Formalization of Mathematics

Masahiko Sato

Graduate School of Informatics, Kyoto University

Workshop on Logic and Computation
Kanazawa, Japan
February 8, 2011

Contents

- My view of mathematics
- What is formalized mathematics?
- Why formalize mathematics?
- History of formalization
- Mathematical Objects
- Our approach

My view of mathematics

Mathematics is a human linguistic activity.

- Mathematics has been and will be developed by humans.
- Mathematics is communicated by language.
- Mathematics always changes dynamically.
- Mathematics is open ended.

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

- Verifcation of proofs.
- Verifcation 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 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)
- de Bruijn (Automath 1967 $-$)
- Mizar (1973 $-$)
- Coq
- Isabelle
- Agda

History of formalization (cont.)

- Decimal notation for natural numbers (0, 1, ...)
- Use of symbols for functions on numbers ($+, -$ etc.)
- Use of equality sign ($=$) for the equality proposition.
- Use of "$f(x)$" for function application.
- Use of TEX, LATEX for writing mathematical papers.
- Use of CAS (computer algebra system) by mathematicians.
- Use of proof assistance system by mathematicians is not yet realized.

History of formalization (cont.)

- Decimal notation for natural numbers (0, 1, ...)
- Use of symbols for functions on numbers ($+, -$ etc.)
- Use of equality sign ($=$) for the equality proposition.
- Use of "$f(x)$" for function application.
- Use of TEX, LATEX for writing mathematical papers.
- Use of CAS (computer algebra system) by mathematicians.
- Use of proof assistance system by mathematicians is not yet realized.

We can observe that what mathematicians consider to be mathematical objects have formal notations. In other word, mathematicians today do not consider proofs and most of propostions to be mathematical objects.

Mathematics is already partially formalized, and I call it semi-formalization.

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
Math Objects	Set	Type	Class

Objects of the first kind

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:

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

Objects of the first kind (cont.)

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

Two objects:

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

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:

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

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:

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

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.

Moreover, given a creation method M and a sequence of (already created) objects, it is decidable whether M may be applied to these objects to create a new object. (Decidability of side condition.)

Objects of the first kind (cont.)

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 second kind

Let C be a class whose members are objects of the first kind, and let $=_2$ be a (partial) equivalence relation on C.

We can obtain objects of the second kind by identifying a and b in C if $a =_2 b$. When $=_2$ is a partial equivalence relation, an object a of the first kind in C is considered to be an object of the second kind if $a =_2 a$ holds.

In this setting, functions and relations on these objects must be defined so that the equality $=_2$ becomes congruence relation with respect to these functions and relations.

Well-definedness of these functions and relations are sometimes nontrivial.

Also, inductive arguments are not as smooth as for objects of the first kind, or even impossible.

Objects of the second kind (cont.)

Example: Rational numbers.

Let $\mathbb{Z}$ be the class of integers, and let $\mathbb{Z} \times \mathbb{Z}$ be the class whose members are a/b where $a, b \in \mathbb{Z}$. Define $=_2$ on $\mathbb{Z} \times \mathbb{Z}$ by:

$$a/b =_2 c/d \iff ad = bc \text{ and } b \neq 0 \text{ and } d \neq 0$$

We can define addition $(+)$ on rational numbers by putting:

$$a/b + c/d := (ad + bc)/bd.$$

This is a well-defined operation, since we have $a/b + c/d =_2 a'/b' + c'/d'$ if $a/b =_2 a'/b'$ and $c/d =_2 c'/d'$.

However, taking the denominator of a rational number is not a well-defined function on rational numbers. That is, from $1/2 =_2 2/4$, it does not follow that $2 = 4$.

Our way of formalization

- Define formal language as a subset of natural language (such as English).
- The formalized language is reflective, and can talk about itself easily and naturally.
- Formalize minimal mathematics which can talk about and compute *any* formal system.
- We do not formalize full mathematics directly but do this indirectly by formalizing finite and symbolic mathematics which naturally includes metamathematics.
- We can then develop any mathematics within this framework.

Basic concepts

Even if we try to boot strap 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*:

Tuple, Symbol, 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.

