MathLang, a framework for computerising mathematics

Fairouz Kamareddine

Tuesday 14 September 2006

Computerisation?

- Logic is *OLD*. Mathematics is *OLD*. But, *SO IS* computer science.
- Assume a problem Π,
  - If you *give* me an algorithm to solve Π, I can check whether this algorithm really solves Π.
  - But, if you ask me to *find* an algorithm to solve Π, I may go on forever trying but without success.
- But, this result was already found by Aristotle: Assume a proposition Φ.
  - If you *give* me a proof of Φ, I can check whether this proof really proves Φ.
  - But, if you ask me to *find* a proof of Φ, I may go on forever trying but without success.
- In fact, *programs* are *proofs* and much of computer science in the early part of the 20th century was built by mathematicians and logicians.
- There were also important inventions in computer science made by physicists (e.g., von Neumann) and others, but we ignore these in this talk.

An example of a computable function/solvable problem

- E.g., 1.5 chicken lay down 1.5 eggs in 1.5 days.
- How many eggs does 1 chicken lay in 1 day?
- 1.5 chicken lay 1.5 eggs in 1.5 days.
- Hence, 1 chicken lay 1 egg in 1.5 days.
- Hence, 1 chicken lay 2/3 egg in 1 day.

Unsolvability of the Barber problem

- *which man barber in the village shaves all and only those men who do not shave themselves?*
- If John was the barber then
  - *John shaves Bill* $\iff$ *Bill does not shave Bill*
  - *John shaves x* $\iff$ *x does not shave x*
  - *John shaves John* $\iff$ *John does not shave John*
- *Contradiction.*

Unsolvability of the Russell set problem

- Another unsolvable problem: Give me the Russell set $R = \{x \mid x \notin x\}$.
- If R existed then $x \in R$ *iff* $x \notin x$.
  - If $R \in R$ then
    $(x \notin x)[x := R]$ and so $R \notin R$.
    Contradiction.
  - If $R \notin R$ then
    $(x \notin x)[x := R]$ and so $R \in R$.
    Contradiction.
- What about the problem:
- *Find an algorithm which takes any program P and input x and tells you whether P halts or loops with input x.*

- Aristotle already knew that for a proposition Φ.
  - If you *give* me a proof of Φ, I can check whether this proof really proves Φ.
  - But, if you ask me to *find* a proof of Φ, I may go on forever trying but without success.

- Aristotle used logic to reason about everything (law, farming, medicine,...)

- At the glorious times when the Arabs were leading in science, princes used to study mathematics for pleasure. Their courts had musicians, poets, math teachers, etc. These teachers already insisted that Maths must be taught and developed using logic. This is one of the main themes of the research of Frege and Russell.

- In the 17th century, Leibniz wanted to use logic to prove the existence of God.

Why did computer science kick off in the 20th century?

In the 19th century, the *need for a more precise* style in mathematics arose, *because controversial results* had appeared in *analysis*.

- 1821: Many of these controversies were solved by the work of Cauchy. E.g., he introduced *a precise definition of convergence* in his *Cours d'Analyse* [4].
- 1872: Due to the more *exact definition of real numbers* given by Dedekind [9], the rules for reasoning with real numbers became even more precise.
- 1895-1897: Cantor began formalizing *set theory* [2, 3] and made contributions to *number theory*.

Formal systems in the 19th century

- 1889: *Peano* formalized *arithmetic* [26], but did not treat logic or quantification.
- 1879: *Frege* was not satisfied with the use of *natural language in mathematics*:

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

  (*Begriffsschrift*, Preface)

  Frege therefore presented *Begriffsschrift* [11], the first formalisation of logic giving logical concepts via symbols rather than natural language.

Formal systems in the 19th century

> "[Begriffsschrift's] first purpose is to *provide us with the most reliable test of the validity of a chain of inferences* and to point out *every presupposition* that tries to sneak in unnoticed, so that its origin *can be investigated*."
> (*Begriffsschrift*, Preface)

- 1892-1903 Frege's *Grundgesetze der Arithmetik* [13, 17], could handle elementary arithmetic, set theory, logic, and quantification.
- Also in 1900, Hilbert, posed a list of problems at a conference in Paris.
- One very important question was: Can any logical statement have a proof or be disproved.
- More than 30 years later, this question was negatively answered by Turing (Turing machines), Goedel (incompleteness results) and Church (λ-calculus).

Can we solve/compute everything?

- Turing answered the question in terms of a *computer*. Turing's machines are so powerful: *anything that can ever be computed even on the most powerful computers, can also be computed on a Turing machine.*

- Church invented the λ-calculus, a *language for programming*. λ-calculus is so powerful: *anything that can ever be computed can be described in the λ-calculus.*

- Goedel's result meant that no absolute guarantee can be given that many significant branches of mathematics are entirely free of contradictions.

- This meant that: we can compute a very small (countable) amount compared to what we will never be able to compute (uncountable).

- Hilbert's dream was shattered. According to the great historian of Mathematics Ivor Grattan-Guinness, Hilbert behaved coldly towards Goedel.

How did Logic and mathematics influence programming languages?

- Frege was the first most precise logician. He wanted symbols to replace natural language everywhere.
- *Self-application of functions* was at the heart of *Russell's paradox 1902* [30].
- To *avoid paradox* Russell controled function application via *type theory*.
- Russell [31] *1903* gives the first type theory: the *Ramified Type Theory* (RTT).
- But, *type theory* existed since the time of *Euclid* (325 B.C.).
- RTT is used in Russell and Whitehead's Principia Mathematica [34] 1910–1912.
- *Simple theory of types* (STT): Ramsey [28] *1926*, Hilbert and Ackermann [19] *1928*.

- Church's *simply typed λ-calculus* $\lambda\!\rightarrow$ [7] 1940 $=$ λ-calculus $+$ STT.

