Platonism with a Flavor of Constructivism

Masahiko Sato

Graduate School of Informatics, Kyoto University

Workshop on Constructivism: Logic and Mathematics

May 26, 2008

Motivation

Computer assistance of human mathematical *activities*.

- Formalization of mathematics and metamathematics
- Proof assistance on a computer
- Comparison of various frameworks
- NF (Natural Framework) as meta-frameworks

Motivation (cont.)

Proofs and propositions as mathematical objects are to be considered here.

Some previous attempts:

- Constructive validity ([Scott 1970])
- Propositions as types ([Martin-Löf 1972])
- Frege structure ([Aczel 1980])
- Frege structure with proof objects ([Sato 1991])

Formalist, Constructivist and Platonist

What is a proof?

Formalist, Constructivist and Platonist

What is a proof?

- For a formalist, a proof is

Formalist, Constructivist and Platonist

What is a proof?

- For a formalist, a proof is just a natural number (Gödel).
- For a constuctivist, a proof is

Formalist, Constructivist and Platonist

What is a proof?

- For a formalist, a proof is just a natural number (Gödel).
- For a constuctivist, a proof is a computable function (Bishop and many others).
- For a platonist, a proof is

Formalist, Constructivist and Platonist

What is a proof?

- For a formalist, a proof is just a natural number (Gödel).
- For a constustivist, a proof is a computable function (Bishop and many others).
- For a platonist, a proof is not a mathematical object, but is it really so?

Mathematical Objects

What are mathematical objects, and how they are constructed?

For a platonist, mathematical objects exist independent of his mind. So he is not interested in the latter half of the quesion, or, at least it seems to be so.

For a constructivist, mathematical objects are to be mentally constructed by him. So, he is more interested in the latter half of the question, and try to answer the first half by solving the latter.

We wish to attack this question based on platonistic ontology but from a constructive point of view.

A hint for this approach was given by John H. Conway.

Platonism vs. Constructivism

	Platonism	Constructivism	Formalism

Platonism vs. Constructivism

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism

Platonism vs. Constructivism

	Platonism	Constructivism	Formalism
Philosophy	Realism	Conceptualism	Nominalism
Mathematics	Logicism	Intuitionism	Formalism

Platonism vs. Constructivism

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

Platonism vs. Constructivism

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

Platonism vs. Constructivism

	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

Platonism vs. Constructivism

	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
Mathematician	Classical mathematician	Constructive mathematician	Proof theorist

Platonism vs. Constructivism

	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
Mathematician	Classical mathematician	Constructive mathematician	Proof theorist

Ontology concerns what and computation concerns how.

$\Rightarrow$

Classical mathemtaics became more and more abstract.

$\Rightarrow$

We wish to make classical mathematics more concrete (constructive in a sense).

Mathematicians' Liberation Movement

Conway, in his book "*On Numbers and Games*" (1976), proposed the following way of construction of mathematical objects.

1. Objects may be created from earlier objects in any reasonably constructive fashion.
2. Equality among created objects can be any desired equivalence relation.

This is very similar to Martin-Löf's predicative construction of objects.

Conway also stressed the *open-endedness* of mathematics.

Classical ZFC is good for metamathematics but inadequate for the purpose of actually working in it.

Platonism as Transfinitary Constructivism

- Computation in ordinary sense of the term means computation on natural numbers.
- We extend the notion of computation, and compute on ordinal numbers. (Takeuti already suggested this.)

Ontological Commitment

- We must commit ourselves ontologically as to what objects we accept as entities which exist.
- At the same time, we must accept the limitations which come from Gödel's second incompleteness theorem and from Tarski's theorem on indefinablity of truth.
- In other words, it is impossible to have a fixed formal system in which we can develop all the mathematics.
- This means that we always have to have (at least) two linguistic layers, one for the object-level and the other for the meta-level.
- In this talk, I concentrate mainly on the construction of the meta-level.

Ontological Commitment

- We must commit ourselves ontologically as to what objects we accept as entities which exist.
- At the same time, we must accept the limitations which come from Gödel's second incompleteness theorem and from Tarski's theorem on indefinablity of truth.
- In other words, it is impossible to have a fixed formal system in which we can develop all the mathematics.
- This means that we always have to have (at least) two linguistic layers, one for the object-level and the other for the meta-level.
- In this talk, I concentrate mainly on the construction of the meta-level.
- By the dynamical interaction between the meta and object levels, we can *modify* and *grow* the object-level language.

