RESEARCH CENTRE

Inria Center
at Université Côte d'Azur

2022
ACTIVITY REPORT

Project-Team
STAMP

Safety Techniques based on Formalized Mathematical Proofs

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Inria

Contents

Project-Team STAMP

Creation of the Project-Team: 2019 November 01

Keywords

Computer sciences and digital sciences

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 members, visitors, external collaborators

Research Scientists

- Yves Bertot [Team leader, INRIA, Senior Researcher, HDR]
- Cyril Cohen [INRIA, Researcher]
- Benjamin Grégoire [INRIA, Researcher]
- Laurence Rideau [INRIA, Researcher]
- Enrico Tassi [INRIA, Researcher]
- Laurent Théry [INRIA, Researcher]

PhD Student

- Swarn Priya [INRIA]

Technical Staff

- Pierre Boutry [INRIA, Engineer]
- Maxime Denes [INRIA]
- Jean-Christophe Lechenet [INRIA, Engineer]

Interns and Apprentices

- Quentin Corradi [ENS Lyon, Intern, from May 2022 until Aug 2022]
- Léon Ducruet [ENS Lyon, Intern, from May 2022 until Jul 2022]
- Lucas Tabary-Maujean [ENS Paris-Saclay, Intern, from Jun 2022 until Jul 2022]
- Mohit Kumar Tekriwal [University of Michigan, from Oct 2022, Visit supported by the French Embassy in the USA]

Administrative Assistant

- Nathalie Bellesso [INRIA]

Visiting Scientists

- Andreas Hulsing [UNIV EINDHOVEN, from Sep 2022]
- Matteo Manighetti [UNIV BOLOGNE, from Oct 2022 until Oct 2022]

External Collaborator

- Gilles Barthe [INSTITUT MAX-PLANCK, HDR]

2 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 promoting its usage by academic 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 usually recorded in definitions and theorems, but also the practical knowledge that is recorded in mathematical practice, idioms, 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 their 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 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 a 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 and 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 library 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 countries (Denmark, France, Germany, Japan, Singapore, Spain, Sweden, UK, USA) 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 aspects and the capability to establish relations between several implementations of a problem. The resulting programming logic is called *probabilistic relational Hoare logic*. We also 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 use 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 attacks, 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 develop such a compiler (Jasmin) and show its strength on a variety of applications.

The pair of tools EasyCrypt and Jasmin has also proved its worth in the formal verification of correctness for post-quantum cryptography.

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 kinematics 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 have been subject to formal verification.

5 Highlights of the year

The software tool EasyCrypt and Jasmin that we develop are instrumental for the Libjade library, with a notable application to the formal verification of NIST-approved post-quantum cryptography algorithms like Dilithium.

6 New software and platforms

6.1 New software

6.1.1 Coq

Name: 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 integrated development environment (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 Contributions: Coq version 8.16 integrates changes to the Coq kernel and performance improvements along with a few new features. We highlight some of the most impactful changes here:

The guard checker (see Guarded) now ensures strong normalization under any reduction strategy.

Irrelevant terms (in the SProp sort) are now squashed to a dummy value during conversion, fixing a subject reduction issue and making proof conversion faster.

Introduction of reversible coercions, which allow coercions relying on meta-level resolution such as type-classes or canonical structures. Also allow coercions that do not fullfill the uniform inheritance condition.

Generalized rewriting support for rewriting with Type-valued relations and in Type contexts, using the Classes.CMorphisms library.

Added the boolean equality scheme command for decidable inductive types.

Added a Print Notation command.

Incompatibilities in name generation for Program obligations, eauto treatment of tactic failure levels, use of ident in notations, parsing of module expressions.

Standard library reorganization and deprecations.

Improve the treatment of standard library numbers by Extraction.

See https://coq.inria.fr/refman/changes.html#version-8-16 for a detailed changelog.

News of the Year: Coq version 8.16 integrates changes to the Coq kernel and performance improvements along with a few new features. See the detailed changes at https://coq.inria.fr/refman/changes.html#version-8-16 for an overview of the new features and changes, along with the full list of contributors.

URL: http://coq.inria.fr/

Contact: Matthieu Sozeau

Participants: Yves Bertot, Frederic Besson, Tej Chajed, Cyril Cohen, Pierre Corbineau, Pierre Courtieu, Maxime Denes, Jim Fehrle, Julien Forest, Emilio Jesús Gallego Arias, Gaetan Gilbert, Georges Gonthier, Benjamin Grégoire, Jason Gross, Hugo Herbelin, Vincent Laporte, Olivier Laurent, Assia Mahboubi, Kenji Maillard, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pedrot, Clément Pit-Claudel, Kazuhiko Sakaguchi, Vincent Semeria, Michael Soegtrop, Arnaud Spiwack, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Anton Trunov, Li-Yao Xia, Theo Zimmermann, Gaetan Gilbert

Partners: CNRS, Université Paris-Sud, ENS Lyon, Université Paris-Diderot

6.1.2 Math-Components

Name: 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.