- Untyped λ-calculus was adopted in LISP.

- Simply typed λ-calculus was adopted in theorem provers like HOL and was used to make sense of other programming languages (e.g., pascal).

- Then, simple types were extended to *polymorphic* (and other) types.

- These are used in programming languages like ML.

- And the search continues for better and better programming languages.

- *Types* continue to play an influential role in the design and implementation of programming languages.

Prehistory of Types (Euclid)

- Euclid's *Elements* (circa 325 B.C.) begins with:

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

     ⋮

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

- 1..15 *define* points, lines, and circles which Euclid *distinguished* between.

- Euclid always mentioned to which *class* (points, lines, etc.) an object belonged.

Prehistory of Types (Euclid)

- By distinguishing classes of objects, Euclid prevented *undesired/impossible* situations. E.g., whether two points (instead of two lines) are parallel.

- Intuition implicitly forced Euclid to think about the *type* of the objects.

- As intuition does not support the notion of parallel points, he did not even *try* to undertake such a construction.

- In this manner, types have always been present in mathematics, although they were not noticed explicitly until the late 1800s.

- If you studied geometry, then you have an (implicit) understanding of types.

Prehistory of Types (Paradox Threats)

- From 1800, mathematical systems became less intuitive, for several reasons:
  - Very *complex* or abstract systems.
  - *Formal* systems.
  - Something with *less intuition* than a human using the systems: a *computer* or an *algorithm*.

- These situations are *paradox threats*. An example is Frege's Naive Set Theory.

- *Not enough intuition* to activate the (implicit) type theory *to warn against an impossible situation*.

Prehistory of Types (Begriffsschrift's functions)

The introduction of a *very general definition of function* was the key to the formalisation of logic. Frege defined the Abstraction Principle.

Abstraction Principle 1.

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

(*Begriffsschrift*, Section 9)

Programs (or algorithms) are functions.

Prehistory of Types (Begriffsschrift's functions)

- Frege put *no restrictions* on what could play the role of *an argument*.
- An argument could be a *number* (as was the situation in analysis), but also a *proposition*, or a *function*.
- Similarly, the *result of applying* a function to an argument did not necessarily have to be a number.
- Functions of more than one argument were constructed by a method that is very close to the method presented by Schönfinkel [33] in 1924.

Prehistory of Types (Begriffsschrift's functions)

With this definition of function, two of the three possible *paradox threats occurred*:

1. The generalisation of the concept of function made the system more abstract and *less intuitive*.

2. Frege introduced a *formal* system instead of the informal systems that were used up till then.

Type theory, that would be helpful in distinguishing between the different types of arguments that a function might take, *was left informal*.

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

Prehistory of Types (Begriffsschrift's functions)

Frege was *aware* of some typing rule that does *not* allow to *substitute functions for object variables or objects for function variables*:

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

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

Prehistory of Types (Begriffsschrift's functions)

In *Function and Concept* he was aware of the fact that making a *difference between first-level and second-level objects is essential to prevent paradoxes*:

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

The above discussion on functions and arguments shows that *Frege did indeed avoid the paradox in his Begriffsschrift*.

Prehistory of Types (Grundgesetze's functions)

The *Begriffsschrift*, however, was only a prelude to Frege's writings.

- In Grundlagen der Arithmetik [12] he argued that mathematics can be seen as a branch of logic.

- In Grundgesetze der Arithmetik [13, 17] he described the elementary parts of arithmetic within an extension of the logical framework of *Begriffsschrift*.

- Frege approached the *paradox threats for a second time* at the end of Section 2 of his *Grundgesetze*.

- He did *not* want to *apply a function to itself*, but to its course-of-values.

Prehistory of Types (Grundgesetze's functions)

"the function $\Phi(x)$ has the same *course-of-values* as the function $\Psi(x)$" if:

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

- Note that functions $\Phi(x)$ and $\Psi(x)$ may have equal courses-of-values even if they have different definitions. E.g., $x \wedge \neg x$, and $x \leftrightarrow \neg x$.

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

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

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

Prehistory of Types (Grundgesetze's functions)

- The notation $\grave{\varepsilon}\Phi(\varepsilon)$ may be the *origin* of Russell's notation $\hat{x}\Phi(x)$ for the class of objects that have the property Φ.

- According to a paper by Rosser [29], the notation $\hat{x}\Phi(x)$ has been at the *basis* of the current notation $\lambda x.\Phi(x)$.

- Church is supposed to have written $\wedge x\Phi(x)$ for the function $x \mapsto \Phi(x)$: the hat $\wedge$ in front of the x distinguishes this function from the class $\hat{x}\Phi(x)$.

Prehistory of Types (Grundgesetze's functions)

- Frege treated *courses-of-values* as *ordinary objects*.
- As a consequence, *a function that takes objects as arguments could have its own course-of-values as an argument*.
- In modern terminology: a function that takes objects as arguments can have its own graph as an argument.
- *BUT,* all essential information of a function is contained in its graph.
- A system in which a function can be applied to its own graph should have similar possibilities as a system in which a function can be applied to itself.
- Frege *excluded the paradox threats* by *forbidding self-application*
- but due to his *treatment of courses-of-values* these threats were able to *enter his system through a back door*.

Prehistory of Types (Russell's paradox in *Grundgesetze*)

- In 1902, Russell wrote a letter to Frege [30], informing him that he had *discovered a paradox* in his *Begriffsschrift.*
- *WRONG: Begriffsschrift does not suffer from a paradox*.
- Russell gave his well-known argument, defining the propositional function

  $f(x)$ *by* $\neg x(x)$.

  In Russell's words: "*to be a predicate that cannot be predicated of itself.*"
- Russell assumed $f(f)$. Then by definition of f, $\neg f(f)$, *a contradiction*. Therefore: $\neg f(f)$ holds. But then (again by definition of f), $f(f)$ holds. Russell concluded that *both* $f(f)$ *and* $\neg f(f)$ hold, a *contradiction*.

