Kazimierz Trzęsicki
University in Białystok, Chair of Logic, Informatics and Philosophy of Science
Stanislaw Staszic College of Public Administration in Bialystok

PHILOSOPHY OF MATHEMATICS AND COMPUTER SCIENCE

Abstract: It is well known fact that the foundation of modern computer science were laid by logicians. Logic is at the heart of computing. The development of contemporary logic and the problems of the foundations of mathematics were in close mutual interaction. We may ask why the concepts and theories developed out of philosophical motives before computers were even invented, prove so useful in the practice of computing. Three main programmes together with the constructivist approach are discussed and the impact on computer science is considered.

> Thought has been the father of every advance since time began. 'I didn't think' has cost the world millions of dollars.
>
> Thomas J. Watson, President of IBM

Introduction

The introduction of computers to our daily lives started in the 1950s. At the beginning of the XXI century the information society passed the threshold of unimaginable future. Advances of computer science (CS for short) are amongst the most decisive factors. Three paradigms of CS can be distinguished (Eden 2007):

1. CS is a branch of mathematics,
2. CS is an engineering discipline,
3. CS is a natural (empirical) science.

Re:1. Programs are conceived as a certain kind of mathematical objects and these objects should be investigated according to paradigm of mathematics, i.e. by means of deductive reasoning. *A priori* certain knowledge is the goal of CS. This **rationalist paradigm** is common among theoretical computer scientists.

ISBN 978–83–7431–273–8 ISSN 0860-150X

Re:2. Engineers deal with real objects. They apply theoretical knowledge to develop technology. In the case of CS, real are programs conceived as data. *A posteriori* knowledge about their reliability (practical knowledge) is achieved by empirical testing. This experiential knowledge is only probable. This **technocratic paradigm** is typical for software engineers.

Re:3. The most promising and the most desirable discipline of CS, the AI (**A**rtificial **I**ntelligence) is based on the method of natural sciences. In AI the **scientific paradigm** is prevalent. Programs are conceived as types of mental processes. *A priori* and *a posteriori* knowledge about them is achieved by testifying hypotheses. Induction as well as deduction is applied.

In the following CS will be understood according to the rationalist paradigm, i.e. in the 1st meaning.

CS is essential to almost every human activity, especially science, arts and humanities, even theology. Nevertheless, in the case of mathematics, the expectation that CS will revolutionize it, does not come true. Proof has been considered a fundamental part of the science of mathematical practice since ancient times. Its important status was stressed in Euclidean geometry. The great successes and spectacular achievements of CS are mainly in the field of efficiency, but not in the most important activity of mathematicians: proving new theorems. The situation is well commented by Paul Halmos' statement `http://scidiv.bellevuecollege.edu/math/Halmos.html`:

> Efficiency is meaningless. Understanding is what counts.

and

> I like words more than numbers, and I always did – conceptual more than computational.

However

> I have the world's most intelligent typewriter ... I spend at least six hours a day on my Macintosh.

For him:

> The computer is important, but not to mathematics.

There are different views of mathematics as a science, its subject and methods. The creation of the set theory by Cantor and the discovery of

paradoxes started the discussion of the paradigm of mathematics, i.e. the search for the foundations of mathematics. With the discovery of paradoxes it became evident that mathematics did not live up to the standards of certainty and rigor with which mathematics was over-credited.

The period that started in 1879 with Frege's *Begriffsschrift* and ended in 1931 with Gödel's *Über formal unentscheidbare Satze der Principia Mathematica und verwandter Systeme I* saw the development of three major foundational programmes:

- the logicism of Frege, Russell and Whitehead,
- the intuitionism of Brouwer, and
- Hilbert's formalist and proof-theoretic programme.[1]

Each programme addresses the issues that came to the fore at that time, either attempting to resolve them or claiming that mathematics is not entitled to its status of our most trusted knowledge. To this period, classical period,

- the programme of constructivists mathematics

as a direct legacy of Brouwer's intuitionism can be linked.

Each of the programmes aimed to give a satisfactory firm foundation for mathematics. For logicism:

- mathematics is a branch of logic.

Formalists argued that:

- mathematics is the science of formal systems.

Intuitionists maintained that

- mathematics is about mental constructions.

After Gödel's *Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I* (1931) it was clear that the efforts would not achieve the goal. Though the mathematicians on the whole threw up their hands in frustration and turned away from the philosophy of mathematics, it does not mean that the investigations were fruitless.

We ask about the impact of the programmes on the contemporary CS. We will show that ideas originating in the philosophy of mathematics participated in the rise of CS and have proved very helpful in its development. Philosophy is often thought of as a theoretical activity, which is of little or no practical importance. The influence of philosophy of mathematics on the development of CS demonstrates how this opinion is profoundly mistaken. Philosophical ideas and some kind of philosophical orientation are necessary for many quite practical activities, at least in CS. In the opinion of Gillies (2002):

[1] See (Lindstrom, Palmgren, Segerberg & Stoltenberg-Hansen 2007).

The Wall Street crash of 1929 ushered in the depression of the 1930's. One could say that the Gödel crash of (1931) initiated a period of depression in the philosophy of mathematics. The three main schools all appeared to have failed. Not one had carried out its promise of providing a satisfactory foundation for mathematics. Yet fate was preparing an odd turn of events. In the post-War period the ideas of these philosophical programmes turned out, surprisingly, to be of the greatest possible use in the new and rapidly expanding field of computer science.

1. Logicism and Computer Science

> In any case your discovery is very remarkable and will perhaps result in a great advance in logic, unwelcome as it may seem at first glance.
>
> Gottlob Frege
> From a letter to Bertrand Russell

1.1. The main ideas of logicism

Logicism tries to found the mathematics on basic logical principles. Its main tenet is that mathematical objects are in fact logical constructions. It was started by Frege (1879, 1884, 1893–1903). To accomplish his goal,[2] Frege defined number in terms of purely logical notions. The existing Aristotelian logic was not adequate for his purposes. So he devised a new kind of formal logic which he published in his *Begriffsschrift* (1879) (literally: concept writing). This is essentially the same as the formal logic taught today.[3] Frege then went on to set up a formal system, and tried to show that the whole of arithmetic could be logically deduced within this system using his definition of number. When Frege was nearing completion of the second volume of his 18 years long work, when he was about to see the success of the project, disaster struck. Bertrand Russell, a young logician, wrote him a letter dated June 16, 1902, in which he pointed out on the contradiction (Russell's paradox) derivable from Frege's basic axioms of logic. Frege's system appeared inconsistent. In his replay of June 22, 1902, Frege wrote (1967, pp. 127–8):

> Your discovery of the contradiction caused me the greatest surprise and, I would almost say, consternation, since it has shaken the basis on which I intended to build arithmetic. It seems, then, ... that my Rule V ... is false. ... I must reflect further on the matter. It is all the more serious since, with the

[2] His aim was not to show that the whole of mathematics was reducible to logic, but only that arithmetic was reducible to logic.

[3] Frege used a curious two dimensional notation, which has been abandoned in favor of the more usual one dimensional manner of writing.

> loss of my Rule V, not only the foundations of my arithmetic, but also the sole possible foundations of arithmetic seem to vanish.

Russell himself and Whitehead aimed to reformulated logicism to avoid inconsistencies. They tried to found mathematics on a basis of *ramified theory of types*[4]. This attempt was not entirely successful: they accomplished the creation of a system in which, in principle, most of the mathematics known then could be formalized, but at the cost of blurring the logical simplicity of Frege's primary ideas. When the three huge volumes of *Principia Mathematica* (1910–1913) were published, it looked as if the logicist programme had been brought to a successful conclusion. However, once again, this apparent success proved short-lived. In (1931) Kurt Gödel showed that, if *Principia Mathematica* were consistent, then there was a true arithmetical statement which could not be proved within the system. Thus reducing arithmetic to the logical system of *Principia Mathematica* or any other similar logicist system is impossible, i.e. any logical system is inherently incomplete.

1.2. How logicism has affected CS?

The logicist's approach to mathematics dwindled,[5] yet many of the ideas of logicism have been essential to CS. Though, in CS logic is no longer viewed as a foundation, but as a tool.[6]

The research of Frege and Russell was inspired by philosophical considerations, and they were both either not influenced at all, or to a negligible extent only, by considerations having anything to do with computing. Nevertheless, their work proved useful in CS later on.

Russell's theory of types was conceived as a basis of his logicist programme, in particular the set theory. Today mathematicians prefer to use ZF (and ZFC), the axiomatic set theory developed by Zermelo, Fraenkel and others.

> Indeed type theory is not taught at all in most mathematics departments. The situation is quite different in computer science departments where courses on type theory are a standard part of the syllabus. This is because the theory of types is now a standard tool of computer science. (Gillies 2002)

---

[4] A simpler version of logicism, that avoids the reduction principle and still allows the reconstruction of all classical mathematics, was later presented by Frank Plumpton Ramsey (1931). The simple theory of types was developed by Leon Chwistek (1921, 1948). For more see (Linsky 2009).

[5] The idea of logicism to build mathematics on logical base is continued in projects of computer-aided proving, e.g., the computer system `Mizar http://mizar.org/`.

[6] The relationships between CS and logic could be described in biological term as symbiotic ones.

Two ideas of logicist programme are of special interest to CS:

- universality of the language of logic, and
- theory of types.

Language of logic is a language of computer

First of all, let us state that any programming language is a formal language. After Gilles we may repeat (2002):

> In fact logic has provided the syntactic core for ordinary programming languages. At an even more fundamental level, the *Begriffsschrift* is the first example of a fully formalized language, and so, in a sense, the precursor of all programming languages.

