RESEARCH CENTRE

Inria Centre at Rennes University

IN PARTNERSHIP WITH:

Université de Rennes

2024

ACTIVITY REPORT

Project-Team

EPICURE

Semantic analysis and compilation for secure execution environments

IN COLLABORATION WITH: Institut de recherche en informatique et systèmes aléatoires (IRISA)

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Inria

Contents

Project-Team EPICURE

Creation of the Project-Team: 2022 June 01

Keywords

Computer sciences and digital sciences

A2.1. – Programming Languages

A2.2. – Compilation

A2.2.1. – Static analysis

A2.2.5. – Run-time systems

A2.2.9. – Security by compilation

A2.4. – Formal method for verification, reliability, certification

A2.4.1. – Analysis

A2.4.3. – Proofs

A4.4. – Security of equipment and software

A4.5. – Formal methods for security

Other research topics and application domains

B6.1.1. – Software engineering

B6.4. – Internet of things

B6.6. – Embedded systems

1 Team members, visitors, external collaborators

Research Scientists

- Thomas Jensen [Team leader, INRIA, Senior Researcher]
- Frédéric Besson [INRIA, Researcher]
- Simon Castellan [INRIA, Researcher]
- Benoit Montagu [INRIA, Researcher]
- Alan Schmitt [INRIA, Senior Researcher]

Faculty Members

- Sandrine Blazy [UNIVERSITE DE RENNES, Professor]
- Delphine Demange [UNIVERSITE DE RENNES, Associate Professor]
- Benjamin Farinier [UNIVERSITE DE RENNES, Associate Professor]
- Thomas Genet [UNIVERSITE DE RENNES, Professor]

PhD Students

- Santiago Bautista [ENS Rennes, ATER, until Aug 2024]
- Sebastien Bonduelle [INRIA, from Sep 2024]
- Clement Chavanon [INRIA]
- Alexandre Drewery [INRIA]
- Jean-Loup Hatchikian-Houdot [INRIA]
- Romeo La Spina [UNIVERSITE DE RENNES]
- Tony Law [UNIVERSITE DE RENNES]
- Théo Losekoot [INRIA, from Sep 2024 until Oct 2024]
- Théo Losekoot [UNIVERSITE DE RENNES, until Aug 2024]
- Malo Revel [UNIVERSITE DE RENNES]

Technical Staff

- Aurore Alcolei [INRIA, Engineer, until Aug 2024]
- Pierre Lermusiaux [INRIA, Engineer]
- Victoire Noizet [INRIA, Engineer, until Jun 2024]

Interns and Apprentices

- Sebastien Bonduelle [ENS RENNES, Intern, from Feb 2024 until Jun 2024]
- Yu-Hui Chiang [INRIA, Intern, from May 2024 until Jul 2024]
- Lysa Dahmani [INRIA, Intern, from Apr 2024 until Jul 2024]
- Tom Goalard [CNRS, Intern, from May 2024 until Jul 2024]
- Rayane Jelidi–Daniel [INRIA, Intern, from May 2024 until Jul 2024]
- Sibylle Jullien [INRIA, Intern, from Jun 2024 until Jul 2024]
- Charlotte Thomas [INRIA, Intern, from May 2024 until Jul 2024]

Administrative Assistant

- Lydie Mabil [INRIA]

2 Overall objectives

The security of the software that surrounds us is, more than ever, a scientific challenge of utmost societal importance. More and more software is produced to operate on an increasingly varied number of devices and to provide increasingly complex functionality. There is a pressing need to provide the science and technology for engineering software so that it becomes safe and secure, in addition to providing the desired functionality. This need is not new and a multitude of programming languages, semantic theories, formal methods, verification tools and techniques have been developed and contribute to meet this need. One of the challenges with this state of affairs is exactly the multitude of languages in which to express the algorithms that we develop, and in particular the distance between those languages for which it is comparatively easy to develop correct and secure software, and those that actually get executed in our computers, telephones, pacemakers, cars, smart home IoT devices *etc.*.