Prehistory of Types (Russell's paradox in *Grundgesetze*)

- *6 days later*, Frege wrote [16] that *Russell's derivation of paradox is incorrect*.

- Ferge explained that *self-application $f(f)$ is not possible in* Begriffsschrift.

- $f(x)$ *is a function, which requires an* object *as an argument.*
  A function cannot be an object in the Begriffsschrift.

- Frege explained that *Russell's argument could be amended to a paradox* in Grundgesetze, using the *course-of-values* of functions:

  Let $f(x) = \neg\forall\varphi[(\grave{\alpha}\varphi(\alpha) = x) \longrightarrow \varphi(x)]$
  I.e. $f(x) = \quad \exists\varphi[(\grave{\alpha}\varphi(\alpha) = x) \ \wedge \neg\varphi(x)]$ hence $\neg\varphi(\grave{\alpha}\varphi(\alpha))$

- *Both* $f(\grave{\varepsilon}f(\varepsilon))$ and $\neg f(\grave{\varepsilon}f(\varepsilon))$ hold.

- Frege added an appendix of 11 pages to the 2nd volume of *Grundgesetze* in which he gave a very detailed description of the paradox.

Prehistory of Types (How wrong was Frege?)

- Due to Russell's Paradox, Frege is often depicted as the pitiful person whose system was inconsistent.

- This suggests that Frege's system was the only one that was inconsistent, and that Frege was very inaccurate in his writings.

- On these points, history does Frege an injustice.

- Frege's system was much more accurate than other systems of those days.

- Peano's work, for instance, was *less precise* on several points:

- Peano *hardly paid attention to logic* especially quantification theory;

- Peano *did not make a strict distinction* between his *symbolism* and the *objects underlying this symbolism*. Frege was much more accurate on this point (see Frege's paper *Über Sinn und Bedeutung* [14]);

Prehistory of Types (How wrong was Frege?)

- Frege *made a strict distinction* between a *proposition* (as an object) and the assertion *of a proposition*. Frege denoted a *proposition*, by $-A$, and *its assertion* by $\vdash A$. Peano did not make this distinction and simply wrote A.

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

- Peano had *able collaborators*, and a *better eye for presentation and publicity*.
- Peano bought *his own press* to supervise the printing of his own journals Rivista di Matematica and Formulaire [27]

Prehistory of Types (How wrong was Frege?)

- Peano used a *familiar symbolism* to the notations used in those days.

- Many of *Peano's notations*, like $\in$ for "is an element of", and $\supset$ for logical implication, are used in *Principia Mathematica*, and are actually still in use.

- *Frege's work* did not have these advantages and *was hardly read before 1902*

- When *Peano* published his formalisation of mathematics in 1889 [26] he clearly *did not know* Frege's *Begriffsschrift* as he did not mention the work, and *was not aware* of Frege's formalisation of quantification theory.

Prehistory of Types (How wrong was Frege?)

- Peano considered quantification theory to be "abstruse" in [27]:

  > "In this respect my *[Frege] conceptual notion of 1879 is superior to the Peano one*. Already, at that time, I specified all the laws necessary for my designation of generality, so that nothing fundamental remains to be examined. These laws are few in number, and *I do not know why they should be said to be abstruse*. If it is otherwise with the *Peano conceptual notation*, then this is due to the *unsuitable* notation."
  >
  > ([15], p. 376)

Prehistory of Types (How wrong was Frege?)

- In the last paragraph of [15], Frege concluded:

  "... I observe merely that the *Peano notation* is unquestionably *more convenient for the typesetter*, and in many cases *takes up less room* than mine, but that these advantages seem to me, due to the inferior perspicuity and *logical defectiveness*, to have been paid for too dearly — at any rate for the purposes I want to pursue."
  (*Ueber die Begriffschrift des Herrn Peano und meine eigene*, p. 378)

Prehistory of Types (paradox in Peano and Cantor's systems)

- Frege's system was *not the only paradoxical* one.
- The Russell Paradox can be derived in *Peano's system* as well, by defining the class $K \stackrel{\text{def}}{=} \{x \mid x \notin x\}$ and deriving $K \in K \longleftrightarrow K \notin K$.
- In *Cantor's Set Theory* one can derive the paradox via the same class (or *set*, in Cantor's terminology).

Prehistory of Types (paradoxes)

- Paradoxes were already widely known in *antiquity*.
- The oldest logical paradox: the *Liar's Paradox* "This sentence is not true", also known as the Paradox of Epimenides. It is referred to in the Bible (Titus $1{:}12$) and is based on the confusion between language and meta-language.
- The *Burali-Forti paradox* ([1], 1897) is the first of the modern paradoxes. It is a paradox within Cantor's theory on ordinal numbers.
- Cantor was *aware* of the Burali-Forti paradox but *did not think* it would render his system incoherent.
- *Cantor's paradox* on the largest cardinal number occurs in the same field. It was discovered by Cantor around 1895, but was not published before 1932.

Prehistory of Types (paradoxes)

- Logicians considered these paradoxes to be *out of the scope of logic*:
  - The *Liar's Paradox* can be regarded as a problem of *linguistics*.
  - The *paradoxes of Cantor and Burali-Forti* occurred in what was considered in those days a *highly questionable* part of mathematics: *Cantor's Set Theory*.
- The Russell Paradox, however, was *a paradox that could be formulated in all* the systems of the end of the 19th century (except for Frege's *Begriffsschrift*).
- Russell's Paradox was at the very basics of logic.
- It could not be disregarded, and a solution to it had to be found.
- In 1903-1908, Russell suggested the use of *types* to solve the problem [32].

Prehistory of Types (vicious circle principle)

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

> "I am on the point of finishing a book on the principles of mathematics"
>
> (*Letter to Frege*, [30])

Russell *had to find a solution* to the paradoxes, before finishing his book.