The kind of language needed to communicate with computer was discussed by Turing (2004, p. 392):

> I expect that digital computing machines will eventually stimulate a considerable interest in symbolic logic and mathematical philosophy. The language in which one communicates with these machines . . . forms a sort of symbolic logic. The machine interprets whatever it is told in a quite definite manner without any sense of humour or sense of proportion. Unless in communicating with it one says exactly what one means, trouble is bound to result. Actually one could communicate with these machines in any language provided it was an exact language, i.e. in principle one should be able to communicate in any symbolic logic, provided that the machine were given instruction tables which would enable it to interpret that logical system. This should mean that there will be much more practical scope for logical systems than there has been in the past.

The difference between the ordinary and formal languages is explained by Frege in the *Begriffsschrift* (Berka & Kreiser 1971, p. 49):

> Das Verhältnis meiner Begriffsschrift zu der Sprache des Lebens glaube ich am deutlichsten machen zu können, wenn ich es mit dem des Mikroskops zum Auge vergleiche. Das Letztere hat durch den Umfang seiner Anwendbarkeit, durch die Beweglichkeit, mit der es sich den verschiedensten Umständen anzuschmiegen weiss, eine grosse Ueberlegenheit vor dem Mikroskop. Als optischer Apparat betrachtet, zeigt es freilich viele Unvollkommenheiten, die nur in Folge seiner innigen Verbindung mit dem geistigen Leben gewöhnlich unbeachtet bleiben. Sobald aber wissenschaftliche Zwecke grosse Anforderungen an die Schärfe der Unterscheidung stellen, zeigt sich das Auge als ungenügend. Das Mikroskop hingegen ist gerade solchen Zwecken auf das vollkommenste angepasst, aber eben dadurch für alle anderen unbrauchbar.
> I believe that I can best make the relation of my ideography to ordinary language clear if I compare it to that which the microscope has to the eye. Because

> of the range of its possible uses and the versatility with which it can adapt to the most diverse circumstances, the eye is far superior to the microscope. Considered as an optical instrument, to be sure, it exhibits many imperfections, which ordinarily remain unnoticed only on account of its intimate connection with our mental life. But, as soon as scientific goals demand greater sharpness of resolution, the eye proves to be insufficient. The microscope, on the other hand, is perfectly suited to precisely such goals, but that is just why it is useless for all others. (1879, p. 6)

Similarly the language of formal logic is suited to the scientific goal of communicating with computers, since this task demands great precision of expression. It is less suited, however, to the task of communicating with other human beings.

It is remarked by Gillies that (2002):

> the idea that different languages are suited to different purposes is already to be found in a reputed saying of the multi-lingual emperor Charles V. He is supposed to have said that he found French the most suitable language for talking to men, Italian for women, Spanish for God, and German for horses. If he had lived today, he could have added that the language of formal logic was the most suitable for talking to computers.

The builders' language-game (Wittgenstein 1953, 2), in which a builder and his assistant use exactly four terms (block, pillar, slab, beam), resemblance our communication with computers when we use the formal language of logic. The rules of language (grammar) are analogous to the rules of games; meaning something in a language is thus analogous to making a move in a game. Thus we arrive at the conclusion that computers do understand the meaning of the symbols they process.

Impact of type theory on CS

In CS, a type system defines how a programming language classifies values and expressions into types, how it can manipulate those types and how they interact. Some form of typing is incorporated into most programming languages. The most obvious application of type theory is in constructing type checking algorithms in the semantic analysis phase of compilers for programming languages. Thus three major areas of using of type theory in CS may be pointed out:

- typed programming languages,
- type-driven program analysis and optimization,
- type-aided security mechanisms

Type theory proponents have the following saying, which proclaims that the design of type systems is the very essence of programming language design:

> Design the type system correctly, and the language will design itself.[7]

Davis recapitulates the situation as follows (1988, p. 322):[8]

> Although the role of a hierarchy of types has remained important in the foundations of set theory, strong typing has not. It has turned out that one can function quite well with variables that range over sets of whatever type. So, Russell's ultimate contribution was to programming languages!

The term "type theory" is used to refer to the formal study of type systems for programming languages, although some limit it to the study of more abstract formalizms such as typed lambda-calculi.

Lambda-calculus

The lambda-calculus emerged in Church's famous paper (1936) showing the existence of an "undecidable problem". The Turing machine was invented later, though independently from the lambda-calculus.[9] Alonzo Church's lambda-calculus and Steven Kleene's recursive functions were arguably more elegant, but it was the mechanical action of Turing's machines that most agreed intuitively about how people calculate:

> These (Turing's) machines are *humans* who calculate. (Wittgenstein 1980, 1096)

It also makes the Turing machines a natural object for studying even more powerful models of computation. Church (1937, pp. 42–43) reviewed Turing's paper comparing the Turing machine to other concepts:[10]

---

[7] See eg. `www.javaworld.com/javaworld/jw-07-2007/jw-07-awscripting1.html?page=3`.

[8] Cf. (Gillies 2002).

[9] As regards the tricky question of priority, Church wrote:

> In an appendix, the author [i.e. AMT] sketches a proof of the equivalence of 'computability' in his sense and 'effective calculability; in the sense of the present author [i.e. Church's definition using the lambda-calculus.] The author's result concerning the existence of uncomputable sequences was also anticipated, in terms of effective calculability, in the cited paper [i.e. Church's paper]. His work was, however, done independently ....

[10] Church omitted Post's concept of binormality. For Post (1965, pp. 408, 419) himself it was only a working hypotheses:

> As a matter of fact, there is involved here the equivalence of three different notions: computability by a Turing machine, general recursiveness in the sense of Herbrand-Gödel-Kleene, and the λ-definability in the sense of Kleene and the present reviewer. Of these, the first has the advantage of making the identification with effectiveness in the ordinary (not explicitly defined) sense evident immediately – i.e., without the necessity of proving preliminary theorems. The second and third have the advantage of suitability for embodiment in a system of symbolic logic.

Lambda-calculus was applied to answer Hilbert's *Entscheidungsproblem*, but its idea originated in Russell's type theory. It was conceived by Church (1932, 1933) as part of a general theory of functions and logic, intended as a foundation for mathematics to develop the logicist position of Russell and Whitehead (1910–1913). Church (1940) introduced functions as primitive objects and the lambda-calculus notation was used in his elegant formulation of the simple type theory (Church 1940).

We may point to at least two ideas of CS that the lambda-calculus influenced: the design of the LISP programming language and functional programming languages in general.[11] There are two different families of systems of lambda-calculus: one elaborated by Curry (1934) and another by Church (1940). They correspond to two paradigms in programming (Barendregt 1992, p. 119). In the case originated in Curry's paper (1934) program is written without typing at all, i.e. is implicitly typed. ML (MetaLanguage) is such a language, (Milner 1984). Explicite typing corresponds to the version originated in Church's paper (1932, 1933). In this case, the program is written together with its type. ALGOL 68 and PASCAL are examples of such a language.

> [...] for full generality a complete analysis would have to be made of all the possible ways in which the human mind could set up finite processes for generating sequences. [...] we have to do with a certain activity of the human mind as situated in the universe. As activity, this logico-mathematical process has certain temporal properties; as situated in the universe it has certain spatial properties.

[11] Let us mention a curiosity concerning the lambda-calculus. Under the name *The Knights of the Lambda Calculus* is hidden a semi-mythical organization of wizardly LISP and Scheme hackers. LISP and Scheme are based on lambda-calculus, hence the phrase "lambda-calculus" is used in the name of the group. The word "knights" refers to the Knights Templar. It mostly only exists as a hacker culture in-joke. See `http://www.catb.org/~esr/jargon/html/K/Knights-of-the-Lambda-Calculus.html`.
We may mention the Church encoding, too. In mathematics, Church encoding is a means of embedding data and operators into the lambda-calculus. The method is named for Alonzo Church, who first encoded data in the lambda-calculus this way.

LISP

LISP (for **LIS**t **P**rocessor) is a family of programming languages with a distinctive, fully parenthesized syntax. LISP was invented by John McCarthy in 1958:

> The system was designed to facilitate experiments with a proposed system called the Advice Taker, whereby a machine could be instructed to handle declarative as well as imperative sentences and could exhibit "common sense" in carrying out its instructions. (McCarthy 1960, p. 184)

LISP is the second-oldest high-level programming language in widespread use today; only FORTRAN is older. Today, the most widely known general-purpose LISP dialects are COMMON LISP, Scheme, and their derivative DYLAN. LISP quickly became the favored programming language for AI research. LISP introduced many features now found in functional languages, though LISP is technically a multi-paradigm language. It pioneered many ideas in computer science, including tree data structures, automatic storage management, dynamic typing, and the self-hosting compiler.

Functional programming

It is proved that all recursive functions can be represented in the lambda-calculus (Kleene & Rosser 1935). Moreover, exactly the functions computable by a Turing machine can be represented in the lambda-calculus (Turing 1937). Computable functions as expressions in the lambda-calculus give rise to the so-called functional programming (Thompson 1991, Barendregt 1992, p. 118). Lambda-calculus forms the basis of almost all functional programming languages. These languages can be viewed as elaborations of the lambda-calculus. Functional programming is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids state and mutable data. It emphasizes the application of functions, in contrast to the imperative programming style, which emphasizes changes in state.