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

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

Symbol

A *symbol* is used as a name of some objects.
For example, we can use the symbol `one` as a name of the natural number 1.
We assume that for any objects, there is a uniquely determined symbol.

$$\frac{\boldsymbol{a} : \langle \texttt{object} \rangle}{(\texttt{Symbol/symbol}\ \boldsymbol{a}) : \langle \texttt{Symbol} \rangle}\ \texttt{Symbol/symbol}$$

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 the bottom 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 ⟨`object`⟩ and ⟨⟩ for the top and the bottom class, respectively.
The initial four mother classes are: ⟨`Tuple`⟩, ⟨`Symbol`⟩, ⟨`Function`⟩ and ⟨`Class`⟩.

Primitive Methods

```
    Tuple/nil :                                          → <Tuple>
   Tuple/cons :                <object> <Tuple>          → <Tuple>

 Symbol/symbol :                             <object>    → <Symbol>

 Function/prim :                             <Symbol>    -→ <Function>
Function/defun : <Symbol> <Tuple> <Tuple>    -→ <Function>

    Class/top :                                          → <Class>
  Class/class :                              <Symbol>    -→ <Class>
Class/subclass : <Symbol> <Tuple> <Class>    -→ <Class>
```

Analysis of objects

Any object can be completely analyzed by the following primitive functions.

$$\texttt{mother-name} : \langle \texttt{object} \rangle \to \langle \texttt{Symbol} \rangle$$
$$\texttt{method-name} : \langle \texttt{object} \rangle \to \langle \texttt{Symbol} \rangle$$
$$\texttt{body} : \langle \texttt{object} \rangle \to \langle \texttt{Tuple} \rangle$$

The case function is implemented by these methods.
We can prove properties of objects by induction on the creation of objects.

Natural Number

The class $\langle \texttt{Nat} \rangle$ of natural numbers has two methods.

$$\frac{}{(\texttt{Nat/zero}) : \langle \texttt{Nat} \rangle}\ \texttt{Nat/zero}$$

$$\frac{\boldsymbol{n} : \langle \texttt{Nat} \rangle}{(\texttt{Nat/succ}\ \boldsymbol{n}) : \langle \texttt{Nat} \rangle}\ \texttt{Nat/succ}$$

Error

The class $\langle \texttt{Error} \rangle$ is defined by the following methods.

$$\frac{}{(\texttt{Error/diverge}\ \boldsymbol{a}) : \langle \texttt{Error} \rangle}\ \texttt{Error/diverge}$$

$$\frac{\boldsymbol{a} : \langle \texttt{object} \rangle}{(\texttt{Error/error}\ \boldsymbol{a}) : \langle \texttt{Error} \rangle}\ \texttt{Error/error}$$

Environment

An environmet $(\boldsymbol{\Gamma}, \boldsymbol{\Delta}, \ldots)$ a meta object keeping the assignment of objects to a finite number of symbols.
If $\boldsymbol{\Gamma}$ assigns an object $\boldsymbol{v}$ to a symbol $\boldsymbol{x}$, we will write $\boldsymbol{\Gamma}(\boldsymbol{x})$ for $\boldsymbol{v}$.

An environment also has the collection of objects created so far.

An environment changes by the cycle of evaluation described in the next slide.

Evaluation

An evaluation is a cycle of the followng steps. Before starting a cycle, the system is charcterized by its current environment $\boldsymbol{\Gamma}$.

- Read in an expression, and convert it into an object $\boldsymbol{a}$.
- The object $\boldsymbol{a}$ is evaluated, and if the evaluation succeeds, a value $\boldsymbol{v}$ and a new environment $\boldsymbol{\Delta}$ is obtained.
- The value is printed as an expression.

We write this cycle as follows.

$$\boldsymbol{\Gamma} \vdash \boldsymbol{a} \Downarrow \boldsymbol{v} \dashv \boldsymbol{\Delta}$$

Objects are created during the process of evaluation.
After the evaluation, all the assignments in $\boldsymbol{\Gamma}$ is kept in $\boldsymbol{\Delta}$, and may contain new assignments.

Evaluation (cont.)

There are two cases where the evaluation of an object fails.

- The evaluation continues forever.
- Reference to a symbol unbound in the current enviroment.

In these cases, the failed object $\boldsymbol{a}$ is said to be meaningless. After the failed evaluation of a meaningless object, the environment $\boldsymbol{\Gamma}$ in which $\boldsymbol{a}$ is evaluated is restored.

The evaluation relation has the following properties.

