Foundations of Mathematics from the perspective of Computer Verification

8.12.2007:660

Henk Barendregt
Nijmegen University
The Netherlands

To Bruno Buchberger independently of any birthday

Abstract

In the philosophy of mathematics one speaks about Formalism, Logicism, Platonism and Intuitionism. Actually one should add also Calculism. These foundational views can be given a clear technological meaning in the context of Computer Mathematics, that has as aim to represent and manipulate arbitrary mathematical notions on a computer. We argue that most philosophical views over-emphasize a particular aspect of the mathematical endeavor.

Acknowledgment. The author is indebted to Randy Pollack for his stylized rendering of Formalism, Logicism and Intuitionism as foundational views in connection with Computer Mathematics, that was the starting point of this paper. Moreover, useful information or feedback was gratefully received from Michael Beeson, Wieb Bosma, Bruno Buchberger, John Harrison, Jozef Hooman, Jesse Hughes, Bart Jacobs, Sam Owre, Randy Pollack, Bas Spitters, Wim Veldman and Freek Wiedijk.

1. Mathematics

The ongoing creation of mathematics, that started 5 or 6 millennia ago and is still continuing at present, may be described as follows. By looking around and abstracting from the nature of objects and the size of shapes *homo sapiens* created the subjects of arithmetic and geometry. Higher mathematics later arose as a tower of theories above these two, in order to solve questions at the basis. It turned out that these more advanced theories often are able to model part of reality and have applications. By virtue of the quantitative, and even more qualitative, expressive force of mathematics, every science needs this discipline. This is the case in order to formulate statements, but also to come to correct conclusions.

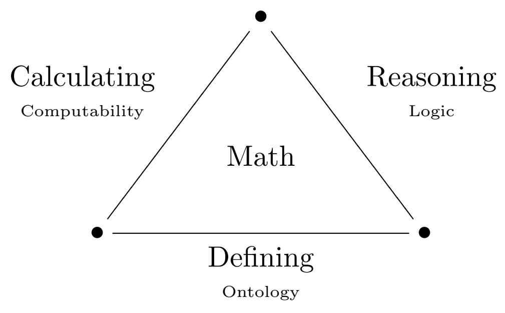

Figure 1: The triangle of mathematical activities

The mathematical endeavor consists in a stylized way of three activities[1]: *defining*, *calculating* and *proving*[2]. The three started in this order, but over the centuries they became more and more intertwined. Indeed, before one can do arithmetic, one has to have numbers and an analogous statement holds for geometry. Having numbers one wants to add and multiply these; having polygons one wants to calculate their area. At some point the calculations became complex and one discovered shortcuts. One role of proofs is that they are an essential tool to establish the correctness of calculations and constructions.

Egyptian-Chinese-Babylonian vs Greek mathematics

Different appreciations of the three sides of the triangle of mathematical activities gave rise to various explicit foundational views. Before entering into these we will argue that different implicit emphases on the sides of the triangle also did lead to different forms of mathematics. In the Egyptian-Chinese-Babylonian tradition emphasis was put on calculation. One could solve e.g. linear and quadratic equations. This was done in a correct way, but a developed notion of proof was lacking. In the Greek tradition the emphasis was on proofs. Using these one can show that there are infinitely many primes, or that $\sqrt{2}$ is irrational, something impossible to do by mere computation alone. But the rigor coming from geometric proofs also had its limitations. Euclid[3] [2002] gives a geometric proof that $(x + y)^2 = x^2 + 2xy + y^2$, but no similar results for $(x + y)^3$ (although such a result could have been proved geometrically) or $(x + y)^4$, let alone $(x + y)^n$.

Then came Archimedes (287-212 BC), who was well versed in both calculating and proving. Another person developing mathematics toward the synthesis of these two traditions was the Persian mathematician al-Khowârizmî (app. 780-850 AD), who showed that the algorithms for addition and multiplication of decimal numbers (as we learn them at school) are provably correct.

When calculus was invented by Newton (1643-1727) and Leibniz (1646-1716) the dichotomy between proving and computing was reinforced. Newton

[1]I learned this from Gilles Barthe [1996].

[2]The activity of *solving* can be seen as a particular instance of computing (or of proving, namely that of an existential statement $\exists x.P(x)$ in a constructive setting).

[3]App. 325-265 BC.

derived Kepler's laws of planetary movement from his own law of gravitation. For this he had to develop calculus and use it in a nontrivial way. He wanted to convince others of the correctness of what he did, and went in his Principia into great detail to arrive at his conclusions geometrically, i.e. on the Greek tradition[4]. Leibniz [1875-1890] on the other hand used calculus with a focus on computations. For this he invented the infinitesimals, whose foundation was not entirely clear. But the method worked so well that this tradition still persists in physics textbooks. Euler could do marvelous things with this computational version of calculus, but he needed to use his good intuitio in order to avoid contradictions. Mathematicians in Britain, on the other hand, "did fall behind" by the Greek approach of Newton, as stated by Kline (1908-1992) [1990], pp. 380-381. Only in the 19th century, by the work of Cauchy (1789-1857) and Weierstrass[5] (1815-1897), the computational and proving styles of doing calculus were unified and mathematics flourished as never before[6].

In the last third of the 20th century the schism between computing and

[4]Newton also did many important things for the synthesis of the two styles of doing mathematics. His binomial formula $(x+y)^n = \sum_{k=0}^{n}\binom{n}{k}x^{n-k}y^k$ involves computing and reasoning. It also makes sense for n a rational number. Also his fast method of computing digits of π, see [Newton 1736] or Beckmann [1971] pp. 142-143, is impressive. By computing twice

$$\int_0^{\frac{1}{4}} \sqrt{x-x^2}dx,$$

one time using calculus, another time using planar geometry and employing the binomial formula for $n = \frac{1}{2}$, Newton derived

$$\begin{aligned}\pi &= 24(\frac{\sqrt{3}}{32} + \frac{1}{12} - \frac{1}{160} - \frac{1}{3584} - \frac{1}{36864} - \frac{5}{1441792} - \frac{7}{13631488}\ldots) \\ &= 24(\frac{\sqrt{3}}{32} + \frac{1}{3}\frac{1}{2^2} - \frac{1}{5}\frac{1}{2^5} - \frac{1}{7}\frac{1}{2^9} - \frac{1}{9}\frac{1}{2^{12}} - \sum_{k=4}^{\infty}\frac{2k-3}{(2k+1)2^{3k+5}}),\end{aligned}$$

using modern notation. Newton knew how to compute $\sqrt{3}$ and this series converges quite fast. In this way he obtained $\pi = 3.14159265897928$, the last two digits are a roundoff error for 32. Ludolph van Ceulen (1539-1610) spent several decades of his life in order to compute 32 digits (later 35 digits published on his tomb in Leiden), see his [1615], while with Newton's method this could have been done in a day or so. As opposed to Newton it should be admitted that van Ceulen was more precise about the validity of the digits he obtained.

[5]Poincaré (1854-1912) made a distinction between logicians using "Analysis", among which he placed Weierstrass and intuitive mathematicians, using "Synthesis", like Klein. He mentioned that the intuitive mathematicians are better in discovery, although some logicians have this capacity as well. Poincaré added that we need both types of mathematicians: *Les deux sortes d'esprits sont également nécessaires aux progrès de la science; les logiciens, comme les intuitifs, ont fait de grandes choses que les autres n'auraient pas pu faire. Qui oserait dire s'il aimerait mieux que Weierstrass n'eût jamais écrit, ou s'il préférerait qu'il n'y eût pas eu de Riemann?* See Poincaré [1905], Ch. 1: L'intuition et la logique en mathématiques.

[6]In the nineteenth century the infinitesimals of Leibniz were abolished (at least in mainstream mathematics). But in the twentieth century they came back as *non-standard* reals. One way of doing this is by considering $h > 0$ as infinitesimal if $\forall n{\in}\mathbb{N}.h{<}\frac{1}{n}$; for this it is necessary to work in a non-Archimedian extension of $\mathbb{R}$, which can be obtained as $\mathbb{R}^I/D$, where I is an infinite set and D is an ultra-filter on $\mathcal{P}(I)$. This approach is due to Robinson (1918-1974), see his [1996]. The other way consist of infinitesimals $h > 0$, such that $h^2 = 0$. This time the trick is to work in an an intuitionistic context where the implication $h^2 = 0 \Rightarrow h = 0$ does not hold, see Moerdijk and Reyes [1991] and Bell [1998].

proving reappeared. Computer Algebra Systems are good at symbolic computing, but they cannot keep track of assumptions and use them to check whether the side conditions necessary for certain computations actually hold, nor provide proofs of the correctness of their results. Proof-verification Systems at first were not good at computing and at providing proofs for the correctness of the result of a computation. This situation is changing now.

Progress on Foundations

During the development of mathematics, notations have been introduced to help the mathematicians to remember what they defined and how and what they did compute and prove. A particularly useful notation came from Vieta (1540-1603), who introduced variables to denote arbitrary quantities. Together with the usual notations for the algebraic operations of addition and multiplication, this made finding solutions to numerical problems easier. The force of calculus consists for a good part in the possibility that functions can be manipulated in a symbolic way.

During the last 150 years general formal systems have been introduced for defining, computing and reasoning. These are the formal systems for ontology, computability and logic. The mathematical notations that had been used throughout the centuries now obtained a formal status. If a student who states the Archimedian axiom as "For all x and all $\epsilon{>}0$ there exists an $n{\in}\mathbb{N}$ such that $n\epsilon$ is bigger" a teacher could say only something like: "I do not exactly understand you." If the student is asked to use a formal statement to express what he or she means and answers "$\forall x \forall \epsilon{>}0\, \exists n{\in}\mathbb{N}. n\epsilon{>}$" the teacher can now say that this is demonstrably not a WFF (well formed formula). This little example is enough to show that these systems do provide help with defining. They also provide help with computing and proving. Often it is felt by working mathematicians, that formalization acts as a kind of chastity belt[7]. There are, nevertheless, good reasons for formalization. It may provide extra clarity to determine whether a certain reasoning is correct. But there is more to it: formalizations allow a strong form of *reflection* over established mathematics, to be discussed below.

It was mentioned that there are mathematical disciplines dealing with one subject, like arithmetic and geometry, and that there are "towers of theories" above these subjects. Examples of the latter are algebraic and analytical number theory. Let us call mathematics of the first kind a *close-up* and of the second kind a *wide-angle* discipline. These two styles will have their counterparts in the foundations of mathematics.

Reflection

There is one aspect of mathematics, present in informal mathematics as well as formalized mathematics, that plays an important role. This is *reflection* over the syntactic form of the mathematics that has been obtained so far. An

[7]On the other hand, some constructivists may call some formal systems, notably those for classical set theory, a license for promiscuity.

early example of reflection in informal mathematics is the duality principle in planar projective geometry discovered by Poncelet (1788-1867), Gergonne (1771-1859) and Plücker (1801-1868): "Given a theorem one obtains another one by replacing the word point by line and vice versa[8]."

Another example comes from Mostowski[9] [1968], who challenged in the 1960's an automated theorem prover by asking whether

$$\begin{array}{l}((((((((((((((((((A \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \\ \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A) \leftrightarrow A\end{array}$$

is a propositional tautology. The machine did not give an answer, but the statement can be seen as a tautology by considering it as A_{20} with

$$\begin{array}{rcl}A_1 & = & A \\ A_{n+1} & = & A_n \leftrightarrow A\end{array}$$

for which sequence one can show by induction on n that A_{2n} holds for all $n \in \mathbb{N}^+$.

Using formal systems also allows a form of reflection on the mathematical activities. This gives metamathematical results, showing e.g. the limitations of the methods involved (incompleteness and undecidability theorems). The incompleteness theorem of Gödel[10] [1931] uses reflection in a fundamental way: provability becomes internalized. Not only this famous theorem, but also the completeness theorem

$$\Gamma \vdash A \;\Leftrightarrow\; \Gamma \models A$$

(a statement A is derivable from the axioms Γ in first order logic iff A holds in all structures in which the axioms Γ are valid). Most other metamathematical results use reflection: within mathematics one speaks about mathematical statements.

Besides pointing to limitations, reflection also enables us to "get more mileage". If we know that a result can be proved using first order logic, then by the compactness theorem we sometimes can come to stronger conclusions. In fact this is the starting point of model theory. This subject was at first mainly based on reflection over the mathematical activities of proving and defining. Later notions like "computably (recursively) saturated models", see Chang and Keisler [1990], showed that model theory fruitfully makes use of reflection over the full triangle of mathematical activities.

Also set theory and category theory know their forms of reflection. In set theory there is Gödel's construction of the class L of constructible sets, which uses internalized definability. In category theory one can internalize the notion of a category inside a topos.

2. Foundational Formalisms

In this section we will describe various formalisms for the mathematical endeavor in the following order: logic, computability and ontology.

[8]It is convenient to first replace the statements '*point P lies on line l*' and '*line l goes through point P*' by the statements '*P touches l*' and '*l touches P*', respectively.

[9]1913-1975.

[10]1906-1978.

Logic

The Greek philosopher Aristotle (384-322 BC) made several fundamental contributions to the foundations of mathematics that are still relevant today. From him we have inherited the idea of the *axiomatic method*[11], not just for mathematics, but for all sciences. A science consists of statements about concepts. Concepts have to be *defined* from simpler concepts. In order to prevent an infinite regression, this process starts from the *primitive concepts*, that do not get a definition. Statements have to be *proved* from statements obtained before. Again one has to start somewhere; this time the primitive statements are called *axioms*. A statement derived from the axioms by pure reason is called a theorem in that axiomatic system. In mathematics one starts from arbitrary primitive notions and axioms, while in science from empirical observations, possibly using (in addition to pure reason) the principle of induction (generalization).

Just a couple of decades after Aristotle and the axiomatic method, Euclid came with his compilation of existing geometry in this form in his *Elements*[12] and was very influential as an example of the use of the axiomatic method. Commentators of Euclid stated that the primitive notions are so clear that they did not need definitions; similarly it was said that the axioms are so true that they did not need a proof. This, of course, is somewhat unsatisfactory.

A couple of millennia later Hilbert (1862-1943) changed this view. For him it did not matter what exactly is the essence of the primitive notions such as point and line, as long as they satisfy the axioms: "*The axioms form an implicit definition of the primitive concepts*". This is a fair statement, even if not all axiom systems determine up to isomorphism the domains of concepts about which they speak. In fact the more traditional mathematical theories (arithmetic, geometry) may have had the intention to describe precisely a domain thought to be unique[13]. In modern mathematics, axioms are often used with almost the opposite intention: to capture what it is that different structures have in common. The axioms of group theory[14] describe groups of which there are many kinds and sizes.

It was again Aristotle who started the quest for logic, i.e. the laws by which scientific reasoning is possible[15]. Aristotle came up with some *syllogisms* (valid

[11]In Aristotle [350 B.C.], Posterior Analytics.