His paper Mathematical logic as based on the theory of types [32] *(1908)*, in which a first step is made towards the Ramified Theory of Types, started with a description of the most important contradictions that were known up till then, including Russell's own paradox. He then concluded:

Prehistory of Types (vicious circle principle)

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

(*Ibid.*)

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

Ramified Type Theory

“*Whatever involves all of a collection must not be one of the collection.*”

(Mathematical logic as based on the theory of types)

- Russell applies this principle *very strictly*.
- He implemented it using *types*, in particular the so-called ramified *types*.
- The type theory of 1908 was elaborated in Chapter II of the Introduction to the famous *Principia Mathematica* [34] (1910-1912).

Ramified Type Theory and Principia

- In the *Principia*, *mathematics was founded on logic*, as far as possible.

- The *logical part* of *Principia* was *based* on the works of *Frege* (acknowledged by Whitehead and Russell in the preface, and can be seen throughout the description of Type Theory).

- The notion of *function is based on Frege's Abstraction Principles*.

- The *Principia* *notation* $\hat{x}f(x)$ for a class looks very *similar to Frege's* $\grave{\varepsilon}f(\varepsilon)$ for course-of-values.

The Simple Theory of Types

- Ramsey [28], and Hilbert and Ackermann [19], *simplified* the Ramified Theory of Types RTT by removing the orders. The result is known as the Simple Theory of Types *(STT)*.

- In *1932 and 1933*, Church presented his (untyped) *λ-calculus* [5, 6]. *In 1940* he combined this theory with STT giving us the simply typed λ-calculus $\lambda{\rightarrow}$.

- $\lambda{\rightarrow}$ is very restrictive.

- Numbers, booleans, the identity function have to be defined at every level.

- We can represent (and type) terms like $\lambda x : o.x$ and $\lambda x : \iota.x$.

- We cannot type $\lambda x : \alpha.x$, where α can be instantiated to any type.

- This led to new (modern) type theories that allow more general notions of functions (e.g, *polymorphic*).

How can Computerisation help mathematics?

- Nowadays, *computerization* is an essential feature of any field.
- What is the influence of computerization on the study of language.
- Which language? Arabic, English, French, German, ...
- Euclid's book on geometry was written in Greek in Alexandria and translated into many languages.
- The impressive library of Alexandria at that time was destroyed later.
- Attempts at recreating this library *electronically* are under way
- We need the computerization of a huge number of texts and books.
- Why computerize books? How do we computerize books? What problems do we encounter?

The Goal: Open borders between mathematics, logic and computation

- Ordinary mathematicians *avoid* formal mathematical logic.
- Ordinary mathematicians *avoid* proof checking (via a computer).
- Ordinary mathematicians *may use* a computer for computation: there are over 1 million people who use Mathematica (including linguists, engineers, etc.).
- Mathematicians may also use other computer forms like Maple, LaTeX, etc.
- But we are not interested in only *libraries* or *computation* or *text editing*.
- We want *freedeom of movement* between mathematics, logic and computation.
- At every stage, we must have *the choice* of the level of formalilty and the depth of computation.

Common Mathematical Language of mathematicians: CML

+ CML is *expressive*: it has linguistic categories like *proofs* and *theorems*.

+ CML has been refined by intensive use and is rooted in *long traditions*.

+ CML is *approved* by most mathematicians as a communication medium.

+ CML *accommodates many branches* of mathematics, and is adaptable to new ones.

– Since CML is based on natural language, it is *informal* and *ambiguous*.

– CML is *incomplete:* Much is left implicit, appealing to the reader's intuition.

– CML is *poorly organised:* In a CML text, many structural aspects are omitted.

– CML is *automation-unfriendly:* A CML text is a plain text and cannot be easily automated.

A CML-text

From chapter 1, § 2 of E. Landau's *Foundations of Analysis* [Lan51].

Theorem 6. [Commutative Law of Addition]

$$x + y = y + x.$$

Proof Fix y, and let $\mathfrak{M}$ be the set of all x for which the assertion holds.
I) We have

$$y + 1 = y',$$

and furthermore, by the construction in the proof of Theorem 4,

$$1 + y = y',$$

so that

$$1 + y = y + 1$$

and 1 belongs to $\mathfrak{M}$.
II) If x belongs to $\mathfrak{M}$, then

$$x + y = y + x,$$

Therefore

$$(x + y)' = (y + x)' = y + x'.$$

By the construction in the proof of Theorem 4, we have

$$x' + y = (x + y)',$$

hence

$$x' + y = y + x',$$

so that x' belongs to $\mathfrak{M}$. The assertion therefore holds for all x. □

What are the options for computerization?

Computers can handle mathematical text at various levels:

- Images of pages may be stored. While useful, this is not a good representation of *language* or *knowledge*.
- Typesetting systems like LaTeX can be used.
- Document representations like OMDoc can be used.
- Formal logics used by theorem provers can be used.

We are gradually developing a system named MathLang which we hope will eventually allow building a bridge between the latter 3 levels.

This talk aims at discussing the motivations rather than the details.

The issues with typesetting systems

+ A system like LaTeX provides good defaults for visual appearance, while allowing fine control when needed.

+ LaTeX supports commonly needed document structures, while allowing custom structures to be created.

– Unless the mathematician is amazingly disciplined, the logical structure of symbolic formulas is not represented at all.

– The logical structure of mathematics as embedded in natural language text is not represented. Automated discovery of the semantics of natural language text is still too primitive and requires human oversight.

