Activity Report 2019

Project-Team STAMP

Safety Techniques based on Formalized Mathematical Proofs

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Proofs and Verification

Table of contents

Project-Team STAMP

Creation of the Project-Team: 2019 November 01

Keywords:

Computer Science and Digital Science:

A2.1.11. - Proof languages
A2.4.3. - Proofs
A4.5. - Formal methods for security
A5.10.3. - Planning
A7.2. - Logic in Computer Science
A7.2.3. - Interactive Theorem Proving
A7.2.4. - Mechanized Formalization of Mathematics
A8.3. - Geometry, Topology
A8.4. - Computer Algebra
A8.10. - Computer arithmetic

Other Research Topics and Application Domains:

B6.1. - Software industry
B9.5.1. - Computer science
B9.5.2. - Mathematics

1. Team, Visitors, External Collaborators

Research Scientists

Yves Bertot [Team leader, Inria, Senior Researcher, HDR]
Cyril Cohen [Inria, Researcher]
Benjamin Grégoire [Inria, Researcher]
José Grimm [Inria, Researcher, until September 2019 (deceased)]
Laurence Rideau [Inria, Researcher]
Enrico Tassi [Inria, Researcher]
Laurent Théry [Inria, Researcher]

Post-Doctoral Fellow

Pierre Boutry [Inria, Post-Doctoral Fellow, from Sep 2019]

PhD Students

Cécile Baritel-Ruet [Inria, PhD Student]
Sophie Bernard [Univ de Nice - Sophia Antipolis, PhD Student, until Sep 2019]
Mohamad El Laz [Inria, PhD Student]
Damien Rouhling [Ministère de l'Enseignement Supérieur et de la Recherche, PhD Student, until Sep 2019]

Interns and Apprentices

Ahmed Khulaif A Alharbi [Inria, from Mar 2019 until Jun 2019]
Julien Lamiroy [École Normale Supérieure de Paris, from Jun 2019 until Jul 2019]

Administrative Assistant

Nathalie Bellesso [Inria, Administrative Assistant]

Visiting Scientists

Reynald Affeldt [AIST, Japan, from Oct 2019]

Manuel Barbosa [University of Porto, from Jun 2019 until Jul 2019]
Christian Doczkal [Univ. Côte d'Azur, from Oct 2019]
Kazuhiko Sakaguchi [University of Tsukuba, Japan]

External Collaborators

Gilles Barthe [Max Planck Institute, Bochum, Germany, HDR]
Loïc Pottier [Ministère de l'Education Nationale, HDR]

2. Overall Objectives

2.1. Overall Objectives

Computers and programs running on these computers are powerful tools for many domains of human activities. In some of these domains, program errors can have enormous consequences. It will become crucial for all stakeholders that the best techniques are used when designing these programs.

We advocate using higher-order logic proof assistants as tools to obtain better quality programs and designs. These tools make it possible to build designs where all decisive arguments are explicit, ambiguity is alleviated, and logical steps can be verified precisely. In practice, we are intensive users of the Coq system and we participate actively to the development of this tool, in collaboration with other teams at Inria, and we also take an active part in advocating its usage by academics and industrial users around the world.

Many domains of modern computer science and engineering make a heavy use of mathematics. If we wish to use proof assistants to avoid errors in designs, we need to develop corpora of formally verified mathematics that are adapted to these domains. Developing libraries of formally verified mathematics is the main motivation for our research. In these libraries, we wish to capture not only the knowledge that is usualy recorded in definitions and theorems, but also the practical knowledge that is recorded in mathematical practice, idoms, and work habits. Thus, we are interested in logical facts, algorithms, and notation habits. Also, the very process of developing an ambitious library is a matter of organisation, with design decisions that need to be evaluated and improved. Refactoring of libraries is also an important topic. Among all higher-order logic based proof assistants, we contend that those based on Type theory are the best suited for this work on libraries, thanks to to they strong capabilities for abstraction and modular re-use.

The interface between mathematics, computer science and engineering is large. To focus our activities, we will concentrate our activity on applications of proof assistants to two main domains: cryptography and robotics. We also develop specific tools for proofs in cryptography, mainly around a proof tool named EasyCrypt.

3. Research Program

3.1. Theoretical background

The proof assistants that we consider provide both a programming language, where users can describe algorithms performing tasks in their domain of interest, and a logical language to reason about the programs, thus making it possible to ensure that the algorithms do solve the problems for which they were designed. trustability is gained because algorithms and logical statements provide multiple views of the same topic, thus making it possible to detect errors coming from mismatch between expected and established properties. The verification process is itself a logical process, where the computer can bring rigor in aligning expectations and guarantees.

The foundations of proof assistants rest on the very foundations of mathematics. As a consequence, all aspects of reasoning must be made completely explicit in the process of formally verifying an algorithm. All aspects of the formal verification of an algorithm are expressed in a discourse whose consistency is verified by the computer, so that unclear or intuitive arguments need to be replaced by precise logical inferences.

One of the foundational features on which we rely extensively is *Type Theory*. In this approach a very simple programming language is equiped with a powerful discipline to check the consistency of usage: types represent sets of data with similar behavior, functions represent algorithms mapping types to other types, and the consistency can be verified by a simple computer program, a *type-checker*. Although they can be verified by a simple program, types can express arbitrary complex objects or properties, so that the verification work lives in an interesting realm, where verifying proofs is decidable, but finding the proofs is undecidable.

This process for producing new algorithms and theorems is a novelty in the development of mathematical knowledge or algorithms, and new working methods must be devised for it to become a productive approach to high quality software development. Questions that arise are numerous. How do we avoid requiring human assistance to work on mundane aspects of proofs? How do we take advantage of all the progress made in automatic theorem proving? How do we organize the maintenance of ambitious corpora of formally verified knowledge in the long term?

To acquire hands-on expertise, we concentrate our activity on three aspects. The first one is foundational: we develop and maintain a library of mathematical facts that covers many aspects of algebra. In the past, we applied this library to proofs in group theory, but it is increasingly used for many different areas of mathematics and by other teams around the world, from combinatorics to elliptic cryptography, for instance. The second aspect is applicative: we develop a specific tool for proofs in cryptography, where we need to reason on the probability that opponents manage to access information we wish to protect. For this activity, we develop a specific proof system, relying on a wider set of automatic tools, with the objective of finding the tools that are well adapted to this domain and to attract users that are initially specialists in cryptography but not in formal verification. The third domain is robotics, as we believe that the current trend towards more an more autonomous robots and vehicles will raise questions of safety and trustability where formal verification can bring significant added value.

4. Application Domains

4.1. Mathematical Components

The Mathematical Components is the main by-product of an effort started almost two decades ago to provide a formally verified proof for a major theorem in group theory. Because this major theorem had a proof published in books of several hundreds of pages, with elements coming from character theory, other coming from algebra, and some coming from real analysis, it was an exercice in building a large library, with results in many domains, and in establishing clear guidelines for further increase and data search.

