Bootstrapping Mathematics

Masahiko Sato

Graduate School of Informatics, Kyoto University

Mathematical Logic:
Development and Evolution into Various Sciences
Kanazawa, Japan
March 9, 2012

Contents

What is formalized mathematics?

- A formalized mathematics is *written* in a formal language.
- Syntax of the language is formally given by, e.g., a context-free grammar.
- Mathematical objects are *represented* by linguistic entities such as nouns.
- Mathematical assersions (propositions) are represented by formulas, which are also linguistic objects.
- Proofs are also formally written in the formal language.
- Given any formula and (formal) proof, it is primitive recursively *decidable* if the proof proves the formula.

Note that there is no formal definition of formalized mathematics. (Cf. Church's Thesis.)

Why formalize mathematics?

- Motivations coming from mathematics.
- Motivations coming from computer science.

Why formalize mathematics? (cont.)

Motivations coming from mathematics

- Proof of unprovability of a proposition.
- Consistency proof.
- Gödel's incompleteness theorem.
- Reverse mathematics.
- Zermelo-Fraenkel set theory.

These motivations are mainly theoretical. Mathematicians usually *talk about* formalized mathematics but *not work in* it.

Formalization of logic is important here.

Why formalize mathematics? (cont.)

Motivations coming from computer science

- Verification of proofs.
- Verification of programs.
- Constructive programming.
- Formalization of metamathematics.

These motivations are mainly practical. Some computer scientists are interested in creating a computer environment for *doing* mathematics in it.

Cf., Isabelle, Coq, Agda, Minlog etc.

Formalization of computation is important here.

History of formalization

- Frege (Begriffsschrift, 1879) Higher order logic, Natural deduction
- Russell (Principia Mathematica (with Whitehead), 1910) Type theory
- Brouwer (Intuitionism) $\longrightarrow$ Heyting
- Hilbert (Formalism) $\longrightarrow$ Gödel, Gentzen
- Zermelo-Fraenkel (Set theory)
- Church ($\boldsymbol{\lambda}$-calculus, Simple theory of types)
- Turing (universal Turing machine, decision problem)
- McCarthy (1961: A basis for mathematical theory of computation) $\longrightarrow$ Milner
- de Bruijn (Automath 1967 –)
- Mizar (1973 –), Coq, Isabelle, Minlog, Agda

Quotation from McCarthy

(1961: A basis for mathematical theory of computation)

Proof-checking by computer may be as important as proof generation. It is part of the definition of formal system that proofs be checkable.

Because a machine can be asked to do much more work in checking a proof than can a human, proofs can be made much easier to write in such systems. In particular, proofs can contain a request for the machine to explore a tree of possibilities for a conventional proof.

The potential applications for computer-checked proofs are very large. For example, instead of trying out computer programs on test cases until they are debugged, one should prove that they have the desired properties.

Quotation from McCarthy (cont.)

(1961: A basis for mathematical theory of computation)

The usefulness of computer checked proofs depends both on the development of types of formal systems in which proofs are easy to write and on the formalization of interesting subject domains.

It should be remembered that the formal systems so far developed by logicians have heretofore quite properly had as their objective that it should be convenient to prove metatheorems about the systems rather than that it be convenient to prove theorems in the systems.

Motivation for bootstrapping mathematics

- Mathematics is human linguistic activity.
- Acceptance of proofs by mathematicians is a social process.
- Hilbert's idea: Finitism and Formalism.
- Finitism = real mathematics = mathematics with meaning
- Formalism = ideal mathematics = mathematics without meaning

Feasibility of bootstrapping mathematics

- The notion of computable function is a stable notion.
- Bootstrap finitary mathematics.
- Then, formalize ideal mathematics as formal systems which are objects of real mathematics.
- In this way, we can mechanically check all the mathematics we create.

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism
Comp. Sci.	Denotational semantics	Operational semantics	Axiomatic semantics

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism
Comp. Sci.	Denotational semantics	Operational semantics	Axiomatic semantics
Ontology	Strong	Weak	Weakest

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism
Comp. Sci.	Denotational semantics	Operational semantics	Axiomatic semantics
Ontology	Strong	Weak	Weakest
Computation	Neglected	Essential	Essential

Ontology concerns what and computation concerns how.

Mathematical Objects

In mathematics we talk about mathematical objects, but what are mathematical objects and how they are constructed?

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism
Comp. Sci.	Denotational semantics	Operational semantics	Axiomatic semantics
Ontology	Strong	Weak	Weakest
Computation	Neglected	Essential	Essential

Ontology concerns what and computation concerns how.

