Proof Assistants and Foundation of Mathematics

Masahiko Sato
Graduate School of Informatics, Kyoto University

SCSS 2024
Tokyo University of Science
August 30, 2024

Plan of the talk

Part I

Introduction

Present, Past and Future

Present, Past, Future = 現在，過去，未来 = Genzai, Kako, Mirai

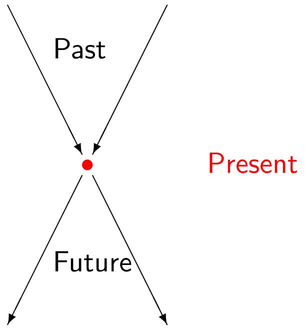
mypath is winding = Mayoi Michi, Kune Kune

Lostway(Mayoi Michi in Japanese) : Lyrics, Compostion and Singing all done by Machiko Watanabe. released in 1977.

Paul McCartney recorded The Long and Winding Road. in 1970. Lyrics, Composition and Singing all done py Paul.

The Long and Winding Road had a goal (which belongs to future), but Lostway had no goal.

Context and environment



- History creates me and I create history.
- Context = History
- An environment = The current part of a context.
- To live is to experience the interaction between I and the environment surrounding me.

Context

History created me and I will create history, but history also created you and you will also create history.

It is crucial to observe that each of us (you and I) were created (born) and raised under different contexts (including birth places, mother lanuages and DNA).

This means that each of us is always carrying a personal context (personal history) which is unique to each person, and the context changes as time goes by.

We apply this idea to mathematics and to mathematicians in this talk.

My history

Below I list a tiny bit of my personal context.

- I was born in Kobe (神戸), Japan, on May 1, 1947.
- I studied mathematics and graduated from the University of Tokyo.
- My teachers, when I was 15-28 years old, include Sin Hitotumatu (一松信), Nobuo Yoneda (米田信夫), Mikio Sato (佐藤幹夫) and John McCarthy.
- My original field was mathematics(object-level) but gradually shifted to metamathematics (meta-level), and then to computer science (meta-meta-level).
- In 1975, I learned the concept of proof assistant from John McCarthy. He statyed in Kyoto for 3 months, as a visiting Professor of Kyoto University. I was Assistant Professor at RIMS then.

My stand point

The title of my talk is:

Proof Assistants and Foundations of Mathematics

But, it has a subtitle:

Language and Mathematics

We can communicate each other only through a language. Since this is so obvious, we often forget this context.

My stand point is:

Every mathematical object is a linguistic object.

Symbolic Computation

My stand point is:

> *Every mathematical object is a linguistic object.*

A quotation from the home page of SCSS 2024:

> *Symbolic computation is the science of computing with symbolic objects (terms, formulae, programs, representations of algebraic objects, etc.).*

The above quotation is an open definition. Because, it contains the open word etc. which can be filled in by other closed words such as *proofs*. If one accepts my stand point, one obtains a closed definition:

> *Symbolic computation is the science of computing with mathematical objects.*

Proof Assistant

A proof assistant is an interactive software tool which we use to write correct formal proofs on a computer.

In this sense, a proof assistant does meta-symbolic computation dealing with metamathematical objects.

In the long history of mathematics, proof assistants appeared only recently after the invention of computers.

Logic and Computation

Logic and Computation, which is more basic?

Please ask yourself.

1. Logic is more basic than Computation.
2. Computation is more basic than Logic.
3. I have no idea.

Part II

Mathematics as social activity

Mathematics as social activity

In Japan, we have The Mathematical Society of Japan.

USA has The American Mathematical Society and its home page says:

> *The American Mathematical Society is dedicated to advancing research and connecting the diverse global mathematical community through publications, meetings and conferences, MathSciNet, professional services, advocacy, and awareness programs.*

SCSS 2024 is also such an activity.

Journal publications

As seen in the previous slide, journal publications are an instance of acitivity of mathematicians.

A paper submitted to a peer-reviewed journal is read and checked by a number of reviewers before publication.

In the case of a mathematical paper, a necessary condition for the acceptance of the paper for publication is that all proofs given in the paper are correct.