This library has proved to be a useful repository of mathematical facts for a wide area of applications, so that it has a growing community of users in many contries (Denmark, France, Germany, Japan, Singapore, Spain, Sweden, UK, USA, at the time of writing these lines in 2019) and for a wide variety of topics (transcendental number theory, elliptic curve cryptography, articulated robot kinematics, recently block chain foundations).

Interesting questions on this library range around the importance of decidability and proof irrelevance, the way to structure knowledge to automatically inherit theorems from one topic to another, the way to generate infrastructure to make this automation efficient and predictable. In particular, we want to concentrate on adding a new mathematical topic to this library: real analysis and then complex analysis (Mathematical Components Analysis).

On the front of automation, we are convinced that a higher level language is required to describe similarities between theories, to generate theorems that are immediate consequences of structures, etc, and for this reason, we invest in the development of a new language on top of the proof assistant (ELPI).

4.2. Proofs in cryptography

When we work on cryptography, we are interested in the formal verification of proofs showing that some cryptographic primitives provide good guarantees against unwanted access to information. Over the years we have developed a technique for this kind of reasoning that relies on a programing logic (close to Hoare logic) with probabilistic apsects and the capability to establish relations between several implementations of a problem. The resulting programming logic is called *probabilistic relational Hoare logic*. In more recent work, we have also started to study questions of *side-channel* attacks, where we wish to guarantee that opponents cannot gain access to protected knowledge, even if they observe specific features of execution, like execution time (to which the answer lies in *constant-time* execution) or partial access to memory bits (to which the answer lies in *masking*).

For this domain of application, we choose to work with a specific proof tool (EasyCrypt), which combines powerful first-order reasoning and uses of automatic tools, with a specific support for probabilistic relational Hoare Logic. The development of this EasyCrypt proof tool is one of the objectives of our team.

When it comes to formal proofs of resistance to side-channel attack, we contend that it is necessary to verify formally that the compiler used in the production of actually running code respects the resistance properties that were established in formally verified proofs. One of our objectives is to describe such a compiler (Jasmin) and show its strength on a variety of applications.

4.3. Proofs for robotics

Robots are man-made artifacts where numerous design decisions can be argued based on logical or mathematical principles. For this reason, we wish to use this domain of application as a focus for our investigations. The questions for which we are close to providing answers involve precision issues in numeric computation, obstacle avoidance and motion planning (including questions of graph theory), articulated limb cinematics and dynamics, and balance and active control.

From the mathematical perspective, these topics require that we improve our library to cover real algebraic geometry, computational geometry, real analysis, graph theory, and refinement relations between abstract algorithms and executable programs.

In the long run, we hope to exhibit robots where pieces of software and part of the design has been subject to formal verification.

5. New Software and Platforms

5.1. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

FUNCTIONAL DESCRIPTION: Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes,

- a critical bug fix related to template polymorphism,

- native 63-bit machine integers,

- a new sort of definitionally proof-irrelevant propositions: SProp,

- private universes for opaque polymorphic constants,

- string notations and numeral notations,

- a new simplex-based proof engine for the tactics lia, nia, lra and nra,

- new introduction patterns for SSReflect,

- a tactic to rewrite under binders: under,

- easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

5.2. Math-Components

Mathematical Components library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: The Mathematical Components library is a set of Coq libraries that cover the prerequiste for the mechanization of the proof of the Odd Order Theorem.

RELEASE FUNCTIONAL DESCRIPTION: This releases is compatible with Coq 8.9 and Coq 8.10 it adds many theorems for finite function, prime numbers, sequences, finite types, bigo operations, natural numbers, cycles in graphs.

- Participants: Alexey Solovyev, Andrea Asperti, Assia Mahboubi, Cyril Cohen, Enrico Tassi, François Garillot, Georges Gonthier, Ioana Pasca, Jeremy Avigad, Laurence Rideau, Laurent Théry, Russell O'Connor, Sidi Ould Biha, Stéphane Le Roux and Yves Bertot
- Contact: Assia Mahboubi
- URL: http://math-comp.github.io/math-comp/

5.3. Semantics

KEYWORDS: Semantic - Programming language - Coq

FUNCTIONAL DESCRIPTION: A didactical Coq development to introduce various semantics styles. Shows how to derive an interpreter, a verifier, or a program analyser from formal descriptions, and how to prove their consistency.

This is a library for the Coq system, where the description of a toy programming language is presented. The value of this library is that it can be re-used in classrooms to teach programming language semantics or the Coq system. The topics covered include introductory notions to domain theory, pre and post-conditions, abstract interpretation, and the proofs of consistency between all these point of views on the same programming language. Standalone tools for the object programming language can be derived from this development.

- Participants: Christine Paulin and Yves Bertot
- Contact: Yves Bertot
- URL: http://www-sop.inria.fr/members/Yves.Bertot/proofs/semantics_survey.tgz

5.4. Easycrypt

KEYWORDS: Proof assistant - Cryptography

FUNCTIONAL DESCRIPTION: EasyCrypt is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs. EasyCrypt can also be used for reasoning about differential privacy.

- Participants: Benjamin Grégoire, Gilles Barthe and Pierre-Yves Strub
- Contact: Gilles Barthe
- URL: https://www.easycrypt.info/trac/

5.5. ELPI

Embeddable Lambda Prolog Interpreter

KEYWORDS: Constraint Programming - Programming language - Higher-order logic

SCIENTIFIC DESCRIPTION: The programming language has the following features

- Native support for variable binding and substitution, via an Higher Order Abstract Syntax (HOAS) embedding of the object language. The programmer needs not to care about De Bruijn indexes.

- Native support for hypothetical context. When moving under a binder one can attach to the bound variable extra information that is collected when the variable gets out of scope. For example when writing a type-checker the programmer needs not to care about managing the typing context.

- Native support for higher order unification variables, again via HOAS. Unification variables of the meta-language (lambdaProlog) can be reused to represent the unification variables of the object language. The programmer does not need to care about the unification-variable assignment map and cannot assign to a unification variable a term containing variables out of scope, or build a circular assignment.

- Native support for syntactic constraints and their meta-level handling rules. The generative semantics of Prolog can be disabled by turning a goal into a syntactic constraint (suspended goal). A syntactic constraint is resumed as soon as relevant variables gets assigned. Syntactic constraints can be manipulated by constraint handling rules (CHR).

- Native support for backtracking. To ease implementation of search.

- The constraint store is extensible. The host application can declare non-syntactic constraints and use custom constraint solvers to check their consistency.