No one single silver bullet will solve the problem of developing secure software worthy of the user's trust. We are however convinced that a cornerstone of the answer is *programming language semantics*, *i.e.*, a mathematically robust yet flexible formalism for defining the behaviour of a program. The goal of the EPICURE project is to contribute with semantics-based methods for producing safe and secure software by

- defining new semantic frameworks that will provide more accurate models of modern execution platforms, and which can facilitate the semantic definition of the above-mentioned multitude of programming languages,
- designing formally verified analysis and compilation schemes, with the specific aim of being able to analyse and verify properties of programs written in high-level languages, and to compile both program and the verified properties down to low-level executable representations,
- demonstrate the impact of language-based tools on software security by showing how they can improve the correctness, safety and security of critical software found in modern execution environments, such as the Java virtual machine, the Tezos blockchain written in OCaml, and small operating systems for the IoT such as RIOT.

3 Research program

The overall goal of the EPICURE project is to guarantee the security and safety of key software components of execution platforms, including those used in the IoT and blockchains. Our contribution to this goal will be to develop semantics-based, formally verifiable program analyses and compilation techniques for improving and enforcing software security and safety. The main open challenges in the field include:

- providing mechanised formalisations of modern programming languages (such as Rust, JavaScript, Web Assembly) which facilitate the reasoning about these languages and their tools,
- faithfully modeling architectures on which they execute, taking into account features such as out-of-order execution and trust-enhancing mechanisms such as enclaves and trust zones,
- designing program processing tools such as analyses and compilers, the correctness of which can be verified mechanically,
- developing scalable analyses for proving security properties of high-level programs, and compiling programs and their proofs down to low-level executables, the security of which is guaranteed by the compilation process.

The EPICURE project is structured into the following research axes:

- Semantics and their mechanisation.
- Program analysis.
- Trustworthy compilation.
- Secure execution platforms.

The axis on semantics and their mechanisation will investigate frameworks for defining semantics, in particular the recently proposed *skeletal* semantics and the notion of causal semantics. We will pay particular attention to the semantics of intermediate representations used in compilers and to the semantic description of low-level languages, *e.g.* eBPF. In the axis on program analysis, we plan to conduct work both on the foundations of static analysis and abstract interpretation and on the development of specific analyses, in particular for higher-order polymorphic functional programs. A special attention will be given to the problem of translating results of an analysis from a high-level language to its compiled (low-level) version. In the strand on trustworthy compilation we will pursue the effort on mechanised verification of optimising compilers. We will also examine the security impact of compilation with respect to different (passive and active) attacker models. The intended application areas for these techniques are the Internet of Things and high-assurance block chains.

4 Application domains

The intended application of the scientific results outlined in the previous sections is to improve the safety and security of execution platforms, taken in a broad sense ranging from virtual machines to hardware processors. We will improve on analyses and compilation techniques for verifying and producing safer code, as we will improve on the key software tools and components that implement the execution platform. In this section we outline a number of more concrete applications that we intend to investigate.

4.1 Internet of Things

The Internet of Things offers a large and diverse domain of application for our formal methods. The limitations of the devices populating the IoT mean that a different kind of algorithms are deployed but the security and privacy concerns remain, and are even accentuated by the relative weak protection mechanisms offered by the underlying hardware. In particular, the IoT relies on cryptographic primitives for secure communication and software updates but these primitives are often different from what is used on standard execution platforms due to the limited computing resources. The question of secure compilation and the techniques that we expect to develop can be transferred to the IoT but the security properties might be harder to verify because of optimisations.

On the application level, the distributed and asynchronous nature of the IoT has led to new programming paradigms and novel uses of existing languages (such as JavaScript) that pose new verification challenges, in particular the verification of coordinating programs written in different complex languages in a multitier framework. A multitier language unifies within a single formalism and a single execution environment the programming of the different tiers of distributed applications. On the web, this

paradigm unifies the client tier, the server tier, and the database tier. We thus want to investigate how our techniques can be brought to bear on multitier programming languages. In particular, we propose to investigate the design of program analyses for a multitier language for the IoT.