Quine's view

In 1948, Quine published a very influential paper:

On what there is

In this paper, Quine wrote the following famous sentence:

To be is to be the value of a variable.

This dictum (almost) implies that function application must be done by call-by-value and not by call-by-name.

Quine's view (cont.)

For example, in set theory, instead of introducing a specific constant $\emptyset$ for the empty set, one can do without it by introducing an axiom which guarantees the existence and uniqueness of some object which satifies the properties of the empty set.

$$\exists x.\ \forall y.\ \neg y \in x$$

Name and object

Quine stressed that names (terms) may not always denote objects. (Example, 'Pegasus'.)
There are (at least) three different approaches to names and objects.

- First-order logic assumes that names always have values.
- Constructive type theories use contexts to control the usage of names, so that when they are used they always have values.
- Logic of partial terms (Scott, Beeson etc.) allows undefined terms.

Name and object (cont.)

$$\frac{A(b)}{\exists x.\ A(x)} \qquad \frac{b : B \quad p : A(b)}{(b, p) : \exists(x : B).\ A(x)} \qquad \frac{A(b) \quad b = b}{\exists x.\ A(x)}$$

Our approach adopts the first inference rule, but we can also explicitly talk about names and objects at the same level.

The logic of partial terms cannot directly talk about names. In our approach names are also objects.

We remark that H. Ono (1977) proposed a first-order theory of names and objects.

Name and object (cont.)

In traditional systems, terms are constructed as follows.

In first-order logic:

$$\frac{\text{`}\boldsymbol{f}\text{'} : \text{unary-fn-symbol} \quad \text{`}\boldsymbol{a}\text{'} : \textbf{term}}{\text{`}\boldsymbol{f}(\boldsymbol{a})\text{'} : \textbf{term}}$$

In type theory:

$$\frac{\boldsymbol{f} : \boldsymbol{A} \to \boldsymbol{B} \quad \boldsymbol{a} : \boldsymbol{A}}{\boldsymbol{f}(\boldsymbol{a}) : \boldsymbol{B}}$$

Type theory *confuses* syntax and semantics in a sense. This confusion is carried over to Edinburgh LF for instance.

Our approach is similar to first-order logic.

Name and object (cont.)

Our approach is to clearly distinguish names and objects by introducing the notion of kinds which are used to classify objects.

An object whose kind is expression is used to name an object.

An *expression* is an *object* and we can talk about it directly within our system.

Moreover, our system has a binary relation

$$e \downarrow a$$

which means that e is an expression denoting a. For example, we have

$$\text{`}2 + 3\text{'} \downarrow 5, \quad \text{``}2 + 3\text{''} \downarrow \text{`}2 + 3\text{'}, \quad \ldots$$

Name and object (cont.)

Consider division of a by b where a and b are rational numbers. It is well defined if b is not 0.

What if $b = 0$?

In first-order logic, since functions are always total, $\mathtt{div}(a, 0)$ is usually defined by assigning an aribitrary value, say, 0.

In (dependent) type theory, the division function has the following type:

$$\mathtt{div} : \mathbb{Q} \to (b : \mathbb{Q}) \to (b \neq 0) \to \mathbb{Q}$$

In our system, '$\mathtt{div}(a, 0)$' denotes an error object.

Meta Language and Object Languge

We use English as the meta language for defining the formal object language which will be used to formally define our NF (Natural Framework).

It is important to remark that our object language will be defined as a sub-language of English. That is, although it is a formal language, it is at the same time, a part of a natural language, namely, English.

We will call the object language NF English.

So, a sentence of NF English is also an English sentence, and we can always read it aloud.

Meta-level objects and object-level objects

In our meta language we will use informal platonistic mathematics freely.

The ontology of the meta language will be strictly stronger than that of the object language.

- We commit ourselves to the existence of the inaccessible cardinals $\boldsymbol{\Omega_1, \Omega_2, \ldots}$.
- The collection of all object-level objects is a meta-level object but not an object-level object.

Construction of objects

We must presuppose *time* and *space* so that we can construct objects.

In type theory, we define natural numbers as follows.

$$\frac{}{\mathbf{0} : \mathbb{N}}\ \texttt{zero} \qquad \frac{\boldsymbol{n} : \mathbb{N}}{\mathrm{s}(\boldsymbol{n}) : \mathbb{N}}\ \texttt{succ}$$

Then natural numbers are *constructed* in time and space as follows.

$$\mathbf{0}, \boldsymbol{s}(\mathbf{0}), \boldsymbol{s}(\boldsymbol{s}(\mathbf{0})), \cdots$$

They are obtained by *applying* the methods `zero` and `succ` as follows.

$$\boldsymbol{apply}(\texttt{zero}, ()), \boldsymbol{apply}(\texttt{succ}, (\mathbf{0})), \boldsymbol{apply}(\texttt{succ}, (\boldsymbol{s}(\mathbf{0}))), \cdots$$

Construction of objects (cont.)