URL: https://math-comp.github.io/

Contact: Assia Mahboubi

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, Yves Bertot

6.1.3 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.

News of the Year: In 2021, Benjamin Gregoire, Adrien Koutsos and Pierre-Yves Strub extended the EasyCrypt proof assistant to reason about complexity, by adding a Hoare logic to prove computational complexity (execution time and oracle calls) of adversarial computations. This Hoare logic is built on top of EasyCrypt module system used to model adversaries, which has been extended to support complexity restrictions.

URL: https://www.easycrypt.info/trac/

Publications: hal-03352062, hal-03469015

Contact: Gilles Barthe

Participants: Benjamin Grégoire, Gilles Barthe, Pierre-Yves Strub, Adrien Koutsos

6.1.4 ELPI

Name: 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 does not need to care about technical devices to handle bound variables, like De Bruijn indices.

- 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 get 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 uses 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 features 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 Contributions: - Support for recent ocaml - New parser based on menhir - New tracing facility and corresponding GUI for vscode

News of the Year: New parser based on Menhir. New trace browsing facility.

URL: https://github.com/lpcic/elpi/

Publications: hal-01176856, hal-01410567, hal-01897468

Contact: Enrico Tassi

Participants: Enrico Tassi, Claudio Sacerdoti Coen

6.1.5 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 Contributions: - support for universe polymorphism - support for raw or elaborated arguments - new APIs for reduction and modules - new schema to derive equality tests

News of the Year: Experimental support for universe polymorphism. New schema for efficient synthesis of equality tests. New API- for modules and functors.

Publications: hal-01897468, hal-01637063

Contact: Enrico Tassi

Participant: Enrico Tassi

6.1.6 Jasmin

Name: Jasmin compiler and analyser

Keywords: Cryptography, Static analysis, Compilers

Functional Description: The Jasmin programming language smoothly combines high-level and low-level constructs, so as to support "assembly in the head" programming. Programmers can control many low-level details that are performance-critical: instruction selection and scheduling, what registers to spill and when, etc. The language also features high-level abstractions (variables, functions, arrays, loops, etc.) to structure the source code and make it more amenable to formal verification. The Jasmin compiler produces predictable assembly and ensures that the use of high-level abstractions incurs no run-time penalty.

The semantics is formally defined to allow rigorous reasoning about program behaviors. The compiler is formally verified for correctness (the proof is machine-checked by the Coq proof assistant). This justifies that many properties can be proved on a source program and still apply to the corresponding assembly program: safety, termination, functional correctness...

Jasmin programs can be automatically checked for safety and termination (using a trusted static analyzer). The Jasmin workbench leverages the EasyCrypt toolset for formal verification. Jasmin programs can be extracted to corresponding EasyCrypt programs to prove functional correctness, cryptographic security, or security against side-channel attacks (constant-time).

Release Contributions: It contains the following major improvements: - a new instruction "#random-bytes" to fill an array with "random" data, - access to mmx registers, - support for Windows calling convention, in addition to Linux, - "else if" blocks for readability, - strict preservation of source-level intrinsics, - an option to extract all the functions of a file to EasyCrypt, - many fixes to the extraction to EasyCrypt.

News of the Year: In 2022, two major versions were released. The first one, 2022.04.0, is the result of more than two years of development, and thus brings major new features to the language, including the support for local functions and sub-arrays. The second one, 2022.09.0, adds in particular the support for system calls, which allows for instance to call an operating system function returning random data.

Work to support multiple architecture has progressed well. The support for ARM 32 bits was merged and is being polished.

Work on "Speculative Load Hardening", a transformation making a program resistant to some Spectre attacks, are ongoing. In parallel, another line of work tries to add arrays whose length is not known at compile time.

URL: https://github.com/jasmin-lang/jasmin

Publications: hal-03844366, hal-03430789, hal-03352062, hal-02404581, hal-02974993, hal-01649140

Contact: Benjamin Grégoire

Participants: Gilles Barthe, Benjamin Grégoire, Adrien Koutsos, Vincent Laporte, Jean-Christophe Lechenet, Swarn Priya

Partners: The IMDEA Software Institute, Ecole Polytechnique, Universidade do Minho, Université de Porto, Max Planck Institute for Security and Privacy

6.1.7 Math-comp-analysis

Name: Mathematical Components Analysis

Keyword: Proof assistant

Functional Description: This library adds definitions and theorems to the Math-components library for real numbers and their mathematical structures.

Release Contributions: The main change is the split into two packages coq-mathcomp-classical and coq-mathcomp-analysis.

News of the Year: In 2022 were added - the theory of simple functions, Lebesgue measure and Lebesgue integral, - Arzelà–Ascoli theorem, - support for subspace topology - part of the theory of continuity of real functions on a segment, - the theory of finitely supported sums and infinite sums of extended reals, - the theory of partial and total surjections, injections and bijections, - fast and ad-hoc automated positivity proofs, - splitting in two sub-packages mathcomp-classical and mathcomp-analysis.