[12]As was already observed in antiquity the theorems in the Elements were not always proved from the axioms by logic alone. Sometimes his arguments required extra assumptions. The axiomatization of Hilbert [1900] corrected the subtle flaws in Euclid.

[13]By Gödel's incompleteness theorem the axioms of arithmetic do not uniquely determine the set of natural numbers. By the existence of different forms of geometry and its applicability in physics we know that Euclidean geometry is not the only variant and not even the true theory about space.

[14]Actually this well-known axiom system (of a set with a binary operation such that $\forall x, y\, \exists z\, x \cdot z = y$) is a close-up theory of statements that are valid in arbitrary groups. Next to this there is also the much more interesting the wide-angle theory of groups studied with their interconnections and occurrences in mathematical situations.

[15]In Aristotle [350 B.C.], Prior Analysis. One may wonder whether his teacher Plato (427-347 BC) was in favor of this quest (because we already *know* how to reason correctly).

reasoning step based on syntactical form) like

$$\frac{\text{No } A \text{ is a } B}{\text{No } B \text{ is a } A}\,.$$

Aristotle explains this by the particular case

$$\frac{\text{No horse is a man}}{\text{No man is a horse}}\,.$$

Another of his syllogisms is

$$\frac{\text{No } A \text{ is a } C \quad \text{All } B \text{ are } C}{\text{No } A \text{ is a } B}\,.$$

Take e.g. men, birds and swans for A, B and C respectively. Aristotle also makes a distinction between such syllogisms and so called *imperfect* syllogisms, that require more steps (nowadays these are called admissible rules). The idea of specifying formal rules sufficient for scientific reasoning was quite daring and remarkable at the time. Nevertheless, from a modern perspective the syllogisms of Aristotle have the following shortcomings. 1. Only unary predicates are used (monadic logic). 2. Only composed statements involving $\rightarrow$ and $\forall$ are considered (so $\&, \vee, \neg$ and $\exists$ are missing). 3. The syllogisms are not sufficient to cover all intuitively correct steps.

In commentators of Aristotle one often finds the following example.

$$\frac{\text{All men are mortal} \quad \text{Socrates is a man}}{\text{Socrates is mortal}}\,. \tag{1}$$

Such 'syllogisms' are not to be found in Aristotle, but became part of the traditional logical teaching. They have an extra disadvantage, as they seem to imply that they do need to lead from true sentences to true sentences. This is not the case. Syllogism only need to be truth preserving, even if that truth is hypothetical. So a more didactic (and more optimistic) version of (1) is

$$\frac{\text{All sentient beings are happy} \quad \text{Socrates is a sentient being}}{\text{Socrates is happy}}\,. \tag{2}$$

This example is more didactic, because one of the premises is not true, while the rule is still valid. Aristotle was actually well aware of this hypothetical reasoning.

It was more than 2300 years later that Frege[16] [1971] completed in 1879 the quest for logic and formulated (first-order) predicate logic. That it was sufficient for the development of mathematics from the axioms was proved by Skolem[17] [1922] and independently Gödel [1930] as the completeness theorem for first order logic; see Jervell [1996] for the priority of Skolem.

The final fundamental 'contribution' of Aristotle to modern foundations of mathematics is the distinction attributed to him between proof-checking

[16] 1848-1925.
[17] 1887-1963.

and theorem-proving. If someone would claim to have a proof of a statement and present it, then that proof always could be checked line by line for its correctness. If someone would claim that a statement is provable, without presenting the proof, then it is much harder to verify the correctness of this assertion. From a modern perspective these remarks may be restated as follows: proof-checking is decidable, while theorem testing is (in general) impossible, i.e. undecidable.

Rules of logic

The rules of logic as found by Frege have been given a particularly elegant form by Gentzen[18] [1969], in his system of natural deduction, see Fig. 2. Predicate logic gives in the first place a basis for close-up mathematics (as defined in section 1). Using the axioms of (Peano) arithmetic or Euclidean geometry (as provided rigorously by Hilbert) together with the deductive power of logic one can prove a good deal. For wide-angle mathematics one needs a stronger formal system. Below we will meet three candidates for this: set theory or category theory combined with logic, or type theory (in which a sufficient amount of logic is built in).

Elimination rule	Introduction rule
$\dfrac{\Gamma \vdash A \quad \Gamma \vdash A{\rightarrow}B}{\Gamma \vdash B}$	$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A{\rightarrow}B}$
$\dfrac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash A} \quad \dfrac{\Gamma \vdash A \,\&\, B}{\Gamma \vdash B}$	$\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}$
$\dfrac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}$	$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash A \vee B}$
$\dfrac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[x := t]}$ t is free in A	$\dfrac{\Gamma \vdash A}{\Gamma \vdash \forall x.A}\; x{\notin}\Gamma$
$\dfrac{\Gamma \vdash \exists x.A \quad \Gamma, A \vdash C}{\Gamma \vdash C}\; x{\notin}C$	$\dfrac{\Gamma \vdash A[x := t]}{\Gamma \vdash \exists x.A}$

Start rule	False rule	Double-negation rule
$\dfrac{}{\Gamma \vdash A}\; A{\in}\Gamma$	$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A}$	$\dfrac{\Gamma \vdash \neg\neg A}{\Gamma \vdash A}$

Figure 2: Predicate Logic Natural Deduction Style

Several comments are in order. Γ stands for a set of formulas and $\Gamma, A = \Gamma \cup \{A\}$. The formula $\bot$ stands for absurdum, i.e. the false statement. Negation is defined as $\neg A = (A{\rightarrow}\bot)$. The double negation rule is left out for Intuitionistic logic

[18] 1909-1945.

to be discussed later. The condition $x \notin \mathrm{FV}(C)$ ($x \notin \mathrm{FV}(\Gamma)$) means that x is not among the free variables in C (in a formula of Γ respectively).

Computability

A theoretical foundation for the act of calculating was given relatively late. Following work of Grassmann (1809-1877), Dedekind (1831-1916), Peano (1858-1932) and Skolem it was Hilbert [1926] who formally introduced a class of schematically defined numerical functions (i.e. on the set $\mathbb{N}$ of natural numbers). In Gödel [1931] this class was called the *recursive* functions but, he was aware that it did not encompass all computable functions (by a human calculator). Their usual name *primitive recursive functions* was given later by Rózsa Péter[19] [1934]. This class can be defined as follows.

$$\begin{array}{|rcl|}
\hline
Z(x) & = & 0; \\
S(x) & = & x+1; \\
P^n_k(x_1,\ldots,x_n) & = & x_k; \\
\hline
f(\vec{x}) & = & g(h_1(\vec{x}),\ldots,h_n(\vec{x})); \\
\hline
f(\vec{x},0) & = & g(\vec{x}); \\
f(\vec{x},y+1) & = & h(\vec{x},y,f(\vec{x},y)). \\
\hline
\end{array}$$

Figure 3: Schemes for the primitive recursive functions

Here the variables like x range over the natural numbers $\mathbb{N}$. The scheme states that Z, S and the P^n_k are (primitive) recursive functions and that and if g, h denote earlier obtained (primitive) recursive functions then so is f defined in terms of these.

General computability

Sudan[20] [1927] and Ackermann[21] [1928] independently defined a computable function that was not primitive recursive. A simplification of such a so-called Ackermann function, with the property of not being primitive recursive but nevertheless computable, was given by Péter [1967].

$$\begin{array}{rcl}
\psi(0,y) & = & y+1; \\
\psi(x+1,0) & = & \psi(x,1); \\
\psi(x+1,y+1) & = & \psi(x,\psi(x+1,y)).
\end{array}$$

Gödel (based on an idea of Herbrand (1908-1931)) Gödel [1931], Church[22] [1932] and Turing[23] [1936] introduced richer computational mechanisms: respectively systems of equations, lambda calculus and Turing machines. Kleene[24] [1936] and Turing [1936] proved that these systems all represent the same class of numerical

[19] 1905-1977.
[20] 1889-1977.
[21] 1896-1962.
[22] 1903-1995.
[23] 1912-1954
[24] 1909-1994.

functions. This class of functions computable by either of these formalisms is now generally considered as the class of humanly computable functions. Besides this, both Church [1936] and Turing [1936] indicated how to specify a non-computable function. This showed that an old ideal of Leibniz could not be fulfilled. Leibniz wanted 1. to formulate a language in which all problems could be stated; 2. to construct a machine that could decide the validity of such statements[25]. Turing showed in fact that the problem to determine whether a logical formula A is derivable from some axioms Γ, (i.e. whether $\Gamma \vdash A$ holds) is undecidable.

Term Rewrite Systems

The notion of Herbrand-Gödel computability generalizes the class of primitive recursive functions and yields the (partial) computable functions. The idea is presented in the format of Term Rewriting Systems (TRSs) in Klop et al. [2003]. Rather than presenting the precise definitions we give a representative example. The following is called Collatz' problem[26].

$$\boxed{\begin{array}{lll} \text{Define } f:\mathbb{N}\to\mathbb{N} \text{ by} & f(n) \;=\; \begin{cases} \frac{1}{2}n, & \text{if } n \text{ is even;} \\ 3n+1, & \text{else.} \end{cases} \\ \text{Does one have} & \forall n \geq 1 \exists k. f^k(n) = 1? \end{array}}$$

Figure 4: Collatz' problem

The following TRS describes this problem. It uses the following constants and function symbols

$$\{0, S, \texttt{true}, \texttt{false}, \texttt{even}, \texttt{odd}, \texttt{if}, \texttt{half}, \texttt{threeplus}, \texttt{syracuse}, \texttt{perpetual}\}.$$

We write $\underline{0} = 0$, $\underline{n+1} = S(\underline{n})$ and add comments for the intended meanings.

[25]It is said that the existence of God was the first question about which Leibniz wanted to consult his machine. This is an early example of a striking confidence in high technology.

[26]The problem is still open, and a prize of 1000 UK pounds is offered for its solution. For $1 \leq n \leq 10^{17}$ the conjecture has been verified by Oliveira e Silva using in total 14 CPU years on 4 computers (average 200 MHz), see Chamberland [2003] *An update on the* $3x+1$ *problem* `<www.math.grinnell.edu/~chamberl/papers/survey.ps>`.

$$
\begin{array}{lcl}
\texttt{even}(\texttt{0}) & \to & \texttt{true} \\
\texttt{even}(\texttt{S}(x)) & \to & \texttt{odd}(x) \\
\texttt{odd}(\texttt{0}) & \to & \texttt{false} \\
\texttt{odd}(\texttt{S}(x)) & \to & \texttt{even}(x) \\
\texttt{threeplus}(\texttt{0}) & \to & \texttt{S}(\texttt{0}) \\
\texttt{threeplus}(\texttt{S}(x)) & \to & \texttt{S(S(S(threeplus(x))))} \\
\texttt{if}(\texttt{true}, x, y) & \to & x \\
\texttt{if}(\texttt{false}, x, y) & \to & y \quad \% \ \texttt{if}(b, x, y) = \texttt{if}\ b\ \texttt{then}\ x\ \texttt{else}\ y \\
\texttt{syracuse}(x) & \to & \texttt{if}(\texttt{even}(x), \texttt{half}(x), \texttt{threeplus}(x)) \\
 & & \% \ \texttt{syracuse}(n) = \begin{cases} \texttt{half}(n) & \text{if } \texttt{even}(n) \\ \texttt{threeplus}(n) & \text{else} \end{cases} \\
\texttt{perpetual}(\texttt{0}) & \to & \texttt{0} \\
\texttt{perpetual}(\texttt{S}(\texttt{0})) & \to & \texttt{S}(\texttt{0}) \\
\texttt{perpetual}(\texttt{S}(\texttt{S}(x))) & \to & \texttt{perpetual}(\texttt{syracuse}(\texttt{S}(\texttt{S}(x)))) \\
 & & \qquad \%\text{Conjecture: } \forall n{\in}\mathbb{N}^{+}.\texttt{perpetual}(\underline{n}) \twoheadrightarrow 1??
\end{array}
$$

Figure 5: A TRS for Collatz' problem

Note that there are no rewrite rules with $0, S,$ **true**, **false** at the 'head' of a LHS (left hand side): these are the so called *constructors*. The other elements of the alphabet are (auxiliary) function symbols. One may reduce (use the rewrite rules $\to$) within terms at arbitrary positions of the *redexes*, i.e. left hand sides with for the variables (x, y) substituted arbitrary terms. One easily shows that in this TRS ($\twoheadrightarrow$ denotes many step reduction)

$$\texttt{perpetual}(\underline{n}) \twoheadrightarrow 1 \ \Leftrightarrow\ \exists k{\in}\mathbb{N}.f^{k}(n) = 1.$$

The Collatz conjecture now becomes $\forall n{>}0.\ \texttt{perpetual}(\underline{n}) \twoheadrightarrow 1$. Erdös remarked: "Mathematics is not yet ready for these kinds of problems", referring to the Collatz conjecture. This fashion of defining computable partial functions via Term Rewrite Systems is now the standard in functional programming languages, such as Clean[27] and Haskell[28].

Combinatory Logic

As usual when dealing with universal computability, there is in fact a universal mechanism: a single TRS in which all computable functions can be represented. This is CL, combinatory logic, due to Schönfinkel (1889-1942?) and Curry (1900-1982), see Klop et al. [2003]. This TRS has two constants S,K and a binary function symbol app.

$$
\begin{array}{lcl}
\mathsf{app}(\mathsf{app}(\mathsf{app}(\mathsf{S}, x), y), x) & \to & \mathsf{app}(\mathsf{app}(x, z), \mathsf{app}(y, z)) \\
\mathsf{app}(\mathsf{app}(\mathsf{K}, x), y) & \to & x
\end{array}
$$

Figure 6: CL, functional notation

It is more usual to give this TRS in an equivalent different notation.

[27]`<www.cs.kun.nl/~clean>`

[28]`<www.haskell.org>`

$\mathsf{S}\cdot x\cdot y\cdot x$	$\to$	$x\cdot z\cdot(y\cdot z)$	$\mathsf{S}xyx$	$\to$	$xz(yz)$
$\mathsf{K}\cdot x\cdot y$	$\to$	x	$\mathsf{K}xy$	$\to$	x

Figure 7: CL infix notation and applicative notation

The price for being universal is non-termination. Indeed, it is easy to give a non-terminating CL expression: $(\mathsf{SII})(\mathsf{SII})$, where $\mathsf{I} = \mathsf{SKK}$. A more interesting one is $\mathsf{S}(\mathsf{SS})\mathsf{SSSS}$[29]. Potentially non-terminating TRSs are important if a proof-search is involved[30].

Ontology

The branch of philosophy that deals with existence is called *ontology*. Kant stated that 'being' is not a predicate. Indeed, if we would state $B(x)$ with as intended meaning that x exists, we already need to assume that we have the x whose existence is asserted. But in axiomatic theories the notion of 'being' makes sense. One has formal expressions and one claims for only some of these that they make sense, i.e. exist. For example $\frac{1}{0}$ and $\{x \mid x\notin x\}$ are expressions in respectively arithmetic and set theory whose existence cannot be consistently asserted.