However, even if the paper contain some errors, human reviwers may fail to detect them.

In the late 1950s, John McCarthy had the idea that the above problem can be avoided by letting computer do the job of proof checking.

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.

Part III

Formalization of mathematics

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.

What is formalized mathematics? (cont.)

A crucial property of a formalized mathematics is that it can be implemented using only finitary objects (in the sense of Hilbert) and these finitary objects can be constructed and manipulated by applying computable functions.

Computable functions are also (intensional) finitary objects and they are well-understood informally.

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, Lean etc.

Formalization of computation is important here.

History of formalization

- Frege (Begriffsschrift, 1879) Higher order logic
- 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)
- de Bruijn (Automath 1967 –)
- Mizar (1973 –), Coq, Isabelle, Theorema, Agda, Lean

Part IV

Future of mathematics and proof assistants

Mathematicians who influenced me

Many mathematicians, including Dana Scott, Bruno Buchberger and Piet Hut (Physicist at IAS and study Husserlian phenomenology), influenced me a lot.

But, here, I will talk about:

McCarthy, Martin-Löf, Yoneda and Mikio Sato.

This is because the way in which they do mathematics, directly had some influences on the design of my new proof assistant which I call NM (for Natural Mathematics).

This is also the case for Buchberger, but omitted him since he already showed his way of doing mathematics on the first day of this conference.

John McCarthy (1927 - 2011)

McCarthy developed Lisp which has the following properties never or rarely found in other programming languages.

1. Lisp used symbolic expressions, which has binary tree structure, to represent both syntax and data of Lisp.
2. Introduced `quote` which provides quotation mechanism to Lisp.
3. Lisp's *interpreter* can be written succinctly in Lisp (meta-circular interpreter).

He also introduced the notion of first-oreder abstract syntax.

He was also intersted in the representation of someone knowing something using modal logic. This was the topic of my Ph.D thesis.

Tresidder Union, Stanford University, 1977

Vera Watson (John's wife), John McCarthy, Satoru Takasu, S

Per Matin-Löf (1942 -)

Per Matin-Löf introduced ITT (Intuitionistic Type Theory, 1984), which later became bases of several proof assistants including Coq and Agda.

In the year ITT book was published, Matin-Löf came to Japan at the invitation of ICOT where the 5th Generation Computer Systems project was carried out.

I met him on this occasion and he gave me a copy of just published ITT book.

Later, in 2004, I invited him to come to Kyoto Univerity for 3 months.

He is also a philosopher and deeply understands Frege's philosophy.

RIMS, Kyoto University, 2004

S and Per Martin-Löf

Nobuo Yoneda (1930 - 1996)

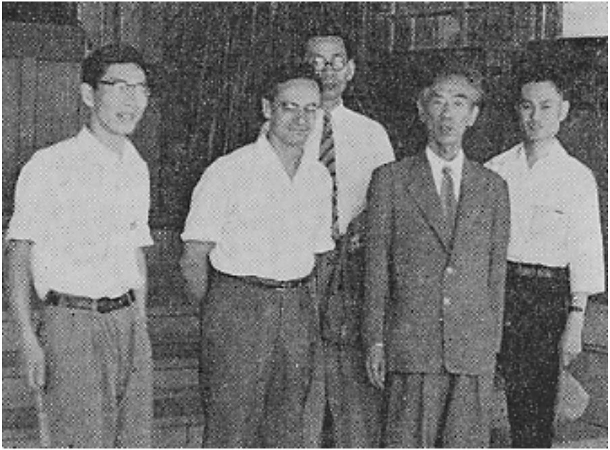
Yoneda is famous for his Yoneda Lemma which is a fundamental lemma in category theory.

Yoneda and I came to know each other when I was 15. This was only by postal communication.

Incidentallay, Hitotumatu (一松) and I also came to know each other when I was 15 through postal communicatio.

Both Yoned and Mikio Sato (佐藤幹夫) graduated from Mathematics Department of the University of Tokyo in 1952.