URL: https://github.com/math-comp/analysis

Publications: hal-02463336, hal-03917948, hal-01719918

Contact: Cyril Cohen

Participants: Cyril Cohen, Georges Gonthier, Marie Kerjean, Assia Mahboubi, Damien Rouhling, Pierre Roux, Laurence Rideau, Pierre-Yves Strub, Reynald Affeldt, Laurent Théry, Yves Bertot, Zachary Stone

Partners: Ecole Polytechnique, AIST Tsukuba, Onera

6.1.8 Hierarchy Builder

Keywords: Coq, Metaprogramming

Scientific Description: It is nowadays customary to organize libraries of machine checked proofs around hierarchies of algebraic structures. One influential example is the Mathematical Components library on top of which the long and intricate proof of the Odd Order Theorem could be fully formalized. Still, building algebraic hierarchies in a proof assistant such as Coq requires a lot of manual labor and often a deep expertise in the internals of the prover. Moreover, according to our experience, making a hierarchy evolve without causing breakage in client code is equally tricky: even a simple refactoring such as splitting a structure into two simpler ones is hard to get right. Hierarchy Builder is a high level language to build hierarchies of algebraic structures and to make these hierarchies evolve without breaking user code. The key concepts are the ones of factory, builder and abbreviation that let the hierarchy developer describe an actual interface for their library. Behind that interface the developer can provide appropriate code to ensure retro compatibility. We implement the Hierarchy Builder language in the hierarchy-builder addon for the Coq system using the Elpi extension language.

Functional Description: Hierarchy Builder is a high level language for Coq to build hierarchies of algebraic structures and to make these hierarchies evolve without breaking user code. The key concepts are the ones of factory, builder and abbreviation that let the hierarchy developer describe an actual interface for their library. Behind that interface the developer can provide appropriate code to ensure retro compatibility.

Release Contributions: Support for hierarchy of morphisms and bugfixes. Adding compatibility with Coq 8.16

News of the Year: Support for structures on function spaces (eg morphisms). New HB.howto command to find missing instances on a key.

URL: https://github.com/math-comp/hierarchy-builder

Publication: hal-02478907

Contact: Enrico Tassi

Participants: Enrico Tassi, Cyril Cohen

Partners: University of Tsukuba, Onera

6.1.9 Abel - Ruffini

Name: A proof of Abel-Ruffini theorem.

Keywords: Number theory, Formalisation, Proof assistant

Functional Description: A proof of Galois Theorem (equivalence between being solvable by radicals and having a solvable Galois group) and Abel - Ruffini Theorem (unsolvability of quintic equations) in the Coq proof-assistant and using the Mathematical Components library.

Release Contributions: This is a full proof Coq/mathcomp of Galois and Abel-Ruffini theorem about the unsolvability of the quintic. It is compatible with mathcomp version 1.12 to 1.15 and Coq from 8.10 to 8.16.

URL: https://github.com/math-comp/Abel

Contact: Cyril Cohen

Partner: Ecole Polytechnique

6.1.10 Semantics

Keywords: Semantic, Programming language, Coq

Functional Description: A didactical Coq development to introduce various semantics styles. Shows how to derive an interpreter, a compiler, 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, compilation, and the proofs of consistency between all these points of view on the same programming language. Standalone tools for the object programming language can be derived from this development.

Release Contributions: This version now contains an example of small compiler and a partial correctness proof (completeness).

URL: https://github.com/coq-community/semantics

Contact: Yves Bertot

Participants: Christine Paulin, Yves Bertot

7 New results

7.1 A logic for expectation

Participants: Martin Avanzini *(Focus)*, Gilles Barthe *(MPI-SP, Germany, IMDEA, Spain)*, Benjamin Grégoire, Georg Moser *(University of Innsbruck)*, Gabriele Vanoni *(Focus)*.

We have developed a new logic for bounding the expectation of a function in a probabilistic program. This logic has been integrated in the tool Eassycrypt and used to bound the expected cost of two algorithms: randomized qselect (a variant of quick sort) and the skip-list data-structure.

7.2 Formal verification of Dilithium

Participants: Manuel Barbosa *(University of Porto, INESC TEC, Portugal)*, Gilles Barthe *(MPI-SP, Germany, IMDEA, Spain)*, Christian Doczkal *(MPI-SP, Germany)*, Jelle Don *(Centrum voor Wiskunde en Informatica, the Netherlands)*, Serge Fehr *(Centrum voor Wiskunde en Informatica, Leiden University, the Netherlands)*, Benjamin Grégoire, Andreas Hülsing *(Eindhoven University of Technology, the Netherlands)*, Yu-Hsuan Huang *(Centrum voor Wiskunde en Informatica, the Netherlands)*, Yi Lee *(University of Maryland, USA)*, Pierre-Yves Strub *(Meta)*, Xiaodi Wu *(University of Maryland, USA)*.