In mathematics before roughly 1800 only less than a dozen basic domains are needed: the number systems $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, the Euclidean and projective plane and space and perhaps a few more. Since Descartes, one used the construction of products of domains. But that was more or less all. In the 19-th century a wealth of new spaces were created that, as stated in section 1, have their impact for proofs of properties of the elements in the more familiar spaces. Examples are groups, non-Euclidean spaces, Riemann surfaces, function spaces and so on. Therefore the need for a systematic ontology arose.

Set theory

The first systematic ontology for mathematics was created by Cantor (1845-1918) in the form of set theory. This (informal) theory, contained an inconsistency, as discovered by Russell (1872-1970), when it was embedded in a formal theory by Frege: the 'set' $R = \{x \mid x\notin x\}$ satisfies $R\in R \leftrightarrow R\notin R$. Zermelo (1871-1953) removed the possibility of this proof of inconsistency and the theory was later extended by Fraenkel (1891-1965) resulting in **ZF** set theory. It is a theory formulated in first order logic with equality with $\in$ and $=$ as only predicates. The axioms claim the existence of the sets $\emptyset$ and $\mathbb{N}$ and that other sets are constructed from given sets.

[29]Due to M. Baron and M. Duboué, see Barendregt [1984], exercise 7.4.5(i).

[30]In Geuvers, Poll and Zwanenburg [1999] it is proved that one can safely add the fixed-point combinators Y (one for each type) satisfying

$$\mathsf{Y}f \twoheadrightarrow f(\mathsf{Y}f)$$

to the proof-assistant Coq. 'Safely' means that if a putative proof containing Y normalizes, than that normal form is a proof in the system without the Y.

Set existence[31]			
$\emptyset$	$=$	$\{x \mid \bot\}$	(empty set)
$\bigcup x$	$=$	$\{z \mid \exists u{\in}x\ z{\in}u\}$	(union)
$\{x, y\}$	$=$	$\{z \mid z = x \ \vee\ z = y\}$	(pair)
$\mathcal{P}(x)$	$=$	$\{z \mid z \subseteq x\}$	(power set)
$\mathbb{N}$	$=$	$\{x \mid \forall z\ [[\emptyset{\in}z\ \&\ (\forall y{\in}z.(y \cup \{y\}){\in}z)] \ \Rightarrow\ x \subseteq z]\}$	(infinity)
$A(x, y)$	$=$	$\{z \mid z{\in}x\ \&\ \varphi(z, y)\}$	(separation)
$\psi``z$	$=$	$\{y \mid \exists x{\in}z\ \psi(x, y)\}$	(replacement)
Set properties			
$\forall x, y\ [x = y \leftrightarrow \forall z\,(z{\in}x \leftrightarrow z{\in}y)]$			(extensionality)
$\forall x \exists y{\in}x\ \neg\exists z\ [z{\in}x\ \&\ z{\in}y]$			(foundation)

Figure 8: A modern version of the axioms of **ZF** set theory.

This theory is powerful, in the sense that it gives a place to almost all needed domains in modern mathematics[32], but in some sense it is too powerful. One can form huge sets like

$$\mathcal{P}(\bigcup_{n\in\mathbb{N}} \mathcal{P}^n(\mathbb{N})),$$

and much bigger. Pondering about such large sets one may feel a "*horror infiniti*"[33]. This term was coined by Cantor [1885] referring to Gauss (1777-1855) who held that infinite is only a way of speaking[34]. In set theory with the generalized continuum hypothesis this monster looks tame in a deceptive way, as it has as cardinality 'only' $\aleph_{\omega+1}$. Perhaps it was Cantor's liking of neo-Thomistic thinking that made him comfortable with the infinite.

We will discuss how a notion like the (generalized) Cartesian product can be encoded in set theory. If $x \mapsto B_x$ is a class-function that assigns to a set x a unique set B_x, then

$$\Pi_{x\in A} B_x = \{f{\in}A{\to}\bigcup\{B_x \mid x{\in}A\} \mid \forall x{\in}A.f(x){\in}B_x\}.$$

One needs the union, pair, power set, separation and replacement axioms. (Just

[31]In the list of axioms of set theory, φ is a predicate on sets and ψ is a predicate that is a 'class function', i.e. one can prove $\forall x \exists! y.\psi(x, y)$. Here $\exists!$ means unique existence, making the intended meaning of $\psi``z = \text{Range}(\psi{\restriction}z)$. Moreover, $x \subseteq y \ \Leftrightarrow\ \forall z\,[z{\in}x \ \Leftrightarrow\ z{\in}y]$, $x \cup y = \bigcup\{x, y\}$ and $\{x\} = \{x, x\}$. The precise formulation of the separation and replacement axioms using first order definable formulas φ and ψ is due to Skolem.

[32]An exception is the category of all small categories (that are sets).

[33]"Horror for the infinite." Actually people have this existential experience already just thinking about $\mathbb{N}$ as a completed totality. Aristotle rejected the actual infinite, but did recognize the potential infinite. The Platonist view of mathematics likes to consider infinite sets as an actuality from which arbitrary subsets can be taken.

[34]"*But concerning your proof, I protest above all against the use of an infinite quantity as a completed one, which in mathematics is never allowed. The infinite is only a* façon de parler, *in which one properly speaks of limits*". See Gauss [1862].

the ordinary Cartesian product

$$A \times B = \{\{\{a, a\}, \{a, b\}\} \mid a{\in}A,\ b{\in}B\}$$

already needs several times the pair and replacement axioms.) In type theory the formation of the Cartesian product is just one axiom (but other notions, easy to formulate in set theory, will be more involved).

Often the axiom of choice is added to ZF. It is equivalent with

$$\forall x{\in}I.B_x \neq \emptyset \ \Rightarrow\ \Pi_{x\in I} B_x \neq \emptyset.$$

Also the existence of large cardinals is often assumed, among other things in order to ensure that there exists a category of all small categories.

Type theory

Type theory provides an ontology less permissive than set theory. It should be be emphasized that there are several non-equivalent versions of type theory. Intensional, extensional; first-, second- and higher-order (the second and higher-order ones often called *impredicative*; intuitionistic and classical; with none, some and full computational power; with freely generated data-types and induction/recursion principles over these. This multitude should be seen as an advantage: by choosing a particular type theory one is able to provide a foundation for particular proofs in mathematics, e.g. first- or second-order arithmetic. It is known that one can prove more in the latter, but at the price of having to believe in the quantifying over predicates over the natural numbers. This in contrast to **ZF** set theory, where one 'buys all the features in one package'.

A type A is like a set as it may have inhabitants; if a is an inhabitant of A one writes $a : A$. Of course the transition from the notation and terminology in set theory (if $a{\in}A$, then a is an element of A) is just conventional. But there is more to it. Type theory is usually *intensional*, i.e. such that $a : A$ is decidable. For this reason there is no separation axiom stating that

$$A' = \{a{:}A \mid P(a)\}$$

is another type. Indeed, the predicate P may be undecidable and this would entail the undecidability of $a{:}A'$. For similar reasons there is no quotient type $A/{\sim}$, where $\sim$ is an equivalence relation on A. Indeed,

$$a : [b],$$

where $[b]$ is the equivalence class of b, which means that $a \sim b$ and this could be undecidable. In spite of this, it is possible to represent 'subtypes' and 'quotient types' in type theory, but then proofs come to play a role. Intuitively,

$$\begin{aligned} A' &= \{(a, p) \mid a{:}A \ \&\ p \text{ is a proof of } P(a)\}; \\ A/{\sim} &= (A, {\sim}) \text{ considered as structure with a different equality.} \end{aligned}$$

Extensional type theory on the other hand does not have the *a priori* requirement that $a : A$ is decidable. In this version of type theories one can form arbitrary sub- and quotient types. But then one needs to consider triples (p, a, A), where p is a proof of $a : A$. The extra effort is similar to the simulation of subtypes and quotient types using proofs as was discussed above.

Another feature of type theory is that a statement like $a : A$ can be interpreted in more than one way. The first was already discussed: a is an inhabitant of A. The second interpretation is that now A is considered as a proposition and a is a proof of A (*propositions-as-types* interpretation). In this way proofs become explicit whereas in ordinary logical systems they remain implicit. Indeed, in logic one writes $\vdash_L A$, meaning that A is provable in the logic L. In a type theory T one has to furnish a complete expression p such that $\vdash_T p : A$. Note that this is related to the decidability of p being a proof of A and the (in general) undecidability of provability in a theory L, as was foreseen by Aristotle. For many logics L one has a corresponding type theory T such that

$$\vdash_L A \;\Leftrightarrow\; \exists p \vdash_T p : A.$$

Formal proof-verification for a statement A then consists of the construction of a (fully formalized) proof p (a so-called *proof-object*) and the mechanical verification of $p : A$. As was emphasized by the Bruijn this should be done by a reliable (hence preferably small) program in order to be methodologically sound.

Axiomatization of type theory: pure type systems

As was stated before, there is an axiom in type theory that states directly the existence of the generalized Cartesian product: if A is a type and $B(x)$ is type (with parameter $x{:}A$), then $\Pi x{:}A.B(x)$ is a type. If $B(x)$ does not depend on x (i.e. if there is no occurrence of x in $B(x)$), then $\Pi x{:}A.B(x) = A{\to}B$ ($= B^A$ in set-theoretic notation[35]), the type of functions from (the inhabitants of) A to (the inhabitants of) B.

Types come in 'kinds'. For example, in some type theories there is a kind $*^s$ whose inhabitants are those types that denote 'sets', and a kind $*^p$ for those that types that denote 'propositions'. The kind $*^s$ is therefore the 'super type' of ordinary types intended to denote sets. But also these kinds appear in various version, so we need 'super kinds', etcetera. To fix terminology, these are called collectively *sorts*, including the lower kinds like $*^s$. The various type theories differ as to what is allowed as Cartesian product. If s_1, s_2, s_3 are sorts, then

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.B) : s_3} \quad \text{product } (s_1, s_2, s_3)$$

is called the product-rule, parametrized by the three sorts. This rule works in collaboration with the application and abstraction rules

[35]This is similar to the arithmetic statement $\prod_{i=1}^{3} b_i = b_1.b_2.b_3 = b^3$ if $b_1 = b_2 = b_3 = b$.

$$\frac{\Gamma \vdash F : (\Pi x{:}A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash (F\ a) : B[x{:}=a]} \qquad \text{application}$$

$$\frac{\Gamma, x{:}A \vdash b : B \quad \Gamma \vdash (\Pi x{:}A.B) : s}{\Gamma \vdash (\lambda x{:}A.b) : (\Pi x{:}A.B)} \qquad \text{abstraction}$$

The most simple type theory has just one sort $*$ and as product rule $(*, *, *)$. This is a simplification, due to Ramsey[36] (1903-1930), of Russell's ramified theory of types with sorts $\{*_n \mid n{\in}\mathbb{N}\}$ and product rules $(*_n, *_m, *_{\max(n,m)})$, see Laan [1997].

In type theories with *dependent types*, de Bruijn [1970], one allows types to 'depend' on elements of (other) types. An intuitive example is the vector space $\mathbb{R}^n$ with $n{\in}\mathbb{N}$. This kind of type theory has as sorts $*, \Box$ with $* : \Box$ and as product rules $(*, *, *)$ and $(*, \Box, \Box)$. In type theories with *higher order types*, see Girard, Taylor and Lafont [1989], one has the same two sorts, but now as rules $(*, *, *)$ and $(\Box, *, *)$ (second order) or $(*, *, *)$, $(\Box, *, *)$ and $(\Box, \Box, \Box)$ (higher order). In second order type theory one has for example the inhabited type

$$(\lambda\beta{:}*\ \lambda x{:}(\Pi\alpha{:}*.\alpha).x\beta) : (\Pi\beta{:}*.(\Pi\alpha{:}*.\alpha)\to\beta),$$

corresponding to the statement that from the false statement anything follows (*ex falso sequitur quodlibet*). In the calculus of constructions, which underlies the proof assistant Coq, one has as product rules $(*, *, *)$, $(*, \Box, \Box)$, $(\Box, *, *)$ and $(\Box, \Box, \Box)$.

A faithful description of predicate logic is given by the type theory λPRED, with sorts $\{*^s : \Box^s, *^p : \Box^p\}$ and rules $(*^p, *^p, *^p)$ for implication, $(\Box^s, *^p, *^p)$ for quantification, $(*^s, \Box^p, \Box^p)$ for the formation of predicates.

DEFINITION. The *specification of a* PTS *(pure type system)* consists of a triple $\mathrm{S} = (\mathcal{S}, \mathcal{A}, \mathcal{R})$ where

1. $\mathcal{S}$ is a subset of the constants C, called the *sorts*;

2. A is a set of *axioms* of the form $c : s$ with $c{\in}\mathcal{C}$ and $s{\in}\mathcal{S}$;

3. $\mathcal{R}$ is a set of rules of the form (s_1, s_2, s_3) with $s_1, s_2, s_3{\in}\mathcal{S}$. The rule (s_1, s_2) is an abbreviation of (s_1, s_2, s_2).

DEFINITION. The PTS determined by the specification $\mathrm{S} = (\mathcal{S}, \mathcal{A}, \mathcal{R})$, notation $\lambda\mathrm{S}{=}\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$, is defined as follows. Expressions $\mathcal{T}$ are given by the following abstract syntax

$$\mathcal{T} = V \mid C \mid \mathcal{T}\mathcal{T} \mid \lambda V{:}\mathcal{T}.\mathcal{T} \mid \Pi V{:}\mathcal{T}.\mathcal{T}$$

A *statement* (with subject M) is of the form $M : A$, with $M, A{\in}\mathcal{T}$. A context Γ is an ordered sequence of statements with as subjects distinct variables. $\langle\ \rangle$ denotes the empty sequence. The notion of type derivation $\Gamma \vdash_{\lambda\mathrm{S}} A : B$ (we often just write $\Gamma \vdash A : B$) is defined by the following axioms and rules.

[36]As Peter Aczel pointed out to me this simple type theory was already present in Frege.

$$
\boxed{\begin{array}{lll}
\text{(axioms)} & \langle\,\rangle \vdash c : s, & \text{if } (c:s)\in\mathcal{A};\\[2ex]
\text{(start)} & \dfrac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}\,, & \text{if } x \text{ is fresh;}\\[2ex]
\text{(weakening)} & \dfrac{\Gamma \vdash A : B \quad \Gamma \vdash C : s}{\Gamma, x : C \vdash A : B}\,, & \text{if } x \text{ is fresh;}\\[2ex]
\text{(product)} & \dfrac{\Gamma \vdash A : s_1 \quad \Gamma, x{:}A \vdash B : s_2}{\Gamma \vdash (\Pi x{:}A.B) : s_3}\,, & \text{if } (s_1,s_2,s_3)\in\mathcal{R};\\[2ex]
\text{(application)} & \dfrac{\Gamma \vdash F : (\Pi x{:}A.B) \quad \Gamma \vdash a : A}{\Gamma \vdash Fa : B[x := a]}\,; & \\[2ex]
\text{(abstraction)} & \dfrac{\Gamma, x{:}A \vdash b : B \quad \Gamma \vdash (\Pi x{:}A.B) : s}{\Gamma \vdash (\lambda x{:}A.b) : (\Pi x{:}A.B)}\,; & \\[2ex]
\text{(conversion)} & \dfrac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B}\,, & \text{if} \quad A =_R B
\end{array}}
$$

Figure 9: The PTS $\lambda_R(\mathcal{S},\mathcal{A},\mathcal{R})$

Here $=_R$ is a conversion relation corresponding to a notion of reduction R including at least β, i.e. one always has

$$(\lambda x.M)N \ =_R \ M[x{:}=N]$$

and the deductive power of equational logic (making $=_R$ an equivalence relation compatible with application and λ- and Π-abstraction).
Examples of PTSs.

