this is the main reason why Frege rejected Russell's initial proposal of a theory of types. For Frege, it must be determined for every object whether or not it falls under a concept. Formally speaking, in the concept-script, delimiting the scope of application of a well-formed predicate $f(\xi)$ to only certain terms while excluding others such as $\grave{\epsilon}f(\epsilon)$ would be to disrespect the law of the excluded middle, given that the forbidden sentence $f(\grave{\epsilon}f(\epsilon))$ would be neither true nor false. In a letter sent to Russell on 29 June 1902, Frege writes that

> It seems that you want to prohibit formulas like '$\phi(\grave{\epsilon}\phi(\epsilon))$' in order to avoid the contradiction. But if you admit a sign for the extension of a concept (a class) as a meaningful proper name and hence recognize a class as an object, then the class itself must either fall under the concept or not; *tertium non datur*. If you recognize the class of square roots of 2, then you cannot evade the question whether this

> class is a square root of 2. (Frege, 1980, p.135)

Frege's objection simply ignores the possibility that the sentence $f(\grave{\epsilon}f(\epsilon))$ does not represent a thought at all. As noted in Vanrie (2020), this is precisely what Russell explicitly suggests in the letter Frege is responding to. In modern type theory, we do not say that typing incurs a violation of the law of excluded middle in view of the fact that ill-typed statements are not taken to be propositions. Frege is not prepared to endorse this view because he fails to distinguish closed terms from object terms. $f(\grave{\epsilon}f(\epsilon))$ must express a proposition in the concept-script for any first-level function $f(\xi)$ because $\grave{\epsilon}f(\epsilon)$ is a closed term and therefore must be assigned to the all-inclusive type of individuals. Frege's objection against the use of types is thus pointless because either $f(\grave{\epsilon}f(\epsilon))$ or its negation would be ill-formed if we were not to consider value-range terms as object terms.

6.2 Open terms and function terms

Frege's resolve to view every well-formed term with no argument places or Roman letters (closed term) as an object term is tied to his tendency to mistreat function terms as open terms instead of closed terms of the function type. It is clear that a function term like $\xi^2 = 1$ cannot be closed, since it will always have an argument place. From this perspective, the primitive first-level function terms of the concept-script can be seen to behave like primitive open terms:

$$x : \iota \vdash \text{——} x : o, \qquad x : \iota \vdash \neg x : o,$$

$$x, y : \iota \vdash x \supset y : o, \qquad x, y : \iota \vdash x = y : o,$$

$$x : \iota \vdash \backslash x : \iota.$$

Under this alternative view of first-level function terms as open terms, the concept-script becomes a unityped formal system, whose only type is the type of individuals ι. In other words, just as Frege seems to have in mind in the passages mentioned in the previous subsection, every closed term is an object term and first-level functions are open terms whose argument places are designated by free typed variables.

If we insist in following the above suggestion that first-level functions are open terms, Frege's actual conception of function abstraction becomes readable in a way that is closer to Church's own formulation. The main difference is that value-range terms are taken to refer to objects and that the resulting rule of inference cannot be generalized to hold for an arbitrary context Γ, not if we think of open terms under a context of n-typed variables as n-ary first-level Fregean functions because for Frege function abstraction is restricted to first-level unary functions.[11] So instead we have:

$$\frac{x : \iota \vdash f : \iota}{\vdash \grave{x} f : \iota.} \tag{7}$$

However, this reading of Frege's function terms as open terms is not fully consistent with all the aspects of the concept-script I have discussed so far. In particular, Frege's account of functions of second and third level is not amenable to this interpretation, since open terms, which are represented as general hypothetical judgments, cannot be part of other general hypothetical judgments in the same way that functions of higher level may take functions of lower level as arguments. Well, there is a quite simple way to circumvent this problem for second-level functions, since I have just explained the value-range operator as an inference rule that takes an open term to a closed term. The same method can be used to explain first-order quantification:

$$\frac{x : \iota \vdash f : o}{\vdash \forall(x : \iota) f : o}. \tag{8}$$

Thus, under this prospective extended interpretation for second level functions, the first-order quantifier function becomes quite close to the inference rule described in §17 as a way to obtain quantifiers from Roman markers, except that here we see a restriction to only one variable. The treatment of first-level functions as open terms thus overloads our idea that Roman markers are open terms. It introduces some ambiguity in our reading of rules like (8) in terms of the concept-script, since according to our interpretations proposed so far, the premise could either be a function term or a Roman marker. But I prefer to see this ambiguity as evidence that Frege's distinction between Roman markers and function terms is ultimately superfluous when seen through the lenses of modern type theory.