"Haskell", Curry's first name is used to dub a programming language that is rooted in his work (1934, 1958) and his intellectual descendants (de Bruijn 1968, Howard 1980) observations that:

> a proof is a program; the formula it proves is a type for the program.[12]

In Haskell, "a function is a first-class citizen" (Burstall 2000) of the pro-

[12] The Curry–Howard correspondence is the direct relationship between computer programs and proofs in constructive mathematics. Some researchers tend to use the term Curry–Howard–de Bruijn correspondence in place of Curry–Howard correspondence.

gramming language. Haskell is an advanced purely functional programming language, with non-strict semantics and strong static typing.[13]

Javascript, one of the most widely employed languages today, incorporates functional programming capabilities.

Unlambda

The combinatory logic was founded by Schönfikel's article *Über die Bausteine der mathematischen Logik* (1924). It is more abstract than lambda-calculus and preceded it in invention. Both, combinatory logic and lambda-calculus, are theoretically equivalent (Curry, Feys & Craig 1958). A lambda expressions can be easily transformed into combinator expressions, and combinator reduction is much simpler than lambda reduction. Combinatory logic has been used to model some non-strict functional programming languages and hardware. Combinatory logic is used in some esoteric languages including Unlambda.[14] A functional language based on the combinatory logic is a language without variables or lambda expressions. Unlambda is of some theoretical interest.

Lambda calculus and combinatory logic are now studied as idealized programming languages.

2. Formalizm and Computer Science

> Aus dem Paradies, das Cantor uns geschaffen hat, soll uns niemand vertreiben können.
> No one will drive us from the paradise which Cantor created for us.
>
> David Hilbert
> *Über das Unendliche*

2.1. The main postulates of formalizm

The formalist philosophy of mathematics known as Hilbert's Program was developed by David Hilbert in order to "dispose of the foundational questions in mathematics once and for all". The concept of formal system was taken from the logicists. But for Hilbert each branch of mathematics could be based on different formal system,[15] which should be:

---

[13] For more see `http://www.haskell.org/`.

[14] For the official Unlambda distribution, including several (other) implementations and documentation of the language, see the Unlambda Homepage by David Madore: `http://www.madore.org/~david/programs/unlambda/`.

[15] Hilbert also realized that axiomatic investigations required a well worked-out logical formalizm. He, with the assistance of Bernays and Behmann, made significant new contributions to formal logic.

1. consistent, i.e. free from contradictories;
2. complete, i.e. each theorem (true statement) has to be provable in it;
3. decidable, i.e. in case of any proposition the answer about its truth or about its falsity should be available with the help of "finitary" methods.[16]

If reading or writing a formalized text is concerned, it matters little whether this or that meaning is attached to the formulas. The only important point is the correct observance of the rules of syntax.[17] Hilbert told (in a railway station restaurant):

> Man muß jederzeit an Stelle von 'Punkte, Geraden, Ebenen' 'Tische, Stühle, Bierseidel' sagen können.
> It must be possible to replace the words 'point, line, plane' with 'table, chair, beer mug'.

Properties of formalized mathematics should be proven in the metalanguage via finitistic methods ('metamathematica').[18] The only part of mathematics to have meaningful content is finitary mathematics, that is, the analysis of concrete discrete objects and their decidable properties. The consistency coincides with satisfiability, that is, the existence of a mathematical universe in which all the derivable statements are true (Hilbert 1926, p. 370):

> So in recent times we come upon statements like this: even if we could introduce a notion safely (that is, without generating contradictions) and if this were demonstrated, we would still not have established that we are justified in introducing the notion. Is this not precisely the same objection as the one formerly made against complex numbers, when it was said that one could not, to be sure, obtain a contradiction by means of them, but their introduction was nevertheless not justified, for, after all, imaginary magnitudes do not exist? No, if justifying a procedure means anything more than proving its consistency, it can only mean determining whether the procedure is successful in fulfilling its purpose.

Kurt Gödel (1930) demonstrated that it is enough to prove the consistency of a theory, to be sure that it is meaningful (that is, it has a model). Since

---

[16] The problem of "decidability of every mathematical question" traces back to Hilbert's (1900) address.

[17] For Hilbert mathematics is the science of formal systems, consisting of a well-described syntax and a derivation criterium. But mathematics itself is no formal system; it only studies formal systems.

[18] In Weyl's opinion (1967, p. 483): Hilbert "... succeeded in saving classical mathematics by a radical reinterpretation of its meaning without reducing its inventory, namely by ... transforming it in principle from a system of intuitive results into a game with formulas that proceeds according to fixed rules."

finitary mathematics is the only solid ground to start from, the proof of consistency must be carried out by finitary means. Gödel (1931) discovered that a proof of consistency for a theory that – roughly speaking – contains arithmetic, always requires the use of a more powerful, and thus less reliable theory. Infinitary mathematics cannot be proved consistent by finitary means. Gödel's result showed that there can be no absolute consistency proof of all of mathematics; hence work in proof theory after Gödel concentrated on relative results. Starting with the work of Gerhard Gentzen in the 1930s, the Relativized Hilbert Programs[19] have been central to the development of proof theory.

2.2. Contribution of formalizm to CS

Turing machine

One of the important problems that was raised by Hilbert's Program was the question if there was a mechanical procedure for separating mathematical truths from mathematical falsehoods. It is the so-called *Entscheidungsproblem*.[20] David Hilbert's formulation of the *Entscheidungsproblem*, i.e. classical problem of decidability, produced a plethora of ideas that – in particular – gave rise to CS and is still abundant in opening new horizons to computer science, mathematics and philosophy.

In the 20th century, the idea of computation, the basic idea of CS, was developed in connection with the problem of decidability. The Church–Turing thesis is a statement that characterizes the nature of computation. The thesis (conjecture) claims that an intuitive notion of calculus is adequate to the notion of the Turing machine[21] (or equivalent notions such as: recursive functions, the lambda-calculus by A. Church, the canonical systems by E. Post, the normal algorithms by A. A. Markov, the Minsky machines, the Kolmogorov algorithms, etc.). The Church–Turing thesis cannot be formally proven. Despite this fact, it now has near-universal acceptance.

Alan Turing's work on the *Entscheidungsproblem* followed Kurt Gödel's work on the incompleteness theorems, and the notion of general purpose computers that came from this work, was of fundamental importance to the designers of the computer machinery in the 1940s. The Turing machine is

[19] For more, see (Murawski 1999, 4.4. Relativized Hilbert's Program vs. Reverse Mathematics).

[20] For more about *Entscheidungsproblem* and its implications see (Trzęsicki 2006).

[21] By Turing the machine was named a logical computing machine, or "a-machine" (automatic machine). It has subsequently become known as the "Turing Machine". For the first time this term was used by Church (1937).

a theoretical device that computes all the functions that are computable in any reasonable sense. Conversely, it is also believed that if a computation cannot be performed by the Turing machine, then it cannot be computed at all. In other words, the thesis states that all effective computational models are equivalent to, or weaker than, the Turing machine (Shoenfield 1991, p. 26). It has long been assumed that the Turing machine is able to do all that computers equipped with recursive algorithms do and all that can be done by the Turing machine may be executed by such computers (if they have enough time[22]). Thus the Turing-Church thesis states that any computation solvable by a precisely stated set of instruction (an algorithm) can be run on the Turing machine or a digital process computer.[23]

The idea of computability executed by the Turing machine is very fertile and has originated many ideas such as von Neumann's classical computer. In Turing's paper, there appeared very fruitful notions of "input-output", "memory", "kompiler/interpreter", "finite-state machine", "coded program", and "algorithm". Turing's definition of computability remains a classic paper in the elucidation of an abstract concept into a new paradigm.

In recent years, the number of people who maintain that the Turing machine cannot capture the entire spectrum of applications of computers has been growing. There are important constraints on the ability of the Turing machine. Generally speaking, there are problems related to tractability, commensurability and computability. Moreover, some of these limitations do not concern physical restrictions. There are well-stated problems that are not computable by the Turing machine. Thus the question arises if there are possible devices which can compute more than the Turing machine.

The theory of computation became an independent academic discipline and was separated from mathematics. It is one of the disciplines of CS. Computability theory is closely related to recursion theory. Recursion theory is not restricted to consider models of computation that are reducible to the Turing model.

For formalists mathematical reasoning as captured in a formal system means manipulating strings of symbols. Manipulating symbols is what computers do. Thus the formalists approach encourages projects of systematic encoding in computer-readable format of mathematical knowledge so as to facilitate automated proof checking of mathematical proofs and the use

[22] The notion of the "speed of computation" makes little sense in the classical understanding of mathematics. Its importance has been recognized with the advent of modern computers and their applications, especially such that need to be accomplished in real time.

[23] For more about the Turing-Church Thesis, its history and implication, see (2009).

of interactive theorem proving in the development of mathematical theories and computer software.[24] Moreover, if human knowledge could be expressed in a formal language, it would be possible – as some researchers predicted in the 1950s and 1960s – an artificial intelligence, a machine that "thinks"[25]. Today AI is a flourishing branch of CS.

Logic, mathematics, and CS are strongly related to each other. They are the only genuine formal sciences in the sense that they can be carried out purely formally. Programs are of course nothing more than rather large and very complicated formal objects.

The consequences of Hilbert's program, unexpected by himself, are summarized by Chaitin (2004):

> As I said, formal systems did not succeed for reasoning, but they succeeded wonderfully for computation. So Hilbert is the most incredible success in the world, but as technology, not as epistemology. [...] Hilbert's idea of going to the limit, of complete formalization, which was for epistemological reasons, this was a philosophical controversy about the foundations of mathematics – are there foundations? And in a way this project failed, as I've explained, because of the work of Gödel and Turing. But here we are with these complete formalizations which are computer programming languages, they're everywhere!