(i) ($\lambda{\to}$, Church [1940]) The simply typed lambda calculus with one ground type o can be specified as PTS as follows.

$$\lambda{\to}\ \boxed{\begin{array}{ll}\mathcal{S} & *,\Box\\ \mathcal{A} & o : *, * : \Box\\ \mathcal{R} & (*,*)\end{array}}$$

(ii) ($\lambda 2$, also called *system F*, Girard, Taylor and Lafont [1989]) The second order polymorphic lambda calculus can be specified as PTS as follows.

$$\lambda 2\ \boxed{\begin{array}{ll}\mathcal{S} & *,\Box\\ \mathcal{A} & * : \Box\\ \mathcal{R} & (*,*), (\Box,*)\end{array}}$$

(iii) (The λ-cube, Barendregt [1992]) The calculus of constructions as a PTS is specified by

$$\lambda C\ \boxed{\begin{array}{ll}\mathcal{S} & *,\Box\\ \mathcal{A} & * : \Box\\ \mathcal{R} & (*,*), (*,\Box), (\Box,*), (\Box,\Box)\end{array}}$$

(iv) λPRED, logic as a PTS, is determined by the following specification.

$$\lambda\text{PRED}\ \boxed{\begin{array}{ll} \mathcal{S} & *^s, *^p, *^f, \Box^s, \Box^p \\ \mathcal{A} & *^s : \Box^s, *^p : \Box^p \\ \mathcal{R} & (*^p, *^p), (*^s, *^p), (*^s, \Box^p), \\ & (*^s, *^s, *^f), (*^s, *^f, *^f) \end{array}}$$

(v) (The inconsistent type theory $* : *$)

$$\lambda *\ \boxed{\begin{array}{ll} \mathcal{S} & * \\ \mathcal{A} & * : * \\ \mathcal{R} & (*, *) \end{array}}$$

(vi) The Curry-Howard 'isomorphism' is the map $\theta : \lambda\text{PRED} \rightarrow \lambda C$ given by

$$\begin{array}{rcl} \theta(*^i) & = & * \\ \theta(\Box^i) & = & \Box \end{array}$$

See Barendregt [1992] for a discussion of such and similar 'Pure Type Systems'. These were introduced by Berardi and Terlouw as a generalization of the lambda cube.

Inductive types

For representation of mathematics we need also inductive types. These are freely generated data types like

```
nat  :=  O:nat | S:nat→nat
```

for the representation of freely generated data types. Inductive types come together with terms for primitive recursion and at the same time induction.

```
nat_rec_s :  ΠP:(nat->s).
             ((P O)->(Πn:nat.(P n)->(P (S n)))->(Πn:nat.P n))
```

for which the following rewrite rules are postulated.

```
nat_rec_s P a b O       →_ι  a
nat_rec_s P a b (S n)   →_ι  b n (nat_rec_s P a b n)
```

Also predicates can be defined inductively.

```
le [n,m:nat]  :=  le_n : (Πn:nat.(le n n) |
                  le_S : (Πn,m:nat.((le n m)→(le n (S m)))).
```

This means that we have the axiom and rule (writing n≤m for (Lt n m))

$$\frac{}{\texttt{n}\leq\texttt{n}}\ (\texttt{le\_n n}) \qquad\qquad \frac{\texttt{n}\leq\texttt{m}}{\texttt{n}\leq\texttt{(S m)}}\ (\texttt{le\_S n m}).$$

Properties of $\leq$, like transitivity, can be proved by induction on the generation of this inductive relation.

Inductive types in the context of type theory have been first proposed in Scott [1970]. Their presence makes formalization easier. A technical part of Gödel's incompleteness proof is devoted to the coding of finite sequences of natural numbers via the Chinese remainder theorem. Using an inductive type for lists makes this unnecessary. See Paulin-Mohring [1993], Dybjer [2000] and Capretta [2003] for a formal description of inductive types, related to those used at present in the mathematical assistant Coq.

The Poincaré Principle

The reduction rules generated by the primitive recursion ι-contractions, together with β-reduction from lambda calculus and δ-reduction (unfolding definitions) play a special role in type theory.

$$\frac{\Gamma \vdash p : A \quad \Gamma \vdash B : s}{\Gamma \vdash p : B} \quad A =_{\beta\delta\iota} B$$

This has, for example, the consequence that if p is a proof of $A(4!)$, then that same p is also a proof of $A(24)$. This is called the *Poincaré Principle*[37] for $\beta\delta\iota$-reduction, see Barendregt [1997], Dowek [2001]. It has a shortening effect on the lengths of proofs, while in practice the simple decidability of $p : A$ is not much increased. In fact

$$\Gamma \vdash p : A \;\Leftrightarrow\; \mathtt{type}_\Gamma(p) =_{\beta\delta\iota} A.$$

Now in principle already the complexity of $=_\beta$ is expensive. In fact it is PSPACE complete, as Statman [1979] has shown. But for terms coming from humans it turns out that this complexity upper limit is far from being used in natural situations (i.e. in from proofs given by hand).

The Poincaré Principle can be postulated for different classes $\mathcal{R}$ of rewrite relations. Below we will discuss that in some mathematical assistants (HOL) $\mathcal{R} = \emptyset$, in others (PVS) essentially one has that $\mathcal{R}$ corresponds to a large set of decision procedures.

Propositions-as-types

The already mentioned propositions-as-types interpretation was first hinted at by in Curry and Feys [1958], p. , and later described by Howard [1980]. It is related to the intuitionistic interpretation of propositions. A proposition A is interpreted as the collection (type) of its proofs $[\![A]\!] = \{p \mid p \text{ is a proof of } A\}$. Martin-Löf [1984] completed the interpretation by showing how inductive types can be used to give a very natural interpretation to this.

[37]Speaking about "2+2=4" and its proof in some logical system, Poincaré [1902], p. 12, states: *Mais interrogez un mathématicien quelconque: "Ce n'est pas une démonstration proprement dite", vous répondra-t-il,"c'est une vérification."* .

```
A× B      :=  pair:A→B→(A× B)
A+B       :=  in left:A→(A+B)|in right:B→(A+B)
∅         :=
Σx:X.B    :=  pair:Πx:X.(B→Σx:X.B)
```

Using these one has the following.

$$\begin{array}{rcl} [\![A{\to}B]\!] & := & [\![A]\!]{\to}[\![B]\!] \\ [\![A \mathbin{\&} B]\!] & := & [\![A]\!] \times [\![B]\!] \\ [\![A \vee B]\!] & := & [\![A]\!] + [\![B]\!] \\ [\![\bot]\!] & := & \emptyset \\ [\![\forall x{:}X.B]\!] & := & \Pi a{:}X.[\![B[x{:}=a]]\!] \\ [\![\exists x{:}X.B]\!] & := & \Sigma a{:}X.[\![B[x{:}=a]]\!] \end{array}$$

Predicative type theory has been argued by Martin-Löf to be the right kind of foundation.

Category theory

Category theory can be axiomatized in a two sorted predicate logic (one for objects and another for arrows) with partial terms (composition of arrows is not always defined) and equational logic. For every object A there is an arrow id(A) with dom(id(A))=cod(id(A))=A; for every arrow f there are objects dom(f) and cod(f). If f, g are arrows and cod(f)=dom(g), then $g \circ f$ (sometimes also written as $f; g$) is defined and dom$(g \circ f)$=dom(f), cod$(g \circ f)$=cod(g). The following axioms hold $f \circ (g \circ h) = (f \circ g) \circ h$, $\mathrm{id}(\mathrm{dom}(f)) \circ f = f$ and $f \circ \mathrm{id}(\mathrm{cod}(f)) = f$. In a diagram one has ($f, g, h, \ldots$ range over the arrows, $A, B, \ldots$ over the objects and $g \circ f \downarrow$ means that $g \circ f$ is defined) the following. If an equation is postulated it has to be interpreted as: if the LHS is defined then so is the RHS and conversely and in either case both are equal.

$$\begin{tikzcd} \text{Objects} \arrow[r, "\mathrm{id}"] & \text{Arrows} \arrow[l, bend right, "\mathrm{cod}"'] \arrow[l, bend left, "\mathrm{dom}"] \end{tikzcd}$$

$$\begin{array}{rcl} (g \circ f)\downarrow & \Leftrightarrow & \mathrm{cod}(f) = \mathrm{dom}(g) \\ \mathrm{dom}(g \circ f) & = & \mathrm{dom}(f) \\ \mathrm{cod}(g \circ f) & = & \mathrm{cod}(g) \end{array}$$

$$\begin{array}{rcl} f \circ (g \circ h) & = & (f \circ g) \circ h \\ \mathrm{id}(\mathrm{dom}(f)) \circ f & = & f \\ f \circ \mathrm{id}(\mathrm{cod}(f)) & = & f \end{array}$$

Figure 10: The axioms of category theory

The equational reasoning is often done in a pictorial way (using diagrams). For an implication like

$$g \circ f = k \circ h,\ \ q \circ p = r \circ k\ \Rightarrow\ r \circ g \circ f = q \circ p \circ h$$

one draws a commutative diagram:

$$\begin{tikzcd}
A \arrow[r, "h"] \arrow[d, "f"'] & B \arrow[r, "p"] \arrow[d, "k"] & E \arrow[d, "q"] \\
C \arrow[r, "g"'] & D \arrow[r, "r"'] & F
\end{tikzcd}$$

Category theory as foundation

One can view category theory as a foundation for mathematics. By imposing extra axioms, e.g. there exists only one object, the category under consideration becomes a particular mathematical structure (in this case a monoid). This method is strong enough to be a foundation for close-up mathematics. By imposing that we work in a more complex category, e.g. a topos that has a rich structure of objects and arrows, one obtains a wide-angle mathematical view with many (internal) categories in which the various mathematical structures that exists can be embedded and compared. In this respect category theory makes little ontological commitment: things are valid depending on the particular category one starts with.

The interpretation of an intuitive mathematical statement "Given situation Γ, then one has A" can be interpreted both in the way as in logic or as in type theory. In the first case the (translation of the) statement A becomes valid in a category enjoying some properties depending on Γ; in the second case one takes into account the proof p of A and validity becomes (after translation into the right *fibered* category) $p(A) = 1$, see Jacobs [1999].

For some of the categories, e.g. the effective topos and several of the fibered categories, the elements have a computational flavor. This, however, has not been exploited in a system for computer mathematics. On the other hand the functional programming language Ocaml[38], is implemented on the Categorical Abstract Machine, Cousineau, Curien and Mauny [1987], inspired by category theory. Interestingly, Ocaml is the implementation language of Coq.

A comparison between the three foundations

Categories plays a mediating role in the foundations of mathematics. Some of the type theories are quite wild, notably the higher order ones. By providing category theoretic semantics for type theory, see Jacobs [1999], some clarity can be obtained.

As the needed categories do exist within set theory, one has bridged the "ontological gap" between set theory and type theory: if one believes in set theory, one also ought to believe in type theory. Aczel argues that one should do the opposite, i.e. found the reliability of set theory on type theory. In a series of papers Aczel [1978], [1982], [1986] and [2000/2001] CZF (constructive, i.e. predicative, ZF set theory) is based on the predicative type theory of Martin-Löf

[38] `<caml.inria.fr/ocaml>`

[1984]; also (the much stronger) IZF on a type theory containing an analogue of the sub-object classifier in a topos[39].

As a foundation for mathematics category theory stands between set theory and type theory. It needs equational logic in order to deal with equality between expressions consisting of composed arrows (using composition). In this respect it is much like most type theories that also need equational logic. But in a category one sometimes wants to state e.g. the existence of a pull-back and for these one needs quantifiers. Type theory, on the other hand, does not need logic using quantifiers; these are built in.

The analogy with set theory is somewhat different. Both category and set theory use first order logic with equality. Both have enough ontological expressive force to be able to describe the necessary spaces in mathematics. There is a noticeable difference in that category theory (like group theory) is polyvalent in intension: there are many categories (and groups). Set theory has a fixed ontological commitment: it is (or at least, was originally) intended to describe the structure of the platonic universe. But it does not succeed in this, as there are many independent statements like $2^{\aleph_0} = \aleph_1$, even under the assumption of extra large cardinal axioms. Given this state of affairs, for the foundation of mathematical theorems it may be equally arbitrary to choose a model of set theory as choosing a topos. Relativity is also evidenced in type theory by its many possible choices of sorts (universes), product rules and sets $\mathcal{R}$ for which the Poincaré Principle holds. On the other hand in the predicative Martin-Löf type theory a very precise ontological commitment is made.

In set theory mathematical objects have a specific 'implementation', namely as a set. This may cause unnatural questions for example whether the real number 0 is an element of the real number π[40]. In the category and type theoretic foundations one has a structuralist point of view: one does not say what 0 or π are, only what relations they form with other mathematical objects. This is a strong point in favor of category and type theory. Therefore category theory (and one could add type theory) is sometimes said to conform to the structuralist view of mathematics; see Makkai [1999]. Also in set theory with Ur-elements like KPU, see Barwise[41] [1975], there is a more structuralist point of view.

3. Foundational views and criticism

Formalism

In section 1 it was mentioned that mathematics consists of a tower of theories with at the bottom elementary ones like geometry and arithmetic. In the higher theories notions of an infinite nature play a definite role. One considers sets like $\mathbb{N}, \mathbb{N}{\to}\mathbb{N}$ and $\mathcal{P}(\mathbb{N})$ as a completed entities, since one may quantify over them.

[39]It is open whether IZF can be interpreted in some extension of the calculus of constructions with inductive types and universes.

[40]In the system for Computer Mathematics Mizar, see section 5, the answer is "Yes", but the answer depends on the fact that π is irrational.

[41]1942-2000.

For example

$$D = \{n\in\mathbb{N} \mid \exists x, y\in\mathbb{N}.\, 7y^2 = nx^3 + 1\}$$

consists of those parameters n for which the given Diophantine equation is solvable. The existence of such sets D seems to imply that we know them for what values of the parameter n this holds[42]. But in fact such Diophantine[43] sets are in general undecidable. Even more problematic is the following notion.

$$P(n) \;\Leftrightarrow\; \forall X\subseteq\mathbb{N}.\, [n\in X \;\&\; \ldots \;\Rightarrow\; 0\in X].$$

This is called an impredicative definition, because P, essentially a subset of $\mathbb{N}$ is defined in terms of all possible subsets of $\mathbb{N}$. So in this case the property $P(n)$ depends on all possible subsets of $\mathbb{N}$, including the one determined by P.