LaTeX example

draft documents	✓
public documents	✓
computations and proofs	✗

```
\begin{theorem}[Commutative Law of Addition]\label{theorem:6}
  $$x+y=y+x.$$
\end{theorem}
\begin{proof}
  Fix $y$, and $\mathfrak{M}$ be the set of all $x$ for which the
  assertion holds.
  \begin{enumerate}
  \item We have $$y+1=y',$$
    and furthermore, by the construction in
    the proof of Theorem~\ref{theorem:4}, $$1+y=y',$$
    so that $$1+y=y+1$$
    and $1$ belongs to $\mathfrak{M}$.
  \item If $x$ belongs to $\mathfrak{M}$, then $$x+y=y+x,$$
    Therefore
    $$(x+y)'=(y+x)'=y+x'.$$
    By the construction in the proof of
    Theorem~\ref{theorem:4}, we have $$x'+y=(x+y)',$$
    hence
    $$x'+y=y+x',$$
    so that $x'$ belongs to $\mathfrak{M}$.
  \end{enumerate}
  The assertion therefore holds for all $x$.
\end{proof}
```

The differences of OMDoc

OMDoc attempts to solve some of the difficulties of typesetting systems.

+ Translation to LaTeX (still needed) or MathML can handle visual appearance.
– Precise appearance control must work *through* a translation (difficult!).
+ OMDoc supports commonly needed document structures.
+ The tree structure of symbolic formulas is represented.
– The semantics of symbolic formulas is not represented.
– Type checking symbolic formulas (beyond arity) must be outside OMDoc.
– The logical structure of mathematics as embedded in natural language text is still not represented. There are ways to associate symbolic formulas with natural language text, but no way to check their consistency.

The beginnings of computerized formalization

- In 1967 the famous mathematician de Bruijn began work on logical languages for complete books of mathematics that can be *fully* checked by machine.
- People are prone to error, so if a machine can do proof checking, we expect fewer errors.
- Most mathematicians doubted de Bruijn could achieve success, and computer scientists had no interest at all.
- However, he persevered and built *Automath* (AUTOmated MATHematics).
- Today, there is much interest in many approaches to proof checking for verification of computer hardware and software.
- Many theorem provers have been built to mechanically check mathematics and computer science reasoning (e.g. Isabelle, HOL, Coq, etc.).

The problem with formal logic

- No logical language has the criteria expected of a language of mathematics.
  - A logical language does not have *mathematico-linguistic* categories, is *not universal* to all mathematicians, and is *not a good communication medium*.
  - Logical languages make fixed choices (*first versus higher order, predicative versus impredicative, constructive versus classical, types or sets*, etc.). But different parts of mathematics need different choices and there is no universal agreement as to which is the best formalism.
  - A logician reformulates in logic their *formalization* of a mathematical-text as a formal, complete text which is structured considerably *unlike* the original, and is of little use to the *ordinary* mathematician.
  - Mathematicians do not want to use formal logic and have *for centuries* done mathematics without it.

- *So, mathematicians kept to* CML.

- We would like to find an alternative to CML which avoids some of the features of the logical languages which made them unattractive to mathematicians.

Full formalization difficulties: choices

A CML-text is structured differently from a fully formalized text proving the same facts. *Making the latter involves extensive knowledge and many choices:*

- The choice of the *underlying logical system.*
- The choice of *how concepts are implemented* (equational reasoning, equivalences and classes, partial functions, induction, etc.).
- The choice of the *formal foundation*: a type theory (dependent?), a set theory (ZF? FM?), a category theory? etc.
- The choice of the *proof checker*: Automath, Isabelle, Coq, PVS, Mizar, ...

An issue is that one must in general commit to one set of choices.

Full formalization difficulties: informality

Any informal reasoning in a CML-text will cause various problems when fully formalizing it:

- A single (big) step may need to expand into a (series of) syntactic proof expressions. Very long expressions can replace a clear CML-text.
- The entire CML-text may need *reformulation* in a fully *complete* syntactic formalism where every detail is spelled out. New details may need to be woven throughout the entire text. The text may need to be "turned inside out".
- Reasoning may be obscured by *proof tactics*, whose meaning is often *ad hoc* and implementation-dependent.

Regardless, ordinary mathematicians do not find the new text useful.

Coq example

draft documents	✗
public documents	✗
computations and proofs	✓

From Module `Arith.Plus` of Coq standard library (`http://coq.inria.fr/`).

```
Lemma plus_sym : (n,m:nat)(n+m)=(m+n).
Proof.
Intros n m ; Elim n ; Simpl_rew ; Auto with arith.
Intros y H ; Elim (plus_n_Sm m y) ; Simpl_rew ; Auto with arith.
Qed.
```

Where do we start? de Bruijn's Mathematical Vernacular MV

- *De Bruijn's Automath not just [...] as a technical system for verification of mathematical texts, it was rather a life style with its attitudes towards understanding, developing and teaching mathematics.... The way mathematical material is to be presented to the system should correspond to the usual way we write mathematics. The only things to be added should be details that are usually omitted in standard mathematics.*

- MV is faithful to $\mathrm{C}\textsc{ml}$ yet is formal and avoids ambiguities.

- MV is close to the usual way in which mathematicians write.

- MV has a syntax based on linguistic categories not on set/type theory.

- MV is weak as regards correctness: the rules of MV mostly concern *linguistic* correctness, its types are mostly linguistic so that the formal translation into MV is satisfactory *as a readable, well-organized text*.

Problems with MV

- MV makes many logical and mathematical choices which are best postponed.
- MV incorporates certain correctness requirements, there is for example a hierarchy of types corresponding with sets and subsets.
- MV is already *on its way* to a full formalization, while we want the option of remaining *closer to* a given informal mathematical content.
- We want a *formal* language MathLang which •has the advantages of CML but not its disadvantages and •respects CML content.
- *MV does not respect* CML *content.*

What is the aim for MathLang?

Can we formalise a CML text, avoiding as much as possible the ambiguities of natural language, while still guaranteeing the following four goals?

1. The formalised text looks very much like the original CML text (and hence the content of the original CML text is respected).

