논리연구 20-1(2017) pp. 97-112

Frege’s influence on the modern practice of doing mathematics

Gyesik Lee

【Abstract】 We discuss Frege’s influence on the modern practice of doing mathematical proofs. We start with explaining Frege’s notion of variables. We also talk of the variable binding issue and show how successfully his idea on this point has been applied in the field of doing mathematics based on a computer software.

【Key Words】 Frege, Begriffsschrift, variable binding, mechanization, formal proofs

Received: Nov. 11, 2016. Revised: Feb. 10, 2017. Accepted: Feb. 11, 2017.

1. Introduction

Around the turn of the 20th century, mathematicians and logicians were interested in a more exact investigation into the foundation of mathematics and soon realized that ordinary mathematical arguments can be represented in formal axiomatic systems. One prominent figure in this research was Gottlob Frege. His main concern was twofold:

(1) whether arithmetical judgments could be proved in a purely logical manner, and
(2) how far one could progress in arithmetic merely by using the laws of logic.

In *Begriffsschrift* (Frege, 1879), he invented a special kind of language system, where statements can be proved as true based only upon some general logical laws and definitions. This system is then used later in the two volumes of *Grundgesetze der Arithmetik* (Frege, 1893, 1903), where he provided and analyzed a formal system for second order arithmetic.

Although the system in (Frege, 1893, 1903) is known to be inconsistent, it contains all of the essential materials to provide a fundamental basis for dealing with propositions of arithmetic based on an axiomatic system. Indeed, it addresses all *three types of concern that can attend a mathematical proof* as mentioned in (Avigad and Harrison, 2014):

(1) whether the methods it employs are appropriate to mathematics,
(2) whether the proof itself represents a correct use of these methods, and
(3) whether a proof delivers an appropriate understanding of the mathematics in question.

In particular, it is Frege who first showed that a rigorous and detailed proof could be given for each provable statement and that the validity of each logical inference can be checked when it is required. This is the main reason why people claim that Frege’s work initiated an era of applying rigorous scientific method to mathematics. Following his work, logicians and mathematicians began to consider mathematical systems as axiomatic ones, examples of which are Peano’s *The principles of arithmetic* (Peano, 1889), Hilbert’s *Grundlagen der Geometrie* (Hilbert, 1899), Whitehead and Russel’s *Principia Mathematica* (Whitehead and Russell, 1910, 1912, 1913), Zermelo’s axiomatic set theory (Zermelo, 1908), and Church’s type theory (Church, 1940). We refer to van Heijennoort's *From Frege to Gödel* (van Heijenoort, 1967) for more about Frege’s influence on the development of modern mathematics and logic.

This paper addresses an issue pointed out by Frege in *Begriffsschrift*. It is about the *use* and *role* of variables in proving meta-theories of first-order predicate logic. And we focus on Frege’s influence on the modern practice of applying formal proofs.

2. Computer-based formalization of mathematics

A *formal proof* is a proof which is written in an artificial language and in which every step of the proof can be checked according to some fixed logical rules and axioms. Note that this is exactly what Frege had in mind when he developed his system. Indeed, Frege said the following:

> The gaplessness of the chains of inferences contrives to bring to light each axiom, each presupposition, hypothesis, or whatever one may want to call that on which a proof rests; and thus we gain a basis for an assessment of the epistemological nature of the proven law. (Ebert and Rossberg, 2013, p. VII).

The only difference between his concept and the present-day practice is that computer software has developed sufficiently to assist humans in writing down and proving mathematical statements. There are various computer programs that can check and (partially) construct proofs written in their specific programming languages.

When mathematicians talk about a *formal proof*, it is generally meant that the mentioned proof followed *some level of a rigorous scientific method* acknowledged by a group of mathematicians. The group which decides the appropriateness of given proofs could be a group of mathematicians with some authority. Our interest lies not in identifying the group, but in the meaning of a rigorous scientific method in connection with the modern practice of mathematics using a computer.

To explain the modern practice of mathematics using a computer we look at the case of Hales' proof of the Kepler conjecture. When in 2003 Hales submitted his proof, the referees could not verify the correctness of the computer programs which are used to solve 1,039 complicated inequalities. In 2004, Hales himself announced his intention to have a formal version of his original proof. His aim was to remove any uncertainty about the validity of his proof by creating a formal proof that can be verified by some automated proof checking software, that is by some computer programs. His intention was then realized through a project called *Flyspeck* on 10th August 2014, 10 years after his announcement. In January 2015, Hales and 21 collaborators published a paper entitled with *A formal proof of the Kepler conjecture* (Hales et al., 2015). The paper describes a formal proof of the Kepler conjecture in a combination of two proof assistants, *HOL Light* (Harrison, 2009) and *Isabelle* (Nipkow et al., 2002).