We classify mathematical objects into the following two kinds.

1. Mathematical objects of the first kind.
2. Mathematical objects of the second kind.

Mathematical Objects (cont.)

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism
Comp. Sci.	Denotational semantics	Operational semantics	Axiomatic semantics
Ontology	Strong	Weak	Weakest
Computation	Neglected	Essential	Essential

Mathematical Objects (cont.)

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism
Comp. Sci.	Denotational semantics	Operational semantics	Axiomatic semantics
Ontology	Strong	Weak	Weakest
Computation	Neglected	Essential	Essential
Classification of Math Objects	Set	Type	Class

How can we bootstrap mathematics?

How can we teach first-order Peano Arithmetic to someone who knows nothing about mathematics?

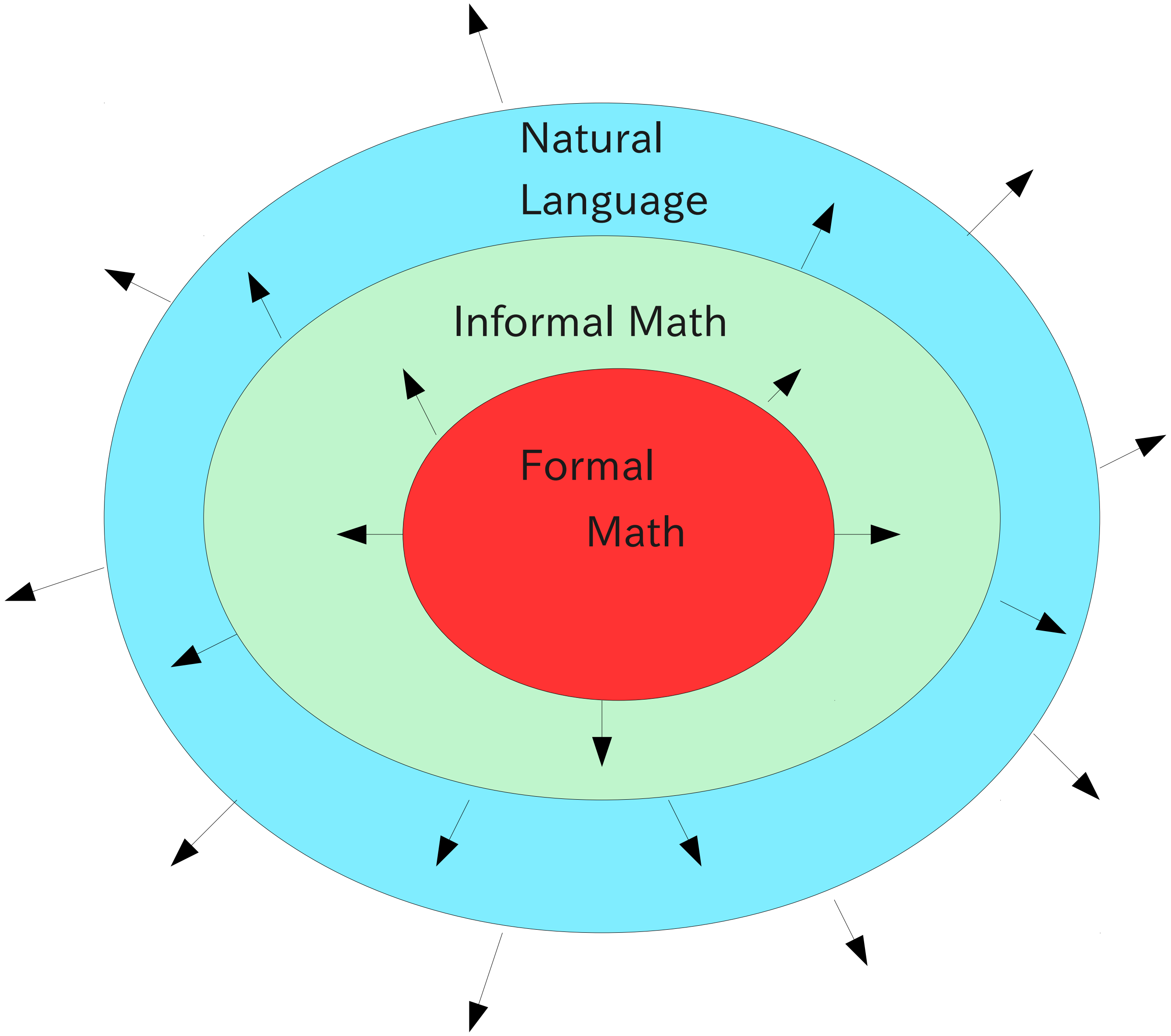
1. $x = x$.
2. $x = y \supset y = x$.
3. $x = y \supset y = z \supset x = z$.
4. $x = y \supset \mathrm{S}(x) = \mathrm{S}(y)$.
5. $\mathrm{S}(x) \neq \mathbf{0}$.
6. $\mathrm{S}(x) = \mathrm{S}(y) \supset x = y$.
7. $x + \mathbf{0} = x$.
8. $x + \mathrm{S}(y) = \mathrm{S}(x + y)$.
9. $x * \mathbf{0} = \mathbf{0}$.
10. $x * \mathrm{S}(y) = x * y + x$.
11. $P(\mathbf{0}) \wedge (\forall x.P(x) \supset P(\mathrm{S}(x))) \supset \forall x.P(x)$.