In 2016, NIST initiated a competition for standardizing cryptographic algorithms that could withstand quantum adversaries. The competition recently reached an important milestone with the selection of four standards: one KEM (Kyber) and three signature algorithms (Dilithium, Falcon, Sphincs+). Dilithium is a lattice-based digital signature based on the Fiat-Shamir with aborts (FSa) paradigm introduced by Lyubashevsky. We identified a subtle gap that appears in several ROM and QROM security proofs of Dilithium and other schemes based on FSa. Second, we provided fixed proofs, both for the ROM and the QROM. Third, we mechanized the ROM proof completely in the EasyCrypt proof assistant. This work is based on the logic for expectation presented in the previous section.

7.3 Resistance to timing attacks and Spectre

Participants: Basavesh Ammanaghatta Shivakumar *(MPI-SP, Germany)*, Gilles Barthe *(MPI-SP, Germany, IMDEA, Spain)*, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira *(University of Porto, INESC TEC, Portugal)*, Swarn Priya, Peter Schwabe *(MPI-SP, Germany and Radboud University, the Netherlands)*, Lucas Tabary-Maujean *(Ens Paris Saclay)*.

The current gold standard of cryptographic software is to write efficient libraries with systematic protections against timing attacks. In order to meet this goal, cryptographic engineers increasingly use high-assurance cryptography tools. However, high-assurance tools reason about overly simple execution models that elide transient execution leakage. Thus, implementations validated by high-assurance cryptography tools remain potentially vulnerable to transient execution attacks such as Spectre or Meltdown. Moreover, proposed countermeasures are not used in practice due to performance overhead. We have proposed, analyzed, implemented and evaluated an approach for writing efficient cryptographic implementations that are protected against Spectre v1 attacks in Jasmin. Our approach ensures speculative constant-time. Speculative constant-time is enforced by means of a (value-dependent) information flow type system. We implemented our approach in the Jasmin framework for high-assurance cryptography, and use it for protecting all implementations of an experimental cryptographic library named Libjade that includes highly optimized implementations of symmetric primitives, of elliptic-curve cryptography, and of Kyber, a lattice-based KEM recently selected by NIST for standardization. The performance impact of our protections is very low; for example, less than 1% for Kyber and essentially zero for the curve X25519.

7.4 Enforcing fine-grained constant-time policies

Participants: Basavesh Ammanaghatta Shivakumar *(MPI-SP, Germany)*, Gilles Barthe *(MPI-SP, Germany, IMDEA, Spain)*, Benjamin Grégoire, Vincent Laporte, Swarn Priya.

Cryptographic constant-time (CT) is a popular programming discipline used by cryptographic libraries to protect themselves against timing attacks. The CT discipline aims to enforce that program execution does not leak secrets, where leakage is defined by a formal leakage model. Constant-timeness can be checked automatically by many tools. However, most sound tools are focused on a baseline (BL) leakage model where memory addresses and conditional branches leak but other basic operations do not leak. In other models, operation like division can leak the value of its arguments or only the cache line of a memory address can leak. We have developed a verification infrastructure, which proves that source programs are constant-time, and a compiler infrastructure, which provably preserves constant-timeness for these fine-grained policies. By making these infrastructures parametric in the leakage model, we achieve the first approach that supports fine-grained constant-time policies. We implemented the approach in the Jasmin tool, and we evaluated our approach with examples from the literature: OpenSSL and wolfSSL. We found a bug in OpenSSL and provided a formally verified fix. This work has been published in CCS 2022 [8].

7.5 Fast equality tests with coq-elpi

Participants: Benjamin Grégoire, Jean-Christophe Léchenet, Enrico Tassi.

We studied how to use coq-elpi to implement fast equality tests for inductive datatypes in Coq. The complete work, including benchmarks to verify asymptotic behavior with respect to numbers of constructors, is ready for publication and will appear in 2023 [11].

7.6 Formal study of Double-word arithmetic algorithms

Participants: Laurence Rideau, Jean-Michel Muller *(CNRS, ENS de Lyon)*, Joris Picot *(ENS Lyon)*, Nicolas Louvet *(ENS Lyon)*, Vincent lefèvre.

The article describing the work on the formalization of "basic building blocks of double-word arithmetics" has been published in *Transactions on Mathematical Software* [5].

Our collaboration continues on the formalization of algorithms for Euclidean norms. This year we formalized the algorithm presented in the article in the case without overflow or underflow. This algorithm uses a result of Rump and Lange on iterated sums of floating-point numbers, which we also formalized. The article describing this work has just been published (Transactions on Mathematical Software) [4].

7.7 Formal study of Fast Fourier Transforms

Participants: Laurence Rideau, Laurent Théry.

Currently, within the ANR project Nuscap, we have undertaken a work of formalization of the FFT.