Proof assistants are computer software specialized in doing mathematics on a computer. Using a proof assistant, one can set up a meta-language system where mathematical concepts like terms and formulas can be defined and properties like theorems can be proved. The way how it works is very similar to what a mathematician does in everyday mathematics. In addition, one can use many proof assistants including *HOL Light* and *Isabelle* as a programming language. Using this aspect, Hales could verify the correctness of his computer programs as well as that of his proofs.

In order to understand the mechanism of a proof assistant, it is necessary to understand how mathematicians set up a theory and how they define and prove mathematical properties in a theory. For more details we refer however to (Geuvers, 2009) which gives a detailed and kind explanation exactly on this point. In this paper we instead pay our attention to Frege's influence on the development of formalized mathematics.

3. Quantification theory and notions of variables

One of Frege's contributions to the formalization of mathematics is the invention of a full-fledged form of quantification theory. The importance of this aspect is well expressed by van Heijenoort in the introductory note on *Begriffsschrift* (van Heijenoort, 1967, p. 3):

> "When the slowness and the wavering of the propositional calculus are remembered, one cannot but marvel at seeing quantification theory suddenly coming full-grown into the world." Many years later (*1894*, p. 21) Peano still finds quantification theory "abstruse" and prefers to deal with it by means of just a few examples. Frege can proudly answer (*1896*, p. 376) that in 1879 he had already given all the laws of quantification theory; "these laws are few in number, and I do not know why they should be said to be abstruse".

This comments implies that nobody had known how to deal with quantification in a general form before Frege found out that several laws were enough for mathematicians to go one step further beyond the propositional logic. However, it is not our intention to explain in detail the system with quantification introduced by Frege. We instead focus only on his understanding and use of variables.

Logic with quantification is nowadays called predicate logic. And in predicate logic, two sorts of variable binding are involved. First, bound variables are used for representation of universal quantification, as in

$$\vdash \forall x\, P(x)$$

while free variables are used for representation of parametric derivations, as in

$$A(x) \vdash B(x).$$

It is very common to use the same set of variables for both of them. The main issue with this approach is that bound variables clashes sometimes with free variables, otherwise said, free variables can be captured suddenly by bound variables. Here is an example. Let $A := \forall x\, A_0(x)$, where $A_0(x) := \exists y\, (y = x + x)$, be a formula of the first-order Peano arithmetic (PA). Then A is provable in PA, hence the following holds:

$$PA \vdash A_0(t)$$

where t is an arbitrary term. But, this claim holds under the condition that y does not occurr free in t. Otherwise, we e.g. have for $t = y$ that

$$PA \vdash \exists y\,(y = y + y)$$

which would imply that PA is not sound.

This kind of variable capture can always occur when free and bound variables are not distinguished. For instance, the standard definition of unrestricted substitution for the lambda calculus in (Curry and Feys, 1958, p. 94) causes that variable capture can occur during substitution and that many proofs involving substitution are notoriously tedious because substitution is not defined by a structural induction.

A typical way of addressing this issue is to work with *alpha-conversion* in order to ensure that all free variables are always distinct from the bound variables. The *Barendregt Variable Convention* expresses exactly this point:

> If M_1, ..., M_n occur in a certain mathematical context (e.g. definition, proof), then in these terms all bound variables are chosen to be different from the free variables. (Barendregt, 1981)

Note however that this requires another, this time semantic, convention that *alpha-equivalent terms can be identified*. Using these two syntactic and semantic conventions one can have very slick *informal* arguments. Informal means here traditional pen-and-paper arguments.

4. Frege's influence on mechanization of mathematics

Dealing with alpha-conversion formally has turned out to be not so feasible. It just requires huge amount of extra work. So it has become a key issue in mechanical developments of formal meta-theory. It concerns the representation and manipulation of terms with variable binding.

There are two main approaches to address this issue: *first-order* and *higher-order* approaches. In first-order approaches variables are typically encoded using names or natural numbers, whereas higher-order approaches such as higher-order abstract syntax (HOAS) use the function space in the meta-language to encode binding of the object language. Higher-order approaches are appealing because issues like capture-avoidance and alpha-equivalence can be handled once and for all by the meta-logic. This is why such approaches are used in logical frameworks such as Abella (Gacek, 2008), Hybrid (Momigliano at al., 2008) or Twelf (Pfenning and Schürmann, 1999).

The main advantage of first-order approaches, and the reason why they are so popular in practice, is that terms with binders are easy to manipulate and understand; and they work well in general-purpose theorem provers like Coq (Bertot and Castéran, 2004). Here we mention two major first-order approaches whose idea goes back to Frege: the locally nameless style and the locally-named style. Both styles uses different sets of variables for bound and free variables in order to mainly avoid the variable capture phenomenon which inevitably occurs when only one sort of variables are used. And this idea of distinguishing two kinds of variables was proposed and used by Frege for the first time.

