Formal Proofs for Security

David Nowak

JFLI

CNRS & The University of Tokyo

Outline

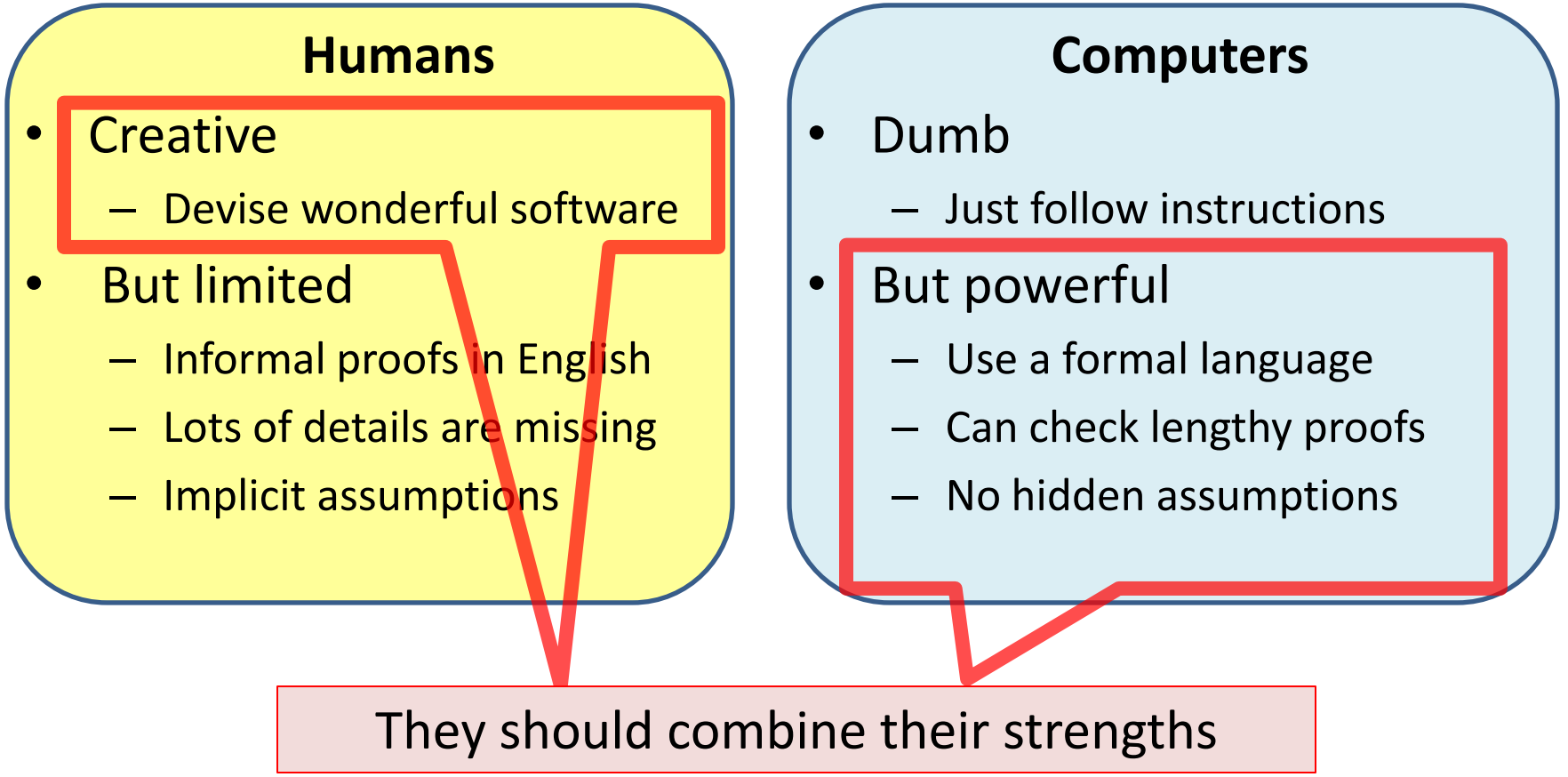
- A brief introduction to formal proofs
- A few outstanding applications
- Projects I have been recently involved in