As a first milestone, we formalized the correctness proof of the classic algorithm that works on an arbitrary integral ring [12].

We also formalized algorithms for the multiplication of complex numbers, as well as the proofs of different algorithms of an article of Kahan concerning the efficient multiplication/sum of four floating point numbers "a*b+c*d". This is useful for the multiplication of complex numbers.

7.8 A generic library for injective, surjective, and bijective functions

Participants: Reynald Affeldt, Cyril Cohen.

In order to support function handling in the development of Lebesgue integral, we developed a library of injective/surjective/bijective functions from a subset to another, with automated inference of compositions, and overloading of the function inverse symbol and its theory.

7.9 Simple automatic positivity

Participants: Cyril Cohen, Pierre Roux *(ONERA)*.

We formalized a dedicated type for positive elements of a numeric domain (like the rationals, real numbers or complex numbers), but also for the positive extended reals. We designed it with enough generality so as to encompass positive, negative, non-positive, non-negative and non-zero elements, with fast automated inference of positivity in many cases. This makes it possible to discharge automatically many trivial proofs of positivity.

7.10 Lebesgue measure and Lebesgue integral for Mathematical Components

Participants: Reynald Affeldt, Cyril Cohen.

After we provided a construction of the Lebesgue measure, we continued with Lebesgue integration. We proved the monotone convergence theorem, the dominated convergence theorem, and Fubini's theorem. This is part of MathComp analysis.

7.11 Semantics of Probabilistic Programs using s-Finite Kernels in Coq

Participants: Reynald Affeldt *(AIST, Japan)*, Cyril Cohen, Ayumu Saito *(Tokyo Institute of Technology, Japan)*.

We extend an existing formalization of measure and integration theory with s-finite kernels, a mathematical structure to interpret typing judgments in the semantics of a probabilistic programming language. The resulting library makes it possible to reason formally about transformations of probabilistic programs and their execution. This is published in a paper to appear in early 2023 [7].

7.12 A stratified variant of univalent parametricity

Participants: Cyril Cohen, Assia Mahboubi *(Gallinette)*, Enzo Crance *(Misubishi Electric, France)*.

In this ongoing work, we rephrase and restructure the results from "The Marriage of Univalence and Parametricity", in order to retain translations even without univalence.

7.13 A new design pattern for the formalization of subsets in mathematics

Participant: Cyril Cohen.

We explore an alternative to the packaging of signatures and sets, based on a mixture of class based inference and structure based inference. The result of this experiment is available as teasing material for the LiberAbaci Inria challenge.

7.14 Formalizing network sorting algorithms

Participants: Benjamin Grégoire, Laurent Théry.

We got interested in the djbsort library, a library for sorting numbers in a cryptographic context. As an initial step, we formalized the proof of the main network sorting algorithms : bitonic, odd-even merge, odd-even exchange [13].

7.15 Toward a type class engine for Coq written in Elpi

Participants: Matteo Manighetti *(University of Bologna)*, Claudio Sacerdoti Coen *(University of Bologna)*, Enrico Tassi.

The mechanism of type classes is a typical addition to dependent type theory that makes the overloading of notations and the re-use of generic theorems elegant. In this work, we explore how such a system can be implemented using the ELPI language. With such a high-level language description we hope to clarify the key points of the mechanism and ease its fine-tuning and improvements.

Two prototypes have been developed: one for Coq and one for Lambdapi (the basis of Dedukti).

7.16 Hierarchy Builder

Participants: Cyril Cohen, Enzo Crance *(Mitsubishi Electric)*, Enrico Tassi.

Hierarchy builder is still being improved, aiming for complete integration in released version of mathematical components and math-comp-analysis.

7.17 Jasmin development

Participants: Benjamin Grégoire, Vincent Laporte, Jean-Christophe Léchenet, Santiago Arranz Olmos *(MPI-SP, Germany)*.

We added a new feature to the Jasmin compiler: clearing the stack when exiting functions. The proof of correctness for this feature is still ongoing work. We finished the work on generalizing the compiler so that it will be able to generate code for different architectures. We started implementing an ARM code producer, but it is not completely functional yet. Three releases of Jasmin were published in 2022.

7.18 CryptoVerif to EasyCrypt

Participants: Bruno Blanchet, Pierre Boutry, Christian Doczkal, Benjamin Grégoire, Pierre-Yves Strub *(Meta)*.

We continue our study of approaches to combine two mechanized tools to verify protocols.

We developed a translation from CryptoVerif to EasyCrypt that allows cryptographic assumptions that cannot be proved in CryptoVerif to be translated to EasyCrypt and proved there. We used the translation to prove different hypotheses assumed in CryptoVerif:

- The reduction of the N query formulation of the Computational/Gap Diffie-Hellman (CDH/GDH) games in CryptoVerif to the standard, single-query formulation. The obtained bounds are better than what can be obtained by a direct hybrid argument.
- The reduction from the N participant games (e.g. insider or outsider adversaries) for authenticated Key encapsulation mechanisms (KEM) to 1 or 2 participant games (almost done).