2. The formalised text can be fully manipulated and searched in ways that respect its mathematical structure and meaning.

3. Steps can be made to do computation (via computer algebra systems) and proof checking (via proof checkers) on the formalised text.

4. This formalisation of text is not much harder for the ordinary mathematician than LaTeX. *Full formalization down to a foundation of mathematics is not required*, although allowing and supporting this is one goal.

(No theorem prover's language satisfies these goals.)

Starting point for MathLang: MV and WTT

- MV was an initial inspiration for MathLang. But MV fails on goal 1.
- Weak Type Theory, WTT [21], is MV minus the added logic.
- Although in many ways WTT succeeds and improves on MV, it still fails on goal 1. A WTT text is not close to its CML original.
- With MathLang, we start from WTT, add some features, and investigate how to integrate it with natural language text.
- Our ongoing development of MathLang is driven by testing it in translating a set of sample texts chosen to cover a large portion of CML usages, both current and historical.

MathLang

draft documents	✓
public documents	✓
computations and proofs	✓

- A MathLang text captures the grammatical and reasoning aspects of mathematical structure for further computer manipulation.

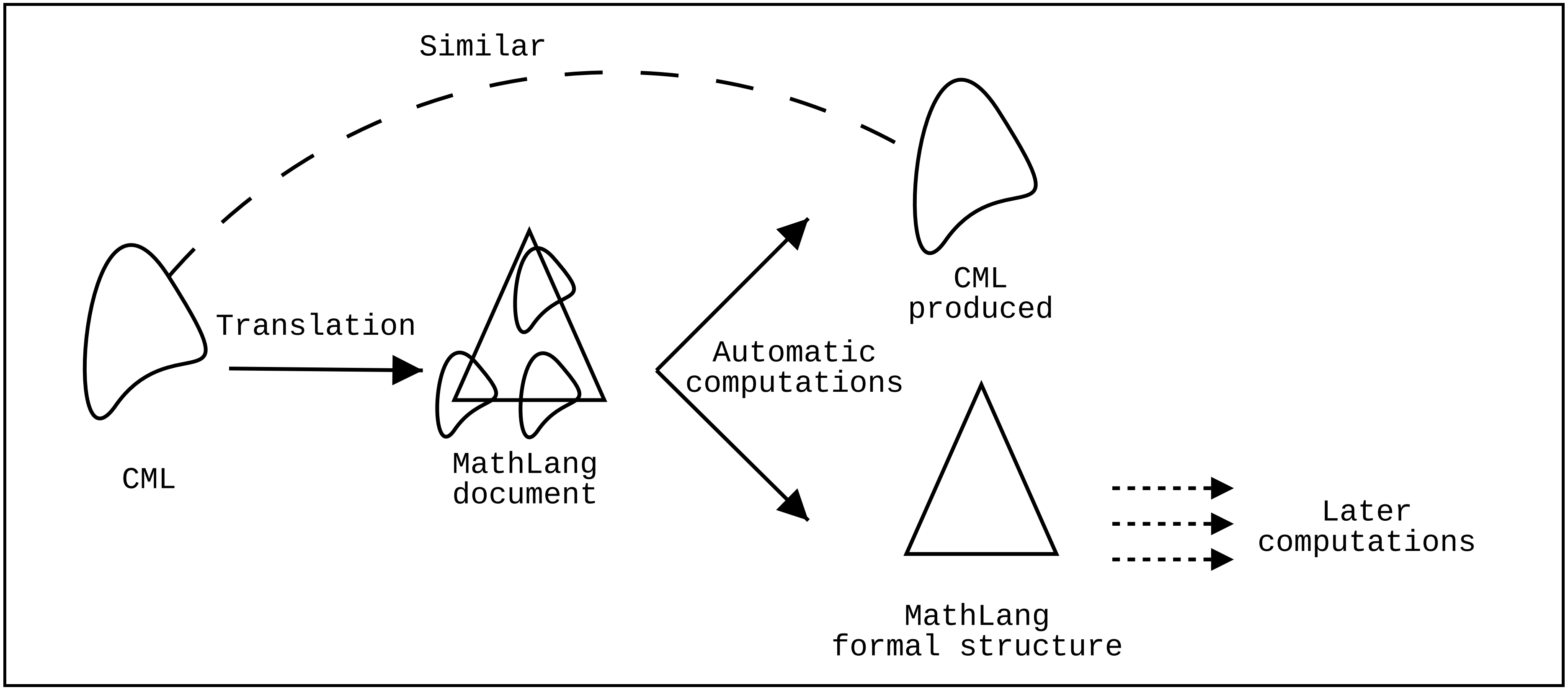
- A *weak type system* checks MathLang documents at a grammatical level.

- A MathLang text remains *close* to its CML original, allowing confidence that the CML has been captured correctly.

- We have been developing ways to weave natural language text into MathLang.

- MathLang aims to eventually support *all encoding uses*.

Process of translation into MathLang



- The CML view of a MathLang text should match the mathematician's intentions.
- The formal structure should be suitable for various automated uses.

Linguistic categories in WTT and MathLang

- At the *atomic* level, WTT has separate syntactic categories for *variables*, *constants*, and *binders*. The latest MathLang uses one syntactic category and instead distinguishes these roles via weak types.
- At the *phrase* level, there are *terms*, *sets*, *nouns*, and *adjectives*. (Manuel's talk will give details on how this is handled in the latest MathLang.)
- At the *sentence* level, there are *statements* and *definitions*.
- At the *discourse* level, WTT has *contexts*, *lines*, *books*, and *prefaces*. The latest MathLang replaces these by *blocks* and *scoping* operators.

Generally, each syntactic category has a corresponding *weak type*.