3. Intuitionism and Computer Science

> The question where mathematical exactness does exist, is answered differently by the two sides; the intuitionist says: in the human intellect, the formalist says: on paper.
>
> L. E. J. Brouwer (Brouwer 1913, p. 56)

3.1. Fundamentals of intuitionism

Intuitionism originates with the Dutch mathematician L. E. J. Brouwer.[26] The main ideas of his philosophy were formulated in his doctoral dissertation *Over de Grondslagen der Wiskunde* (1907). Brouwer's philosophy of mathematics is embedded in a general philosophy, the essentials of which are found already in (Brouwer 1905). Intuitionism as a philosophy of

[24] Because of their close connection with computer science, this idea is also advocated by some mathematical intuitionists and constructivists.

[25] "Think" is the motto of IBM.

[26] For more about history of intuitionism see (van Atten, Boldini, Bourdeau & Heinzmann 2008).

mathematics is based on the rejection of the Platonic notion of the existence of mathematical objects. For Brouwer (1975) mathematics is not formal; the objects of mathematics are mental constructions in the mind of the (ideal) mathematician. Only the thought constructions of the (idealized) mathematician are exact. Mathematics is a stand-alone activity concerning mental constructions according to self-evident rules, independent of experience in some outside reality, and mathematics is in principle also independent of language:

> The first act of intuitionism completely separates mathematics from mathematical language, in particular from the phenomena of language which are described by theoretical logic, and recognizes that intuitionist mathematics is an essentially languageless activity of the mind. (Brouwer 1952, pp. 141–2)

Numbers and functions are mental constructions and mathematical theorems express the capacity of the human mind to perform certain operations on these constructions. Brouwer accepted only one basic notion: time,[27] mathematically seen as the real line. For set theorists the real numbers are set theoretical construction, e.g. Dedekind cuts, but for Brouwer the reals are there without the need of any further qualification. Objects of mathematics have meaning only inside the human mind. Mathematics is the study of mental mathematical constructions realized by a creative subject.

Mathematics does not depend on logic; on the contrary, logic is part of mathematics. Mathematics precedes logic: the logic we use in mathematics grows from mathematical practice, and is not something *a priori* given before mathematical activity can be undertaken. For intuitionists the truth of a proposition must be constructive: it must rest on proof (a certain kind of mental construction). This led to the rejection of some principles of the classical logic. In (1908) Brouwer explicitly noted that intuitionism required a different logic. The principle of the *excluded middle*[28] and the principle of *double negation elimination* were rejected. A question has an answer when we find it; if we are unable to construct a proof or a confutation of a proposition, we cannot assume that:

---

[27] For some protagonists of the role of insight in mathematics intuition of space was the source of mathematical concepts, e.g. Poincaré rejected purely logical or linguistic descriptions as the only source for mathematics and stressed the role of geometric insight.

[28] Propositions restricted to a finite collections are still regarded as being either true or false, as they are in classical mathematics, but this bivalence does not extend to propositions which refer to infinite collections. L. E. J. Brouwer viewed the law of the excluded middle as abstracted from finite experience, and then applied to the infinite without justification. To him the law of the excluded middle was tantamount to assuming that every mathematical problem has a solution.

- it must be either that a proposition is true or that a proposition is false;
- the truth of doubled negation of a proposition is not the sufficient reason of the truth of that proposition.[29]

The principle of impredicativity[30] that allows the construction of a new object of a certain class by referring to the whole class as a completed whole is not accepted. It means that the principle of the classical first-order logic that general proposition is a sufficient reason of a particular (existential) proposition: $\forall x P(x) \rightarrow \exists x P(x)$ is rejected, too. The intuitionistic interpretation (of truth) of propositions gives rise to a non-classical logic.

In (1930), Brouwer's most famous pupil, Arend Heyting, published the first Hilbert style formalization of the logic used by intuitionists which so clearly characterized the logic that it has become universally known as the axioms for intuitionistic logic (1930, 1956). This formal system captured the informal intuitionistic comprehension of the connectives and quantifiers. The standard informal interpretation of logical operators in intuitionistic logic is the so-called proof-interpretation or the Brouwer–Heyting–Kolmogorov interpretation, (BHK for short).[31]

Intuitionism as the only of the three major programs was not affected by Gödel's incompleteness theorems. Nevertheless, the intuitionistic mathematics turned out to be more involved and intricate than standard mathematics, and, as a result, it was rejected by most mathematicians as just too complicated to be acceptable.

Traditionally, some mathematicians have been suspicious, if not antagonistic, towards intuitionism. David Hilbert forcefully expressed his attitude. In *Die Grundlagen der Mathematik* he wrote (1928)[32]:

> I am astonished that a mathematician should doubt that the principle of excluded middle is strictly valid as a mode of inference. I am even more astonished that, as it seems, a whole community of mathematicians who do the same has so constituted itself. I am most astonished by the fact that even in mathematical circles the power of suggestion of a single man, however full of

[29] The converse holds: the truth of a proposition is the sufficient reason of the truth of its double negation.

[30] See (1906, 1916, 1925) and Weyl (1918).

[31] The meaning of a connective is described by explaining what is to be regarded as a proof of a compound statement assuming one knows what counts as proof of each of its arguments. Such an interpretation is implicit in Brouwer's writings (e.g. (1908, 1924), and has been made explicit by Heyting (1934) for predicate logic, and by A. N. Kolmogorov (1932) for propositional logic.

[32] See `www.marxists.org/reference/subject/philosophy/works/ge/hilbert.htm`

> temperament and inventiveness, is capable of having the most improbable and eccentric effects.[33]

Many mathematicians probably shared Bourbaki's view (Bourbaki 1991, p. 38):

> The intuitionistic school, of which the memory is no doubt destined to remain only as an historical curiosity, would at least have been of service by having forced its adversaries, that is to say definitely the immense majority of mathematicians, to make their position precise and to take more clearly notice of the reasons (the ones of a logical kind, the others of a sentimental kind) for their confidence in mathematics.

Intuitionism, in Brouwer's form, doesn't lend itself to computer implementation, since Brouwer rejected the idea that mathematical reasoning is a formal activity. He even refused to use logical symbolism, preferring, whenever possible, to express himself in natural language. The human mind, he contended, is a *creative subject*, that can perform constructions that elude any formal process. He even contested that mathematics is based on logic. Thus, intuitionism is radically opposed to both formalizm and logicism. Nevertheless, intuitionism gave rise to critical review of

- the notion of an (existence) proof,
- the notion of a computable function.

According to Troelstra (1999)

> Ideas developed in the context of intuitionism turned out to have a relevance which transcends the original setting.

In the contemporary philosophy of mathematics with the posthumous publication of Wittgenstein's *Remarks on the Foundations of Mathematics* in 1956, Brouwer's ideas again sparked a lively interest among philosophers.

With the publication, in 1967, of Errett Bishop's *Foundations of Constructive Analysis* perception of Brouwer's legacy changed dramatically (Bridges & Reeves 1997). The intuitionist philosophy of mathematics gave rise to different streams of constructivism and due to that fact intuitionism has an impact on CS. Constructivism has many connections to CS and has found numerous applications in CS (Troelstra & van Dalen 1988, Beeson 1985).

---

[33] Brouwer struggled, perhaps too aggressively, to overcome sometimes harsh polemic against him and the antipathy of Hilbert and his followers to intuitionistic mathematics. See (van Stigt 1990) for more details of the history of that period. See also `http://en.wikipedia.org/wiki/Brouwer%E2%80%93Hilbert_controversy`

3.2. Constructivism in computer science

> In mathematics everything is algorithm and nothing is meaning; even when it doesn't look like that because we seem to be using words to talk about mathematical things. Even these words are used to construct an algorithm.
>
> Ludwig Wittgenstein (1974, p. 468)

When Brouwer proposed its new foundation of mathematics, the modern computer had not appeared yet. But soon later, the first cornerstones of theoretical computer science began to be laid by Turing (1936–37), Church (1932, 1941) and Kleene (1936, 1952). The convergence of some of the basic ideas of intuitionism and computer science became soon evident. The starting point can be identified in the work of Arend Heyting (1956), Brouwer's student.

Constructivism

Constructivism[34] is often identified with intuitionism, although intuitionism is not only a constructivist program.

Constructivism emerges in the final quarter of the 19th century, and may be regarded as a reaction to the rapidly increasing use of highly abstract concepts and methods of proof in mathematics. In the course of time, the focus of activity in constructive mathematics has shifted from Brouwerian intuitionism to Markov's constructivism, then to Bishop's constructivism. Together with the growth of interest in CS, recursive mathematics attracted more researches. The subject is still flourishing.[35]

Characteristic for the constructivist trend is the insistence that mathematical objects are to be constructed (mental constructions) or computed. Leopold Kronecker and Henri Poincaré may be described as the first conscious constructivists. For Kronecker, only the natural numbers were 'God-given'; all other mathematical objects ought to be explained in terms of natural numbers (at least in algebra).[36] From the philosophical angle,

[34] Constructivism is also the label given to a set of theories about learning which fall somewhere between cognitive and humanistic views. It is a theory, which claims that students construct knowledge rather than merely receive and store knowledge transmitted by the teacher. In CS is practised by, e.g. Ben-Ari (2001). In the case of mathematics, and – by analogy – CS could be based on social constructivist philosophy of mathematics:

> In fact, whether one wishes it or not, all mathematical pedagogy, even if scarcely coherent, rests on a philosophy of mathematics. (Thom 1973, p. 204)

[35] About history of constructivism till ca. 1965, see (Troelstra 1991).