One fundamental difficulty with this extended interpretation is that it has an air of arbitrariness, given that first-level functions are interpreted as open terms but the interpretation of second-level functions is stated in terms of rules of inference from open terms. It is unlikely that this strategy can be generalized to third-level functions. But although the analogy of functions as open terms is quite limited in scope, it sheds light on an important relation between function abstraction and quantification in dependent type theory. From an open term that indicates a sentence we can always either obtain a value-range from (7) or a quantified sentence from (8), but never both simultaneously in a single rule. In dependent type theory, this conflict is nicely resolved by keeping function abstraction and quantification apart in one rule:

$$\frac{\Gamma, x : \sigma \vdash f : P}{\Gamma \vdash \lambda x.f : \forall(x : \sigma)P}. \tag{9}$$

This is possible because under the propositions-as-types correspondence quantification $\forall(x : \sigma)P$ can be seen as a generalized function type $(x : \sigma) \to P$ where the image of $a : \sigma$ under any function of this type has type $P[x/a]$. P is thus an indexed family of types that associates every term of a base type to another type. For Frege a predicate $P(\xi)$ over the domain of individuals has type $\iota \to o$, or more precisely, we have $x : \iota \vdash P(x) : o$. In dependent type theory, we say instead that P has type $\iota \to \mathcal{U}$, where $\mathcal{U}$ is a universe type, a type whose terms are themselves types. Clearly, Frege would not be prepared to accept a sophisticated rule such as (9) were it proposed to him. Regardless, looked from this angle, it seems fair to say that the germ of the idea that abstract function terms and quantified sentences can be obtained in parallel from open terms is found in the concept-script.

7 Concluding remarks

In the final analysis, it can be seen that Frege vacillates between a treatment of his function terms $f(\xi)$ as closed terms of a function type $\sigma \to \tau$ and as open terms of type τ depending on variables of type σ. His hierarchy of function levels only makes sense under the former interpretation where we have a function type. It allows us to represent, for example, unary third-level function terms as closed terms of type $((\iota \to \iota) \to \iota) \to \iota$. Another benefit of endorsing the former interpretation is that we can understand Frege's differentiation between Roman markers and function terms in terms of the modern type-theoretic distinction between open terms and closed terms of the function type.

But as we saw in the previous section, his remarks on the unsaturated nature of function terms, especially his conviction that a function term should have argument places occurring in the body of the expression, seems to commit him to the latter interpretation, that is, the view of function terms as open terms. Under this interpretation, the concept-script only contains a single type of objects, which then is assigned to every closed term of the language by virtue of the fact that no argument places or Roman letters are contained in the body of the expression. It follows that typing is for Frege a syntactic property of a term. It is of no importance that number, truth value, and value-range terms all denote different categories of entities; they share the property of being saturated so they are all object terms. The interpretation of function terms as open terms cannot be understood systematically for all function levels admitted in the concept-script, but it at least gives an indication of how Frege's conception of abstraction and quantification is not entirely in opposition to modern type theory.

In the end, Frege failed to develop his naive type theory further because he refused to see beyond the limitations of his syntactic style of typing and recognize the benefits of a semantic account of typing. Even after learning about the contradiction from Russell, Frege was reluctant to accept that a closed term like a value-range term should be assigned to the type of functions. I have covered some of the main reasons for his rejection of the proposal: it would reduce the type of individuals ι of the concept-script into the type of booleans o; Basic Law V would have to be replaced with the second-order function correspondence relation; and it would compromise his definition of numbers as value-ranges, since numbers are supposed to be logical objects. Still, Frege came remarkably close to the formulation of simple type theory as we know it, as close as he could considering his peculiar logical views and logicist ambitions.

Notes

[1] In this paper, I will focus exclusively on the first volume of *Grundgesetze*, where Frege establishes the definitive version of his formal theory, which I shall hereafter refer to as the concept-script.

[2] See especially *Grundgesetze* §§29–30. I am omitting some other principles of definitions laid down in §32 that are not pertinent here.

[3] See Frege (1980, p.131–133).

[4] For some of Martin-Löf's comments on *Grundgesetze* §29, see Martin-Löf (2001, p.13).

[5] For simplicity, I am omitting functions of two arguments and of second order here. I have dealt with Frege's criterion of referentiality in detail elsewhere (Bentzen, 2019).

[6] It is believed that Frege's explanation of this new notation was given in the lost first part of the manuscript (see Frege (1979, p.121, fn.1)). I took the liberty to modernize Frege's quantifier notation.

[7] Surprisingly, many forms of dependent type theory are unable to prove function extensionality, even though the principle is validated by its intended semantics, the meaning explanations (Bentzen, 2020b). Homotopy type theory (UFP, 2013) has been gaining acceptance as a foundational language for mathematics strong enough for proving not only function extensionality but also that isomorphic objects are equal. However, as the theory has to abandon the meaning explanations as its informal interpretation, its philosophical coherence is open to question (Ladyman and Presnell, 2016; Bentzen, 2020a).

[8] Actually, when y is not a value-range, $x \cap y$ refers to the value-range of a function whose value for every argument is the false, which, according to the transsortal stipulations of §10 is the false itself.

[9] I have stressed this point in Bentzen (2020b). For a more focused investigation of Frege's theory of sense and reference in the setting of type theory see Martin-Löf (2001) and Bentzen (2020c).

[10] At least until Frege's acknowledgment of Russell's paradox. See Ruffino (2001) and Bentzen (2019).