- Clauses are graftable. The user is free to extend an existing program by inserting/removing clauses, both at runtime (using implication) and at "compilation" time by accumulating files.

Most of these feature come with lambdaProlog. Constraints and propagation rules are novel in ELPI.

FUNCTIONAL DESCRIPTION: ELPI implements a variant of lambdaProlog enriched with Constraint Handling Rules, a programming language well suited to manipulate syntax trees with binders and unification variables.

ELPI is a research project aimed at providing a programming platform for the so called elaborator component of an interactive theorem prover.

ELPI is designed to be embedded into larger applications written in OCaml as an extension language. It comes with an API to drive the interpreter and with an FFI for defining built-in predicates and data types, as well as quotations and similar goodies that come in handy to adapt the language to the host application.

RELEASE FUNCTIONAL DESCRIPTION: improvement to the parser (parsing negative numbers) improvement to the foreign function interface (accepting ternary comparison, instead of equality) adds ternary comparisons to the standard library provides a builtin comparison cmp_term provides a builtin to check whether a term is ground

NEWS OF THE YEAR: There were 7 releases in 2019. Work done mostly in these areas:

- consolidation (documentation, bug fixes, test suits)

- API and FFI (making it easier to export host applications to ELPI)

- standard library

- Participant: Claudio Sacerdoti Coen
- Contact: Enrico Tassi
- Publications: ELPI: fast, Embeddable, λProlog Interpreter - Implementing Type Theory in Higher Order Constraint Logic Programming - Deriving proved equality tests in Coq-elpi: Stronger induction principles for containers in Coq
- URL: https://github.com/lpcic/elpi/

5.6. Coq-elpi

KEYWORDS: Metaprogramming - Extension

SCIENTIFIC DESCRIPTION: Coq-elpi provides a Coq plugin that embeds ELPI. It also provides a way to embed Coq's terms into lambdaProlog using the Higher-Order Abstract Syntax approach (HOAS) and a way to read terms back. In addition to that it exports to ELPI a set of Coq's primitives, e.g. printing a message, accessing the environment of theorems and data types, defining a new constant and so on. For convenience it also provides a quotation and anti-quotation for Coq's syntax in lambdaProlog. E.g. {{nat}} is expanded to the type name of natural numbers, or {{A -> B}} to the representation of a product by unfolding the -> notation. Finally it provides a way to define new vernacular commands and new tactics.

FUNCTIONAL DESCRIPTION: Coq plugin embedding ELPI

RELEASE FUNCTIONAL DESCRIPTION: Minor relase for extra API for global reference data types

NEWS OF THE YEAR: Releases 1.0, 1.1, and 1.2 were made in 2019, they constitute the first public release with tutorials and examples.

Work done in 2019 is mostly in these areas:

- expose a complete set of API to script Coq's vernacular language

- take advantage or recent ELPI API and FFI to convert back and forth terms containing existential variables (Evars)

- Contact: Enrico Tassi
- Publications: Deriving proved equality tests in Coq-elpi: Stronger induction principles for containers in Coq - Elpi: an extension language for Coq (Metaprogramming Coq in the Elpi λProlog dialect)

5.7. AutoGnP

KEYWORDS: Formal methods - Security - Cryptography

FUNCTIONAL DESCRIPTION: autoGnP is an automated tool for analyzing the security of padding-based public-key encryption schemes (i.e. schemes built from trapdoor permutations and hash functions). This years we extended the tool to be able to deal with schemes based on cyclic groups and bilinear maps.

- Participants: Benjamin Grégoire, Gilles Barthe and Pierre-Yves Strub
- Contact: Gilles Barthe
- URL: https://github.com/ZooCrypt/AutoGnP

5.8. MaskComp

KEYWORD: Masking

FUNCTIONAL DESCRIPTION: MaskComp is a compiler generating masked implémentation protected against side channel attack based on differential power analysis. It take a unmasked program in a syntaxe close to C and generate a new C protected program. We did not claim that the generate C program will be secure after compilation (C compiler can break protection), but it provide a good support for generating masked implementation.

- Contact: Benjamin Grégoire
- URL: https://sites.google.com/site/maskingcompiler/home

5.9. Jasmin

Jasmin compiler and analyser

KEYWORDS: Cryptography - Static analysis - Compilers

FUNCTIONAL DESCRIPTION: Analysing the execution time of a cryptographic code can be a way to discover the secret protected by this code. To avoid this pitfall, Jasmin proposes a high-level language and an analyzer for this language that makes it possible to predict when the execution of this code will happen in constant time and thus does not unveil the secret (for instance, the cryptographic key). Once the Jasmin code is valid with respect to the analyzer, the compiler produces assembly code that still preserves this property of constant time.

- Contact: Benjamin Grégoire

5.10. MaskVerif

KEYWORDS: Masking - Hardware and Software Platform

FUNCTIONAL DESCRIPTION: MaskVerif is a tool to verify the security of implementations protected against side channel attacks, in particular differential power analysis. It allows to check different security notions in the probing model: - Probing security - Non Interference - Strong Non Interference. The tool is able to analyse software implementations and hardware implementations (written in Verilog). It can prove the different security notions in presence of glitch or transition.

- Contact: Benjamin Grégoire
- URL: https://sites.google.com/view/maskverif/home

5.11. CoqEAL

The Coq Effective Algebra Library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: This library contains formal developments in algebra and optimized algorithms on mathcomp data structures and a framework to ease change of data representation during a proof.

RELEASE FUNCTIONAL DESCRIPTION: First release

- Contact: Cyril Cohen

5.12. math-comp-analysis

Mathematical Components Analysis

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: This library adds definitions and theorems for real numbers and their mathematical structures

RELEASE FUNCTIONAL DESCRIPTION: Compatible with mathcomp 1.8.0, 1.9.0, and 1.10.0

NEWS OF THE YEAR: In 2019, there were 3 releases.

- Partners: Ecole Polytechnique - AIST Tsukuba
- Contact: Cyril Cohen
- Publication: Formalization Techniques for Asymptotic Reasoning in Classical Analysis
- URL: https://github.com/math-comp/analysis

5.13. math-comp-finmap

Finite maps and ordered types library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: Support for reasoning about finite maps and ordered types

RELEASE FUNCTIONAL DESCRIPTION: This release is solely an update of order.v and set.v in order to integrate the changes in math-comp/math-comp#270

- Contact: Cyril Cohen

5.14. math-comp-real-closed

Real Closed Fields

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: Theorems for real closed fields

RELEASE FUNCTIONAL DESCRIPTION: First release

- Contact: Cyril Cohen
- URL: https://github.com/math-comp/real-closed

6. New Results

6.1. Hol-Light and Elpi

Participants: Enrico Tassi, Marco Maggesi [University of Florence, Italy].

We implemented an elaborator for HOL-Light in Elpi. In particular the new elaborator supports coercions and overloaded notations for algebraic structures.