Our ontology accepts the existence of all the platonistic *ordinals* (time) and *sequences of objects of arbitrary ordinal length* (space).

A sequence of *length* α can be visualized as follows.

sequence a:	a_0	a_1	a_2	$\cdots\cdots$	a_β	$\cdots\cdots$	$(\beta < \alpha)$

We will write '$|a|$' for the length of a.

A sequence may be considered as a generalisation of a Turing Machine's tape where each cell can contain any object, and cells are indexed by ordinals bounded by another ordinal.

Transfinitary inductive definition

We *construct* new objects inductively from *already constructed* objects by applying *constructors* (which are methods).

On day α, we construct new objects using objects created before day α.
Namely, all the objects created before day α are available, and more over, we assume that blank tapes of length β are available for each $\beta \leq \alpha$.

If an object a is created, for the first time, on day δ, then δ is called its *birthday* and we write '$\|a\|$' for it.

Kind

We categorize objects into following *kinds*:

1. Ordinal
2. Sequence
3. Character
4. String
5. Set
6. Quotient
7. Function
8. Proposition
9. Arity
10. Expression
11. Abstract
12. Error

Kind (cont.)

The kinds above are all mutually disjoint and we also have the kind `Obj` of all the NF-objects.

The kind `Obj` will be stratified into $\mathtt{Obj}_{\boldsymbol{k}}$ $(\boldsymbol{k} = \mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots)$ so that:

$$\mathtt{Obj}_{\mathbf{0}} \subset \mathtt{Obj}_{\mathbf{1}} \subset \cdots, \mathtt{Obj} = \bigcup_{\boldsymbol{k}} \mathtt{Obj}_{\boldsymbol{k}} \text{ and } \boldsymbol{a}\, \mathtt{Obj}_{\boldsymbol{k}} \Leftrightarrow \|\boldsymbol{a}\| < \boldsymbol{\Omega}_{\boldsymbol{k}+\mathbf{1}}.$$

Each kind will be stratified similarly.

A sentence of NF English will be called *judgments*.

For example, '$\mathbf{0}$ `Ord`' (read: 0 is an Ordinal) is a judgment.

Well Ordering of Objects and ID Number

We can well-order

$$\mathtt{Obj} \triangleq \bigcup_{k} \mathtt{Obj}_{k}$$

in such a way that if $\|a\| < \|b\|$, then a will come before b.

We will write '$\mathtt{ID}(a)$' for the ordinal assigned to a by this well-ordering.

We have:

$$a \in \mathtt{Obj}_{k} \;\Leftrightarrow\; \|a\| \in \mathtt{Obj}_{k} \;\Leftrightarrow\; \mathtt{ID}(a) \in \mathtt{Obj}_{k} \;\Leftrightarrow\; \|a\| < \Omega_{k+1}.$$

Ordinal

$$\frac{\cdots \quad \alpha_i \ \mathtt{Ord} \quad \cdots \quad (0 \leq i < \gamma)}{\mathtt{ord}(\cdots, \alpha_i, \cdots) \ \mathtt{Ord}} \ \mathtt{ord}$$

On day δ, one may apply this rule if $\gamma \leq \delta$, and if $\alpha_i < \alpha_j$ whenever $i < j$.

Order relation and equality are defined as follows.

$$\begin{aligned} \mathtt{ord}(\alpha_i) \leq \mathtt{ord}(\beta_j) \quad &\Leftrightarrow \quad \forall \alpha_i \exists \beta_j \ \alpha_i \leq \beta_j. \\ \alpha = \beta \quad &\Leftrightarrow \quad \alpha \leq \beta \wedge \beta \leq \alpha. \end{aligned}$$

We have: $0 = \mathtt{ord}(), 1 = \mathtt{ord}(0), 2 = \mathtt{ord}(0,1) = \mathtt{ord}(1), \ldots$ and $\omega = \mathtt{ord}(0, 1, \ldots, i, \ldots)$ $(0 \leq i < \omega)$ and so on.

In general, on day δ, we can construct all the ordinals less than or equal to δ.