7.19 Models of nominal groups

Participant: Pierre Boutry.

For the formalization of the proof of the reduction linked to the Computational/Gap Diffie-Hellman (CDH/GDH) games we relied on a set of properties, denoted as nominal groups by Bruno Blanchet, common to prime-order groups, Curve25519 and Curve448. We worked on verifying that Curve25519 and Curve448 are actually models of nominal groups using Easycrypt as a proof-checker and we spotted two problems in the formalization of these properties as used in 7.18. The first one was a mistake on our part. The second one was a property that turned out to be too strong and was not holding for Curve25519 and Curve448. However, the proof from 7.18 did not need to be modified since the restrictions that we needed to add to fix these problems were already met.

In the process we found another needed correction. There are several definitions of the concept of statistical distance between distributions. The pen-and-paper proof was using them as if they were equivalent. Though they are indeed related, they are not equivalent. Beside allowing to fix three problems in

the proof of these reductions, showing yet again the benefit of a complete formal proof, the formalization of properties of Curve25519 and Curve448 will make it possible to increase the level of confidence in some of the proofs in Libjade. These curves are commonly used in reference cryptographic algorithms but until now there was no Easycrypt proofs of their properties which then needed to be axiomatized. This work will be completed shortly after the completion of 7.20, making it easier to justify the minor remaining gaps.

7.20 Towards a library of field extensions

Participants: Pierre Boutry, Antoine Séré *(École Polytechnique)*, Pierre-Yves Strub *(Meta)*.

As an effort to complete Easycrypt with proofs of properties that should be needed for future developments, we started working on a library of field extensions. Most of the remaining work is about adding missing lemmas about permutation and polynomial Euclidean division, the latter being the main focus locally. We are done with the properties of the division when the considered polynomials are defined over rings, while the case of fields still needs to be completed. We take inspiration from the development about polynomial Euclidean division done in the Mathcomp library for the choice of the statements of the lemmas. However, the definition of the operations differ. In the Mathcomp library, the operators that capture these definitions are defined recursively, while in Easycrypt recursion is not allowed. So we use iterators to do so. since the operations do not have the same definitions, the Mathcomp proof is mostly just a source of inspiration as one can obviously not prove that the same property holds for two distinct operations in a unique way. Some of the lemmas about polynomial Euclidean division will also be used to conclude the proof of 7.19.

7.21 Easycrypt library

Participants: Pierre Boutry, Benjamin Grégoire.

In the library there were two different definitions of cyclic groups that were not taking advantage of existing material. So we gave a third definition that supersedes the previous ones and modified the rest of the library to make sure the new one is used everywhere. This made it possible to detect maintenance issues with the library, especially with respect to continuous integration.

7.22 Collisions between Bezier Curves and straight line segments

Participants: Yves Bertot, Quentin Vermande *(ENS Paris)*, Reynald Affeldt *(AIST, Japan)*.

We are working on a formal description of Bezier Curves and the formal verification that curves of that kind do not collide with obstacles given by straight line segments.

7.23 Document management for the Coq system

Participants: Enrico Tassi, Maxime Dénès.

We have been redesigning the communication protocol between Coq and its user-interface software to make it compliant with the LSP protocol used in Visual Studio Code. We are now exploring the use of event-based programming for this integrated development environment.

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

Participants: Benjamin Grégoire, Swarn Priya, Yves Bertot.

The STAMP team participates with the Grace team (Inria Saclay) in the JASMIN contract funded in the framework of the Inria-Nomadic Labs collaboration for research related to the Tezos blockchain. This contract funds the PhD thesis of Swarn Priya.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Inria associate team not involved in an IIL or an international program

FLAVOR

Participants: Yves Bertot, Cyril Cohen, Laurence Rideau, Enrico Tassi, Laurent Théry.

Title: Formal Library of Analysis for the Verification of Robots

Duration: 2020 ->

Coordinator: Reynald Affeldt (reynald.affeldt@aist.go.jp)

Partners:

- National Institute of Advanced Industrial Science and Technology Tokyo (Japon)

Inria contact: Yves Bertot

Summary: The objective is to apply formal methods based on Coq to software and designs that are concerned with robots. Covered topics concern mathematical formalization for real analysis, control theory, kinematic chains, and motion planning.

9.2 International research visitors

9.2.1 Visits of international scientists

Reynald Affeldt

Status: researcher

Institution of origin: AIST

Country: Japan

Dates: June 27-July 8, 2022 and December 5-12, 2022

Context of the visit: Equipe associée FLAVOR

Takafumi Saikawa

Status: post-doctoral researcher

Institution of origin: Nagoya University

Country: Japan

Dates: December 5-12, 2022

Context of the visit: Equipe associée FLAVOR

Reynald Affeldt (AIST, Japan) visited the STAMP team from June 27 to July 8, to discuss the development of the Math-comp-analysis library and the project to verify algorithms computing trajectories.