Bugs can be expensive and dangerous

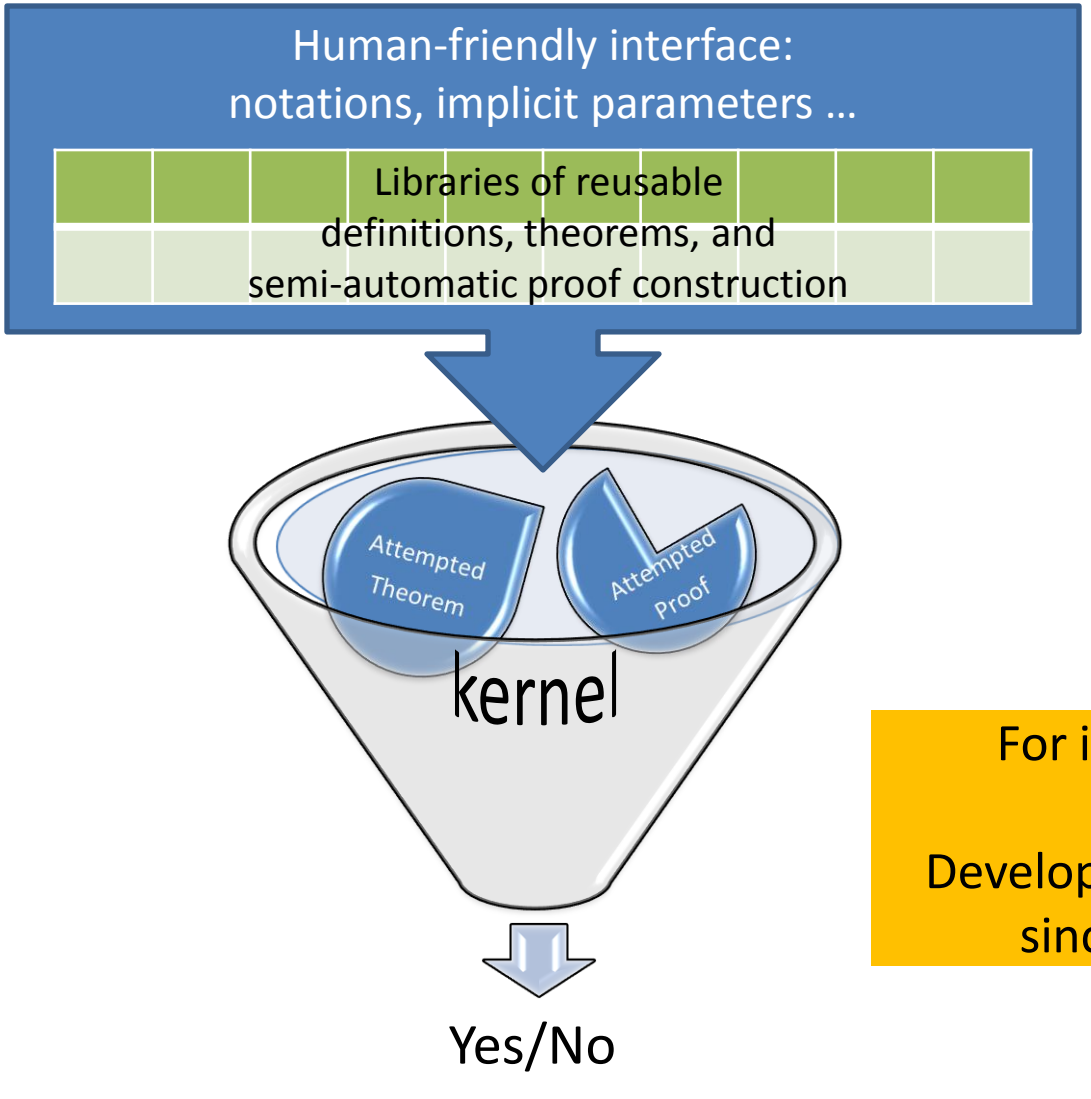
Pentium FDIV bug	US$475 million
Ariane 5 flight 501 crash	US$370 million

"*a math error in a widely used computing chip places the security of the global electronic commerce system at risk*"
(Adi Shamir, New York Times, November 17, 2007)

Computers can help



How can a computer help?