Sequence

$$\frac{\cdots \quad a_i\ \texttt{Obj} \quad \cdots \quad (0 \leq i < \gamma)}{\mathsf{seq}(\cdots, a_i, \cdots)\ \texttt{Seq}}\ \texttt{seq}$$

On day δ, one may apply this rule if $\gamma \leq \delta$.

Equality is defined by:

$$\mathsf{seq}(a_i) = \mathsf{seq}(b_j) \iff |(a_i)| = |(b_j)| \wedge \forall i\ a_i = b_i.$$

We will write:

$$\begin{array}{rcl} `(a_0, a_1, \ldots)' & \text{for} & \mathsf{seq}(a_0, a_1, \ldots) \\ `\mathsf{seq}(a_i)[j]' & \text{for} & a_j. \end{array}$$

Character

$$\frac{i\ \texttt{Ord}}{\texttt{char}(i)\ \texttt{Char}}\ \texttt{char}$$

On day δ, one may apply this rule if $i \leq \delta$ and $i < \omega$.
Equality is defined by:

$$\texttt{char}(i) = \texttt{char}(j) \ \Leftrightarrow\ i = j.$$

String

$$\frac{\cdots \quad c_i\ \texttt{Char} \quad \cdots \quad (0 \le i < n)}{\texttt{str}(\cdots, c_i, \cdots)\ \texttt{Str}}\ \texttt{str}$$

On day δ, one may apply this rule if $n \le \delta$ and $n < \omega$.

Equality is defined by:

$$\texttt{str}(c_i) = \texttt{str}(d_j) \iff |(c_i)| = |(d_j)| \wedge \forall i\ c_i = d_i.$$

We will write '$\texttt{"}c_0 c_1 \cdots c_n\texttt{"}$' for $\texttt{str}(c_0, c_1, \ldots, c_n)$.

Set

$$\frac{\cdots \quad a_i\ \texttt{Obj} \quad \cdots \quad (0 \le i < \gamma)}{\texttt{set}(\cdots, a_i, \cdots)\ \texttt{Set}}\ \texttt{set}$$

In the above rule, we must have: $a_i < a_j$ if $i < j$.

On day δ, one may apply this rule if $\gamma \le \delta$.

Equality and membership relations and length of a set are defined by:

$$\begin{aligned} \texttt{set}(a_i) = \texttt{set}(b_j) \quad &\Leftrightarrow \quad |(a_i)| = |(b_j)| \wedge \forall i\ a_i = b_i. \\ b \in \texttt{set}(a_i) \quad &\Leftrightarrow \quad \|b\| < \|\texttt{set}(a_i)\| \wedge \exists i\ b = a_i. \\ |\texttt{set}(a_i)| \quad &\triangleq \quad |(a_i)|. \end{aligned}$$

We will write '$\texttt{set}(a_i)[j]$' for a_j.

Quotient Object

$$\frac{a \ \mathtt{Obj} \quad R \ \mathtt{Set}}{\mathtt{qobj}(a, R) \ \mathtt{Qobj}} \ \mathtt{qobj}$$

In this rule, R must an equivalence relation and a is an object such that $(a, a) \in R$.

Equality is defined by:

$$\mathtt{qobj}(a, R) = \mathtt{qobj}(b, S) \ \Leftrightarrow \ R = S \wedge (a, b) \in R.$$

We will write '$[a]_R$' for $\mathtt{qobj}(a, R)$.

Function

$$\frac{A \ \texttt{Set} \quad s \ \texttt{Seq}}{\texttt{fun}(A, s) \ \texttt{Fun}} \ \texttt{fun}$$

This rule may be applied when $|A| = |s|$.

Equality and function application are defined as follows.

$$\begin{aligned} \texttt{fun}(A, s) = \texttt{fun}(B, t) \quad &\Leftrightarrow \quad A = B \land s = t \\ \texttt{apply}(\texttt{fun}(A, s), x) = y \quad &\Leftrightarrow \quad \exists i \ x = A[i] \land y = s[i]. \end{aligned}$$