Reynald Affeldt (AIST, Japan), Takafumi Saikawa (Nagoya University, Japan), and Kazuhiko Sakaguchi (University Tsukuba, Japan) visited the STAMP team from December 5 to December 12 to participate in the Mathematical Components winter school that was organized in Sophia Antipolis and discuss research projects around the formal verification of probability theory.

Ralf Hulsing

Status: Professor

Institution of origin: Eindhoven University

Country: the Netherlands

Dates: September-December2022

Context of the visit: no information

Ralf Hulsing from University Eindhoven visited the team from September to December to collaborate with Benjamin Grégoire on post-quantum cryptography.

9.3 National initiatives

9.3.1 ANR

- 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.
- 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.
- NuSCAP "Numerical Safety for Computer-Aided Proofs", started on February 1st, 2021 for 48 months, with a grant covering traveling costs. Other partners are CNRS-LIP, Sorbonne University LIP6, and CNRS-LAAS. The corresponding researcher for this contract is Laurence Rideau.

9.3.2 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 partner (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.

9.3.3 PEPR

- SVP PEPR Cybersecurity. We participate in a project concerned with the verification of security protocols. Partners in this project are CNRS IRISA Rennes (coordinator Stéphanie Delaune), Inria, University of Paris-Saclay, University of Lorraine, University of Côte d'Azur, ENS Rennes. The funds allocated to our team in this collaboration are 333 kEuros. The corresponding researcher for this contract is Benjamin Grégoire.

9.3.4 Inria Challenges

- Liber Abaci. Yves Bertot coordinated the creation of Inria challenge Liber Abaci on the use of a Type-theory based proof assistant to improve mathematics education for the first years of higher education (undergraduate mathematics).

10 Dissemination

Participants: Yves Bertot, Pierre Boutry, Cyril Cohen, Benjamin Grégoire, Laurence Rideau, Enrico Tassi, Laurent Théry.

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

Participants: Cyril Cohen, Enrico Tassi.

Member of the organizing committees Enrico Tassi organized a winter school on formalization using the Mathematical Components Library from December 5 to December 9, 2022. Yves Bertot, Cyril Cohen, Laurence Rideau, and Laurent Théry were also speakers at this school.

Assia Mahboubi and Enrico Tassi organized a workshop on the Mathematical Components Library on December 7.

Cyril Cohen was workshop chair for the conference ITP 2022 (Interactive Theorem Proving).

10.1.2 Scientific events: selection

Participants: Yves Bertot, cyril Cohen, Enrico Tassi, Laurent Théry.

Member of the conference program committees Enrico Tassi was a member of the program committee for the workshop F-IDE (Formal Integrated Development Environment).

Reviewer Yves Bertot and Cyril Cohen reviewed papers for ITP 2022 (Interactive Theorem Provers). CSL (Computer Science Logic) CICM (Conference on intelligent computer mathematics), CoqPL (Coq for programming Languages).

10.1.3 Journal

Participants: Yves Bertot, Pierre Boutry, Cyril Cohen, Laurent Théry.

Reviewer - reviewing activities Yves Bertot, Cyril Cohen, Pierre Boutry, and Laurent Théry reviewed articles for JAR (Journal of Automated Reasoning), MSCS (Mathematical Structures in Computer Science), and LMCS (Logical Methods in Computer Science).

10.1.4 Invited talks

Benjamin Grégoire gave an invited talk on EasyCrypt at "Journées Codage & Cryptographie" in April 2022.

10.1.5 Leadership within the scientific community

Yves Bertot is a member of the steering committee for the conference "Interactive Theorem Provers" (ITP).

10.1.6 Research administration

Yves Bertot is coordinator of the Inria Challenge "Liber Abaci" on formal proofs for first years of higher mathematical higher education.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Master : Yves Bertot, “Proofs and reliable programming using Coq”, 12hours ETD, Nov-Dec 2022, Master Informatique et Interactions, Université Côte d’Azur, France.
- Benjamin Grégoire was a speaker at the Summer School on Cybersecurity in Nancy, July 4-8, 2022

10.2.2 Supervision

Yves Bertot and Benjamin Grégoire supervise the thesis of Swarn Priya (Université Côte d’Azur).
Cyril Cohen spervised the thesis of Chris Hughes until September 2022.

10.2.3 Juries

Yves Bertot was member of the Jury for the thesis of Lucien Rakotomalala (Université de Toulouse).

10.3 Popularization

10.3.1 Internal or external Inria responsibilities

Laurence Rideau is a member of the editorial board of Interstices, the Inria medium for outreach.

10.3.2 Interventions

Yves Bertot and Maxime Dénès gave outreach talks at the OSXP conference (Open-Source Experience) in Paris in November 2022.