6.2. Generating equality tests for inductive types

Participant: Enrico Tassi.

We show how to derive in a modular fashion equality tests for a wide variety of inductive type definitions. This makes an instrumental use of parametricity. This work has been published in an international conference [10]. This is also an interesting case study for the Elpi language [2].

6.3. Re-designing the state machine of Coq

Participants: Enrico Tassi, Maxime Dénès.

We redesigned the state machine of Coq to improve its support for LSP-based user interfaces [1]. In particular, we decoupled the representation of the document as seen by the User-Interface and the structured document as used by the STM to decide what to compute and how (in which order).

6.4. Formal proofs on session types

Participants: Enrico Tassi, Cinzia Di Giusto [University of Nice], Marco Giunti [New University of Lisbon], Kirstin Peters [University of Darmstadt], Antonio Ravara [New University of Lisbon].

We formalized in Coq and Isabelle a linear, monadic Pi calculus, its labelled transition system, and type system. We proved the properties of subject reduction and absence of linearity violation. These are based on De Bruijn levels and Nominals with the objective of comparing the approaches and provide automation for recurrent goals. This is done in Project PROSE (Provers for Sessions).

6.5. Formal proofs of an axiomatization of graphs with tree-width two

Participants: Christian Doczkal, Damien Pous [CNRS, ENS de lyon].

We finished the formalization of a completeness proof for an axiomatization of graphs of treewidth at most two in Coq+MathComp. This work was submitted for publication in a conference [11]. We are also revising the article presenting our proof of the Minor-Exclusion property for Treewidth-Two graphs [15] for publication in a journal. Most of the formal proofs are available from the following web-site https://perso.ens-lyon.fr/damien.pous/covece/graphs/.

6.6. Formal study of double-word arithmetic algorithms

Participants: Laurence Rideau, Jean-Michel Muller [CNRS, ENS de Lyon].

We finished the formalization of double-word arithmetic algorithms, described in the article *Tight and rigourous error bounds for basic building blocks of double-word arithmetic* [16].

Thanks to the formalization, errors were found in the proofs, but the stated results (correction of algorithms and error limits) were proven correct. On the other hand, for the purposes of this formalization, we had to develop a more general version of the proof of the Fast2Sum algorithm, which should soon be integrated into the Flocq library.

An article describing this work of formalisation is being written.

6.7. Approximations using Chebyshev polynomials

Participants: Laurent Théry, Florian Steinberg [Inria Saclay, Toccata project-team].

Florian Steinberg and Laurent Théry have been working on polynomial approximations using Chebyshev polynomials. This works has been presented at the ANR FastRelax final meeting (Lyon, June 2019) and is available as a library at https://github.com/FlorianSteinberg/Cheby.

6.8. Formalizing computational analysis

Participants: Laurent Théry, Florian Steinberg [Inria Saclay, Toccata project-team], Holger Thies [Kyushu University, Fukuoka].

[1] LSP stands for Language Server Protocol

Florian Steinberg, Holger Thies, and Laurent Théry have been working on formalizing computational analysis. This work is described in a paper to be submitted for publication [13]. A shorter version was published in a conference [9].

6.9. Formal study of probabilistic programs

Participants: Cécile Baritel-Ruet, Benjamin Grégoire, José Bacelar Almeida [INESC TEC], Manuel Barbosa [INESC TEC], Gilles Barthe [IMDEA], Sonia Belaïd [CryptoExpert], Matthew Campagna [AWS], Gaëtan Cassiers [UCL], Sunjay Cauligi [UC San Diego], Ernie Cohen [AWS], François Dupressoir [University of Surrey], Pierre-Alain Fouque [Université Rennes 1], Charlie Jacomme [LSV], Steve Kremer [Inria Nancy Grand-Est, PESTO project Team], Adrien Koutsos [LSV], Vincent Laporte [Inria], Tiago Oliveira [INESC TEC], Vitor Pereira [INESC TEC], Bernardo Portela [INESC TEC], Alley Stoughton [Boston University], François-Xavier Standaert [UCL], Deian Stefan [UC San Diego], Pierre-Yves Strub [Ecole Polytechnique], Serdar Tasiran [AWS].

We provide two differents tools:

- EasyCrypt (see http://www.easycrypt.info/) is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs.
- Jasmin (see https://github.com/jasmin-lang/jasmin) is certified compiler to generate high-speed and high-assurance cryptographic code.

6.10. Security of a key management service

Participants: Benjamin Grégoire, José Bacelar Almeida [INESC TEC], Manuel Barbosa [INESC TEC], Gilles Barthe [IMDEA], Matthew Campagna [AWS], Vitor Pereira [INESC TEC], Bernardo Portela [INESC TEC], Pierre-Yves Strub [Ecole Polytechnique], Serdar Tasiran [AWS].

We have developed a machine-checked proof of security for the domain management protocol of Amazon Web Services' KMS (Key Management Service) a critical security service used throughout AWS and by AWS customers. Domain management is at the core of AWS KMS; it governs the toplevel keys that anchor the security of encryption services at AWS. We show that the protocol securely implements an ideal distributed encryption mechanism under standard cryptographic assumptions. The proof is machine-checked in the EasyCrypt proof assistant and is the largest EasyCrypt development to date. This work corresponds to a contract with AWS and has been published in a major computer security conference [3].

6.11. High-assurance and high-speed SHA-3

Participants: Cécile Baritel-Ruet, Benjamin Grégoire, José Bacelar Almeida [INESC TEC], Manuel Barbosa [INESC TEC], Gilles Barthe [IMDEA], François Dupressoir [University of Surrey], Vincent Laporte [Inria], Tiago Oliveira [INESC TEC], Alley Stoughton [Boston University], Pierre-Yves Strub [Ecole Polytechnique].

We have developed a high-assurance and high-speed implementation of the SHA-3 hash function. Our implementation is written in the Jasmin programming language, and is formally verified for functional correctness, provable security and timing attack resistance in the EasyCrypt proof assistant. Our implementation is the first to achieve simultaneously the four desirable properties (efficiency, correctness, provable security, and side-channel protection) for a non-trivial cryptographic primitive. Concretely, our mechanized proofs show that:

1. The SHA-3 hash function is indifferentiable from a random oracle, and thus is resistant against collision, first and second preimage attacks;
2. The SHA-3 hash function is correctly implemented by a vectorized x86 implementation.

Furthermore, the implementation is provably protected against timing attacks in an idealized model of timing leaks. The proofs include new EasyCrypt libraries of independent interest for programmable random oracles and modular indifferentiability proofs. This work has been published at an international conference [4].

6.12. A domain-specific language for timing sensitive computation

Participants: Benjamin Grégoire, Sunjay Cauligi [UC San Diego], Gilles Barthe [IMDEA], Deian Stefan [UC San Diego].