Oka and Serre, Nara Hotel, August 1955



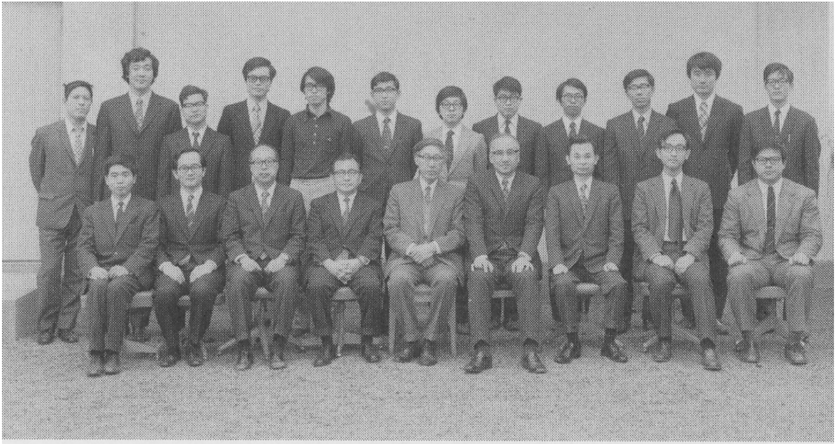
Shigeo Nakano, J.P. Serre [28 years old. Awarded the Fields medal jointly with Kodaira]

Akizuki (a class mate of Oka at the Third High School in Kytoto)

Oka[54], Hitotumatu[29]

This photo was taken by Yoneda[25]

I got a Masters Degree in 1973



東京大学修士 1973 年卒業生と.

I am the shortest in the second row,
just behind Iyanaga(彌永), the oldest (66) in the photo.
Kodaira(小平邦彦) is the shortest person in the front row.

A remark about Iyanaga

Iyanaga was a teacher of ipressive numbers of good mathematicians including :

> *Kiyoshi Ito, Kodaira, Mikio Sato, Nobuo Yoneda, Gaisi Takeuti and Satoru Takasu.*

International Sympsium on Algorithmic Languages
CWI, Amsterdam, 1981

Dana Scott also attended the symposium.

Nobuo Yoneda and S

Mikio Sato (1928 - 2023)

He was a member of my Ph.D degree examination committee. Other two members were Sin Hitotumatu (chair) and Satoru Takasu.

He introduced hyper-function, D module, prehomogenious vector space etc. into mathematics

He stressed the importance of using algebraic aproach (which mainly rely on reasoning with equation) in other area of mathematics including analysis.

He refused to be called a philosopher, but I think he is a great philsopher.

He is a natural mathematician.

Drawbacks of current proof assistants

Here, I discuss only proof assistants based on type theory. Set theory has more serious drawbacks. I just remark here that formalization of both type theory and set theory started after Russel Paradox was found in one of Frege's book.

There are many type theories, but they all have the following two serious drawawbacks, in my opinion.

1. So far, no type-theorist could define concept of type. Namely, no one could answer the question: "What is type?".
2. Type theories handle only terminating computaion. Thus, you cannot write non-terminating programs in type theory.

A proof assistant is a softare which should keep on running as long as its users wish to use it. So, all the proof assistants are implemented using languages which support non-terminating computation. A proof assistant cannot bootstrap itself.

How and What

This extension is based on the following observations.

In each item below, the word set in this color represents some *action* to be performed on the following word set in this color.

How and what	[Mikio Sato]
Verb and noun	[J.L. Austin's speech act]
Function and argument	[Gotolob Frege and Martin-Löf]
Dynamic and static	[Lisp and scheme]
Proof and theorem	[Frege, Gentzen, Martin-Löf]
Definition and class/object	[definition is a speech act (S)]
Meta-Level and object-level	[Wolfram, Buchberger and S]

Note that the observation was done at meta-meta-level (meta-meta-meta-level in the case of Definition).

Part V

Conclusion

Coclusion

I cannot getaway from my Present since I am here now

I cannot also getway from my Past, since it is always with me as my own personal history, wich I call my Context.

But, fortunately (or, unfortunately), my Future is yet to come, and has possibilities of making it as I wish to be.

So, I always have a goal in my mind and trying to get to the Future in which the goal will come true.