11 Scientific production

11.1 Major publications

[1] R. Affeldt, C. Cohen, M. Kerjean, A. Mahboubi, D. Rouhling and K. Sakaguchi. ‘Competing inheritance paths in dependent type theory: a case study in functional analysis’. In: IJCAR 2020 - International Joint Conference on Automated Reasoning. Paris, France, 29th June 2020, pp. 1–19. URL: https://hal.inria.fr/hal-02463336.

[2] R. Affeldt, C. Cohen and D. Rouhling. 'Formalization Techniques for Asymptotic Reasoning in Classical Analysis'. In: *Journal of Formalized Reasoning* (Oct. 2018). URL: https://hal.inria.fr/hal-01719918.

[3] J. B. Almeida, M. Barbosa, G. Barthe, A. Blot, B. Grégoire, V. Laporte, T. Oliveira, H. Pacheco, B. Schmidt and P.-Y. Strub. 'Jasmin: High-Assurance and High-Speed Cryptography'. In: *CCS 2017 - Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. Dallas, United States, Oct. 2017, pp. 1–17. URL: https://hal.archives-ouvertes.fr/hal-01649140.

11.2 Publications of the year

International journals

[4] V. Lefèvre, N. Louvet, J.-M. Muller, J. Picot and L. Rideau. 'Accurate calculation of Euclidean Norms using Double-word arithmetic'. In: *ACM Transactions on Mathematical Software* (25th Oct. 2022). URL: https://hal.science/hal-03482567.

[5] J.-M. Muller and L. Rideau. 'Formalization of double-word arithmetic, and comments on "Tight and rigorous error bounds for basic building blocks of double-word arithmetic"'. In: *ACM Transactions on Mathematical Software* 48.1 (Mar. 2022), pp. 1–24. DOI: 10.1145/3484514. URL: https://hal.science/hal-02972245.

National journals

[6] C. Icubam, L. Bonnasse-Gahot, M. Dénès, G. Dulac-Arnold, S. Girgin, F. Husson, V. Iovene, J. Josse, A. Kimmoun, F. Landes, J.-P. Nadal, R. Primet, F. Quintao, P. G. Raverdy, V. Rouvreau, O. Teboul and R. Yurchak. 'ICU Bed Availability Monitoring and analysis in the Grand Est region of France during the COVID-19 epidemic'. In: *Statistique et Société* (17th Mar. 2022). DOI: 10.1101/2020.05.18.20091264. URL: https://hal.science/hal-02620018.

International peer-reviewed conferences

[7] R. Affeldt, C. Cohen and A. Saito. 'Semantics of Probabilistic Programs using s-Finite Kernels in Coq'. In: CPP 2023 - Certified Programs and Proofs. Boston, United States, 16th Jan. 2023. DOI: 10.1145/3573105.3575691. URL: https://hal.inria.fr/hal-03917948.

[8] B. Ammanaghatta Shivakumar, G. Barthe, B. Grégoire, V. Laporte and S. Priya. 'Enforcing Fine-grained Constant-time Policies'. In: CCS '22: 2022 ACM SIGSAC Conference on Computer and Communications Security. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22). Los Angeles CA, United States: ACM, 7th Nov. 2022, pp. 83–96. DOI: 10.1145/3548606.3560689. URL: https://hal.inria.fr/hal-03844366.

[9] G. Barthe, A. Koutsos, S. Mirliaz, D. Pichardie and P. Schwabe. 'Semantic foundations for cost analysis of pipeline-optimized programs'. In: Static Analysis - 29th International Symposium. Static Analysis - 29th International Symposium, SAS 2022, Auckland, New Zealand, December 5-7, 2022, Proceedings. Auckland, New Zealand, 8th Dec. 2022. URL: https://hal.inria.fr/hal-03779257.

Conferences without proceedings

[10] K. Palmskog, E. Tassi and T. Zimmermann. 'Reliably Reproducing Machine-Checked Proofs with the Coq Platform'. In: RRRR 2022 - Workshop on Reproducibility and Replication of Research Results. Munich, Germany, 2nd Apr. 2022. URL: https://hal.inria.fr/hal-03592675.

Reports & preprints

[11] B. Grégoire, J.-C. Léchenet and E. Tassi. *Practical and sound equality tests, automaticallyDeriving eqType instances for Jasmin's data types with Coq-Elpi*. 6th Oct. 2022. DOI: 10.1145/nnnnnnn.nnnnnnn. URL: https://hal.inria.fr/hal-03800154.

[12] L. Théry. *A Formalisation of a Fast Fourier Transform*. 13th Oct. 2022. URL: https://hal.inria.fr/hal-03807965.

[13] L. Théry. *A Formalisation of Algorithms for Sorting Network*. 2nd Mar. 2022. URL: https://hal.inria.fr/hal-03585618.

[14] L. Théry. *Primality Tests and Prime Certificate*. 8th Mar. 2022. URL: https://hal.inria.fr/hal-03601611.