[36] The thought of Kronecker is summarized in his famous saying: Die ganzen Zahlen hat der Liebe Gott gemacht, alles andere ist Menschenwerk (God created the integers, all else is the work of man).

constructivism asserts that it is necessary to find (or "construct") a mathematical object to prove that it exists. The proof of the existence of a mathematical object is tied to the possibility of its construction. Even though most mathematicians do not accept the constructivist's thesis, that only mathematics performed on the basis of constructive methods is sound, constructive methods are increasingly of interest on non-ideological grounds. Although investigations in the area of constructive logic and mathematics do not belong to the main stream of research, the concepts and techniques of constructive systems play a significant role in theoretical computer science and artificial intelligence, e.g. the notion of 'formulas-as-types'[37] and Martin-Löf-style type theories. The most celebrated contemporary mathematical constructivism, at least before 1967, was Brouwerian (Dutch or neo-) intuitionism.

Because the meaning "to construct" is not clearly defined, that has led to many forms of constructivism. In particular – and this is of interest to us – to a refinement of intuitionistic mathematics based on the reinterpretation of Brouwer's *mental constructions* as *computer programs*. From constructive proofs one can, at least in principle, extract algorithms that compute the elements and simulate the constructions whose existence is established in the proof.

Almost all constructivists accept the same logic, namely that axiomitized by Heyting. Intuitionistic connectives and quantifiers have computational interpretations. A surprising fact is that the rejection of the principles of excluded middle and impredicativity had very important consequences for computability. It is also possible to reconcile intuitionism with its competitors: the core ideas of formalizm, logicism and intuitionism can be synthesized in constructive foundation of mathematics.

Recursive realizability

Stephen Cole Kleene[38] with his notion of *recursive realizability* (Kleene 1945, Kleene 1952) thoroughly investigated the nature of intuitionism. Recursive realizability establishes a connection between the notion of computable (recursive) function and intuitionistic logic. Every proposition is

[37] The expression "formulas-as-types" is widely widely used but its precise meaning is not sharply delimited.

[38] In 1990 he was awarded the National Medal of Science.

To precision, to the clear definition of the notion of constructable and recursive functions, and to the application of these notions to intuitionism, in computer science, and in logic generally. `http://www.nap.edu/html/biomems/skleene.pdf`

For more about history of realizability, see (Oosten 2000).

interpreted as a natural number, by using the encoding of pairs of natural numbers and of recursive functions as natural numbers. Proofs become themselves mathematical objects (natural numbers) about which we can reason. Such objects as natural numbers can be implemented on a computer. The system Automath (**auto**mating **math**ematics) devised by Nicolaas Govert de Bruijn in and after 1967 is the first such an implementation (de Bruijn 1970, de Bruijn 1980).[39] This project can be seen as the predecessor of type theoretical proof assistants such as the well known Nuprl and Coq.

Recursive constructive mathematics

In the former Soviet Union, A. A. Markov,[40] who initiated the approach around 1950, and his collaborators developed what was essentially recursive mathematics using intuitionistic logic. The school maintained that mathematics (Aberth 1980, p. 2):

> may be informally described as an analysis wherein a computation algorithm is required for every entity employed. The functions, the sequences, even the numbers of computable analysis are defined by means of algorithms. In this way definition and evaluation are made inseparable.

New constructivism

New constructivism (or Bishop constructivism) originated with Errett Albert Bishop's influential *Foundations of Constructive Analysis* (1967. In *A Constructivist Manifesto* (1967, Chapter 1, p. 2) we read:

> Mathematics belongs to man, not to God. We are not interested in properties of the positive integers that have no descriptive meaning for finite man. When a man proves a positive integer to exist, he should show how to find it. If God has mathematics of his own that needs to be done, let him do it himself.

Bishop insisted that new constructive mathematics capture the 'numerical meaning' of mathematical claims.

The numerical meaning was to be cashed out in predicting the outcomes of performable computations on the integers, computations that come from realizing constructive theorems as computer programs. Bishop encouraged

---

[39] For more, see `http://www.win.tue.nl/automath/`, `http://www.cs.ru.nl/~freek/aut/`. For an overview of systems implementing "mathematics in the computer" see `http://www.cs.ru.nl/~freek/digimath/bycategory.html#tacticprover`.

[40] An excellent reference for the work of the Markov School is (Kushner 1985). See also `http://logic.pdmi.ras.ru/Markov/Markov.html`.

a comparison between formalizations of new constructivism and high-level specification and programming languages, ones whose proofs could be compiled into implementable code. This general idea of 'programming constructive mathematics' has since been adopted by a number of computer scientists.

Martin-Löf's Constructive Type Theory

Intuitionistic type theory, or constructive type theory, or Martin-Löf type theory is a logical system and a set theory based on the principles of mathematical constructivism. It was developed by Per Martin-Löf in the early 1970s, as a constructive foundation for mathematics. A similar approach has been developed by Constable (1971) and Beeson (1983).

This theory is based on extended[41] Curry–Howard isomorphism between propositions and types to include disjunction, existential quantification, and induction: a proposition is represented as a type: namely, the type of proofs of the proposition (Martin-Löf 1998, Martin-Löf 1982*a*, Martin-Löf 1984*b*).[42] The types of Martin-Löf type theory play a similar role as sets in set theory but functions definable in type theory are always computable.

The possibility of using constructive mathematics as a basis for programming was suggested in (Bishop 1970) by Bishop. Per Martin-Löf was the first who explicitly and directly used intuitionistic logic in connection with computer science[43]. Intuitionistic logic in the form of Martin-Löf's theory of types (1982*a*) is well suited as a theory for program construction since it is possible to express both specifications and programs within the same formalizm. It provides a complete theory of the process of program specification, construction, and verification, i.e. theory of programs correctness.

Furthermore, the proof rules can be used to derive a correct program from a specification as well as to verify that a given program has a certain property. Type theory is more than a programming language and it should not be compared with programming languages, but with formalized programming logics.[44]

---

[41] By introducing dependent types, that is types which contain values.

[42] Some of the basic ideas were already present in (Scott 1970), who was inspired by de Bruijn's work on Automath

[43] His first paper on the subject *Constructive Mathematics and Computer Programming* (1982*a*), later reprinted as (Martin-Löf 1984*a*, Martin-Löf 1985), was presented at the 6th International Congress for Logic, Methodology and Philosophy of Science in Hannover in August 1979. This paper was preceded by the first expositions of Martin-Löf 's ideas in (Martin-Löf 1982*b*) and in some lecture notes, made by Sambin during a course in 1980, published as (Martin-Löf 1984).

[44] NordstromPeterssonSmith1990

For Martin-Löf (1982*a*):

> the whole conceptual apparatus of programming mirrors that of modern mathematics (set theory, that is, not geometry) and yet is supposed to be different from it. How come? The reason for this curious situation is, I think, that the mathematical notions have gradually received an interpretation, the interpretation which we refer to as classical, which makes them unusable for programming. Fortunately, I do not need to enter the philosophical debate as to whether the classical interpretation of the primitive logical and mathematical notions . . . is sufficiently clear, because this much at least is clear, that if a function is defined as a binary relation satisfying the usual existence and unicity conditions, whereby classical reasoning is allowed in the existence proof . . . then a function cannot be the same thing as a computer program . . . Now it is the contention of the intuitionists. . . that the basic mathematical notions, above all the notion of function, ought to be interpreted in such a way that the cleavage between mathematics, classical mathematics, that is, and programming that we are witnessing at present disappears. In the case of the mathematical notions of function and set, it is not so much a question of providing them with new meanings as of restoring old ones . . .

One major advantage of Martin-Löf's formal approach to constructive mathematics is that it greatly facilitates the extraction of programs from proofs. This has led to serious work on the implementation of constructive mathematics (Martin-Löf 1982*a*, Constable, Allen, Allen, Bromley, Cleaveland, Cremer, Harper, Howe, Knoblock, Mendler, Panangaden, Smith, Sasaki & Smith 1986, Hayashi & Nakano 1988).

A number of popular computer-based proof systems are based on Martin-Löf type theory, for example NuPRL,[45] LEGO,[46] Coq,[47] Agda[48] and Twelf.[49]

The link between constructive mathematics and programming holds great promise for the future implementation and development of abstract mathematics on the computer. Martin-Löf 's work made it evident that: *computer science is equivalent to completely formalized mathematics that uses only intuitionistic logic.*

---

[45] See `http://www.cs.cornell.edu/info/projects/nuprl/book/doc.html`

[46] `http://www.dcs.ed.ac.uk/home/lego/`

[47] `http://coq.inria.fr/`

[48] See `http://www.cs.chalmers.se/~catarina/agda/`

[49] `http://www.cs.cmu.edu/~twelf/`

Constructive Reverse Mathematics

Reverse mathematics originates with Friedman's paper *Some systems of second order arithmetic and their use* (1975). The research programme is aiming to classify mathematical theorems according to their equivalence to one of a small number of set-theoretic principles. A programme of reverse mathematics based on intuitionistic logic was initiated independently by Veldman (2005) and Ishihara (2005, 2006).

Social constructivism

Although it seems that social constructivism has no direct impact on CS, to name all the contemporary kinds of constructivism, we will point out some its theses.

The most significant bases of the social constructivist theory were laid down by Vygotsky (1962). For social constructivists all knowledge, including mathematics is actually socially constructed in support of particular values and understandings. Mathematics as a social construction, a cultural product, is fallible corrigible and changing like any other kind of human cognition. Paul Ernest works (Ernest 1991, Ernest State University of New York Press) are significant for social constructivism in mathematics (Piotrowska 2008).