The question arose whether the results that can be proved from these higher ("*infinitary*") concepts remain provable without them ("*by finitistic means*"). Hilbert started a program with the intention to prove this. In Simpson [1988] this is described as follows. Hilbert was not precise as to what is a theory with infinitary character and what is a finitary theory. Simpson argues convincingly that second order arithmetic Z_2 and primitive recursive arithmetic PRA may be seen as a good operationalizations of these concepts respectively. Indeed most of mathematics may be formalized in Z_2, while PRA is definitely an innocent theory, as it has no explicit quantifiers. In this view Hilbert's program wants to establish that statements in PRA provable in Z_2 are already provable in PRA. This is called 'conservativity' (of Z_2 over PRA). The way Hilbert wanted to achieve this is nowadays called reflection. By seeing provability as a game with symbols one could describe it within PRA and then establish mentioned conservativity[44].

As Hilbert's program was the first serious attempt of reflection via a formal description of mathematics, he was said to adhere to the formalist philosophy: mathematics devoid of any meaning. I think this is unfair, as Hilbert was very much aware of the meaning of finitistic mathematics and just wanted to establish the same for "higher" mathematics.

Logicism

Whereas in formalism mathematics is reduced to a game with symbols, in logicism the rules of logic are seen as having a meaning and being valid. According to this view mathematics developed using the axiomatic method depends on logic. This seems plausible, as (first order) logic proves all statements that hold in all models of the axioms (completeness, see Gödel [1930], Skolem [1922]).

[42]Looking for $n\in D$ one has at least as possibilities $6, 27, 62, 111, 174, \ldots, 7k^2 - 1, \ldots$ (take $x = 1$), and also $1, 162$, 218, 701 and 813 (take $x = 3$), but it is not immediate whether there are infinitely many more solutions.

[43]After Diophantos of Alexandria (app. 200-284 AD)

[44]Gödel's famous incompleteness result showed that conservativity does not hold. But Simpson argues that Hilbert's program is partly successful. He estimates that about 85% of mathematics can be proved in the system PRA+WKL_0^+, where WKL_0^+ is some form of König's Lemma stating that a finitely branching infinite tree has an infinite path, and this system *is* conservative over PRA.

Traditionally Frege and Russell are considered as logicists. When Frege had completed the quest for logic initiated by Aristotle, he started to formalize mathematics within his system. For this he had to use some ontology and his choice here was Cantorian set theory that he gave a particular logical basis. The axioms of set existence, mentioned in section 2, all could be postulated by the comprehension axiom:

$$\exists x \forall y.[y{\in}x \leftrightarrow P(y)],$$

for arbitrary formulas $P(y)$. Frege's system was shown to be inconsistent by Russell, who together with Whitehead (1861-1947) started a different formal system, some form of Type Theory, in which (parts of) mathematics could be formalized. Although the logical aspects of this system where somewhat sloppy (they failed to treat the difference between free and bound variables, causing the reader to make systematic disambiguations), they succeeded for the first time in history to provide a formalized version of number theory. So actually Russell and Whitehead (and also Frege) could have been called formalists as well. The impact of Principia Mathematica was not mathematical, but meta-mathematical. Since the collection of provable statements in arithmetic in the context of Principia became a well-defined notion, one could wonder whether it was complete (for each numerical sentence A one has that either A or $\neg A$ is provable) or decidable (a machine could decide whether A is provable). Both questions turned out to have negative answers as we learned from Gödel on the one hand and Church and Turing on the other hand. The Gödel incompleteness results in this respect should be seen as a limitation of the axiomatic method (but we do not have something better).

Serious arguments against logicism came from Poincaré and Skolem. In order to prove that $2 + 2 = 4$ one needs too many steps in a logicistic system like Peano arithmetic formulated in first order predicate logic. As mentioned before, Poincaré stated that this is "just a verification." The Poincaré Principle in type theory aims to overcome this.

Platonism

In Platonism the mathematical objects are taken for real. Logic is only a way to reveal some of the truths about these real objects. Set Theory is a theory that describes these objects. Mathematics is a walk through the paradise of Cantor. One sees better and better what theorems are valid in this world. One consequence of the belief in the existence of actual infinity is the belief in the following principle[45]:

$$\neg\neg\exists x{:}\mathbb{N}.P(x) \;\Rightarrow\; \exists x{:}\mathbb{N}.P(x).$$

Indeed, assume $\neg\neg\exists x{:}\mathbb{N}.P(x)$. If $\mathbb{N}$ exists as a totality, and if P is a well-defined property over $\mathbb{N}$ that can be represented as a subset, then one just needs to see whether there is an element of this subset. If there is none, then we obtain a contradiction. So there must be an element! Of course, this reasoning is circular,

[45]For decidable P this is called *Markov's Principle*, after A.A. Markov Jr. (1903-1979).

as it depends on the double negation law (closely related to the excluded third), that we basically want to prove. But the reasoning shows how compelling it is.

Criticism against the Platonist view has been formulated by Feferman in a series of papers collected in Feferman [1998]

(i) abstract entities are assumed to exist independently of any means of human definition or construction;

(ii) classical reasoning (leading to nonconstructive existence results) is admitted, since the statements of set theory are supposed to be about such an independently existing reality and thus have a determinate truth value (true or false);

(iii) completed infinite totalities and, in particular, the totality of all subsets of any infinite set are assumed to exist;

(iv) in consequence of (iii) and the Axiom of Separation, impredicative definitions of sets are routinely admitted;

(v) the Axiom of Choice is assumed in order to carry through the Cantorian theory of transfinite cardinals.

Mostowski [1968] once said: "Peano arithmetic is probably consistent. Impredicative set theory may very well be inconsistent. In that case the inconsistency may already be present in second order arithmetic. Also the notions of forcing play already a role in this theory. That is why I like to work in this field[46]." One should add, that in this respect Girard's system $\lambda 2$, see Girard, Taylor and Lafont [1989] where it is called 'System F', is very interesting. Its strong normalization is equivalent to the consistency of second order arithmetic and can be proved using impredicative methods. But since these principles are dubious, the strong normalization is not so reliable as that of the system T of Gödel, see Troelstra [1973]. In fact, for the system $\lambda *$ (the PTS with $* : *$ and $(*, *)$) strong normalization was proved using the methods of the system itself, while the system is not strongly normalizing, see Hurkens [1995] simplifying a proof of Girard.

One of the advantages of (impredicative) set theory is that it is so strong that it is able to embed all results from most other foundations. This gives mathematics a unity, as has been emphasized by Girard. (The exception consists of those parts of Intuitionism depending on continuity principles and other non-classical statements to be discussed below. But there are classical interpretations of those parts of intuitionism.)

Calculism

In Calculism the emphasis is put on computing. At first there was the overly optimistic belief in the power of computing by Leibniz as discussed above. Even

[46]Later he added: "But when the secret police comes to ask on what grounds I choose my object of research, then I tell them: 'I study it, because the Americans study it'; and the next year I say: 'I study it, because the Russians study it'!"

at the dawn of the twentieth century Hilbert believed that e.g. solvability of Diophantine equations could be decided in a computable way (In Hilbert [1901-1902] his 10-th problem was to establish an algorithm to do just that; but as we mentioned above this is impossible.) After the notion of general computability had been captured, it was proved by Church [1936] and Turing [1936] that validity or provability of many mathematical questions was not decidable. Certain important theories are decidable, though. Tarski [1951] showed that the theory of real closed fields (and hence elementary geometry) is decidable. An essential improvement was given by Collins [1975]. In Buchberger [1965] a method to decide membership of finitely generated ideals in certain polynomial rings was developed. For polynomials over $\mathbb{R}$ this can be done also by the Tarski-Collins method, but much less efficiently so. Moreover, "Buchberger's algorithm" was optimized by several people, e.g. Bachmair and Ganzinger [1994]. It has impressively many applications ranging from the fields of robotics to differential equations, see Buchberger and Winkler [1998], Saito, Sturmfels and Takayama [2000][47]. Another use of calculism is the automated theorem prover for geometry, see Chou [1988], based on translating putative theorems into algebra, where they are decided by manipulating polynomial inequalities. See also Börger, Grädel and Gurevich [2001] for other examples of decidable theories. It is the success of calculism, even if partial, that has been neglected in mathematics based on set theory or logic alone.

Intuitionism

Intuitionism is a foundational view initiated by Brouwer[48] [1908]; see van Dalen [2000] and Troelstra and van Dalen [1988] for later developments. It proposes to use a sharper language than in Classical Mathematics. Brouwer's observation was the following. If one claims that $A \vee B$ holds one also wants to be able indicate which one is the case. Similarly, if $\exists x.A$ holds, then one wants to be able to find a 'witness' and a proof of $A[x := a]$. In classical mathematics, based on Aristotelian logic, this is not always the case. Take for example the arithmetic statement (where GC is an open problem like the Goldbach Conjecture)

$$P(x) \;\Leftrightarrow\; (x = \underline{0} \;\&\; \mathrm{GC}) \vee (x = \underline{1} \;\&\; \neg\mathrm{GC}).$$

Now one can prove $\vdash \exists x.P(x)$ (indeed if GC holds take $x = 0$ else $x = 1$), without having $\vdash P(\underline{0})$ or $\vdash P(\underline{1})$ because the GC is still open (and certainly one has $\not\vdash P(\underline{n})$ for $n > 1$). Similarly one has $\vdash \mathrm{GC} \vee \neg\mathrm{GC}$, without having $\vdash \mathrm{GC}$ or $\vdash \neg\mathrm{GC}$. One may object that sooner or later GC may be settled. But then one can take instead of GC an independent Gödel sentence that for sure will not be settled (if arithmetic is consistent). Brouwer analyzed that this imperfection was caused by the law of excluded middle $A \vee \neg A$. Heyting formulated a logical system (intuitionistic[49] predicate logic) that remedied this

[47]The Buchberger algorithm and that of Knuth and Bendix [1970] are closely related, see Middeldorp and Starčević [1991] and Marché [1998].

[48]1881-1966.

[49]It is interesting that there is a set theoretic semantics of intuitionistic propositional logic comparable to that of the classical version. The latter theory can be interpreted in Boolean

effect, as proved by Gentzen (who also gave intuitionistic logic a nicer form: see the system in fig. 2 leaving out the double negation rule from classical logic).

Another criticism of Brouwer (against the logicistic view this time) is that logic does not precede mathematics. For example if one wants to formulate logic, one needs to define the context free language of formulas. This criticism has been dealt with somewhat in type theory where next to the logical axioms there are axioms concerning data types.

The intuitionistic school of mathematics at first did not gain much interest. One of the reasons was that mathematics had to be reproved and possibly modified. Now that this work has obtained a serious start one collects the fruits. In set theory a theorem like

$$\forall n{\in}\mathbb{N}\exists m{\in}\mathbb{N}.P(n,m) \tag{1}$$

does not imply that the m can be found computably form n. If one wants to express this computability, then it is not even enough to state and prove

$$\exists f \text{ computable } \forall n{\in}\mathbb{N}.P(n,f(n)),$$

as the $\exists f$ may not lead to a witness for a computable function. The only way to state that in (1) the m is computable in n is to actually give the algorithm, which in set theory is not very practical. In intuitionistic mathematics provability of (1) automatically implies computability. And if computability does not hold one can reformulate (1) as

$$\forall n{\in}\mathbb{N}\neg\neg\exists m{\in}\mathbb{N}.P(n,m) \qquad (2).$$

For these reasons Constable [1997] stated "Intuitionism nowadays has become technology". A challenging subject is to extract programs from fully formalized $\forall\exists$ statements, see Paulin-Mohring [1989], Letouzey [2003]. Although this is possible in principle, there is space for optimizations. As pointed out by Kreisel [1985], see also Schwichtenberg [2002] and Cruz-Filipe and Spitters [2003], the information of proofs of negative statements is irrelevant, so that these need to be discarded. Moreover, in several cases classical proofs can be transformed into intuitionistic proofs (for example if the statement is an arithmetic Π^0_2 statement, see Friedman [1978], Coquand and Herbelin [1994]) and widens the scope of the extraction technology, see Berger, Buchholz and Schwichtenberg [2000,2001].

Constructivism vs intuitionism

Brouwer not only criticized the double negation law, he also stated principles that contradict it, for example that all functions $\mathbb{R}{\to}\mathbb{R}$ are continuous[50], see

algebras with as prototype subsets of a given set. The intuitionistic theory as Heyting algebras with as prototype the open subsets of a topological space X. Negation is interpreted as taking the interior of the complement, disjunction as union. And indeed in general one does for $A \subseteq X$ that $A \cup \overline{A}^o \neq X$. Therefore the law of the excluded middle fails in this model. See Rasiowa and Sikorski [1963].

[50]A classical function that contradicts this is the step function $s(x)$ that is 0 for $x < 0$ and 1 otherwise. But intuitionistically s is not definable as total function, as one cannot determine from e.g. a Cauchy sequence whether its limit is < 0 or ≥ 0.

see Troelstra and van Dalen [1988], Ch. 4. Constructivism consist of the part of intuitionism by just leaving out the double negation law, see Bishop and Bridges [1985], Mines, Richman and Ruitenburg [1988]. Although it seems daring to state axioms contradicting classical mathematics, one should realize that with some effort much of classical mathematics can be reformulated in a way such that it becomes constructively valid. This means that there is place for extensions like the continuity theorem. The strong intutionistic principles then can be seen as a welcome additional power. For those functions that are provably total, one can show that they are continuous indeed. This was how Bishop (1928-1983) understood Brouwer's explanation of these axioms, see Bishop [1970]. Finally it should be observed that in mathematical models occurring in physics, e.g. microelectronics, all total discontinuous functions like 'square waves' are just a *façon de parler.*

4. Computer Mathematics

Computer Mathematics (CM) is mathematics done by humans and computers in collaboration. The human gives the development of a theory: definitions and theorems. The computer checks whether the definitions and statements of theorems are well formed; then in an interactive fashion some or all steps of the proof are given; finally the computer checks their correctness. One purpose of CM is to assist with teaching existing mathematics and to develop" new mathematics. Another purpose is to reach the highest degree of reliability. Last but not least, through CM it will be possible to have a certified library of theories, ready for reuse and exchange, see Cohen [2001], Barendregt and Cohen [2001]. At present CM is not yet established, but forms an interesting challenge. See Beeson [2003] for a stimulating discussion of the subject, with examples not covered in this paper[51].

One part of CM is Computer Algebra (CA). It deals with 'computable' objects, often in an equational way. This by now is an established, though developing, subject. Mathematical Assistants deal with side conditions of equations and more general with reasoning that cannot be formulated in CA.