4.2 High-assurance blockchains

Because they enable the distributed management of virtual assets—such as property rights, proofs of payments— blockchain systems play a growing, *critical* role in our societies. Blockchain-based systems, like Ethereum or Tezos, are equipped with so-called *contracts*. A contract is a program which is executed by a *virtual machine* (VM) over the blockchain. The effect of a contract is to update values and assets stored in the blockchain. Thus, any failure in the safety, availability, or security in the VM of a system like Tezos could have dramatic consequences on industries, on public infrastructures, and eventually on people. The pieces of code that lie at the foundations of the Tezos system are entrusted with the safety and security of all the managed assets. The Tezos core software is thus expected to attain the highest levels of clarity and quality, and to get as close as possible to zero defects. This is where formal methods—and in particular static analyses—can help, by giving guarantees about the dynamic behaviour of programs, in an automatic way. The expressive type system of OCaml—the implementation language of Tezos—already provides static safety guarantees by ensuring data is used in a consistent way. In collaboration with Nomadic Labs, we will provide OCaml programs with additional guarantees, by answering questions such as *"can a program raise an exception?"*, *"can a program break some user-defined invariant?"*, or *"which data might be modified by a program?"*. Those questions are beyond the scope of the OCaml type system, but are within reach of abstract interpretation-based static analyses. The endeavour of supporting all the features of OCaml is beyond the scope of this project. Instead, we will target a representative subset of the pure fragment of the OCaml language, in which the core of Tezos's VM is written.

5 Social and environmental responsibility

EPICURE runs the INRIA exploratory action "Back to the trees" which aims to use probabilistic programming and Bayesian inference to produce a plant identification tool that is reliable, educational and convivial, built together with botanist collectives.

6 New software, platforms, open data

6.1 New software

6.1.1 necro

Name: necro

Keywords: Semantics, Programming language, Specification language

Functional Description: The goal of the project is to provide a tool to manipulate skeletal semantics, a format to represent the semantics of programming languages. This tool has been mostly developed by Victoire Noizet.

URL: http://skeletons.inria.fr/necro.html

Publication: tel-03855276v1

Contact: Alan Schmitt

Participant: Alan Schmitt

6.1.2 Timbuk

Keywords: Automated deduction, Ocaml, Program verification, Tree Automata, Term Rewriting Systems

Functional Description: Timbuk is a tool designed to compute or over-approximate sets of terms reachable by a given term rewriting system. The library also provides an OCaml top-level with all usual functions on Bottom-up Nondeterministic Tree Automata.

URL: http://people.irisa.fr/Thomas.Genet/timbuk/index.html

Contact: Thomas Genet

Participant: Thomas Genet

6.1.3 dmap

Name: dependent maps library in OCaml

Keywords: Ocaml, Library, Data structures

Functional Description: dmap is an OCaml library that implements immutable maps, for which the type of data may depend on the key they are associated with.

URL: https://gitlab.inria.fr/bmontagu/dmap

Contact: Benoit Montagu

Participant: Benoit Montagu

6.1.4 sexp_decode

Keywords: Ocaml, Library

Functional Description: sexp_decode is an OCaml library of monadic combinators for decoding S-expressions (as defined in the Csexp library) into structured data.

URL: https://gitlab.inria.fr/bmontagu/sexp_decode

Contact: Benoit Montagu

Participant: Benoit Montagu

6.1.5 CompcertSSA

Keywords: Optimizing compiler, Formal methods, Proof assistant, SSA

Functional Description: CompcertSSA is built on top of the Compcert verified C compiler, by adding a middle-end based on the SSA form (Static Single Assignment) : conversion to SSA, SSA-based optimizations, and destruction of SSA.