Generally it is not controversial that the origins of mathematics are social or cultural (Bishop 1991, Wilder 1953, Wilder 1968, Wilder 1981). The new wave in the philosophy of mathematics with such representatives as Lakatos (1976, 1978), Davis and Hersh (1980), Kitcher (1983), Tymoczko (1986) and Wittgenstein (1978) advocates the thesis that the justification of mathematical knowledge rests on its quasi-empirical basis. This view is causing controversy.

According to Lakatos (1978), the quest for certainty in mathematics leads inevitably to an infinite regression. Any mathematical system depends on a set of assumptions, and there is no way of escaping them. We cannot establish the certainty of mathematics without assumptions. Only from an assumed basis do the theorems of mathematics follow. Therefore theorems of mathematics are conditional and not absolute certain.

For social constructivist the truths of mathematics is a question of social agreement. According to Wittgenstein (1978) mathematical certainty rests on socially accepted rules of discourse embedded in 'forms of life'.

The social constructivism view of mathematics is completely opposed to the absolutists belief that mathematical truths are universal, independent of humankind (mathematics is discovered, not invented), and culture- and value-free.

Conclusions

Gödel blew away the dream of establishing firm foundation of mathematics. Nevertheless "no good tree bears bad fruit": the efforts of the creators of the programmes in the foundation of mathematics resulted in much newer and more beautiful mathematics, and – what was the subject of our considerations – have had great impact on the origin and development of computer science. Once again it is true that: Extinguished philosophies lie about the cradle of every science as the strangled snakes beside that of Hercules.[50]

Today we know that we cannot expect to find a firm foundation for our science. Such a foundation is not necessary for technical research at all. It does not mean that no philosophy is needed. Conversely, "thought has been the father of every advance since time began". Now in computer science we may see an increasing influence of ideas from the philosophy and methodology of science and humanities. Such ideas as those connected with probability, induction, and causality have opened new perspectives in computer science.

REFERENCES

Aberth, O. (1980), *Computable Analysis*, McGraw-Hill, New York. An attractive technical introduction to analysis in the style of Russian constructivism.

Barendregt, H. (1992), Lambda calculi with types, *in* T. S. E. M. S. Abramsky (Editor), Dov M. Gabbay (Editor), ed., 'Handbook of logic in Computer Science', Vol. 2, Oxford University Press, Oxford, pp. 117–309.

Beeson, M. J. (1983), Proving programs and programming proofs, *in* 'International Congress on Logic, Methodology and Philosophy of Science. Salzburg, Austria', North-Holland, Amsterdam.

Beeson, M. J. (1985), *Foundation of Constructive Mathematics*, Springer-Verlag.

Berka, K. & Kreiser, L., eds (1971), Akademie-Verlag, Berlin. Zweite durchgesehene Auflage, 1973.

Berka, K. & Kreiser, L., eds (1973), *Logik-Texte: kommentierte Auswahl zur Geschichte der modernen Logik*, 2 edn, Akademie Verlag, Berlin.

Bishop, A. J. (1991), *Mathematical Enculturation*, Kluwer, Dordrecht.

Bishop, E. (1967), *Foundations of Constructive Analysis*, McGraw-Hill, New York. The bible of new constructivism. A technical work with a lucid nontechnical introduction.

[50] Adopted from T. H. Huxley.

Bishop, E. (1970), Mathematics as a numerical language, *in* J. M. A. Kino & R. Vesley, eds, 'Intuitionism and Proof Theory', North-Holland, Amsterdam, pp. 53–71.

Bourbaki, N. (1991), *Elements of the History of Mathematics*, Springer-Verlag, Heidelberg. translated from the French by John Meldrum.

Bridges, D. & Reeves, S. (1997), 'Constructive mathematics, in theory and programming practice'.

Brouwer, L. E. J. (1907), Over de Grondslagen der Wiskunde, Doctoral thesis, University of Amsterdam, Amsterdam. Reprinted with additional material (D. van Dalen, ed.) by Matematisch Centrum, Amsterdam, 1981.

Brouwer, L. E. J. (1908), 'Over de onbetrouwbaarheid der logische principes', *Tijdschrift voor Wijsbegeerte* **2**, 152–158. English translation: *On the unreliability of the principles of logic* (Heyting & Freudenthal 1975, 107–111).

Brouwer, L. E. J. (1913), 'Intuitionism and formalism', *Bull. Amer. Math. Soc.* **20**, 81–96. Inaugural address at the University of Amsterdam, read October 14, 1912. Translated by Arnold Dresden. Reprinted in Bulletin (New Series) of the American Mathematical Society, Volume 37, Number 1, Pages 55–564. Tłumaczenie angielskie w: P. Benacerraf, H. Putnam, *Philosophy of Mathematics. Selected Readings,* (ed.) Cambridge University Press, 1987.

Brouwer, L. E. J. (1924), Beweis dass jede volle Funktion gleichmässig stetig ist, *in* 'Koninklijke Nederlandse Akademie van Wetenschappen. Proceedings of the Section of Sciences', Vol. 27, pp. 189–193. Also (Heyting & Freudenthal 1975, 274–280).

Brouwer, L. E. J. (1952), 'Historical background, principles and methods of intuitionism', *South African Journal of Science* .

Browder, F. E., ed. (1976), *Mathematical Developments Arising from Hilbert Problems, Proceedings of Symposia in Pure Mathematics*, Vol. 28, American Mathematical Society, Providence. two parts.

Burstall, R. (2000), 'Christopher strachey–understanding programming languages', *Higher-Order and Symbolic Computation* **13**(52).

Chaitin, G. J. (2004), 'Leibniz, randomness & the halting probability'. `http://www.cs.auckland.ac.nz/CDMTCS/chaitin/turing.html`.

Church, A. (1932), 'A set of postulates for the foundation of logic', *Annals of Mathematics* **33**, 346–66.

Church, A. (1933), 'A set of postulates for the foundation of logic (second paper)', *Annals of Mathematics* **34**, 839–864.

Church, A. (1936), 'An unsolvable problem of elementary number theory', *American Journal of Mathematics* **58**, 345–363. Presented to the American Mathematical Society, April 19, 1935; abstract in B. Am. Math. S., vol(41), May 1935. Reprinted in (Davis 1965, s. 89–107).

Church, A. (1937), 'Reviews of (Turing 1936–37) and (Post 1936)', *Journal of Symbolic Logic* **2**(1), 42–43.

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

Church, A. (1941), *The Calculi of Lambda-Conversion*, Princeton University Press, Princeton.

Chwistek, L. (1921), 'Antynomje logiki formalnej', *Przegląd Filozoficzny* **24**, 164–171.

Chwistek, L. (1948), *The Limits of Science: Outline of Logic and of the Methodology of the Exact Sciences*, Routledge and Kegan Paul, London.

Constable, R. L. (1971), Constructive mathematics and automatic programs writers, *in* 'Proceedings of IFIP Congress', Ljubljana, pp. 229–233.

Constable, R. L., Allen, S. F., Allen, S. F., Bromley, H. M., Cleaveland, W. R., Cremer, J. F., Harper, R. W., Howe, D. J., Knoblock, T. B., Mendler, N. P., Panangaden, P., Smith, S. F., Sasaki, J. T. & Smith, S. F. (1986), 'Implementing mathematics with the NuPRL proof development system'.

Curry, H. (1934), Functionality in combinatory logic, *in* 'Proceedings of the National Academy of Sciences', Vol. 20, pp. 584–590.

Curry, H. B., Feys, R. & Craig, W. (1958), *Combinatory Logic*, Vol. 1 of *Studies in logic and the foundations of mathematics*, North-Holland Publishing Company, Amsterdam. with 2 sections by William Craig.

Davis, M. (1965), *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems, and Computable Functions*, Raven Press, Hewlett, N.Y.

Davis, M. (1988), Influences of mathematical logic on computer science, *in* R. Herken, ed., 'The Universal Turing Machine. A Half-Century Survey', Oxford University Press, pp. 315–26.

Davis, M., ed. (1994), *Solvability, Provability, Definability: The Collected Works of Emil L. Post*, Birkhuser, Boston.

Davis, P. & Hersh, R. (1980), *The Mathematical Experience*, Penguin, Harmondsworth.

de Bruijn, N. (1968), Automath, a language for mathematics, Technical Report TH-report 68-WSK-05, Department of Mathematics, Eindhoven University of Technology. Reprinted in revised form, with two pages commentary, in: (Siekmann 1983).

de Bruijn, N. G. (1970), The mathematical language AUTOMATH, its usage and some of its extensions, *in* 'Symposium on automatic demonstration', Vol. 125 of *Lecture Notes in Mathematics*, p. 29.

de Bruijn, N. G. (1980), A survey of the AUTOMATH project, *in* Seldin & Hindley (1980).

Eden, A. H. (2007), 'Three paradigms of computer science', *Minds and Machines* **17**(2), 135–167. Special issue on the Philosophy of Computer Science.

Ernest, P. (1991), *The Philosophy of Mathematics Education*, Routledge, London.

Ernest, P. (State University of New York Press), *Social Constructivism as a Philosophy of Mathematics*, 1998, Albany, New York.

Ewald, W., ed. (1996), *From Kant to Hilbert: A Source Book in the Foundation of Logic*, Vol. 1–2, Clarendon Press, Oxford.

Frege, G. (1879), *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, Halle. Introduction dated: 18 XII 1878. Reprinted in (Frege 1993). Shortened version in (Berka & Kreiser 1971). English translation in (van Heijenoort 1967, pp. 1–82).

Frege, G. (1884), *Die Grundlagen der Arithmetik. Eine logisch-mathematische Untersuchung ber den Begriff der Zahl*, W. Koebner, Breslau. Unveränderter Neudruck: M. & H. Marcus, Breslau, 1934. English translation: (Frege 1968), (Frege 1980).