How can we bootstrap mathematics? (cont.)

We provide a framework in which we can formalize real mathematics.

We will treat finitary objects as objects of the first kind.

See the following picture.



My view of mathematics

Inadequacy of type theories for bootstrapping

There are at least 5 reasons why type theories are inadequate for bootstrapping mathematics. (Of course, these theories are good for other purposes.)

1. Type : Type does not hold. In my framework Class : Class does hold.
2. The notion of 'subtype' is difficult to handle, compared to the notion of 'subset' or 'subclass'.
3. Quotation and evaluation are not fully supported.
4. Cannot define partial functions.
5. Logically complex. (For example, to use a type theory to formalize Peano arithmetic is an overkill.)

A framework for doing mathematics in it

We begin with collecting key concepts to be used in building such a framework.

We view mathematics as human collaboration of linguistic activity. We will create a programming environment (called Natural Framework) to support such a linguistic activity.

- Object
- Creation of objects
- Name (Symbol)
- Naming
- Notion of 'as'
- Class
- Classification

Objects of the first kind

Mathematical objects of the first kind are constructed by the fundamental priciple of object creation:

- An object of the first kind is created from finitely many already created objects of the first kind.
- The creation is done by applying a creation method to existing objects.
- Both the creation method and the created object belongs to a specific class.
- The class is called the mother class of the created object.
- Thus, any object is created as an instance of its mother class.
- The equality relation $(=)$ on objects of the first kind is called the equality of the first kind.

Objects of the first kind (cont.)

Objects of the first kind are created by the fundamental principle of object creation:

> *Every object $\boldsymbol{a}$ is created from already created $\boldsymbol{n}$ objects $\boldsymbol{a_1, \ldots, a_n}$ $(\boldsymbol{n \geq 0})$ by applying a creation method M .*

We can visualize this *act* of creation by the following figure:

$$\frac{\boldsymbol{a_1} \quad \cdots \quad \boldsymbol{a_n}}{\boldsymbol{a}}\ M$$

or, by the equation:

$$\boldsymbol{a} = M\,(\boldsymbol{a_1, \ldots, a_n})$$

The method M is a partial function, in general, but it is decidable whether M may be applied to given obects.

Objects of the first kind (cont.)

Equality and inequality relation on objects are defined simultaneously with the creation of objects.

Two objects:

$$M\,(\boldsymbol{a_1, \ldots, a_m}) \text{ and } N\,(\boldsymbol{b_1, \ldots, b_n})$$

are equal $(=)$ if and only if M and N are the same method, $\boldsymbol{m = n}$ and $\boldsymbol{a_i = b_i}$ $(\boldsymbol{1 \leq i \leq m})$.

Objects of the first kind (cont.)

Equality and inequality relation on objects are defined simultaneously with the creation of objects.

Two objects:

$$\mathit{M}\,(a_1, \ldots, a_m) \text{ and } \mathit{N}\,(b_1, \ldots, b_n)$$

are equal $(=)$ if and only if M and N are the same method, $m = n$ and $a_i = b_i$ $(1 \leq i \leq m)$.

In other words, two objects are equal if they are created in exactly the same way, and the equality relation is decidable.

Objects of the first kind (cont.)

Equality and inequality relation on objects are defined simultaneously with the creation of objects.

Two objects:

$$M\,(\boldsymbol{a_1, \ldots, a_m}) \text{ and } N\,(\boldsymbol{b_1, \ldots, b_n})$$

are equal ($=$) if and only if M and N are the same method, $\boldsymbol{m = n}$ and $\boldsymbol{a_i = b_i}$ $(\boldsymbol{1 \leq i \leq m})$.

In other words, two objects are equal if they are created in exactly the same way, and the equality relation is decidable.

However, there is only one excecption: *abstractions* which are objects of the class $\langle \texttt{Abs} \rangle$.