- If $\boldsymbol{\Gamma} \vdash \boldsymbol{a} \Downarrow \boldsymbol{v} \dashv \boldsymbol{\Delta}$, then $\boldsymbol{\Gamma} \subseteq \boldsymbol{\Delta}$.
- If $\boldsymbol{\Gamma} \vdash \boldsymbol{a} \Downarrow \boldsymbol{v} \dashv \boldsymbol{\Delta}$ and $\boldsymbol{\Gamma} \subseteq \boldsymbol{\Gamma'}$, then $\boldsymbol{\Gamma'} \vdash \boldsymbol{a} \Downarrow \boldsymbol{v} \dashv \boldsymbol{\Delta'}$ for some $\boldsymbol{\Delta} \subseteq \boldsymbol{\Delta'}$.

Evaluation (cont.)

When $\Gamma \vdash a \Downarrow v \dashv \Delta$ holds, the relation

$$\Gamma \rightarrow_a \Delta$$

also holds. We read the relation: Evaluation of a in Γ creates Δ.
We write $\Gamma_1 \rightarrow_{a_1,\ldots,a_n} \Gamma_n$ for the obvious compositions of this relation.
This relation has the following confluence property (after some renaming of symbols not in Γ).

If

$$\Gamma \rightarrow_{a_1,\ldots,a_m} \Pi \text{ and } \Gamma \rightarrow_{b_1,\ldots,b_n} \Sigma,$$

then

$$\Pi \rightarrow_{b_1,\ldots,b_n} \Delta \text{ and } \Sigma \rightarrow_{a_1,\ldots,a_m} \Delta$$

for some Δ.

Abstract

We have a class ⟨logicvar⟩ of logic variables which is a subclass of ⟨Symbol⟩.
The class ⟨Abs⟩ of abstracts is defined by the following method.

$$\frac{x : \langle \texttt{logicvar} \rangle \quad a : \langle \texttt{object} \rangle}{(\texttt{Abs/abs}\ x\ a) : \langle \texttt{Abs} \rangle}\ \texttt{Abs/abs}$$

The application of the method succeeds only when the height of x in a is x.

Proposition

We have only one atomic proposition of equalily from which all the propositions are constructed.

$$\frac{\boldsymbol{a} : \langle \texttt{object} \rangle \quad \boldsymbol{b} : \langle \texttt{object} \rangle}{(\texttt{Prop}/= \boldsymbol{a}\ \boldsymbol{b}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop}/=$$

It is decidable whether a given object $\boldsymbol{a}$ is a proposition.

Each proposition will be either meaningless or has the truth value true or false.

The equality proposition above is meaningful iff both $\boldsymbol{a}$ and $\boldsymbol{b}$ are meaningful. It is then true of the values of $\boldsymbol{a}$ and $\boldsymbol{b}$ are eaual.

Conjunction

$$\frac{\boldsymbol{A} : \langle \texttt{Prop} \rangle \quad \boldsymbol{B} : \langle \texttt{Prop} \rangle}{(\texttt{Prop}/\wedge\ \boldsymbol{A}\ \boldsymbol{B}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop}/\wedge$$

$\boldsymbol{A} \wedge \boldsymbol{B}$ is meaningful if both $\boldsymbol{A}$ and $\boldsymbol{B}$ are meningful.
A meaningful conjuction is true iff both $\boldsymbol{A}$ and $\boldsymbol{B}$ are true.

Disjunction

$$\frac{\boldsymbol{A} : \langle \texttt{Prop} \rangle \quad \boldsymbol{B} : \langle \texttt{Prop} \rangle}{(\texttt{Prop}/\vee\ \boldsymbol{A}\ \boldsymbol{B}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop}/\vee$$

$\boldsymbol{A} \vee \boldsymbol{B}$ is meaningful if both $\boldsymbol{A}$ and $\boldsymbol{B}$ are meningful.
A meaningful disjunction is true iff either $\boldsymbol{A}$ or $\boldsymbol{B}$ is true.

Implication

$$\frac{\boldsymbol{A} : \langle \texttt{Prop} \rangle \quad \boldsymbol{B} : \langle \texttt{Prop} \rangle}{(\texttt{Prop}/\supset\ \boldsymbol{A}\ \boldsymbol{B}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop}/\supset$$

$\boldsymbol{A} \supset \boldsymbol{B}$ is meaningful if both $\boldsymbol{A}$ and $\boldsymbol{B}$ are meningful.
A meaningful implicaiton is true iff $\boldsymbol{B}$ is true whenever $\boldsymbol{A}$ is true.