Real-world cryptographic code is often written in a subset of C intended to execute in constant-time, thereby avoiding timing side channel vulnerabilities. This C subset eschews structured programming as we know it: if-statements, looping constructs, and procedural abstractions can leak timing information when handling sensitive data. The resulting obfuscation has led to subtle bugs, even in widely-used high profile libraries like OpenSSL. To address the challenge of writing constant-time cryptographic code, we have participate to the development of FaCT, a crypto DSL that provides high-level but safe language constructs. The FaCT compiler uses a secrecy type system to automatically transform potentially timing-sensitive high-level code into low-level, constant-time LLVM bitcode. While the language and the type system has been developed by our collaborator, we have formalized the constant-time transformation. We have performed an empirical evaluation that uses FaCT to implement core crypto routines from several open-source projects including OpenSSL, libsodium, and curve25519-donna. Our evaluation shows that FaCT's design makes it possible to write readable, high-level cryptographic code, with efficient, constant-time behavior. This work has been published at an international conference [7].

6.13. Proving equivalence between probabilistic programs

Participants: Benjamin Grégoire, Gilles Barthe [IMDEA], Steve Kremer [Inria Nancy Grand-Est, PESTO project Team], Pierre-Yves Strub [Ecole Polytechnique].

We have developed principled methods for proving equivalence between probabilistic programs that operate over finite fields and related algebraic structures. We have focused on three essential properties: program equivalence, information flow, and uniformity. We give characterizations of these properties based on deducibility and other notions from symbolic cryptography. We use (sometimes improve) tools from symbolic cryptography to obtain decision procedures or sound proof methods for program equivalence, information flow, and uniformity. A partial implementation of our approach is integrated in EasyCrypt and in MaskVerif. This work has been published at an international conference [6].

6.14. MaskVerif: automated verification of higher-order masking in presence of physical defaults

Participants: Benjamin Grégoire, Gilles Barthe [IMDEA], Sonia Belaïd [CryptoExpert], Gaëtan Cassiers [UCL], Pierre-Alain Fouque [Université Rennes 1], François-Xavier Standaert [UCL].

Power and electromagnetic based side-channel attacks are serious threats against the security of cryptographic embedded devices. In order to mitigate these attacks, implementations use countermeasures, among which masking is currently the most investigated and deployed choice. Unfortunately, commonly studied forms of masking rely on underlying assumptions that are difficult to satisfy in practice. This is due to physical defaults, such as glitches or transitions, which can recombine the masked data in a way that concretely reduces an implementation's security. We have developed and implemented an automated approach for verifying security of masked implementations in presence of physical defaults (glitches or transitions). Our approach helps to recover the main strengths of masking: rigorous foundations, composability guarantees, automated verification under more realistic assumptions. This work contributes to demonstrate the benefits of language-based approaches (specifically probabilistic information flow) for masking. This work was published at an international conference [5].

6.15. Frame type theory

Participants: Cyril Cohen, Assia Mahboubi [Inria Rennes Bretagne Atlantique, Gallinette project-team], Xavier Montillet [University of Nantes].

Writing modular programs in proof assistants is notoriously difficult. A significant literature and implementation effort is devoted to the topic, with approaches ranging from adding new constructions to the underlying logic, to adding features to the proof assistant. However, all current options (including records, sections and modules) are unsatisfactory in one way or another. In this work in progress we aim at reconciling several options using frames. The central idea is to consider records where some fields do not have a value yet. We will call these generalized records frames, and will say that a field is a definition (resp. abstraction) if it has (resp. does not have) a value. Frames can also be thought of as a reification of the contexts of CiC, as presented in the Coq manual.

6.16. Automated refinements on algorithms in Lean

Participants: Cyril Cohen, Tobias Grosser [ETH Zurich], Utz Haus [CRAY EMEA Research Lab], Chris Hughes [Imperial college].

We have experimented with Applying manual and automated program refinements techniques to a simple algorithm, in Lean, with Tobias Grosser, Utz Haus and Chris Hughes, in Zürich. Experiments on this topic are available at the following address https://github.com/ChrisHughes24/LP.

This work also includes investigations on parametricity in Lean as visible at the following address https://github.com/CohenCyril/mathlib/tree/param.

6.17. Parametricity in Template Coq

Participants: Cyril Cohen, Damien Rouhling, Assia Mahboubi [Inria Rennes Bretagne Atlantique, Gallinette project team], Nicolas Tabareau [Inria Rennes Bretagne Atlantique, Gallinette project team].

We study the implementation of parametricity in Template Coq and improve on the work proposed the article *Equivalence for free!* [17]. This work is available at https://github.com/CoqHott/parametricity-a-la-carte.

6.18. A hierarchy builder

Participants: Kazuhiko Sakaguchi, Cyril Cohen.

We are studying how to generate mathematical structures from their axioms using the high-level language provided by the Coq-Elpi experiment. Ongoing experiments are visible at the following address https://github.com/math-comp/hierarchy-builder.

6.19. Adding measure theory to mathematical components analysis

Participants: Cyril Cohen, Damien Rouhling, Laurence Rideau, Reynald Affeldt [AIST, Japan], Georges Gonthier [Inria Saclay Ile de France, Specfun project team], Marie Kerjean [Inria Rennes Bretagne Atlantique, Gallinette project team], Assia Mahboubi [Inria Rennes Bretagne Atlantique, Gallinette project team], Pierre-Yves Strub [Ecole Polytechnique].

We started extending mathematical components analysis [14] with measure theory and Lebesgue-Stieljes integral. We are taking inspiration from work done on Coquelicot and in the MILC project (DIM-RFSI).

6.20. A formal description of exact real arithmetic

Participants: Yves Bertot, Nicolas Magaud [University of Strasbourg].

We revisited an old package available in the contributions to the Coq system, where algorithms to perform real number computations were described. This package was using primitives described using axioms. We showed that these axioms were faulty and proposed solutions to salvage the package and make it more safely usable in the future.

6.21. Formal study of a triangulation algorithm

Participant: Yves Bertot.

We wish to describe a triangulation algorithm in a way that respects both a high level of abstraction and a precise account of pointer manipulations. Using refinements approaches as in CoqEAL, we hope that this can lead to efficient implementation that are derived from the formal description.

6.22. Formal study of Voronoi diagrams and Fortune's algorithm

Participants: Ahmed Khulaif A Alharbi, Yves Bertot.

Voronoi diagrams are an example of data that can be used to solve problems in robot motion planning. In this experiment, we provided a formal description of Fortune's algorithm to compute such diagrams, together with a framework to animate this algorithm. Formal proofs of correctness will be the next step.

6.23. Formal study of a cell-decomposition algorithm

Participants: Julien Lamiroy, Yves Bertot.