Computer Algebra

Systems of CA, like the commercial packages Maple and Mathematica or the systems more directed toward mathematical research like Gap, Li, Magma and Pari, all represent "computational" mathematical objects and helps the human user to manipulate these. Computational objects not only consists of numbers and polynomials, but also of expressions built up from roots, integrals, transcendental functions, for example the elliptic integral of the first kind

$$f(\alpha) = \int_0^\alpha \frac{1}{\sqrt{1 - \frac{1}{4}\sin^2\varphi}}\,d\varphi.$$

[51]This paper went to press with some delay. In the meantime Georges Gonthier [2005] had established a fully formalized proof of the four colour theorem that was verified by the assistant Coq.

The more advanced systems (like Magma) represent groups, e.g. $\mathrm{Aut}_{\mathbb{F}_7}(C)$, with C being the hyper elliptic curve $\{\langle x,y\rangle \in \mathbb{F}_7^2 \mid y^2 = x^4 + 4\}$ over the finite field with 7 elements[52]. That it is possible to represent on a computer an object like $\sqrt{2}$, that has infinitely many digits in its decimal representation, follows from the fact that it can be represented by a single symbol, but we know how to manipulate these symbols. For this reason we call mentioned objects computable.

Mathematical Assistants

In systems of Computer Mathematics one even can represent arbitrary mathematical notions. Moreover, the systems that can handle these, the mathematical assistants, help the human user to define new mathematical notions and make deductions with them. The reason that the constraint of computability now can be dropped is the following. Even if for a property P and object c it may be undecidable whether $P(c)$ holds, it is decidable whether a putative proof p of this statement is a valid proof indeed.

Formal systems

There is a choice of formal system in which mathematics is being represented. Frege made a start, but when formalizing Cantorian set theory in (his) predicate logic, the system unfortunately became inconsistent as shown by Russell through his paradox. Then Russell and Whitehead chose a form of type theory and made a reasonable start. Their description of the theory lacks rigor though[53]. Curry [1930] worked with extensions of the untyped lambda calculus, but suffered from either weakness or inconsistencies of the systems involved[54]. Church [1940] introduced the theory of simple types that is the basis of the mathematical assistant HOL.

McCarthy [1962] made a plea for formalization together with a computer verification, using first order logic. He did not get far, because of the lack of force of this logic (one can represent the close-up theory of statements valid in all groups, but not the wide-angle theory of groups, unless one formalizes set theory) and because of his proposal to represent proofs in the Hilbert way (a sequence of formulas that either are axioms or follow from previous formulas) was cumbersome. But McCarthy had good ideas about combining formal proofs and symbolic computations in the Babylonian style.

Important progress came from de Bruijn [1970], see for a survey Nederpelt, Geuvers and de Vrijer [1994], with his family of Automath languages and corresponding proof-checkers. These languages all are based on some form of type theory extended by the dependent types already mentioned. The admittedly

[52]It writes this group as a product of simpler groups and tells us how the elements act on the points of the curve C.

[53]For example free and bound variables are used in such a way that the reader has to insert in many places a binder. See Laan [1997].

[54]Only in the 1990's adequate systems of *Illative Combinatory System* have been given, see Barendregt, Bunder and Dekkers [1993], Dekkers, Bunder and Barendregt [1998]. In some sense these are simpler than the PTSs, in another sense they are more complicated.

already quite formal "Grundlagen der Analysis" by Landau[55] [1960] has been formalized in AUT-68, a member of the Automath family, by van Benthem Jutting [1977] and exactly one error was found in it. It was emphasized by de Bruijn emphasized that a mechanical proof-checker should have a small program (so that it can be seen "by hand" to be correct), otherwise the correctness of the verification becomes a point. This is called the *de Bruijn criterion*.

Poincaré Principle

Another 'parameter' relevant for Mathematical Assistants is the way in which calculations are supported. If the formal system has as derivation rule

$$\frac{A(f(t))}{A(s)}\, f(t) = s,$$

then we say that it satisfies the Poincaré Principle (PP) for f. The class $\mathcal{P}$ of functions for which the Poincaré Principle holds varies over the formal systems for CM. If $\mathcal{P} = \emptyset$, then formal proofs involving computations become quite long. We discussed that Poincaré and Skolem criticized logicism for this reason. In that case the proof-objects become so large (they essentially contain the traces of necessary computations) that they will not be stored. The way these will be checked is still quite reliable. The proofs are being represented bit by bit in a working memory and local correctness is checked. As soon as a part turns out to be correct it is being erased. We speak about *ephemeral proof-objects*. For these systems only the proof-script that generates the ephemeral proof-object will be stored. In Pollack [1995] a stronger proposal is made by viewing decision methods as admissible rules with (semi) decidable side conditions.

Reflection

The method of reflection, that had its applications in projective geometry, metamathematics, set theory, model theory and category theory, also becomes important in computer mathematics in order to provide formal proofs for statements otherwise obtained by intuition or computation. A particularly fruitful use of reflection is as follows. If we want to prove a property $A(t)$, where A is some predicate and t is some term, then sometimes the method of *generalization* simplifies matters: one first proves $\forall x.A(x)$ in order to conclude $A(t)$. The method of pattern generalization is more useful. If we want to prove $A(t)$, then we can often write $t = f(s)$ and prove also $\forall x.A(f(x))$. As soon as we can prove $t = f(s)$ (employing the Poincaré Principle or using ephemeral proofs) we are done. The terms s are often of a syntactical nature and the map f involves semantic interpretation $s \mapsto [\![s]\!]$. An example of this use of reflection is the following. In order to prove that the elliptic integral $f(\alpha)$ defined above is continuous in α an attentive student can see this immediately from the defining expression. If the expressions become more complex it is a burden to provide formal proofs of these facts. It can, however, be done in a light way,

[55] 1877-1938.

closely following our intuition. This is done by introducing a formal language L containing expressions for functions like f, together with an interpretation function $[\![\ ]\!]$ transforming this expression in the intended actual function. One only needs to prove once and for all

$$\forall e{:}L.[\![e]\!] \text{ is continuous}$$

and then one can apply this to the *quote* of f. For this a provable computation is needed to show $f = [\![\text{quote } f]\!]$, but that can be done either via the Poincaré Principle or ephemeral proofs. In a similar way a computational statement like

$$\begin{aligned}(xy - x^2 + y^2)(x^3 - y^3 + z^3) &= x^4y - xy^4 + xyz^3 - x^5 + \\ &\quad x^2y^3 - x^2z^3 + y^2x^3 - y^5 + y^2z^3.\end{aligned}$$

can be proved by reflection and primitive recursive computation. See Barendregt and Barendsen [2002] for more details. The first place where reflection occurred in proof-assistants is in Howe [1992]. In [1996], Barras [1999], the kernel of Coq has been reflected in Coq itself.

Systems

Various systems evolved. In most of them the human user constructs the formal proof (the so called *proof-object*) assisted by the computer, by interactively writing a *proof-script*. The resulting proof-object will be verified by the small proof checker. The reason for the requirement that the system be small, the so-called *de Bruijn criterion*, is that even if a formal proof (a so called *proof-object*) is huge, it is reliable, because we are able to check the correctness of the software that performs the verification by hand. Even so, several of the kernels of the present systems of Computer Verification have had bugs. These were caused by logical inconsistencies, faulty module systems, or more technical defects[56].

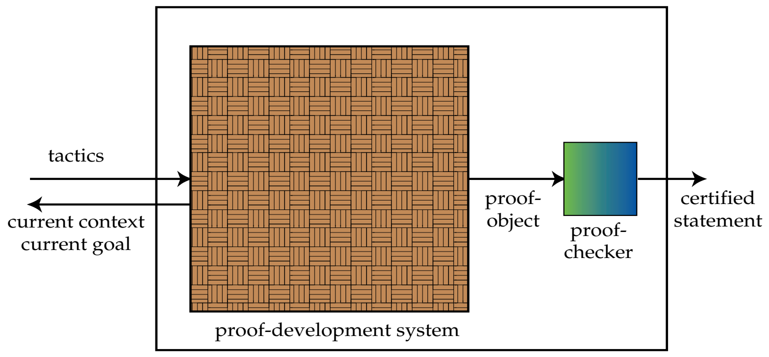

Figure 11: A Mathematical Assistant

Some of the systems come with a fixed foundational formal system. Other ones are *logical frameworks*, in which the foundational system may be specified and used.

[56]The mechanism of α-conversion (changing names of bound variables) is often implemented wrongly. In Gabbay and Pitts [2001] this mechanism is described in a very succinct way so that it will be helpful if implementers use this as foundation of their implementation.

The main systems for CM in which some substantial theories (i.e. on the order of >10 Megabytes, but still small from a working mathematician's point of view) have been represented are the following.

1. HOL[57] and Isabelle[58];
2. Coq[59] and Nuprl[60];
3. Mizar[61];
4. PVS[62].

Many other systems can be found at Freek Wiedijk's homepage[63].

HOL is based on higher-order logic. Its class of functions satisfying the Poincaré Principle is empty. Therefore the proof-objects are huge and therefore made ephemeral and the proof-scripts are being stored. Isabelle is a logical framework that has HOL as one of its main implementations. For this reason the two systems are listed together.

Coq is based on higher order intuitionistic intensional type theory with the Poincaré Principle for $\beta\delta\iota$-reduction. Nuprl is similar, but based on extensional type theory, as discussed in section 2. This has the advantage that subtypes and quotient types can be represented in a natural way and the disadvantage that belonging to a type requires a proof obligation. Mizar is based on ZF with choice and some large cardinal assumptions. It has almost no Poincaré Principle and cannot do interesting computations (there is no support for ephemeral proofs). Also the attempt of Bourbaki to found mathematics on set theory suffered from this and eventually it did lead to the termination of that project. PVS is based on primitive recursive arithmetic and the corresponding Poincaré Principle and is userfriendly for not too abstract mathematics. One nice feature of Mizar is that it has a *mathematical mode*: proofs are relatively close to the informal proofs. This feature has been added to a variant of Isabelle, Isar, and is presently in consideration for Coq, see Barendregt [2003].

Case studies in Coq include a constructive development of mathematical analysis including the Fundamental Theorems of Algebra and Analysis, the correctness of an algorithm for obtaining Gröbner bases and the FFT, the prime number theorem Avigad, Donnely, Gray and Raff [2005] and the Four Color Theorem Gonthier [2005]. Moreover primality of 40 digit numbers has been established by Pocklington's criterion and using the Computer Algebra System Gap for the factorization and congruences (Gap gave input that has been fully verified in Coq).

[57]`<www.cl.cam.ac.uk/Research/HVG/HOL>`

[58]`<www.cl.cam.ac.uk/Research/HVG/Isabelle>`

[59]`<pauillac.inria.fr/coq>`

[60]`<www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html>`

[61]`<www.mizar.org>`

[62]`<pvs.csl.sri.com>`

[63]`<www.cs.kun.nl/~freek/digimath/index.html>`

5. Foundations from a Computer Mathematics perspective

There is a view of Randy Pollack [1994] on formalism, logicism and intuitionism, that gives them a different perspective. These three philosophies correspond to the way mathematics is represented in a Mathematical Assistant and depend on which side(s) of the triangle obtains received ample attention in the basic checker while mathematical activities the other sides have to be provided by the person who formalizes mathematical texts.

Formalism

If the basic program is such that all aspects of mathematics (defining, reasoning and computing) have to be programmed into the underlying proof assistant, then one may speak of a mathematical assistant in the formalist style. Examples of such assistants are the original Automath system and Isabelle[64] in which simple things like implication have to be programmed by the user. The following is a piece of text in AUT-68.

```
1.  BOOLEANS

1.1         @   bool         :=  PN                           :   'type'
1.2         @   x            :=  ---                          :   bool
1.3   x     @   TRUE         :=  PN                           :   'type'
1.4         @   CONTR        :=  [v:bool]TRUE(v)              :   'type'
1.5         @   a            :=  ---                          :   CONTR
1.6   a     @   b            :=  ---                          :   bool
1.7   b     @   then_1       :=  <b>a                         :   TRUE(b)
1.8         @   ksi          :=  ---                          :   'type'
1.9   ksi   @   nonempty     :=  PN                           :   bool
1.10  ksi   @   a            :=  ---                          :   ksi
1.11  a     @   then_2       :=  PN                           :   TRUE(nonempty)
1.12  ksi   @   a            :=  ---                          :   TRUE(nonempty)
1.13  a     @   then_3       :=  PN                           :   ksi
1.14  ksi   @   EMPTY        :=  [u:ksi]CONTR                 :   'type'
1.15  ksi   @   x            :=  ---                          :   ksi
1.16  x     @   u            :=  ---                          :   EMPTY(ksi)
1.17  u     @   then_4       :=  <x>u                         :   CONTR
1.18  x     @   then_5       :=  [t:EMPTY(ksi)] then_4(t)     :   EMPTY(EMPTY(ksi)
1.19  ksi   @   PARADISE_II  :=  [t:EMPTY(EMPTY(ksi))] ksi    :   'type'
```

Figure 12: Booleans in AUT-68

Logicism

The following contains a proof in HOL of Peirce's law (valid in classical logic)

$$((A \to B) \to A) \to A).$$

The proof is immediate because classical logic is wired in. Also a proof of $- - (x + y) = (- - x) + (- - y)$ is displayed. As this proof requires some calculation and HOL does not have a Poincaré Principle, there are procedures

[64]This system is a so-called logical framework, i.e. designed in order to represent arbitrary concrete formal systems, even ones that are meaningless.

outside HOL that *generate* the ephemeral proof, which is checked on the fly in all details.

```
let PEIRCE = prove
    (‘((A ==> B) ==> A) ==> A’,
      ASM_CASES_TAC ‘A:bool’ THEN ASM_REWRITE_TAC[ ]);;

let REAL_NEG_ADD = prove
 (‘!x y. --(x + y) = --x + --y‘,
  REPEAT GEN_TAC THEN
  MATCH_MP_TAC(GEN_ALL(fst(EQ_IMP_RULE(SPEC_ALL REAL_EQ_ADD_RCANCEL))))
  THEN EXISTS_TAC ‘x + y‘ THEN REWRITE_TAC[REAL_ADD_LINV] THEN
  ONCE_REWRITE_TAC[AC REAL_ADD_AC
                  ‘(a + b) + (c + d) = (a + c) + (b + d)‘]
  THEN REWRITE_TAC[REAL_ADD_LINV; REAL_ADD_LID]);;
```

Figure 13: Two simple proof scripts in HOL

Platonism

In Mizar the Platonist view is followed: ZFC is built into the system. There is just a little bit of Poincaré Principle (for functions like addition) and there is no procedure to check ephemeral proofs, so formalizing in this system is comparable to writing Bourbaki style mathematics. On the other hand the system has just enough automated theorem proving that it allows a *mathematical mode*: a mathematician friendly way of interacting with the system[65]. The following example is a formulation and proof of Tarski's theorem.