In order to syntactically deal with quantification, Frege distinguished between two kinds of *signs* when he explained the basic building blocks for constructing syntactic entities like propositions and proofs:

> I therefore divide all signs that I use into *those by which we may understand different* and *those that have a completely determinate meaning*. The former are *letters* and they will serve chiefly to express *generality*. But, no matter how indeterminate the meaning of a letter, we must insist that throughout a given context the letter *retain* the meaning once given to it. (Frege, 1879, p. 11)

Letters represent some objects like numbers or functions left indeterminate and are nowadays called *variables*. And *signs* that have a completely determinate meaning correspond to function symbols in modern terminology of logic.[1] He then distinguished further between two sorts of letters:

- Latin letters a, b, c, etc: to express universal validity of propositions, as in

$$(a + b)\,c = a\,c + b\,c.$$

- Old German letters a, b, c, etc: to state the generality of judgments, as in

$$\forall a \forall b \forall c[(a + b)\,c = a\,c + b\,c]$$

[1] Frege himself rejected to use the word "variable" since it was hardly possible for him to define it properly. See the footnote by Jourdain on page 10 in (Frege, 1879).

In the modern terminology, Latin letters are called *free variables* while German letters are called *bound variables*.[2] Using two kinds of variables is also applied later in (Gentzen, 1934) and (Prawitz, 1965).

Bound variables play the role of *delimiting the scope* that the generality indicated by the letters cover. Indeed, their role is to remember the places within their scope where "something else" might be substituted, resulting in a less general judgment. On the other hand, free variables syntactically play no essential role in Frege's work except when they are replaced by local variables in stating the generality of judgments. General substitution, for instance, is only performed when bound variables are instantiated, that is, when making statements less general.

Coquand (Coquand, 1991) recognized that Frege's idea of distinguishing between the two sorts of variables can be practically applied in machine-checked proofs. He suggested using Frege's idea in order to avoid the need to reason about alpha-conversion. Following him, McKinna and Pollack in (McKinna and Pollcak, 1993, 1999) extensively investigated the main characteristics of using two sorts of variables in proving the meta-theories of lambda calculus and Pure Type Systems. One of the results of their effort is that many important properties of typed lambda calculus can be stated and proved without referring to alpha-conversion, such as Church-Rosser, standardization, and subject reduction. Discussing how Frege's idea is implemented in

[2] This is the reason why people mention Frege as the first who used two disjoint sets of variables, see e.g. (Sato and Pollack, 2010, p. 599).

machine-checked proofs goes beyond the scope of this paper. We instead give here two simple examples demonstrating the effect of using two sorts of variables.

We first remind the reader of the fact that the formula $A_0(x) := \exists y\,(y = x + x)$ from Section 3 can cause a variable capture which would result in the inconsistency of a theory. This kind of problem never occurs when we separate the sets of free and bound variables. Using locally-named style and locally namless style we explain two typical ways of variable separation.

(1) Locally-named style

Let $a, b, c, \dots$ vary over free variables and $x, y, z, \dots$ over bound variables. Then the formula A_0 can be written as

$$A_0(a) := \exists y\,(y = a + a) \ ,$$

and we cannot substitute y for a, simply because y is not a well-formed term. Indeed, y is not allowed to occur unbound in a formula or a term. We refer the reader to (McKinna and Pollack, 1993, 1999) for more detail.

(2) Locally nameless style

Let $a, b, c, \dots$ vary over free variables. Then the formula A_0 can be written as

$$A_0(a) := \exists\ (0 = a + a) \ .$$

In the above formula, 0 stands for the position where a term could be substituted when an instantiation of the existential quantifier is performed. In this way one avoids any use of bound variables, and instead remember the positions where instantiations of quantifiers could happen. The numbers depends on the complexity of the positions. For example, the formula $A := \forall\, x\, A_0(x)$ can be represented in the following way:

$$A := \forall\ \exists\ (0 = 1 + 1)$$

One should remark that 0 and 1 are not numbers which can be manipulated by an operation. They play the role of place holder where instantiations of variables could happen. 0 is bound by $\exists$ and 1 is bound by $\forall$. That is, the position of 0 is one-level deeper than that of 1 with respect to the complexity of the positions. Further details can be found in (Aydemir et al., 2008).

5. Conclusion

We gave a short introduction to Frege’s influence on the modern practice of doing mathematical proofs. In particular, we focused on the variable binding issue and showed that his idea has turned out to be very useful in the field of doing mathematics based on a computer software.

References

Avigad, J. and Harrison, J. (2014), “Formally verified mathematics”, *Communications of the ACM*, 57(4), pp. 66-75.