Proposition

$$\frac{a\ \texttt{Obj} \quad b\ \texttt{Obj}}{(a < b)\ \texttt{Prop}}\ \texttt{lt} \qquad \frac{a\ \texttt{Obj} \quad b\ \texttt{Obj}}{(a = b)\ \texttt{Prop}}\ \texttt{eq}$$

$$\frac{a\ \texttt{Obj} \quad b\ \texttt{Set}}{(a \in b)\ \texttt{Prop}}\ \texttt{in} \qquad \frac{a\ \texttt{Obj} \quad b\ \texttt{Prop}}{(a :: b)\ \texttt{Prop}}\ \texttt{pr} \qquad \frac{a\ \texttt{Exp} \quad b\ \texttt{Obj}}{(a \downarrow b)\ \texttt{Prop}}\ \texttt{dn}$$

$$\frac{P\ \texttt{Prop} \quad Q\ \texttt{Prop}}{P \wedge Q\ \texttt{Prop}}\ \texttt{and} \qquad \frac{P\ \texttt{Prop} \quad Q\ \texttt{Prop}}{P \vee Q\ \texttt{Prop}}\ \texttt{or}$$

$$\frac{P\ \texttt{Prop} \quad Q\ \texttt{Prop}}{P \supset Q\ \texttt{Prop}}\ \texttt{imp} \qquad \frac{P\ \texttt{Prop}}{\neg P\ \texttt{Prop}}\ \texttt{not}$$

$$\frac{f\ \texttt{PropFun}}{\forall f\ \texttt{Prop}}\ \texttt{all} \qquad \frac{f\ \texttt{PropFun}}{\exists f\ \texttt{Prop}}\ \texttt{ex}$$

Propositional Function

$$\frac{\boldsymbol{A}\ \texttt{Set} \quad \boldsymbol{s}\ \texttt{Seq}}{\texttt{fun}(\boldsymbol{A}, \boldsymbol{s})\ \texttt{PropFun}}\ \texttt{propfun}$$

This rule may be applied

- if $|\boldsymbol{A}| = |\boldsymbol{s}|$, and
- if $\boldsymbol{s}[\boldsymbol{i}]$ $\texttt{Prop}$ for all $\boldsymbol{i} < |\boldsymbol{s}|$.

$\texttt{PropFun}$ is a subkind of $\texttt{Fun}$.

If $\boldsymbol{f} = \texttt{fun}(\boldsymbol{A}, \boldsymbol{s})$ is a propositional function, then we will write

'$\forall \boldsymbol{x} \in \boldsymbol{A}\ \boldsymbol{f}(\boldsymbol{x})$' for $\forall \boldsymbol{f}$.

Proofs and Propositions

Recall that:

$$\frac{\boldsymbol{a}\ \texttt{Obj} \quad \boldsymbol{b}\ \texttt{Prop}}{(\boldsymbol{a} :: \boldsymbol{b})\ \texttt{Prop}}\ \texttt{pr}$$

We claim that every propostion $\boldsymbol{P}$ is *true* if and only if it has a proof $\boldsymbol{p}$, namely, '$\boldsymbol{p} :: \boldsymbol{P}$'.

We use platonistic version of propositions-as-sets principle and BHK interpretation to define the provability relation inductively.

So, for example:

$$\begin{aligned} \boldsymbol{f} :: \boldsymbol{P} \supset \boldsymbol{Q} \quad & \Leftrightarrow \quad \boldsymbol{f} \in \mathsf{Proof}(\boldsymbol{P} \supset \boldsymbol{Q}) \\ & \Leftrightarrow \quad \boldsymbol{f} \in \mathsf{Proof}(\boldsymbol{P}) \longrightarrow \mathsf{Proof}(\boldsymbol{Q}) \\ & \Leftrightarrow \quad \forall \boldsymbol{p} \in \mathsf{Proof}(\boldsymbol{P})\ \texttt{apply}(\boldsymbol{f}, \boldsymbol{p}) \in \mathsf{Proof}(\boldsymbol{Q}) \\ & \Leftrightarrow \quad \forall \boldsymbol{p} :: \boldsymbol{P}\ \texttt{apply}(\boldsymbol{f}, \boldsymbol{p}) :: \boldsymbol{Q}. \end{aligned}$$

Summary

- We argued the inadequecy of a fixed language as a universal language for developing (almost) all the mathematcs.
- For example, neither ZFC nor type theory are adequate for this purpose.
- We argued that for the purpose, we need at least two layers of languages for the meta-level and object-level.
- In this setting, object-level language can be *modified* and *extended*.
- The object-language must be *open-ended* and must admit *free structures*.
- The object-language must be able to talk about both *syntax* and *semantics* naturally.

Summary (cont.)

- We proposed a *constructive* way of constructing a universe of mathematical objects.
- Each and every object of the universe is *created one by one* sequentially in time and space.
- We presupposed, unconditionally, the *existence* of time and space whose units are ordinals.
- We have presented a meta language today.
- The meta language naturally contains a constructive sublanguage.
- We wish to use the meta language to implement an object language which can be used as a framework for a proof assistance system.