THEOREM. *Let A be a set and let $f:\mathcal{P}(A)\to\mathcal{P}(A)$ be such that*

$$\forall X,Y{\in}\mathcal{P}(A).X\subseteq Y \;\Rightarrow\; f(X)\subseteq f(Y).$$

Then $\exists Z{\in}\mathcal{P}(A).f(Z)=Z$.

The reader will be able to follow the proof, without any knowledge of the Mizar system (the symbol ‘c=’ stands for $\subseteq$ and expressions like `ZFMISC_1:92` refer to previously proved results).

[65]In `<isabelle.in.tum.de/Isar>` there has been developed also mathematical mode for Isabelle. One for Coq is being considered, see Barendregt [2003].

```
begin
 reserve A for set;
 reserve X,Y for Subset of A;
 reserve F for Function of bool A,bool A;

theorem
 for F st for X,Y st X c= Y holds F.X c= F.Y ex X st F.X = X
proof
 let F;
 assume
A1: for X,Y st X c= Y holds F.X c= F.Y;
 consider P being Subset-Family of A such that
A2: for Y holds Y in P iff Y c= F.Y from SubFamEx;
 set X = union P;
 take X;
 for Y being set st Y in P holds Y c= F.X
 proof
  let Y be set;
  assume
A3: Y in P;
  then reconsider Y as Subset of A;
  Y c= F.Y & Y c= X by A2,A3,ZFMISC_1:92;
  hence thesis by A1;
 end;
 then
A4: X c= F.X by ZFMISC_1:94;
 then F.X c= F.(F.X) by A1;
 then F.X in P by A2;
 then F.X c= X by ZFMISC_1:92;
 hence F.X = X by A4,XBOOLE_0:def 10;
end;
```

Figure 14: Tarski's theorem in Mizar

Calculism

The system PVS has many decision methods built in. This may be seen as a rich form of the Poincaré Principle:

$$\frac{\text{true}}{A}\ A =_R \text{true}$$

where R is a particular decision method.

In the following proof script of PVS it is seen that the system can deal with reasoning with inequalities, showing that in the real numbers the following is valid:

$$\begin{array}{ll}
\forall x, x_1, x_2, x_3 \in \mathbb{R}. & x^2 \geq 0 \\
& \neg(x_1 * x_2 = 5 \ \&\ x_1 = 4 \ \&\ x_2 = 2) \\
& \neg(x_1 * x_2^2 = 5 \ \&\ x_1 * x_2^2 = 5 \ \&\ x_2 = 2) \\
& \neg(x_1^2 * x_2 = 5 \ \&\ x_1 * x_2 = 5 \ \&\ x_2 = 4) \\
& \neg(x_1 * x_2^2 = 5 \ \&\ x_1 * x_2 = 5 \ \&\ x_2 = 2) \\
& \neg(x_1 + 5 * x_2^2 + 20 * x_2 = 0 \ \&\ x_1 > 0 \ \&\ x_2 > 0).
\end{array}$$

This is something that many other systems are not good at.

```
arith: THEORY
BEGIN

  x, x1, x2, x3, x4: VAR real

  Test1: FORMULA
    x * x >= 0

  Test2: LEMMA
    x1 * x2 = 5 and x1 = 4 and x2 = 2 IMPLIES FALSE

  Test3: LEMMA
    x1*x2 = 5 AND x1*x2*x2 = 5 AND x2 = 2 IMPLIES FALSE

  Test4: LEMMA
    x1*x2*x2 = 5 AND  x1*x2 = 5 AND  x1 = 4 IMPLIES FALSE

  Test5: LEMMA
    x1*x2*x2 = 5 AND  x1*x1*x2 = 2 AND  x2 = 2 IMPLIES FALSE

  Test6: LEMMA
    NOT ( x_1 + 5 *x_2 * x_2 + 20*x_2 = 0 AND x_1 > 0 AND x_2 > 0)

END arith
```

Figure 15: A PVS verification based on decision methods

Intuitionism

Division with remainder states the following. Let $d \in \mathbb{N}$ with $0 < d$. Then

$$\forall n \in \mathbb{N} \exists q, r [r < d \;\&\; n = qd + r].$$

This can be proved constructively. From this proof one can automatically derive an algorithm that does the (round off) division. This is done by the (trivial[66]) operation '`choose`' that assigns to a proof of an exists statement a witness. The following is part of a Coq development. The extracted code for dividing is not efficient, better extraction mechanisms and proofs will help, but it plays an explanatory service. The proof, checkable in Coq provided that a module math-mode is included. Boh have been developed by Mariusz Giero and are inspired by Mizar and Barendregt [2003].

[66]The triviality is caused because of the way in an intuitionistic system a proof of a statement $\exists x.P$ is coded as a pair $\langle a, p \rangle$, where p is a proof of $P[x: = a]$.

```
Lemma Euclid : (d:nat)(O<d)->
(n:nat)(EX q:nat|(([q:nat](EX r:nat|((r<d)/\n=((d[x]q)[+]r))))q)).
Proof.
Let_ d be nat. Assume
               (O < d)                                 (A4).
LetTac P:=[n:nat]((EX q:nat|(EX r:nat|((r<d)/\n=((d[x]q)[+]r))))).
Claim
               ((n:nat)(before n P)->(P n))            (A1).
  Let_ n be nat. Assume
               (before n P)                            (A6).
  We need to prove (P n).
  Case 1 (n<d) (A2).
    Take zero and prove (EX r:nat|r<d/\n=d[x]zero[+]r).
    Take n and prove (n<d/\n=d[x]zero[+]n).
    As to (n<d) [by A2].
    Also (n=d[x]zero[+]n) [by times_com].
  Case 2 (n>=d) (A5).
    Claim ( (n-,d) <n).
      Have (O<n) [by A4, A5, lt_le_imp_lt].
    Hence claim done [by A4, pos_imp_mon_lt].
    Then (P (n-,d)) [by A6].
    Then consider q such that
       ([q:nat](EX r:nat|r<d/\n-,d=d[x]q[+]r)).
    Then consider r such that
      ([r:nat](r<d/\n-,d=d[x]q[+]r)) (A8).
    Take (S q) and prove (EX r:nat|r<d/\n=d[x](S q)[+]r).
    Take r and prove (r<d/\n=d[x](S q)[+]r).
    As to (r<d) [by A8].
    Now n=d[x](S q)[+]r).
    We have n  =  ((n-,d)[+]d) [by ge_imp_mon_plus_eq, A5] (Z1).
              _= (d[x]q[+]r[+]d) [by A8].
              _= (d[x](S q)[+]r)  [by compute].
    Hence done.
Hence (P n) [by dichotomy].
So we have proved (A1).
Finally we need to prove ((n:nat)(P n)).
Done [by cv_ind, A1].
Qed.
```

Figure 16: Euclidean division with remainder in Coq

This enables one to extract an algorithm for obtaining the quotient (and the remainder as well).

```
Definition choose [A:Set][P:A->Set][p:(EX a:A|(P a))]:A:=
                  Cases p of
                  (ex_intro a _) => a end.

Definition testimony [A:Set][P:A->Set][p:(EX a:A|(P a))]:
                    (P (choose ? ? p)) :=
              (ex_rec A [a:A](P a)[q:(EX a:A|(P a))](P(choose A P q))
              [a:A; pa:(P a)]pa p).

Definition quotient[d:nat][pos:(O<d)][n:nat] : nat :=
(choose nat ([q:nat](EX r:nat|((r<d)/\(n=(d[x]q[+]r)))))
(Euclid d pos n)).
```

Figure 17: An extracted algorithm for obtaining the quotient

Here `[x:A]B` stands for λ`x:A.B` and `(x:A)B` for Π`x:A.B`. Notice that `quotient 13 3` requires an additional argument that the divisor 3 is positive (i.e. not zero).

6. Discussion

This section really should be called: "opinions". We feel that most of the -isms are overly emphasizing a particular aspect of the mathematical endeavor. At some level, mathematics is indeed a meaningless game with symbols, and although that is not a particularly fruitful view, for the implementation of the first proof-checkers it was. At some level mathematics consists of going from axioms to theorems, following logical rules. Again one forgets one aspect, the computations. Computations alone will not do, as there are many undecidable statements that are provably correct. Considering the Mathematical Universe as a fixed entity gives the working mathematician a strong drive, but one forgets that some properties require a lot of energy to find out (sometimes infinitely much, i.e. one cannot do it). Systems using formal intuitionism for computer mathematics, like Coq and Nuprl have found the right middle way. On the other hand, if intuitionism is considered as a philosophy that states that mathematics only exists in the human mind, one would limit oneself to what may be called in a couple of decades 'pre-historic'[67] mathematics. True, the theories that can be fully run through in our mind constitutes *romantic mathematics*. But the expected results fully checked by computers that have been checked (by computers that have been checked)^n by us will be *cool mathematics*. One does not want this chain to be long (as there is a possibility for erorrors at each relais). A compiled version of Coq (not satisfying the de Bruijn criterion) optimizes the proof of the four color theorem. But this compiled version has been shown correct by using the interpreted version of Coq, that does satisfy the de Bruijn crtiterion. So this is a useful chain of length 2. (The four color theorem has also been verified in the original version of Coq.)

In some sense the five small examples of a formalized proposition are somewhat disappointing: they are all similar. What seems worse, most examples

[67]Expression comes from Zeilberger [2002].

can in essence be run also on the other systems. But I see this as good news. One has found the right way to implement what is needed for a foundation of mathematics. What is lacking in most systems, though, is userfriendliness.

Astronomy and biology have also had their romantic phase of going out in the fields and studying butterflies, plants and stars. The biologist at first could see everything with the naked eye. Then came the phase of the microscope. At present biologists use EM (electro-microscopy) or computers (the latter e.g. for gene sequencing). Very cool. The early astronomers could study the planets with the naked eye. Galileo started using a telescope and found the moons of Jupiter and mountains on the earth's moon. Nowadays there are sophisticated tools for observations from satellites. Again, very cool. Still, even today both biology and astronomy remain romantic subjects, albeit in a different way. In a similar manner the coolness of Computer Mathematics will have its own romantics: human cleverness combined with computer power finding new understandable results.

References

Ackermann, W. [1928]. Zum Hilbertschen Aufbau der reellen Zahlen, *Mathematische Annalen* **99**, pp. 118–133.

Aczel, P. [1978]. The type theoretic interpretation of constructive set theory, *Logic Colloquium '77 (Proc. Conf., Wrocław, 1977)*, Stud. Logic Foundations Math. 96, North-Holland, Amsterdam, pp. 55–66.

Aczel, P. [1982]. The type theoretic interpretation of constructive set theory: choice principles, *The L. E. J. Brouwer Centenary Symposium (Noordwijkerhout, 1981)*, Stud. Logic Found. Math. 110, North-Holland, Amsterdam, pp. 1–40.

Aczel, P. [1986]. The type theoretic interpretation of constructive set theory: inductive definitions, *Logic, methodology and philosophy of science, VII (Salzburg, 1983)*, Stud. Logic Found. Math. 114, North-Holland, Amsterdam, pp. 17–49.

Aczel, P. and M. Rathjen [2000/2001]. Notes on Constructive Set Theory, *Technical report*, Mittag Leffler Institute, URL:
`<www.ml.kva.se/preprints/meta/AczelMon_Sep_24_09_56.rdf.html>`.

Aristotle [350 B.C.]. *Organon*, See also `<classics.mit.edu/Aristotle>`.

Avigad, J., K. Donnely, D. Gray and P. Raff [2005]. A formally verified proof of the prime number theorem, *Technical report*, `http://arxiv.org/abs/cs/0509025v3`.

Bachmair, Leo and Harald Ganzinger [1994]. Buchberger's algorithm: a constraint-based completion procedure, *Constraints in computational logics (Munich, 1994)*, Lecture Notes in Comput. Sci. 845, Springer, Berlin, pp. 285–301.

Barendregt, H. P. [1984]. *The Lambda Calculus, its Syntax and Semantics*, Studies in Logic and the Foundations of Mathematics 103, revised edition, North-Holland Publishing Co., Amsterdam.

Barendregt, H. P. [1992]. Lambda calculi with types, *Handbook of Logic in Computer Science, Vol. 2*, Oxford Sci. Publ., Oxford Univ. Press, New York, pp. 117–309.

Barendregt, H. P. [1997]. The impact of the lambda calculus in logic and computer science, *Bull. Symbolic Logic* **3**(2), pp. 181–215.

Barendregt, H. P. [2003]. Towards an interactive mathematical proof mode, *in:* Fairouz Kamareddine (ed.), *Thirty Five Years of Automating Mathematics*, Kluwer, p. To appear.

Barendregt, H. P. and E. Barendsen [2002]. Autarkic computations in formal proofs, *J. Automat. Reason.* **28**(3), pp. 321–336.

Barendregt, H. P., M. Bunder and W. Dekkers [1993]. Systems of illative combinatory logic complete for first-order propositional and predicate calculus, *J. Symbolic Logic* **58**(3), pp. 769–788.

Barendregt, H.P. and A. Cohen [2001]. Electronic communication of mathematics and the interaction of computer algebra systems and proof assistants, *J. Symbolic Computation* **32**, pp. 3–22.

Barras, B. [1996]. Verification of the interface of a small proof system in coq, *in:* E. Gimenez and C. Paulin-Mohring (eds.), *Proceedings of the 1996 Workshop on Types for Proofs and Programs*, Springer-Verlag LNCS 1512, Aussois, France, pp. 28–45.

Barras, B. [1999]. *Auto-validation d'un système de preuves avec familles inductives*, Thèse de doctorat, Université Paris 7.

Barthe, G. [1996]. Personal communication.

Barwise, J. [1975]. *Admissible sets and structures*, Springer-Verlag, Berlin. An approach to definability theory, Perspectives in Mathematical Logic.

Beckmann, P. [1971]. *A history of* π, St. Martin's Press, New York.

Beeson, M. [2003]. The mechanization of mathematics, *in:* C. Teuscher (ed.), *A. M. Turing Festschrift*, Springer.

Bell, John L. [1998]. *A primer of infinitesimal analysis*, Cambridge University Press, Cambridge.

van Benthem Jutting, L. S. [1977]. *Checking Landau's Grundlagen in the AUTOMATH system*, Technische Hogeschool Eindhoven, Eindhoven. Doctoral dissertation, with a Dutch summary. Also in [Nederpelt, Geuvers and de Vrijer 1994].

Berger, U., W. Buchholz and H. Schwichtenberg [2000,2001]. Refined program extraction from classical proofs, *Preprint Series: Mathematical Logic 14*, Institut Mittag-Leffler, The Royal Swedish Academy of Sciences.

Bishop, E. [1970]. Mathematics as a numerical language, *Intuitionism and Proof Theory (Proceedings of the summer Conference at Buffalo, N.Y., 1968)*, North-Holland, Amsterdam, pp. 53–71.

Bishop, E. and D. Bridges [1985]. *Constructive analysis*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences] 279, Springer-Verlag, Berlin.