Aydemir, B. E., Charguéraud, A., Pierce, B. C., Pollack, R., and Weirich, S. (2008), “Enginerring formal metatheory”, *ACM SIGPLAN Notices,* 43(1), pp. 3-15.

Barendregt, H. (1981), *The Lambda Calculus - Its Syntax and Semantics,* North-Holland.

Bertot , Y. and Castéran, P. (2004), *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions,* Springer.

Church, A. (1940), “A formulation of the simple theory of types”, *Jounal of Symbolic Logic*, 5(2), pp. 56-68.

Coquand, T. (1991), “An algorithm for testing conversion in Type Theory”, in Huet and Plotkin (eds.), *Logical Frameworks*, Cambridge University Press, pp. 255-279.

Curry, H.B. and Feys, R. (1958), *Combinatory Logic Volume 1*, North Holland.

Ebert, P. and Rossberg (2013), M. *Gottlog Frege: Basic Laws of Arithmetic,* Oxford University Press.

Frege, G. (1879), *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens,* Verlag von Nebert.

Frege, G. (1893, 1903), *Grundgesetze der Arithmetik Volume 1 and 2*, Verlag Hermann Pohle.

Gabbay, M. and Pitts, A. (2002), “A new approach to abstract syntax with variable binding”, *Formal aspects of computing*, 13(3-5), pp. 341-363.

Gacek, A. (2008), "The Abella interactive theorem prover (system description)", *Lecture Notes in Computer Science*, 5195, pp. 154-161.

Gentzen, G. (1934), "Untersuchungen über das logische Schließen. I", *Mathematische Zeitschrift*, 39(2), pp. 176-210.

Geuvers, H. (2009), "Proof assistants: History, ideas and future", *Sadhana Journal*, 34(1), pp. 3-25.

Gödel, K. (1958), "Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes", *Dialectica*, 12, pp. 280-287.

Hales, T. et al. (2015), "A formal proof of the Kepler conjecture", arXiv.org, https://arxiv.org/abs/1501.02155.

Harrison, J. (2009), "HOL Light: An overview", *Lecture Notes in Computer Science*, 5674, pp. 60-66.

Hilbert, D. (1899), *Grundlagen der Geometrie,* Teubner Verlag.

Huet, G. and Plotkin, G. (1991), *Logical Frameworks*, Cambridge University Press.

McKinna, J. and Pollack, R. (1993), "Pure type systems formalized", *Lecture Notes in Computer Science,* 664, pp. 69-111.

McKinna, J. and Pollack, R. (1999), "Some lambda calculus and type theory formalized", *Journal of Automated Reasoning,* 23(3-4), pp. 373-*409.*

Momigliano, A., Martin, A. J., and Felty, A. P. (2008), "Two-level hybred: A system for reasoning using higher-order abstract syntax", *Electronic Notes in Theoretical Computer Science,* 196, pp. 85-93.

Nipkow, T., Paulson, L.C., and Wenzel, M. (2002), *Isabelle/HOL:*

A proof assistant for higher-order logic, Springer.

Peano, J. (1899), *Arithmetices principia, nova methodo exposita,* Fratres Bocca.

Pfenning, F. and Schürmann, C. (1999), “System description: Twelf - A meta-logical framework for deductive systems”, *Lecture Notes in Computer Science*, 1632, pp. 202-206.

Prawitz, D. (1965), *Natural Deduction,* Almqvist & Wiksell.

Sato, M. and Pollack, R. (2010), “External and internal syntax of the lambda-calculus”, *Journal of symbolic computation*, 45(5), pp. 598-616.

Urban, C. (2008), “Nominal Techniques in Isabelle/HOL”, *Journal of Automated Reasoning,* 40(4), pp. 327-356.

van Heijennoort, J. (1967), *From Frege to Gödel,* Harvard University Press.

Whitehead, A. N. and Russell, B. (1910, 1912, 1913), *Principia mathematica Vol. 1 - 3,* Cambridge University Press.

Zermelo, E. (1908), “Untersuchungen über die Grundlagen der Mengenlehre. I”, *Mathematische Annalen*, 65, pp. 261-281.

한경대학교 컴퓨터공학과

Department of Computer Science and Engineering,

Hankyong National University

gslee@hknu.ac.kr

현대수학의 정형화에 대한 프레게의 영향

이 계 식

컴퓨터를 이용한 수학적 증명의 정형화는 현대수학의 중요한 연구도구로 활용되고 있다. 본 논문에서는 정형증명에 대한 프레게의 영향을 살펴본다. 이를 위해 자유변항과 구속변항을 정형증명에서 다룰 때 발생하는 문제를 설명한 후, 프레게의 Begriffsschrift에서 언급된 아이디어를 이용하여 변항을 정형적으로 다룰 수 있는 해결책을 소개한다.

주요어: 프레게, Begriffsschrift, 변항다루기, 정형화, 정형증명