Proof Assistant

For instance:
Coq
Developed at INRIA
since 1984

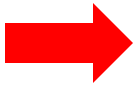
Limitations

The following kind of proofs are not supported:

- proof by intimidation such as
  "*Trivial*" or "*The reader may easily supply the details*"
- proof by hidden assumption
- Proof by vigorous handwaving
- Proof by exhaustion

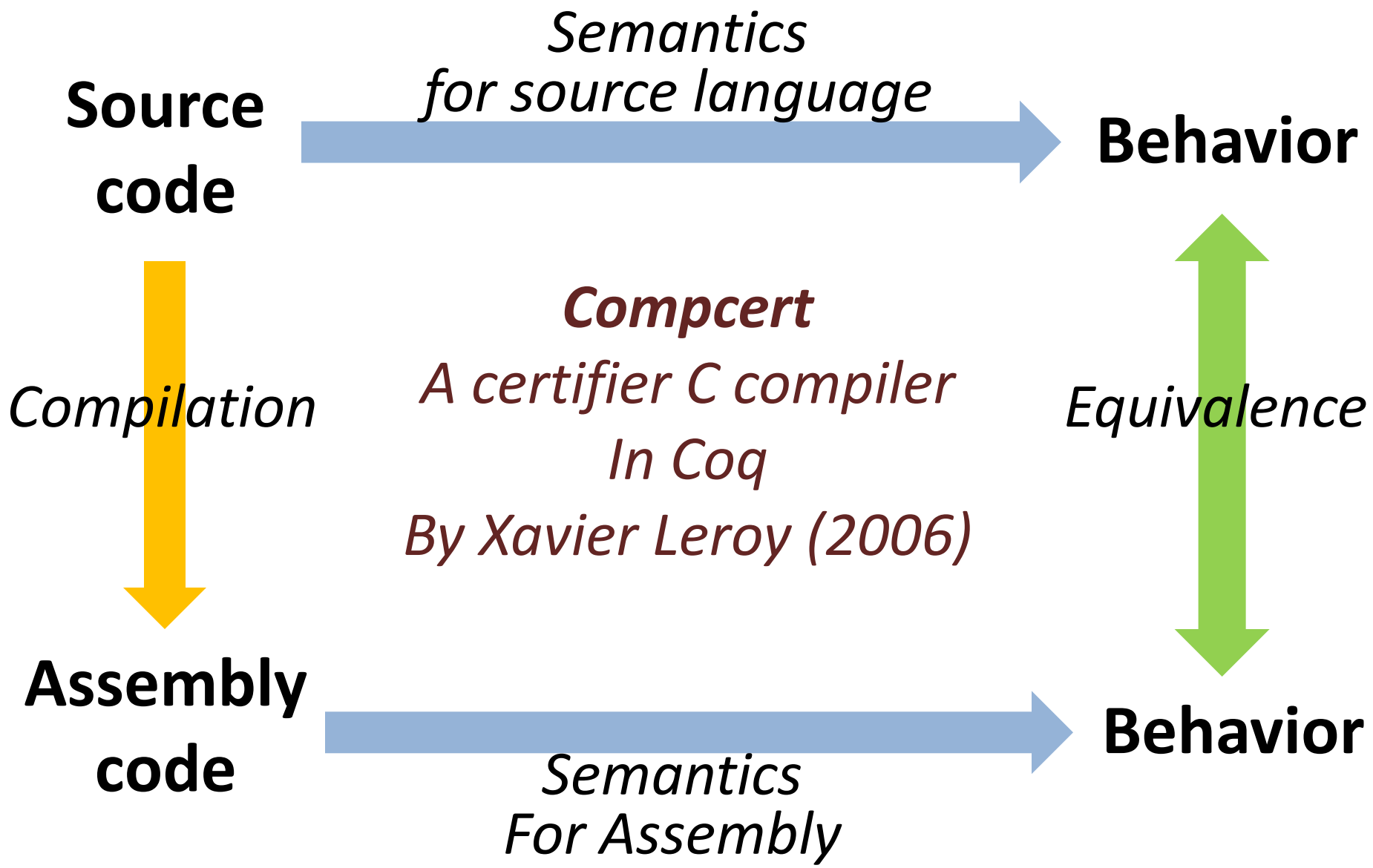
Outline

A brief introduction to formal proofs


A few outstanding applications

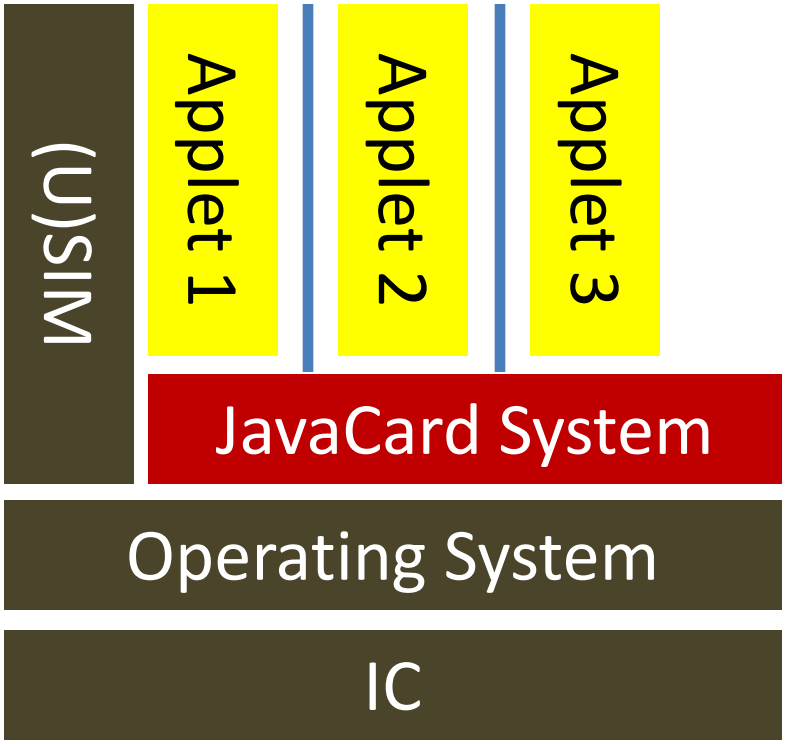
Projects I have been recently involved in

Certified compilation



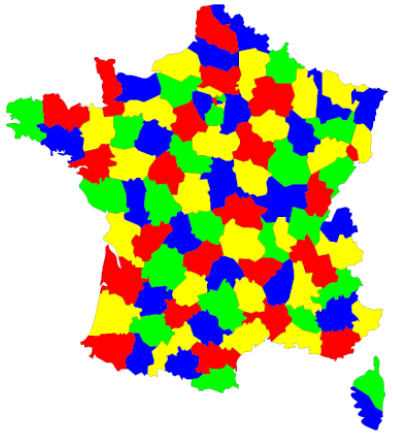
Certification of a Javacard system

In 2003, a JavaCard System was certified secure in Coq by the company Gemalto at the highest level EAL7 of the Common Criteria (ISO/IEC 15408).



The four color theorem

✓ Stated, but not proved, in 1853 by Guthrie.



Four colors are enough to color any geographical map in such a way that no neighboring two areas are of the same color.

✓ Proved in 1976 by Appel and Haken.

- It requires checking many combinations on a computer.
- It is subject to programming error.

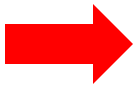
✓ It was proved again in 2004 by Gonthier.

- Proofs and computations are done in Coq.
- There is no risk of programming error.
- You only need to trust the kernel of Coq.

Outline

A brief introduction to formal proofs

A few outstanding applications



Projects I have been recently involved in

Issues with proving security with Coq

Security is often proved of algorithms.

But what about their implementations?

We illustrate our approach on a PRNG for cryptography.

Internet protocols are mostly specified in English.

How can we formally specify them in Coq?

We illustrate our approach on the Internet protocol TLS.

In a proof, the power of an adversary is to be restricted.

How to conveniently and formally handle this in Coq?

We design a proof language based on implicit complexity.

Collaborative work with researchers from Japan, France, China

Funded by Japanese grants and by a CNRS-JST French-Japanese project

Formal security proofs of implementations of cryptographic primitives

Security proofs are usually about algorithms.
How to certify the security of an implementation?

We lift proof techniques for cryptographic algorithms at the level of implementations.

We apply our technique to certify in Coq the security of the implementation of a pseudorandom number generator.