Examples of linguistic categories

- Terms: the triangle ABC; the center of $\boxed{ABC}$; $d(\boxed{x}, \boxed{y})$.
- Nouns: a triangle; an edge of $\boxed{\text{ABC}}$; a group.
- Adjectives: equilateral $\boxed{\text{triangle}}$; prime $\boxed{\text{number}}$; Abelian $\boxed{\text{group}}$.
- Statements: $\boxed{P}$ lies between $\boxed{Q}$ and $\boxed{R}$; $\boxed{5} \geq \boxed{3}$; $\boxed{\text{AB}}$ is $\boxed{\text{an edge of ABC}}$.
- Definition: a number p is prime whenever $\boxed{\cdots}$.

MathLang example

Definition 2. A *Fermat-sum* is a natural number which is the sum of two squares of natural numbers.

Lemma 3. The product of a square and a Fermat-sum is a Fermat sum.

In an older MathLang version, the above text could be translated as the following two *lines*:

$$a\ \mathit{Fermat\text{-}sum}\ := \mathtt{Noun}_{n \in \mathbb{N}} \exists_{k \in \mathbb{N}} \exists_{l \in \mathbb{N}} (n = k^2 + l^2)$$

$$\forall_{u:\ a\ square} \forall_{v:\ a\ \mathit{Fermat\text{-}sum}} (uv :\ \ a\ \mathit{Fermat\text{-}sum})$$

We can also give the following interesting *views* of this example.

MathLang example: Symbolic structure view

`Fermat-sum` () :=

Noun $n : \mathbb{N}$ $\exists\, k : \mathbb{N},\ \exists\, l : \mathbb{N},\ n = k^2 + l^2$ (1)

$\forall\, u : \textit{square},\ \forall\, v : \textit{Fermat-sum},\ u * v : \textit{Fermat-sum}$ (2)

MathLang example: CML view

Definition 4. [Fermat-sum]

A *Fermat-sum* is

a natural number which is the sum of two squares of natural numbers . 1

Lemma 5.

The product of a *square* and a *Fermat-sum* is a *Fermat-sum* . 2

Another MathLang example

T Terms | S Sets | N Nouns | P Statements | Z Declarations | Γ Context

Let $\mathfrak{M}$ be a set ,

y and x are natural numbers ,

if x belongs to $\mathfrak{M}$

then $x + y = y + x$

Another MathLang example: Type checking

T Terms	S Sets	N Nouns	P Statements	Z Declarations	Γ Context

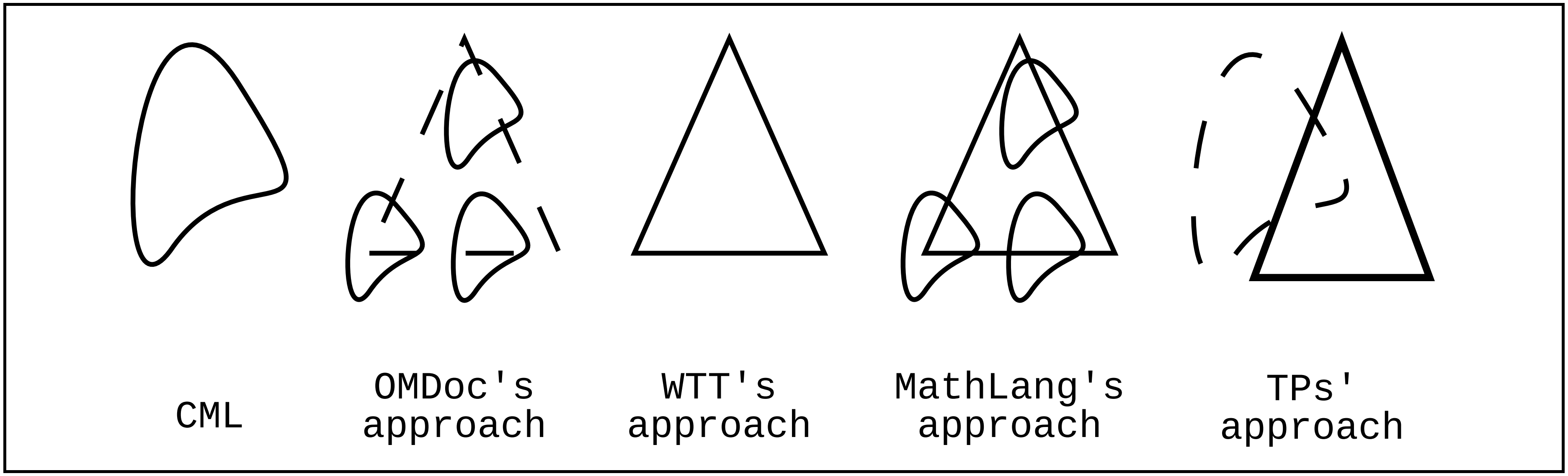
Let $\mathfrak{M}$ be a set ,

y and x are natural numbers ,

if x belongs to $\mathfrak{M}$

then $x + y$ $\Leftarrow$ **error**

Various approaches to representing mathematics



We visually summarize the approaches. Blobs represent natural language text whose structure is not understood by the computer. A broken blob is text maintained separately, not as part of the data structure. Triangles represent a tree-shaped structures understood by the computer. Solid triangles represent additional computer-checked well-formedness conditions. A heavy solid triangle represents full formalization.

Additional comparison with other related work

- *Galina* serves as a *command language* for Coq, aimed at full formalization.
- The *mathematical vernacular* of Ω*MEGA* gives $\textsc{Cml}$-like *views* of fully/partially formalized proofs.
- The basic languages of *Mizar* and Isar preserve the mathematical content. They are aimed at full formalization. Their syntax does not give the same expressive freedom to the mathematician as $\textsc{Cml}$.
- In the *Theorema project* computer algebra systems, the provers are designed to imitate the proof style humans employ in their proving attempts. The proofs can be produced in human-readable style. However, this is done by *post-processing* a fully formal proof.
- The typed functional programming language *GF* can define languages such as fragments of natural languages, programming languages, and formal calculi. GF is based on Martin-Löf's type theory.