Frege, G. (1893–1903), *Grundgesetze der Arithmetik, Begriffsschriftlich abgeleitet*, Vol. 1–2, Jena. Reprint: Darmstadt 1961; English translation (Furth 1964).

Frege, G. (1964), *The basic laws of arithmetic*, University of California Press. Translated by Montgomery Furth.

Frege, G. (1967), Letter to Russell, *in* van Heijenoort 1967, pp. 127–8. 2nd ed., 1971.

Frege, G. (1968), *The Foundations of Arithmetic: A Logico-Mathematical Enquiry into the Concept of Number*, Blackwell. English translation by J. L. Austin.

Frege, G. (1980), *The Foundations of Arithmetic: A logico-mathematical enquiry into the concept of number*, Northwestern University Press, Evanston, Illinois. (Frege 1884), trans. by J. L. Austin.

Frege, G. (1993), *Begriffsschrift und andere Aufsätze*, Hildesheim.

Friedman, H. (1975), Some systems of second order arithmetic and their use, *in* 'Proceedings of the 17th International Congress of Mathematicians', Vancouver, BC.

Gödel, K. (1931), 'Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I', *Monatshefte für Mathematik und Physik* **38**, 173–198. Received: 17. XI. 1930. Translated in: Solomon Feferman (ed.), "Kurt Gdel: Collected Works", Oxford University Press, volume 1 (1986) 144–195. Includes the German text and parallel English translation. English translations in (Davis 1965, 5–38), (van Heijenoort 1967, 595–616). Online version (translation by B. Meltzer, 1962):

`http://www.ddc.net/ygg/etext/godel/` PDF version (translation by M. Hirzel): `http://nago.cs.colorado.edu/~hirzel/papers/canon00-godel.pdf`.

Gillies, D. A. (2002), Logicism and the development of computer science, *in* A. C. Kakas & F. Sadri, eds, 'Computational Logic: Logic Programming and Beyond, Part II', Springer, pp. 588–604. A Paper for Bob Kowalski's 60th Birthday.

Gödel, K. (1930), 'Die Vollständigkeit der Axiome des logischen Funktionskalküls', *Monatshefte für Mathematik und Physik* **37**, 349–360. (Gödels Dissertation, 1929) also with English transl. in *Collected Works*, S. Feferman et al., eds., vol. 1, Oxford University Press, Oxford, 1986, pp. 60–101; English translation in (van Heijenoort 1967, pp. 582–591).

Hayashi, S. & Nakano, H. (1988), *PX: A Computational Logic*, MIT Press, Cambridge MA.

Heyting, A. (1930), 'Die formalen Regeln der intuitionistischen Logik', *S.-Ber. Preuss. Phys.-Math. Kl. II* pp. 42–56.

Heyting, A. (1934), *Mathematische Grundlagenforschung. Intuitionismus. Beweistheorie*, Springer Verlag, Berlin.

Heyting, A. (1956), *Intuitionism. An Introduction*, Amsterdam. Wyd. 2: 1966.

Heyting, A. & Freudenthal, H., eds (1975), *Collected Works of Luitzen Egbertus Jean Brouwer*, North-Holland, Amsterdam.

Hilbert, D. (1899), *Grundlagen der Geometrie*, Teubner-Verlag, Leipzig. 8th amended ed. by P. Bernays with supplements, Stuttgart, 1956-1987; 14th ed., with contributions by H. Kiechle et al., M. Toepell, ed., Stuttgart, 1999; the first English translation *The Foundations of Geometry* by E. J. Townsend, Open Court, Chicago, 1902; new translation by L. Unger, Open Court, La Salle, 1972. [The first German edition of the *Grundlagen* was a contribution to the Festschrift zur Enthllung des Gauss-Weber Denkmals in Gttingen, pp. 1–92; later editions were published separately].

Hilbert, D. (1900), 'Mathematische Probleme', *Nachrichten von der Kniglichen Gesellschaft der Wissenschaften zu Gttingen, Math.-Phys. Klasse* (3), 253–297. Lecture held at the Paris ICM 1900. Revised version in (Hilbert 1901*b*) and (Hilbert 1932–1935), pp. 290–329; English translation in (Hilbert 1901*a*); also in (Browder 1976), pp. 1–34; (Yandell 2002), pp. 324–389; (?), pp. 240–282; selected parts in (Ewald 1996), pp. 1096–1105; (Reid 1970), chap. 9; French translations by M. L. Laugel (with corrections and additions) in *Compte Rendu Deuxième Congrès International des Mathématiciens tenu à Paris du 6 au 12 aoŭt* 1900, E. Duporcq, ed., Gauthier-Villars, Paris, 1902, pp. 58–114; (fragments) L'Enseignement Mathématique 2 (1900) 349–354 and *Revue Générale des Sciences Pures et Appliquées* 12 (1901) 168–174;

annotated Russian edition, *Problemy Gilberta*, P. S. Alexandrov, ed., Nauka, Moscow, 1969; its German edition *Die Hilbertschen Probleme*, Akademische Verlagsgesellschaft, Leipzig, 1976, 2nd ed. 1979, reprint 1998 (both the Russian and German editions give the revised version of (Hilbert 1932–1935)., David Joyce, Clark University, produced a list of Hilbert's problems and a web version of Hilbert's 1900 address in March 1997 – `http://aleph0.clarku.edu/~djoyce/hilbert/problems.html`.

Hilbert, D. (1901*a*), 'Mathematical problems', *Bulletin of the American Mathematical Society* **8**, 437–479. Lecture delivered at the International Congress of Mathematicians at Paris in 1900, translated by Mary Winton Newson. Reprinted in the *Bulletin* 37 (2000) 407–436;.

Hilbert, D. (1901*b*), 'Mathematische Probleme', *Archiv der Mathematik und Physik* **1**, 44–63; 213–237.

Hilbert, D. (1926), 'Über das Unendliche', *Mathematische Annalen* **95**, 161–190. Lecture given in Mnster, 4 June 1925. Also in (Hilbert 1899), 7th ed., pp. 262–288, shortened version in *Jahresber. Deutsch. Math. Verein.* 36 (1927) 201-215; English translation (by S. Bauer-Mengelberg) in (van Heijenoort 1967), pp. 367–392; 2nd ed., 1971; French translation (by A. Weil), *Sur l'infinie*, *Acta Math.* 48 (1926) 91–122; Polish translation (by R. Murawski) in (Murawski 1986, pp. 288–307).

Hilbert, D. (1928), 'Die Grundlagen der Mathematik', *Abhandlungen aus dem Seminar der Hamburgischen Universität* **6**, 65–85. followed by Diskussionsbemerkungen zu dem zweiten Hilbertschen Vortrag by H. Weyl, pp. 86–88, and Zusatz zu Hilberts Vortrag by P. Bernays, pp. 89–95; shortened version in (Hilbert 1998), 7th ed., pp. 289–312; English translation (by S. Bauer-Mengelberg and D. Follesdal) in (van Heijenoort 1967), pp. 464–479. On-line publication `www.marxists.org/reference/subject/philosophy/works/ge/hilbert.htm`.

Hilbert, D. (1932–1935), *Gesammelte Abhandlungen*, Vol. 3, 2nd ed., 1970 edn, Springer-Verlag, Berlin.

Hilbert, D. (1998), *The Theory of Algebraic Number Fields*, Springer Verlag, Berlin.

Howard, W. A. (1980), The formulae-as-types notion of construction, *in* Seldin & Hindley (1980), pp. 479–490. original paper manuscript from 1969.

Ishihara, H. (2005), Constructive reverse mathematics: Compactness properties, *in* L. Crosilla & P. Schuster, eds, 'From Sets and Types to Analysis and Topology: Towards Practicable Foundations for Constructive Mathematics', The Clarendon Press, Oxford.

Ishihara, H. (2006), 'Reverse mathematics in Bishop's constructive mathematics', *Phil. Scientiae, Cahier Special* **6**, 43–59.

Kitcher, P. (1983), *The Nature of Mathematical Knowledge*, Oxford University Press, Oxford.

Kleene, S. C. (1936), 'General recursive functions of natural numbers', *Mathematische Annalen* **112**(1), 727–742.

Kleene, S. C. (1945), 'On the interpretation of intuitionistic number theory', *Journal of Symbolic Logic* **10**, 109–124.

Kleene, S. C. (1952), *Introduction To Metamathematics*, North-Holland Publishing Co, Amsterdam and P. Noordhoff N. V., Groningen.

Kleene, S. C. & Rosser, J. B. (1935), 'The inconsistency of certain formal logics', *Annals of Mathematics* **36**, 630–36.

Kolmogorov, A. N. (1932), 'Zur deutung der intuitionistischen logik', *Mathematische Zeitschrift* **35**, 58–65. Also (Berka & Kreiser 1973, pp. 178–185).

Kushner, B. A. (1985), 'Lectures on constructive mathematical analysis', *Amer. Math. Soc. Providence RI.*

Lakatos, I. (1976), *Proofs and Refutations*, Cambridge University Press, Cambridge.

Lakatos, I. (1978), *Philosophical Papers*, Cambridge University Press, Cambridge.

Lindström, S., Palmgren, E., Segerberg, K. & Stoltenberg-Hansen, V., eds (2007), *Logicism, Intuitionism, and Formalism What Has Become of Them?*, Vol. 341 of *Synthese Library*, Springer.