Universal quantification

$$\frac{\boldsymbol{A} : \langle \texttt{Abs} \rangle}{(\texttt{Prop}/\forall\ \boldsymbol{A}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop}/\forall$$

The application of this method succeeds only when $\boldsymbol{A}$ is an abstract of the form $(\texttt{Abs}/\texttt{abs}\ \boldsymbol{x}\ \boldsymbol{B})$ where $\boldsymbol{B}$ is a proposition.

We write $\forall \boldsymbol{x}.\ \boldsymbol{B}[\boldsymbol{x}]$ for the above proposition.

This propostion is meaningful in $\boldsymbol{\Gamma}$ if $\boldsymbol{B}[\boldsymbol{a}]$ is meaningful in any extension $\boldsymbol{\Delta}$ of $\boldsymbol{\Gamma}$ and for any object $\boldsymbol{a}$ in $\boldsymbol{\Delta}$.

If $\forall \boldsymbol{x}.\ \boldsymbol{B}[\boldsymbol{x}]$ is meaningful, then it is true iff if $\boldsymbol{B}[\boldsymbol{a}]$ is true in any extension $\boldsymbol{\Delta}$ of $\boldsymbol{\Gamma}$ and for any object $\boldsymbol{a}$ in $\boldsymbol{\Delta}$.

Existential quantification

$$\frac{\boldsymbol{A} : \langle \texttt{Abs} \rangle}{(\texttt{Prop}/\exists\ \boldsymbol{A}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop}/\exists$$

The application of this method succeeds only when $\boldsymbol{A}$ is an abstract of the form $(\texttt{Abs}/\texttt{abs}\ \boldsymbol{x}\ \boldsymbol{B})$ where $\boldsymbol{B}$ is a proposition.

We write $\exists \boldsymbol{x}.\ \boldsymbol{B}[\boldsymbol{x}]$ for the above proposition.

This propostion is meaningful in $\boldsymbol{\Gamma}$ if $\boldsymbol{B}[\boldsymbol{a}]$ is meaningful in any extension $\boldsymbol{\Delta}$ of $\boldsymbol{\Gamma}$ and for any object $\boldsymbol{a}$ in $\boldsymbol{\Delta}$.

If $\forall \boldsymbol{x}.\ \boldsymbol{B}[\boldsymbol{x}]$ is meaningful, then it is true iff $\boldsymbol{B}[\boldsymbol{a}]$ is true in some extension $\boldsymbol{\Delta}$ of $\boldsymbol{\Gamma}$ and for some object $\boldsymbol{a}$ in $\boldsymbol{\Delta}$.

Proof

The class ⟨Proof⟩ has the following method for making an assumption.

$$\frac{x : \langle \texttt{logicvar} \rangle \quad A : \langle \texttt{Prop} \rangle}{(\texttt{Proof/var}\ x\ A) : \langle \texttt{Proof} \rangle}\ \texttt{Proof/var}$$

For each proof a, we define the list $(\texttt{hyp}\ a)$ of assumptions on which the proof depends, and also the conclusion $(\texttt{ccl}\ a)$ of assumptions on which of the proof.

In case of the above method, they are $[A]$ and A.
For example, the

Proof (cont.)

The following method discharges an assumption from a proof.

$$\frac{\boldsymbol{A} : \langle \texttt{Pfop} \rangle \quad \boldsymbol{B} : \langle \texttt{Abs} \rangle}{(\texttt{Prop/} \supset \texttt{I}\ \boldsymbol{A}\ \boldsymbol{B}) : \langle \texttt{Prop} \rangle}\ \texttt{Prop/} \supset \texttt{I}$$

The application of this method succeeds only when $\boldsymbol{B}$ is an abstract of the form $(\texttt{Abs/abs}\ \boldsymbol{x}\ \boldsymbol{b})$ where $\boldsymbol{b}$ is a proof and the height of $\boldsymbol{x}$ in $\boldsymbol{b}$ is $\boldsymbol{x}$ and moreover each occurrence of $\boldsymbol{x}$ in $\boldsymbol{b}$ is used as a proof variable for $\boldsymbol{A}$.

Basic judgements

- $e \downarrow a$ (e *evaluates to* a.)
- $a : c$. (a *is an instance of* c.)

Summary

- Initially, there is no formal object.
- The universe of formal objects is extended by the act of applying a creation method to existing objects.
- Thus, the universe is extended by the above unit action which adds one new object to the universe.
- Hence, the universe has only finite number of objects at any moment.
- The future of the universe is open ended and it is extended in a non-deterministic way.
- In other word, we can imagine the tree of possible mathematics, and we mathematicians are walking on the tree by choosing a branch in each step.