Some points to consider

- We do not at all assume/prefer one type/logical theory instead of another.
- The formalisation of a language of mathematics should separate the questions:
  - *which type/logical theory is necessary for which part of mathematics*
  - *which language should mathematics be written in*.
- Mathematicians don't usually know or work with type/logical theories.
- Mathematicians usually *do* mathematics (manipulations, calculations, etc), but are not interested in general in reasoning *about* mathematics.

Conclusions

- The steps used for computerising books of mathematics written in English, as we are doing, can also be followed for books written in Arabic, French, German, or any other natural language.
- MathLang aims to support non-fully-formalized mathematics practiced by the ordinary mathematician as well as work toward full formalization.
- MathLang aims to handle mathematics as expressed in natural language as well as symbolic formulas.
- MathLang aims to do some amount of type checking even for non-fully-formalized mathematics. This corresponds roughly to grammatical conditions.
- MathLang aims for a formal representation of CML texts that closely corresponds to the CML conceived by the ordinary mathematician.

- MathLang aims to support automated processing of mathematical knowledge.
- MathLang aims to be independent of any foundation of mathematics.
- MathLang allows anyone to be involved, whether a mathematician, a computer engineer, a computer scientist, a linguist, a logician, etc.
- MathLang allows more accurate translation between different languages whithin the mathematical dictionary.

References

[1] C. Burali-Forti. Una questione sui numeri transfiniti. *Rendiconti del Circolo Matematico di Palermo*, 11:154–164, 1897. English translation in [18], pages 104–112.

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

[3] G. Cantor. Beiträge zur Begründung der transfiniten Mengenlehre (Zweiter Artikel). *Mathematische Annalen*, 49:207–246, 1897.

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

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

[6] A. Church. A set of postulates for the foundation of logic (2). *Annals of Mathematics*, 34:839–864, 1933.

[7] A. Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5:56–68, 1940.

[8] D.T. van Daalen. *The Language Theory of Automath*. PhD thesis, Eindhoven University of Technology, 1980.

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

[10] D.R. Dowty. *Introduction to Montague Semantics*. Kluwer Academic Publishers, 1980.

[11] G. Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Nebert, Halle, 1879. Also in [18], pages 1–82.

[12] G. Frege. *Grundlagen der Arithmetik, eine logisch-mathematische Untersuchung über den Begriff der Zahl*. , Breslau, 1884.

[13] G. Frege. *Grundgesetze der Arithmetik, begriffschriftlich abgeleitet*, volume I. Pohle, Jena, 1892. Reprinted 1962 (Olms, Hildesheim).

[14] G. Frege. Über Sinn und Bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, new series, 100:25–50, 1892.

[15] G. Frege. Ueber die Begriffschrift des Herrn Peano und meine eigene. *Berichte über die Verhandlungen der Königlich Sächsischen Gesellschaft der Wissenschaften zu Leipzig, Mathematisch-physikalische Klasse 48*, pages 361–378, 1896.

[16] G. Frege. Letter to Russell. English translation in [18], pages 127–128, 1902.

[17] G. Frege. *Grundgesetze der Arithmetik, begriffschriftlich abgeleitet*, volume II. Pohle, Jena, 1903. Reprinted 1962 (Olms, Hildesheim).

[Hea56] Heath. *The 13 Books of Euclid's Elements*. Dover, 1956.

[18] Heijenoort, J. v. (ed.): 1967, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Cambridge, Massachusetts: Harvard University Press.

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

[20] Kamareddine, F., L. Laan, and R. Nederpelt: 2002, 'Types in logic and mathematics before 1940'. *Bulletin of Symbolic Logic* **8**(2), 185–245.

[21] Kamareddine, F., and R. Nederpelt: 2004, A refinement of de Bruijn's formal language of mathematics. Journal of *Logic, Language and Information*. Kluwer Academic Publishers.

[22] Kamareddine, F., Maarek, M., and Wells, J.B.: 2004, MathLang: An experience driven language of mathematics, Electronic Notes in Theoretical Computer Science 93C, pages 123-145. Elsevier.

[23] F. Kamaredine, M Maarek, and J.B. Wells. Flexible encoding of mathematics on the computer. 2004.

[24] F. Kamareddine, T. Laan, and R. Nederpelt. Revisiting the notion of function. *Logic and Algebraic programming*, 54:65–107, 2003.

[Lan30] Edmund Landau. *Grundlagen der Analysis*. Chelsea, 1930.

[Lan51] Edmund Landau. *Foundations of Analysis*. Chelsea, 1951. Translation of [Lan30] by F. Steinhardt.

[25] MacLane, S.: 1972, *Categories for the Working Mathematician*. Springer.

[26] G. Peano. *Arithmetices principia, nova methodo exposita*. Bocca, Turin, 1889. English translation in [18], pages 83–97.

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

[28] F.P. Ramsey. The foundations of mathematics. *Proceedings of the London Mathematical Society*, 2nd series, 25:338–384, 1926.

[29] J.B. Rosser. Highlights of the history of the lambda-calculus. *Annals of the History of Computing*, 6(4):337–349, 1984.

[30] B. Russell. Letter to Frege. English translation in [18], pages 124–125, 1902.

[31] B. Russell. *The Principles of Mathematics*. Allen & Unwin, London, 1903.

[32] B. Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30:222–262, 1908. Also in [18], pages 150–182.

[33] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924. Also in [18], pages 355–366.

[34] Whitehead, A. and B. Russell: 1910^1, 1927^2, *Principia Mathematica*, Vol. I, II, III. Cambridge University Press.

[35] Zermelo, E.: 1908, 'Untersuchungen über die Grundlagen der Mengenlehre'. *Math. Annalen* **65**, 261–281.