Linsky, B. (2009), Chwistek's theory of constructive types, *in* J. M. M. M. W. Lapointe, S.; Wolenski, ed., 'The Golden Age of Polish Philosophy. Kazimierz Twardowski's Philosophical Legacy', Vol. 16 of *Logic, Epistemology and the Unity of Science, Part IV*, Springer, chapter 14, pp. 203–219.

Martin-Löf (1984*a*), 'Constructive mathematics and computer programming', *Philosophical Transactions of the Royal Society of London. Series A, Mathematics and Physical Sciences* **A 312**, 501–518.

Martin-Löf, P. (1982*a*), Constructive mathematics and computer programming, *in* H. P. K.-P. P. L. J. Cohen, J. Łoś, ed., 'Proceedings of the Sixth International Congress for Logic, Methodology and Philosophy of Science in Hannover in August 1979', Vol. 6, North Holland, pp. 153–175. Reprinted in (Martin-Löf 1984*a*, Martin-Löf 1985).

Martin-Löf, P. (1982*b*), An intuitionistic theory of types: predicative part, *in* H. Rose & J. Shepherdson, eds, 'Logic Colloquium 1973', North Holland, pp. 73–118.

Martin-Löf, P. (1984*b*), *Intuitionistic Type Theory*, Bibliopolis, Naples. Notes by Giovanni Sambin of a series of lectures given in Padua, June 1980.

Martin-Löf, P. (1985), Constructive mathematics and computer programming, *in* C. Hoare & J. Shepherdson, eds, 'Mathematical Logic and Programming Languages', Prentice-Hall International, Englewood Cliffs, N. Y.

Martin-Löf, P. (1998), An intuitionistic theory of types, *in* G. Sambin & J. Smith, eds, 'Twenty-Five Years of Constructive Type Theory', Vol. 36 of *Oxford*

Logic Guides, Oxford Science Publications, pp. 127–172. Proceedings of a Congress Held in Venice, October 1995.

McCarthy, J. (1960), 'Recursive functions of symbolic expressions and their computation by machine', *Communications of the ACM* **3**(4), 184–195. Part I, Part II was never published.

Milner, R. (1984), A proposal for standard *ml*, *in* 'Proceedings of the ACM Symposium on LISP and Functional Programming', Austin, pp. 184–197.

Mordechai, B.-A. (2001), 'Constructivism in computer science education', *Journal of Computers in Mathematics and Science Teaching* **20**(1), 45–73. This article is an extended version of a paper that was presented at the *Twenty-Ninth SIGCSE Technical Symposium on Computer Science Education*, Atlanta, GA, 1998.

Murawski, R. (1999), *Recursive Functions and Metamathematics*, Kluwer Academic Publisher, Dordrecht.

Murawski, R., ed. (1986), *Filozofia matematyki. Antologia tekstów klasycznych*, ii wyd. 1994 edn, Poznań. Wybór i opracowanie R. Murawski.

Olszewski, A. (2009), *Teza Churcha. Kontekst historyczno-filozoficzny*, Universitas.

Oosten, J. V. (2000), Realizability: A historical essay, Technical report.

Piotrowska, E. (2008), *Społeczny konstruktywizm a matematyka*, Wydawnictwo Naukowe UAM, Poznań.

Poincaré, H. (1906), *La Science et l'Hypothèse*, Flammarion, Paris.

Poincaré, H. (1916), *Science et Méthode*, Flammarion, Paris.

Poincaré, H. (1925), *La Valeur de La Science*, Flammarion, Paris.

Post, E. (1936), 'Finite combinatory processes – Formulation I', *Journal of Symbolic Logic* **1**, 103–105. received Oct. 7, 1936. Presented to the American Mathematical Society Jan. 1937. Abstract in Bull. of Am. Math. Soc., Nov. 1936. Reprinted in (Davis 1994), pp. 289–291.

Post, E. (1965), Absolutely unsolvable problems and relatively undecidable propositions – account of an anticipation, *in* M. Davies, ed., 'The Undecidable: Basic Papers on Undecidable Propositions, Unsolvable Problems, and Computable Functions', Raven Press, Hewlett, N.Y., pp. 340–433.

Ramsey, F. P. (1931), *The Foundations of Mathematics and other logical essays*, Routledge & Kegan.

Reid, C. (1970), *Hilbert*, Springer-Verlag, New York. reprinted Copernicus, New York, 1996.

Russell, B. & Whitehead, A. N. (1910–1913), *Principia Mathematica*, Vol. 1–3, Cambridge. t. I – 1910, t. II – 1912, t. III – 1913. 2nd edition, with a new introduction and appendices, 1925–1927.

Schönfinkel, M. (1924), 'Über die bausteine der mathematischen logik', *Mathematische Annalen* **92**, 305–316. Translated by Stefan Bauer-Mengelberg as *On the building blocks of mathematical logic* in (van Heijenoort 1967, pp. 355–66).

Scott, D. (1970), Constructive validity, *in* 'Symposium on Automatic Demonstration', Vol. 125 of *Lecture Notes in Mathematics*, Springer-Verlag, pp. 237–275.

Seldin, J. P. & Hindley, J. R., eds (1980), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press.

Shoenfield, J. R. (1991), *Recursion Theory*, Vol. 1 of *Lecture Notes in Logic*, Springer-Verlag, Heidelberg, New York.

Siekmann, J. H., ed. (1983), *Automation of Reasoning: Classical Papers on Computational Logic, 1967–1970*, Vol. 2, Springer Verlag.

Thom, R. (1973), Modern mathematics: does it exist?, *in* A. G. Howson, ed., 'Developments in Mathematical Education', Cambridge University Press, Cambridge, pp. 195–209.

Thompson, S. (1991), *Type Theory and Functional Programming*, Addison-Wesley.

Troelstra, A. S. (1991), A history of constructivism in the 20th century, ITLI Prepublication Series ML-91-05, University of Amsterdam, `http://staff.science.uva.nl//~anne/hhhist.pdf`.

Troelstra, A. S. (1999), 'From constructivism to computer science', *Theor. Comput. Sci.* **211**(1-2), 233–252.

Troelstra, A. S. & van Dalen, D. (1988), *Constructivism in Mathematics*, North Holland.

Trzęsicki, K. (2006), 'From the idea of decidability to the number Ω', *Studies in Grammar, Logic and Rethoric* **9**(22), 73–142.

Turing, A. (2004), *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artifical Life Plus The Secrets of Enigma*, Oxford University Press, Oxford.

Turing, A. M. (1936–37), 'On computable numbers, with an application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society* **42** (Series 2), 230–265. Received May 25, 1936; Appendix added August 28; read November 12, 1936; corrections Ibid. vol. 43(1937), pp. 544–546. Turing's paper appeared in Part 2 of vol. 42 which was issued in December 1936 (Reprint in: (Turing 1965); 151–154). Online version: `http://www.abelard.org/turpap2/tp2-ie.asp`.

Turing, A. M. (1937), 'Computability and λ-definability', *Journal of Symbolic Logic* **2**, 153–163.

Turing, A. M. (1965), On computable numbers, with an application to the Entscheidungsproblem, pp. 116–151.

Tymoczko, T., ed. (1986), *New Directions in the Philosophy of Mathematics*, Birkhauser, Boston.

van Atten, M., Boldini, P., Bourdeau, M. & Heinzmann, G., eds (2008), *One Hundred Years of Intuitionism (1907–2007)*, Birkhauser Verlag AG, Basel-Boston-Berlin.

van Heijenoort, J., ed. (1967), *From Frege to Gödel. A Source Book in Mathematical Logic 1879-1931*, 1 edn, Harvard University Press, Cambridge Mass. 2nd ed., 1971.

van Stigt, W. P. (1990), *Brouwer's Intuitionism*, North-Holland, Amsterdam.

Veldman, W. (2005), Brouwer's fan theorem as an axiom and as a contrast to Kleene's alternative, preprint, Radboud University, Nijmegen, Netherlands.

Vygotsky, L. S. (1962), *Thought and language*, M.I.T. Press, Cambridge Mass.

Weyl, H. (1918), *Das Kontinuum*, Veit, Leipzig.

Weyl, H. (1967), Comments on hilbert's second lecture on the foundations of mathematics, *in* 'From Frege to Gödel. A Source Book in Mathematical Logic 1879-1931', Harvard University Press, Cambridge Mass. 2nd ed., 1971.

Wilder, R. L. (1953), 'The origin and growth of mathematical concepts', *Bull. Amer. Math. Soc.* **59**, 423–448. A retiring address, as Vice President and Chairman of Section A, American Association for the Advancement of Science, delivered before a joint meeting of Section A, the American Mathematical Society, and the Mathematical Association of America, at St. Louis, Missouri, December 29, 1952; received by the editors March 13, 1953.

Wilder, R. L. (1968), *The origin and growth of mathematical concepts*, John Wiley & Sons Inc.

Wilder, R. L. (1981), *Mathematics as a Cultural System*, Pergamon, Oxford.

Wittgenstein, L. (1953), *Philosophical Investigations*, Blackwell, Malden.

Wittgenstein, L. (1974), *Philosophical Grammar*, Basil Blackwell, Oxford.

Wittgenstein, L. (1978), *Remarks on the Foundations of Mathematics*, 3rd, revised and reset edn, Blackwell. English translation by G. E. M. Anscombe.

Wittgenstein, L. (1980), *Remarks on the Philosophy of Psychology*, University of Chicago Press.

Yandell, B. H. (2002), *The Honors Class. Hilberts Problems and Their Solvers*, A. K. Peters, Natick, MA.

Kazimierz Trzęsicki
Chair of Logic, Informatics and Philosophy of Science
University of Białystok
kasimir@uwb.edu.pl