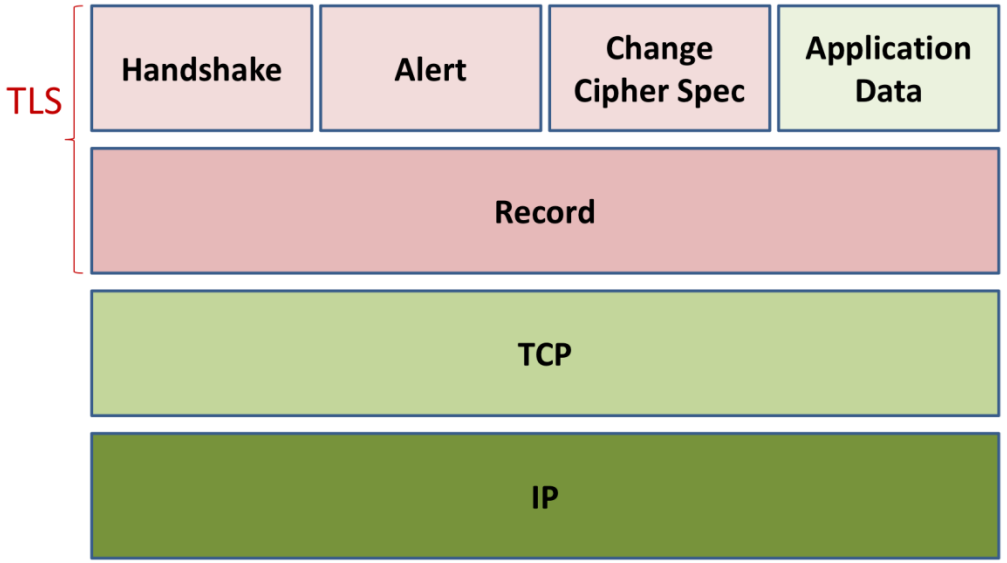
With R. Affeldt and K. Yamada (AIST, Japan)
Funded by the Japanese ministry of education and research

Encoding and decoding of network packets

Internet protocols are specified in natural language that comes with ambiguities.

How to formally specify Internet protocols?

We propose a methodology to specify the syntax of network packets in Coq, and illustrate it on TLS.



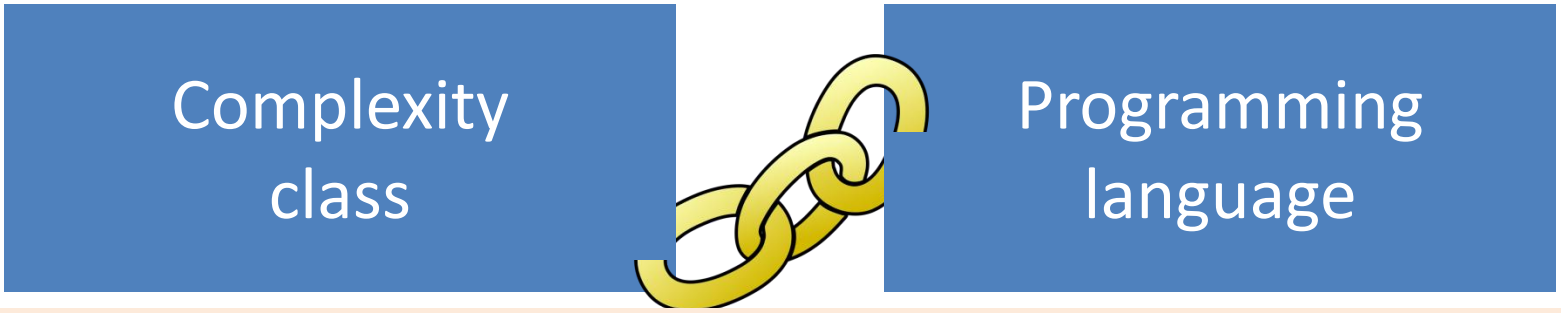
With R. Affeldt and Y. Oiwa (AIST, Japan)
Funded by the Japanese ministry of telecommunications

Implicit Complexity for security proofs

An adversary with unlimited computational power (unrealistic) could for instance break the widely used encryption algorithm RSA.

How to keep the adversary realistic in formal proofs?

We propose a calculus based on implicit complexity theory.



With Yu Zhang (ISCAS, China)
Funded by the Japanese ministry of education and research

Implicit complexity in Coq

Now that we have a calculus on paper, how to integrate and use it in Coq?

We formalize in Coq a calculus that characterizes exactly the class of polytime functions.

It is theoretically complete but it is minimalistic.

It is a necessary first step before formalizing in Coq a richer calculus (previous slide).

With Sylvain Heraud (INRIA)
Funded by a CNRS-JST French-Japanese project