To solve robot motion planning problems, a simple approach is to decompose the available space into obstacle-free cells and to more from one cell to another only by boundaries that are also obstacle free. We developed a formal description of an algorithm producing this kind of decomposition, with the aim of providing formal proofs of correctness in the long run.

6.24. A guide to use Coq for security evaluations

Participants: Maxime Dénès, Yves Bertot, Vincent Laporte, Arnaud Fontaine [ANSSI], Thomas Letan [ANSSI].

Common Criteria are an international standard for computer security certification. Evaluations are rated with Evaluation Assurance Levels, from 1 to 7. Eal6 and EAL7 require developers to conduct a formal analysis of their product with respect to certain security properties.

In France, the Certification Body (the entity emitting Common Criteria certificates) is part of the ANSSI (*l'Agence Nationale de la Sécurité des Systèmes d'Information*, also referred to as the French Cybersecurity Agency), and is one of the few emitters of EAL6 and EAL7 certificates.

Coq has already been used to support Common Criteria formal analysis. The ANSSI and Inria have been collaborating on an authoritative document to introduce guidelines and rules for formal analyses supported by Coq, in order to make these developments easier to read and evaluate.

6.25. Formalization of the Poincaré disk model in Isabelle

Participants: Pierre Boutry, Danijela Simić [University of Belgrade], Filip Marić [University of Belgrade].

The Poincaré disk model is a model that can be shown to satisfy all axioms of Tarski's system of geometry at the exception of the parallel postulate. We developed a formal proof of this fact in the Isabelle system and submitted an article for publication. Reviewers suggested that we add a proof that the postulate of the existence of limiting parallels does hold. This completes neatly the work on this topic, as it allows us to exhibit that the Poincaré disk model is not only a counter-model for the parallel postulate but also a model of hyperbolic geometry. An improved version of the article will be submitted soon.

6.26. Integration of the GeoCoq library to Logipedia

Participants: Pierre Boutry, Gaspard Ferey [Inria Saclay Ile de France, Deducteam project team].

We have proofs of independence of the parallel postulate for several models of hyperbolic geometry (among which the Poincaré disk model). An objective is to provide formal proofs that these models are actually isomorphic. An issue for this objective is the question of re-usability, because the formal proofs that we have so far exist in the realms of different theorem provers. The Logipedia effort is an attempt to make proofs from different proofs systems work together, by using a tool called Dedukti as a go-between. A particular point is to be able to translate proofs already done in Coq, namely the GeoCoq library, into proofs verifiable by Dedukti. This requires handling tactics based on internal computation (reflective tactics), that we used intensively in our Coq proof. However, handling reflective tactics is currently not well supported by Dedukti. This is our current point of attention.

6.27. Performance improvements for a reflective tactic in the GeoCoq library

Participants: Pierre Boutry, Benjamin Grégoire, Enrico Tassi.

The GeoCoq library relies on a reflective tactic. It is an interesting topic to understand how to make such a tactic more efficient. A first pass on the algorithm makes that we manage to gain 15% of performance for the whole library and several orders of magnitude on specific subgoals. Another area of the tactic can also be improved by relying on Coq-Elpi.

6.28. Mutual interpretability of cartesian planes with Tarski's system of geometry

Participants: Pierre Boutry, Cyril Cohen.

A previous result by Pierre Boutry is that cartesian planes over pythagorean ordered fields are mutually interpretable with Tarski's system of geometry without the continuity axiom. This result can be extended by linking cartesian planes over real closed fields and the full Tarski system of geometry, understanding the continuity axiom as an implementation of Dedekind cuts. On the one hand, this requires a new proof that is not already found in the literature, on the other hand, this will result in a verified quantifier elimination procedure for Tarski's system of geometry, thus extending previous work by Cyril Cohen.

6.29. Simplification of a constructive version of Tarski's system of geometry

Participant: Pierre Boutry.

Our long term project is to show the independence of all thirteen axioms in a variant of Tarski's system of geometry. In the current situation, ten axioms have been checked to be independent using counter-models. Specific questions arise around the continuity axiom and decidability of equality between points. This is related to investigations concerning mutual interpretability with cartesian planes and an alternative system proposed by Michael Beeson.

6.30. Formal proofs of Tarjan's strongly connected components algorithm

Participants: Cyril Cohen, Laurent Théry, Ran Chen [Institute of Software, Chinese Academy of Science, Beijing], Jean-Jacques Lévy [Inria Paris, $\pi.r^2$ project-team], Stephan Merz [Inria Nancy Grand Est, Veridis project-team].

Comparing provers on a formalization of the same problem is always a valuable exercise. In this work, we present the formal proof of correctness of a non-trivial algorithm from graph theory that was carried out in three proof assistants: Why3, Coq, and Isabelle. This was published in an international conference [8].

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR

- FastRelax, "Fast and Reliable Approximations", started on October 1st, 2014, for 60 months (ending in September 2019), with a grant of 75 kEuros for Marelle. Other partners are Inria Grenoble (ARIC project-team), LAAS-CNRS (Toulouse), Inria Saclay (Toccata and Specfun project-teams), and LIP6-CNRS (Paris). The corresponding researcher for this contract is Laurence Rideau.
- TECAP "Analyse de protocoles, Unir les outils existants", starting on October 1st, 2017, for 60 months, with a grant of 89 kEuros. Other partners are Inria teams PESTO (Inria Nancy grand-est), Ecole Polytechnique, ENS Cachan, IRISA Rennes, and CNRS. The corresponding researcher for this contract is Benjamin Grégoire.
- SafeTLS "La sécurisation de l'Internet du futur avec TLS 1.3" started on October 1st, 2016, for 60 months, with a grant of 147kEuros. Other partners are Université de Rennes 1, and secrétariat Général de la Défense et de la Sécurité Nationale. The corresponding researcher for this contract is Benjamin Grégoire.
- BRUTUS "Chiffrements authentifiés et résistants aux attaques par canaux auxiliaires", started on October 1st, 2014, for 60 months, with a grant of 41 kEuros for STAMP. Other partners are Université de Rennes 1, CNRS, secrétariat Général de la défense et de la sécurité nationale, and Université des Sciences et Technologies de Lille 1. The corresponding researcher for this contract is Benjamin Grégoire.
- Scrypt "Compilation sécurisée de primitives cryptographiques" started on February 1st, 2019, for 48 months, with a grant of 100 kEuros. Other partners are Inria team Celtique (Inria Rennes Bretagne Atlantique), Ecole polytechnique, and AMOSSYS SAS. The corresponding researcher for this contract is Benjamin Grégoire.

7.1.2. FUI

The acronym *FUI* stands for "fonds unique interministériel" and is aimed at research and development projects in pre-industrial phase. The STAMP team is part of one such project.