Object and Class

In type theory:

$$\mathbf{3} : \mathsf{Nat}$$

is a *judgment*, $\mathbf{3}$ is an *object*, and Nat is a *type*. Here, ‘judgment’, ‘object’ and ‘type’ are concepts used when we talk about type theory in the meta language. These concepts do not have counter-parts in type theory.

Object and Class (cont.)

In NF class theory,

$$\mathbf{3} : \langle \mathtt{Nat} \rangle$$

is a *proposition*, $\mathbf{3}$ is an *object*, and $\langle \mathtt{Nat} \rangle$ is a *class*. Here, 'judgment', 'object' and 'type' are concepts used when we talk about NF in the meta language. These concepts do have counter-parts in NF class theory.
In fact, we have:

$$\begin{aligned}
(\mathbf{3} : \langle \mathtt{Nat} \rangle) &: \langle \mathtt{Prop} \rangle, \\
\mathbf{3} &: \langle \mathtt{Object} \rangle, \\
\langle \mathtt{Nat} \rangle, \langle \mathtt{Prop} \rangle, \langle \mathtt{Object} \rangle, \langle \mathtt{Class} \rangle &: \langle \mathtt{Class} \rangle, \\
\langle \mathtt{Nat} \rangle, \langle \mathtt{Prop} \rangle, \langle \mathtt{Object} \rangle, \langle \mathtt{Class} \rangle &: \langle \mathtt{Object} \rangle, \\
`:\text{'} &: \langle \mathtt{Function} \rangle.
\end{aligned}$$

Object and Class (cont.)

In type theory, fundamental concetps in type theory such as *object*, *type* and *judgment* are not internalized. We can use these concepts only at metalevel and not in the theory itself.

In NF class theory, fundamental concetps in class theory such as *object*, *class* and *proposition* are all internalized. We can use these concepts both at metalevel and in the class theory.

This is possible since NF is designed so that key mathematical concepts can be easily internalized by making a one-to-one correspondece between these metalevel concepts and internal mother classes.

Moreover, since NF is a formal sublanguage of English, it supports real mathematics formally but can be understood directly by humans without further interpretations.

Basic concepts

If we try to bootstrap mathematics from scratch, we must start from somewhere, where we have some intuitive understanding of basic concepts which are, perhaps, absolutely necessary to construct mathematics from them.

Here, we propose the following four *basic concepts*:

Name, *Tuple*, *Function* and *Class*

Tuple

Given any finite sequence of objects $a_1, \ldots, a_n$, we we can form a *tuple*

$$(a_1 \ \ldots \ a_n)$$

of these objects.

Tuples are created by the following two methods.

$$\frac{}{(\texttt{Tuple/nil}) : \langle\texttt{Tuple}\rangle}\ \texttt{Tuple/nil}$$

$$\frac{a : \langle\texttt{Object}\rangle \quad b : \langle\texttt{Tuple}\rangle}{(\texttt{Tuple/cons}\ a\ b) : \langle\texttt{Tuple}\rangle}\ \texttt{Tuple/cons}$$

Function

A *function* is either a primitive function or a defined funtion.
Functions are identified by its name, which is a symbol.
A function is called a *method*, if it can be used to create new objects from existing objects.

Class

A *class* is either a mother class or the top class.
Each class is identified by its name, which is symbol whose first and last character is an angular bracket.
For example, we will write $\langle$`Object`$\rangle$ for the top class.
The initial four mother classes are: $\langle$`Tuple`$\rangle$, $\langle$`Symbol`$\rangle$, $\langle$`Function`$\rangle$ and $\langle$`Class`$\rangle$.

Bootstrapping Natural Framework (NF)

Bootstrapping of NF is done in two steps.

Each object is created interacting with the surrounding universe. The universe is empty initially.

Who creates objects?

In the first step, objects are created by the program `boot.el` written in Emacs Lisp.

In the second step, objects are created by the program `boot.ez` written in Ez which is a language inside the universe of NF.

The bootstrapping process finishes after the loading of these two programs. At this point, the universe is nonempty and the user can directly manipulate and modify the universe by writing programs in Ez (which is part of the universe).

Conclusions

- Hilbert's idea: there are two kinds of mathematics.
  - The first is real mathematics based on the finitistic view of mathematics.
  - The second is ideal mathematics based on the formalistic view of mathematics.
  - Most of platonistic and intuitionistic mathematics can be formalized.
- The notion of computability is stable and accepted by all mathematicians.
  - Proof checking is a decidable process.
  - So, implementing proofs naturally is important
- NF realizes a natural framwork for doing both real mathematics and ideal mathematics.