Börger, Egon, Erich Grädel and Yuri Gurevich [2001]. *The classical decision problem*, Universitext, Springer-Verlag, Berlin. Reprint of the 1997 original.

Brouwer, L. E. J. [1908]. *The Unreliability of the Logical Principles*, North-Holland, 1975, chapter in: A. Heyting, Ed. L. E. J. Brouwer: Collected Works 1: Philosophy and Foundations of Mathematics, pp. 107–111.

de Bruijn, N. G. [1970]. The mathematical language AUTOMATH, its usage, and some of its extensions, *Symposium on Automatic Demonstration (Versailles, 1968)*, Lecture Notes in Mathematics, Vol. 125. Springer, Berlin, pp. 29–61.

Buchberger, B. [1965]. *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ring*, Dissertation, University of Innsbruck.

Buchberger, B. and F. Winkler [1998]. *Gröbner Bases and Applications.*, Cambridge University Press.

Cantor, G. [1885]. Über die verschiedenen Standpunkte in bezug auf das Aktual Unendliche, In: *Gesammelte Abhandlungen mathematischen und philosophischen Inhalts / Georg Cantor*, eds. E. Zermelo and A. Fraenkel, Springer, 1932. 370-377.

Capretta, V. [2003]. *Abstraction and Computation*, Dissertation, Department of Computer Science, Nijmegen University, The Netherlands, 6090 GL Nijmegen.

van Ceulen, Ludolph [1615]. *De arithmetische en geometrische fondamenten, met het ghebruyck van dien in veele verscheydene constighe questien, soo geometrice door linien, als arithmetice door irrationale ghetallen, oock door den regel Coss, ende de tafelen sinuum ghesolveert*, Joost van Colster ende Jacob Marcus, Leyden.

Chang, C. C. and H. J. Keisler [1990]. *Model theory*, Studies in Logic and the Foundations of Mathematics 73, third edition, North-Holland Publishing Co., Amsterdam.

Chou, Shang-Ching [1988]. *Mechanical geometry theorem proving*, Mathematics and its Applications 41, D. Reidel Publishing Co., Dordrecht. With a foreword by Larry Wos.

Church, A. [1932]. A set of postulates for the foundation of logic, *Annals of Mathematics, second series* **33**, pp. 346–366.

Church, A. [1936]. An unsolvable problem of elementary number theory, *American Journal of Mathematics* **58**, pp. 345–363.

Church, Alonzo [1940]. A formulation of the simple theory of types, *J. Symbolic Logic* **5**, pp. 56–68.

Cohen, A. M. [2001]. Communicating mathematics across the Web, *Mathematics unlimited—2001 and beyond*, Springer, Berlin, pp. 283–300.

Collins, G. E. [1975]. Quantifier elimination for real closed fields by cylindrical algebraic decomposition, *Automata theory and formal languages (Second GI Conf., Kaiserslautern, 1975)*, Springer, Berlin, pp. 134–183. Lecture Notes in Comput. Sci., Vol. 33.

Constable, R. L. [1997]. Personal communication.

Coquand, T. and H. Herbelin [1994]. *A*-translation and looping combinators in pure type systems, *J. Funct. Programming* **4**(1), pp. 77–88.

Cousineau, G., P.-L. Curien and M. Mauny [1987]. The categorical abstract machine, *Sci. Comput. Programming* **8**(2), pp. 173–202.

Cruz-Filipe, L. and B. Spitters [2003]. Program extraction from large proof developments, *Proceedings of TPHOLs 2003 (in LNCS proceedings).*

Curry, H. B. [1930]. Grundlagen der kombinatorischen Logic,, *American Journal of Mathematics* **52**, pp. 509–536, 789–834.

Curry, Haskell B. and Robert Feys [1958]. *Combinatory logic. Vol. I*, With two selections by William Craig. Second printing. Studies in Logic and the Foundations of Mathematics, North-Holland Publishing Co., Amsterdam.

van Dalen, D. [2000]. The development of Brouwer's intuitionism, *Proof theory (Roskilde, 1997)*, Synthese Lib. 292, Kluwer Acad. Publ., Dordrecht, pp. 117–152.

Dekkers, W., M. Bunder and H. Barendregt [1998]. Completeness of the propositions-as-types interpretation of intuitionistic logic into illative combinatory logic, *J. Symbolic Logic* **63**(3), pp. 869–890.

Dowek, G. [2001]. The stratified foundations as a theory modulo, *Typed lambda calculi and applications (Kraków, 2001)*, Lecture Notes in Comput. Sci. 2044, Springer, Berlin, pp. 136–150.

Dybjer, Peter [2000]. A general formulation of simultaneous inductive-recursive definitions in type theory, *J. Symbolic Logic* **65**(2), pp. 525–549.

Euclid [2002]. *Euclid's Elements*, Green Lion Press, Santa Fe, NM. All thirteen books complete in one volume, The Thomas L. Heath translation, Edited by Dana Densmore.

Feferman, S. [1998]. *In the Light of Logic*, Oxford University Press, Oxford.

Frege, Gottlob [1971]. *Begriffsschrift und andere Aufsätze*, Georg Olms Verlag, Hildesheim. Zweite Auflage. Mit E. Husserls und H. Scholz' Anmerkungen herausgegeben von Ignacio Angelelli, Nachdruck.

Friedman, H. [1978]. Classically and intuitionistically provably recursive functions, *Higher set theory (Proc. Conf., Math. Forschungsinst., Oberwolfach, 1977)*, Lecture Notes in Math. 669, Springer, Berlin, pp. 21–27.

Gabbay, M. J. and A. M. Pitts [2001]. A new approach to abstract syntax with variable binding, *Formal Aspects of Computing* **13**, pp. 341–363.

Gauss, C.F. [1862]. Letter to H.C. Schumacher, July 12, 1831, In: Briefwechsel zwischen C. F. Gauss und H. C. Schumacher, ed. C. A. F. Peters, von Esch, Altona, p. 269.

Gentzen, G. [1969]. *The collected papers of Gerhard Gentzen*, Edited by M. E. Szabo. Studies in Logic and the Foundations of Mathematics, North-Holland Publishing Co., Amsterdam.

Geuvers, H., E. Poll and J. Zwanenburg [1999]. Safe proof checking in type theory with Y, *Computer science logic (Madrid, 1999)*, Lecture Notes in Comput. Sci. 1683, Springer, Berlin, pp. 439–452.

Girard, J.-Y., P. Taylor and Y. Lafont [1989]. *Proofs and types*, Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press, Cambridge.

Gödel, K. [1930]. Die Vollständigkeit der Axiome des logischen Funktionalkalküls, *Monatshefte für Mathematik und Physik* **37**, pp. 349–360.

Gödel, K. [1931]. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, *Monatshefte für Mathematik und Physik* **38**, pp. 173–198. Translated and commented in **?**. Another English version based on course notes by Kleene and Rosser is in **?**.

Gonthier, G. [2005]. A computer-checked proof of the four colour theorem, *Technical report*, Microsoft Research Cambridge. Available at URL `research.microsoft.com/~gonthier/4colproof.pdf`.

Hilbert, D. [1901-1902]. Mathematical problems, *Bull. Amer. Math. Soc.* **8**, pp. 437–479.

Hilbert, D. [1926]. Uber das unendliche, *Mathematische Annalen* **95**, pp. 161–190.

Howard, W. A. [1980]. The formulae-as-types notion of construction, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, Academic Press, London, pp. 480–490.

Howe, D. [1992]. Reflecting the semantics of reflected proof, *Proof Theory, ed. P. Aczel*, Cambridge University Press, pp. 229–250.

Hurkens, Antonius J. C. [1995]. A simplification of Girard's paradox, *Typed lambda calculi and applications (Edinburgh, 1995)*, Lecture Notes in Comput. Sci. 902, Springer, Berlin, pp. 266–278.

Jacobs, B. [1999]. *Categorical logic and type theory*, Studies in Logic and the Foundations of Mathematics 141, North-Holland Publishing Co., Amsterdam.

Jervell, H. R. [1996]. Thoralf Skolem: pioneer of computational logic, *Nordic J. Philos. Logic* **1**(2), pp. 107–117 (electronic).

Kleene, S. C. [1936]. Lambda-definability and recursiveness, *Duke Mathematical Journal* **2**, pp. 340–353.

Kline, Morris [1990]. *Mathematical thought from ancient to modern times. Vol. 1*, second edition, The Clarendon Press Oxford University Press, New York.

Klop, J.W. et al. (ed.) [2003]. *Term Rewrite Systems*, Cambridge University Press.

Knuth, Donald E. and Peter B. Bendix [1970]. Simple word problems in universal algebras, *Computational Problems in Abstract Algebra (Proc. Conf., Oxford, 1967)*, Pergamon, Oxford, pp. 263–297.

Kreisel, Georg [1985]. Proof theory and the synthesis of programs: Potential and limitations, *in:* Bruno Buchberger (ed.), *EUROCAL '85: European Conference on Computer Algebra*, Lecture Notes in Computer Science 203, Springer-Verlag, pp. 136–150.

Laan, T. [1997]. *The evolution of type theory in logic and mathematics*, Technische Universiteit Eindhoven, Eindhoven. Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1997.

Landau, E. [1960]. *Grundlagen der Analysis (das Rechnen mit ganzen, rationalen, irrationalen, komplexen Zahlen)*, 3rd ed, Chelsea Publishing Co., New York.

Leibniz, G.W. [1875-1890]. De scientia universalis seu calculo philosophico, *in:* C.I. Gerhardt (ed.), *Die Philosophischen Schriften von Gottfried Wilhelm Leibniz*, Vol. VII, Weidmann, Berlin. Reprinted 1960-1961, Georg Olms Verlag, Hildesheim.

Letouzey, P. [2003]. A new extraction for Coq, *Proceedings of the TYPES Conference 2002*, LNCS 2626, Springer-Verlag, pp. 200–219.

Makkai, M. [1999]. On structuralism in mathematics, *Language, logic, and concepts*, Bradford Book, MIT Press, Cambridge, MA, pp. 43–66.

Marché, Claude [1998]. Normalized Rewriting: an unified view of Knuth-Bendix completion and Gröbner bases computation, *Progress in Computer Science and Applied Logic* **15**, pp. 193–208.

Martin-Löf, P. [1984]. *Intuitionistic type theory*, Studies in Proof Theory. Lecture Notes 1, Bibliopolis, Naples. Notes by Giovanni Sambin.

McCarthy, J. [1962]. Computer programs for checking the correctness of mathematical proofs, *Proceedings of a Symposium in Pure Methematics, vol. V.*, American Mathematical Society, Providence, RI, pp. 219–227.

Middeldorp, Aart and Mirjana Starčević [1991]. A rewrite approach to polynomial ideal theory, *Report CS-R9160*, CWI, Amsterdam.

Mines, R., F. Richman and W. Ruitenburg [1988]. *A course in constructive algebra*, Universitext, Springer-Verlag, New York.

Moerdijk, Ieke and Gonzalo E. Reyes [1991]. *Models for smooth infinitesimal analysis*, Springer-Verlag, New York.

Mostowski, A. [1968]. Personal communication.

Nederpelt, R. P., J. H. Geuvers and R. C. de Vrijer [1994]. Twenty-five years of Automath research, *Selected papers on Automath*, Stud. Logic Found. Math. 133, North-Holland, Amsterdam, pp. 3–54.

Newton, I. [1736]. *Method of Fluxions and Infinite Series*, John Nourse, London. Posthumous translation from the unpublished Latin original [1671] by J. Colson.

Paulin-Mohring, Christine [1989]. Extracting F_ω's programs from proofs in the Calulus of Constructions, *Sixteenth Annual ACM Symposium on Principles of Programming Languages*, ACM, Austin.

Paulin-Mohring, Christine. [1993]. Inductive definitions in the system Coq; rules and properties, *Typed lambda calculi and applications (Utrecht, 1993)*, Lecture Notes in Comput. Sci. 664, Springer, Berlin, pp. 328–345.

Péter, Rózsa. [1934]. Über den zusammenhang der verschiedenen begriffe der rekursiven funktion, *Mathematische Annalen.*

Péter, Rózsa. [1967]. *Recursive functions*, Third revised edition. Translated from the German by István Földes, Academic Press, New York.

Poincaré, H. [1902]. *La Science et l'Hypothèse*, Flammarion, Paris.

Poincaré, H. [1905]. *La Valeur de la Science*, Flammarion, Paris.

Pollack, R. [1994]. Personal communication.

Pollack, R. [1995]. On extensibility of proof checkers, *Types for proofs and programs*, Lecture Notes in Computer Science 996, Springer-Verlag, Berlin, pp. 140–161.

Rasiowa, H. and R. Sikorski [1963]. *The mathematics of metamathematics*, PWN-Polish Scientific Publishers.

Robinson, Abraham [1996]. *Non-standard analysis*, Princeton Landmarks in Mathematics, Princeton University Press, Princeton, NJ. Reprint of the second (1974) edition, With a foreword by Wilhelmus A. J. Luxemburg.

Saito, Mutsumi, Bernd Sturmfels and Nobuki Takayama [2000]. *Gröbner deformations of hypergeometric differential equations*, Algorithms and Computation in Mathematics 6, Springer-Verlag, Berlin.

Schwichtenberg, H. [2002]. Minimal logic for computable functionals, *Technical report*, Mathematisches Institut der Universität München.

Scott, D. [1970]. Constructive validity, *Symposium on Automatic Demonstration (Versailles, 1968)*, Lecture Notes in Mathematics, Vol. 125, Springer, Berlin, pp. 237–275.

Simpson, S. G. [1988]. Partial realizations of Hilbert's Program, *J. Symbolic Logic* **53**(2), pp. 349–363.

Skolem, T. [1922]. Über ganzzahlige Lösungen einer Klasse unbestimmter Gleichungen, *Norsk Matematisk Forenings skrifter.*

Statman, R. [1979]. The typed λ-calculus is not elementary recursive, *Theoret. Comput. Sci.* **9**(1), pp. 73–81.

Sudan, G. [1927]. Sur le nombre transfini ω^ω, *Bulletin mathématique de la Société Roumaine des Sciences* **30**, pp. 11–30.

Tarski, A. [1951]. *Decision Method for Elementary Algebra and Geometry*, University of California Press, Berkeley.

Troelstra, A. S. and D. van Dalen [1988]. *Constructivism in mathematics. Vol. I, II*, Studies in Logic and the Foundations of Mathematics 121, 123, North-Holland Publishing Co., Amsterdam. An introduction.

Troelstra, A. S. (ed.) [1973]. *Metamathematical investigation of intuitionistic arithmetic and analysis*, Springer-Verlag, Berlin. Lecture Notes in Mathematics, Vol. 344.

Turing, A.M. [1936]. On Computable Numbers, with an Application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society, Series 2* **42**, pp. 230–265.

Zeilberger, D. [2002]. ENCAPSULATE!, public communication, *in:* A.M. Cohen, X.-S. Gao and N. Takayama (eds.), *Mathematical Software*, First International Conference on Mathematical Software (Beijing), World Scientific, Singapore, p. 318.