- VERISICC (formal verification for masking techniques for security against side-channel attacks). This contract concerns 5 partners: CRYPTOEXPERTS a company from the Paris region (Île de France), ANSSI (Agence Nationale de Sécurité des Systèmes d'Information), Oberthur Technologies, University of Luxembourg, and STAMP. A sixth company (Ninjalabs) acts as a sub-contractant. The financial grant for STAMP is 391 kEuros, including 111kEuros that are reserved for the sub-contractant. This project started in October 2018 for a duration of 4 years. The corresponding researcher for this contract is Benjamin Grégoire.

7.2. European Initiatives

7.2.1. Collaborations in European Programs, Except FP7 & H2020

Program: COST

Project acronym: EUTypes

Project title: The European research network on types for programming and verification (EUTypes)

Coordinator: Prof. Herman Geuvers, Radboud University, The Netherlands

Abstract:This COST Action will give a strong impetus to research on type theory and its many applications in computer science, by promoting (1) the synergy between theoretical computer scientists, logicians and mathematicians to develop new foundations for type theory, for example as based on the recent development of "homotopy type theory", (2) the joint development of type theoretic tools as proof assistants and integrated programming environments, (3) the study of dependent types for programming and its deployment in software development, (4) the study of dependent types for verification and its deployment in software analysis and verification. The action will also tie together these different areas and promote cross-fertilisation.

7.2.2. Collaborations with Major European Organizations

Partner 1: MPI Bochum, Gilles Barthe, Germany

Formally verified cryptography

7.3. International Initiatives

7.3.1. Informal International Partners

We have strong collaborations with AIST in Japan. Reynald Affeldt, a researcher from AIST has been visiting our team since October 1st 2019. The topic of choice is formalization of a variety of topics using the Mathematical Components library, aiming mostly at formalizing robotics.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

We received the visit of Marc Gourjon (Technische Universität Hamburg) in April and from Manuel Barbosa (University of Porto) in June and July.

We received the visit of Reynald Affeldt (AIST, Japan) starting on October 1st.

We received the visit of Kazuhiko Sakaguchi (University of Tsukuba), from January 1st to October 31st.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events: Organisation

8.1.1.1. Member of the Organizing Committees

Yves Bertot and Maxime Dénès organized the Coq User and Developer workshop in Sophia Antipolis, June 3–7, 2019.

8.1.2. Scientific Events: Selection

8.1.2.1. Member of the Conference Program Committees

Enrico Tassi was a member of the PC for PADL and LFMTP. Benjamin Grégoire was a member of the PC for latincrypt and PriSC. Cyril Cohen was a member of the PC for CICM. Yves Bertot was a member of the PC for ITP.

8.1.2.2. Reviewer

Benjamin Grégoire: Conference on Computer and Communication Security (CCS), Certified Programs and Proofs (CPP), Interactive Theorem Proving (ITP). Cyril Cohen: LICS.

8.1.3. Journal

8.1.3.1. Reviewer - Reviewing Activities

Laurent Théry: Journal of Automated Reasoning, Science of Computer Programming, Formal Aspects of Computing. Cyril Cohen: MSCS (Journal on Mathematical Structures in Computer Science), Journal of Automated Reasoning, Theoretical Computer Science, Information Processing Letters.

8.1.4. Invited Talks

Enrico Tassi gave talks at the Coq developer Working Group on the migration of the package index from opam v1 to opam v2 and at the Coq workshop on the improvements to SSReflect in Coq 8.10 (with E. Martin-Dorel from the University of Toulouse). He also gave a talk in the seminar of the Inria Parsifal project-team on synthesizing proved equality tests.

Benjamin Grégoire gave an invited talk at PriSC (Principles of Secure Compilation) in January 2019.

Cyril Cohen gave a talk at the *Lean Together* meeting in 2019, on using unification hints.

8.1.5. Scientific Expertise

Yves Bertot is a member of the steering committee for the Inria-Nomadic Labs collaboration.

Yves Bertot is a member of the steering committee for the ITP series of conferences.

Yves Bertot was a member of the hiring committee for an associate professor at ENSIIE (Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise) in Evry, France.

8.1.6. Research Administration

Benjamin Grégoire is a member of the CUMI committee facilitating communication between IT services, administrative services, and researchers at Inria.

Yves Bertot was coordinator for the evaluation of the theme "Proof and verification" in March 2019. This work involved polling 13 heads of Inria project teams for names of international experts, selecting an appropriate panel, and attributing a small group of experts to each project-team.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Master : Yves Bertot, "Proofs and reliable programming using Coq", 21hours ETD, Sept-Nov 2019, Université Nice Côte d'Azur, France.

Doctorat : Benjamin Grégoire, "EasyCrypt and Jasmin", 14 ETD, summer school on Formal Techniques, Menlo College, California, USA, 18–25 May 2019.

Doctorat : Benjamin Grégoire, "Formal verification of masked implementations" Summer school on security of software/Hardware interfaces, 3 ETD, 8–12 July 2019, Inria, France.

Doctorat : Yves Bertot "Coq introductory course", EUTypes summer school, 6 ETD, Ohrid, Aug. 30–Sep. 4, 2019, North Macedonia.

8.2.2. Supervision

PhD: Damien Rouhling, *Formalization Tools for Classical Analysis – A Case Study in Control Theory*, Université Côte d'Azur, September 2019, supervised by Yves Bertot and Cyril Cohen [1].

Yves Bertot and Laurence Rideau supervise the doctoral thesis of Sophie Bernard.

Yves Bertot and Benjamin Grégoire supervise the doctoral thesis of Cécile Baritel-Ruet.

Benjamin Grégoire and Tamara Rezk (Indes) supervise the doctoral thesis of Mohamad El Laz.

8.2.3. Juries

Yves Bertot was a member of the Jury for the Habilitation to direct research of Guillaume Melquiond.

Laurent Théry was a member of the thesis committee for David Braun (University of Strasbourg).

Laurent Théry was an external reviewer for a PhD at ANU (Australia, anonymity rules apply).

Enrico Tassi was a member of the Jury for the PhD defense of Ulysse Gérard (Inria Saclay).

Yves Bertot was a member of the Jury with report duties for the thesis of Florian Faissole (University of Paris-Saclay).

Yves Bertot was a member of the Jury with report duties for the thesis of Armaël Guéneau (University of Paris Diderot).

Yves Bertot was a member of the Jury for the thesis of Gaëtan Gilbert (Institut Mines Télécom Atlantique, Nantes).

8.3. Popularization

8.3.1. Internal action

- Yves Bertot gave a talk on Coq in the *Café-In* series of seminars for all publics of Inria personnel.
- Yves Bertot gave a general presentation on Coq in the *Doctoral seminar* of Inria Sophia Antipolis Méditerranée.

9. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] D. ROUHLING. *Formalisation Tools for Classical Analysis - A Case Study in Control Theory*, Université Côte d'Azur, September 2019, https://hal.inria.fr/tel-02333396

Articles in International Peer-Reviewed Journals

[2] F. GUIDI, C. SACERDOTI COEN, E. TASSI. *Implementing Type Theory in Higher Order Constraint Logic Programming*, in "Mathematical Structures in Computer Science", March 2019, vol. 29, n^{o} 8, pp. 1125-1150, https://hal.inria.fr/hal-01410567

International Conferences with Proceedings

[3] J. B. ALMEIDA, M. BARBOSA, G. BARTHE, M. CAMPAGNA, E. COHEN, B. GRÉGOIRE, V. PEREIRA, B. PORTELA, P.-Y. STRUB, S. TASIRAN. *A Machine-Checked Proof of Security for AWS Key Management Service*, in "ACM CCS 2019 - 26th ACM Conference on Computer and Communications Security", London, United Kingdom, ACM Press, November 2019, vol. 16, pp. 63-78 [*DOI :* 10.1145/3319535.3354228], https://hal.archives-ouvertes.fr/hal-02404540

[4] J. B. ALMEIDA, C. BARITEL-RUET, M. BARBOSA, G. BARTHE, F. DUPRESSOIR, B. GRÉGOIRE, V. LAPORTE, T. OLIVEIRA, A. STOUGHTON, P.-Y. STRUB. *Machine-Checked Proofs for Cryptographic Standards: Indifferentiability of Sponge and Secure High-Assurance Implementations of SHA-3*, in "CCS 2019 - 26th ACM Conference on Computer and Communications Security", London, United Kingdom, ACM Press, November 2019, pp. 1607-1622 [*DOI :* 10.1145/3319535.3363211], https://hal.archives-ouvertes.fr/hal-02404581

[5] G. BARTHE, S. BELAÏD, G. CASSIERS, P.-A. FOUQUE, B. GRÉGOIRE, F.-X. STANDAERT. *Automated Verification of Higher-Order Masking in Presence of Physical Defaults*, in "ESORICS 2019 - 24th European Symposium on Research in Computer Security", Luxembourg, Luxembourg, September 2019, pp. 300-318 [*DOI :* 10.1007/978-3-030-29959-0_15], https://hal.archives-ouvertes.fr/hal-02404662

[6] G. BARTHE, B. GRÉGOIRE, C. JACOMME, S. KREMER, P.-Y. STRUB. *Symbolic Methods in Computational Cryptography Proofs*, in "CSF2019 - 32nd IEEE Computer Security Foundations Symposium", Hoboken, United States, IEEE, June 2019, pp. 136-13615 [*DOI :* 10.1109/CSF.2019.00017], https://hal.archives-ouvertes.fr/hal-02404701

[7] S. CAULIGI, G. SOELLER, B. JOHANNESMEYER, F. BROWN, R. S. WAHBY, J. RENNER, B. GRÉGOIRE, G. BARTHE, R. JHALA, D. STEFAN. *FaCT: A DSL for Timing-Sensitive Computation*, in "PLDI 2019 - 40th ACM SIGPLAN Conference on Programming Language Design and Implementation", Phoenix, United States, June 2019 [*DOI :* 10.1145/3314221.3314605], https://hal.archives-ouvertes.fr/hal-02404755

[8] R. CHEN, C. COHEN, J.-J. LEVY, S. MERZ, L. THÉRY. *Formal Proofs of Tarjan's Strongly Connected Components Algorithm in Why3, Coq and Isabelle*, in "ITP 2019 - 10th International Conference on Interactive Theorem Proving", Portland, United States, J. HARRISON, J. O'LEARY, A. TOLMACH (editors), Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2019, vol. 141, pp. 13:1 - 13:19 [*DOI :* 10.4230/LIPICS.ITP.2019.13], https://hal.inria.fr/hal-02303987

[9] F. STEINBERG, H. THIES, L. THÉRY. *Quantitative continuity and Computable Analysis in Coq*, in "ITP 2019 - Tenth International Conference on Interactive Theorem Proving", Portland, United States, 2019, The version accepted to the conference can be accessed at https://drops.dagstuhl.de/opus/volltexte/2019/11083/ [*DOI :* 10.4230/LIPICS.ITP.2019.28], https://hal.archives-ouvertes.fr/hal-02426470

[10] E. TASSI. *Deriving proved equality tests in Coq-elpi: Stronger induction principles for containers in Coq*, in "ITP 2019 - 10th International Conference on Interactive Theorem Proving", Portland, United States, September 2019 [*DOI :* 10.4230/LIPICS.CVIT.2016.23], https://hal.inria.fr/hal-01897468

Conferences without Proceedings

[11] C. DOCZKAL, D. POUS. *Completeness of an Axiomatization of Graph Isomorphism via Graph Rewriting in Coq*, in "CPP 2020 - 9th ACM SIGPLAN International Conference on Certified Programs and Proofs", New Orleans, LA, United States, Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP '20), January 2020 [*DOI :* 10.1145/3372885.3373831], https://hal.archives-ouvertes.fr/hal-02333553

Other Publications

[12] B. M. KAPRON, F. STEINBERG. *Type-two polynomial-time and restricted lookahead*, February 2019, https://arxiv.org/abs/1801.07485 - working paper or preprint, https://hal.inria.fr/hal-02018934

[13] F. STEINBERG, L. THÉRY, H. THIES. *Quantitative continuity and computable analysis in Coq*, April 2019, working paper or preprint, https://hal.inria.fr/hal-02088293

References in notes

[14] R. AFFELDT, C. COHEN, D. ROUHLING. *Formalization Techniques for Asymptotic Reasoning in Classical Analysis*, in "Journal of Formalized Reasoning", October 2018, https://hal.inria.fr/hal-01719918

[15] C. DOCZKAL, D. POUS. *Graph Theory in Coq: Minors, Treewidth, and Isomorphisms*, May 2019, working paper or preprint, https://hal.archives-ouvertes.fr/hal-02127698

[16] M. M. JOLDES, J.-M. MULLER, V. POPESCU. *Tight and rigourous error bounds for basic building blocks of double-word arithmetic*, in "ACM Transactions on Mathematical Software", 2017, vol. 44, n^{o} 2, pp. 1 - 27 [*DOI :* 10.1145/3121432], https://hal.archives-ouvertes.fr/hal-01351529

[17] N. TABAREAU, É. TANTER, M. SOZEAU. *Equivalences for Free*, in "Proceedings of the ACM on Programming Languages", September 2018, vol. 2, n^{o} ICFP, pp. 1-29 [*DOI :* 10.1145/3234615], https://hal.inria.fr/hal-01559073