[11] For Frege, the arity of a function depends only on the argument places that actually occur in it. Therefore, weakening, the structural rule that allows us to derive $x : \sigma, y : \upsilon \vdash a : \tau$ from $x : \sigma \vdash a : \tau$, would not be allowed under this interpretation of first-level functions.

References

Bruno Bentzen. Frege on Referentiality and Julius Caesar in Grundgesetze Section 10. *Notre Dame*

Journal of Formal Logic, 60(4):617–637, 2019. URL https://projecteuclid.org/euclid.ndjfl/1569830414.

Bruno Bentzen. What types should not be. *Philosophia Mathematica*, 28(1):60–76, 2020a. URL https://doi.org/10.1093/philmat/nkz014.

Bruno Bentzen. On different ways of being equal. *Erkenntnis*, 2020b. URL https://doi.org/10.1007/s10670-020-00275-8.

Bruno Bentzen. Sense, reference, and computation, 2020c. URL http://philsci-archive.pitt.edu/17408/. Preprint.

Alonzo Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940. URL https://www.jstor.org/stable/2266170.

Alonzo Church. Abstraction. *The Dictionary of Philosophy, New York: Philosophical Library*, 1942.

Haskell B. Curry and Robert Feys. *Combinatory logic*, volume 1. North-Holland Amsterdam, 1958.

Philip A Ebert and Marcus Rossberg. *Gottlob Frege: Basic laws of arithmetic*. Oxford University Press (UK), 2013.

Gottlob Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. L. Nebert, Halle a.S., 1879.

Gottlob Frege. *Funktion und Begriff*. Hermann Pohle, Jena, 1891.

Gottlob Frege. *Grundgesetze der Arithmetik, Begriffsschriftlich Abgeleitet (I/II)*. Georg Olms, Hildesheim, 1962. Translated in Ebert and Rossberg (2013).

Gottlob Frege. Comments on sense and meaning. In Hans Hermes, Friedrich Kambartel, and Friedrich Kaulbach, editors, *Posthumous writings*, pages 118–125. Basil Blackwell, Oxford, 1979.

Gottlob Frege. *Philosophical and Mathematical Correspondence*. University of Chicago Press, Chicago, 1980. Translated by H. Kaal.

Richard K. Heck. Grundgesetze der Arithmetik I §§29–32. *Notre Dame Journal of Formal Logic*, 38(3):437–474, 1997. URL https://doi.org/10.1305/ndjfl/1039700749.

William A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, London, 1980.

Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. Types in logic and mathematics before 1940. *Bulletin of Symbolic Logic*, 8(2):185–245, 2002.

Ansten Klev. Identity and sortals (and caesar). *Erkenntnis*, 82(1):1–16, 2017.

James Ladyman and Stuart Presnell. Does Homotopy Type Theory Provide a Foundation for Mathematics. *The British Journal for the Philosophy of Science*, 69(2):377–420, 2016. URL https://doi.org/10.1093/bjps/axw006.

Per Martin-Löf. An intuitionistic theory of types: predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Logic Colloquium '73 : Proceedings of the logic colloquium, Bristol*, pages 73–118. North-Holland, Amsterdam, New York, Oxford, 7 1975.

Per Martin-Löf. Constructive mathematics and computer programming. In *Logic, methodology and philosophy of science, VI (Hannover, 1979)*, volume 104 of *Stud. Logic Found. Math.*, pages 153–175. North-Holland, Amsterdam, 1982.

Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory. Lecture Notes*. Bibliopolis, Naples, 1984. Notes by Giovanni Sambin.

Per Martin-Löf. The sense/reference distinction in constructive semantics. Paper read at Leiden (transc. by B. Jespersen), 2001.

Timothy C Potts. 'The Grossest Confusion Possible?'– Frege and the Lambda-calculus. *Revue Internationale de Philosophie*, pages 761–785, 1979. URL https://www.jstor.org/stable/23944070.

W. V. Quine. Alonzo Church. A formulation of the simple theory of types. The journal of symbolic logic, vol. 5 (1940), pp. 56–68. *Journal of Symbolic Logic*, 5(3):114–115, 1940. . URL https://doi.org/10.2307/2266866.

W. V. Quine. On Frege's way out. *Mind*, 64(254):145–159, 1955. URL https://www.jstor.org/stable/2251464.

Marco Ruffino. Logical Objects in Frege's Grundgesetze, Section 10. *From Frege to Wittgenstein: Perspectives on Early Analytic Philosophy*, page 125, 2001.

Marco Ruffino. Why Frege Would Not Be a Neo-Fregean. *Mind*, 112(445):51–78, 2003. URL http://www.jstor.org/stable/3093824.

Moses Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische annalen*, 92(3-4):305–316, 1924.

Peter Simons. Double value-ranges. *Essays on Frege's Basic Laws of Arithmetic*, page 167, 2019.

UFP (The Univalent Foundations Program). Homotopy type theory: Univalent foundations of mathematics, 2013.

Wim Vanrie. Why did Frege reject the theory of types? *British Journal for the History of Philosophy*, pages 1–20, 2020.

Bruno Bentzen
b.bentzen@hotmail.com