URL: https://compcertssa.gitlabpages.inria.fr/

Publications: hal-01378393, hal-01193281, hal-02904204, hal-03899435, hal-01110783, hal-01097677, hal-01110779

Contact: Delphine Demange

Participants: Sandrine Blazy, Delphine Demange, Yon Fernandez De Retana, David Pichardie, Leo Stefanesco

6.1.6 plantinator

Name: plantinator

Keywords: Data management, Algebraic Data Types, Decision

Functional Description: Plantinator is a database management software for morphological data about plants as well as automatic identification key generator

URL: https://botascopia.inria.fr

Contact: Simon Castellan

6.1.7 Salto Static Analyser

Keywords: Static analysis, Ocaml, Abstract interpretation

Scientific Description: Static analyser for OCaml programs, that supports recursive algebraic data types (including GADT and non regular types), first-class functions, first-class exceptions, dynamic exceptions, first-class modules, mutable data types (mutable records, arrays), and base types such as integers, floating point numbers, characters, and strings.

The analyser infers for every program point an abstract value that represents an over-approximation of the set of values that this program point can compute, and of the exceptions that can be raised.

Functional Description: Detection of uncaught exceptions, possible exit codes, undefined behaviours in OCaml programs. This static analyser is based on the theory of abstract interpretation.

News of the Year: Extension of the scope of features supported by the analyser, and support for a large part of the OCaml standard library. Analysis of whole projects that are built using the dune build system.

URL: https://salto.gitlabpages.inria.fr/

Publications: hal-04547480, hal-04410771, hal-04769799

Contact: Benoit Montagu

Participants: Benoit Montagu, Pierre Lermusiaux, Thomas Jensen, Thomas Genet

7 New results

7.1 Skeletal Semantics

Participants: Martin Andrieux, , Thomas Jensen, , Victoire Noizet, , Vincent Rébiscoul, , Alan Schmitt.

The work on skeletal semantics [41], a modular and formal way to describe semantics or programming languages, has continued during 2024. Links to papers and tools can be found at the dedicated website.

Victoire Noizet continued her work on the development of Skel, the skeletal semantics language, and on the development of Necro, a tool to manipulate skeletal semantics. Vincent Rébiscoul has continued working on static analyses for skeletal semantics. He is designing a framework that can automatically derive a control-flow analysis from the definition of a language as a skeletal semantics. The goal of the approach is to automatically derive the correctness of the analysis from the correctness of its components. Vincent defended his PhD Thesis in May 2024 [28].

Martin Andrieux (an M1 student) did a research project on a skeletal semantics of Python, based on the formal semantics written by Raphaël Monat [39]. His work was presented at the French conference JFLA 2024 [26].

7.2 Static Analysis of Functional Programs

Participants: Thomas Genet, Thomas Jensen, Pierre Lermusiaux, Benoit Montagu, Tom Goalard.

The Salto project aims at developing a static analyser for OCaml programs based on abstract interpretation. A primary goal is to detect possibly uncaught exceptions in OCaml programs.

In 2024, the scope of the Salto prototype analyser was extended, to the point where it is able to take real OCaml programs as input, that call the OCaml standard library. Large subsets of the OCaml language, of its base types, and of its standard library are now supported. A lot of effort was spent to properly support the alias modules feature. The prototype analyser can automatically discover the structure and dependencies of an OCaml project based on the `dune` build system, and analyse it as a whole. More advanced features, such as objects, recursive modules, and liberal definitions of recursive values remain to be supported.

The internship of Tom Goalard (ENS Rennes, L3), permitted to test, find bugs and implement fixes in the abstract domains used in Salto. The technique of property-based testing was used, relying on the QCheck library implemented in OCaml.

A research article [23] was accepted for publication and presented at ESOP 2024, that describes the theory underlying the Salto static analyser, and that presents some experimental results.

A larger audience article about the Salto project [36] was published in the ERCIM News journal.

7.3 Relational domains for algebraic data types and arrays

Participants: Santiago Bautista, Thomas Jensen, Benoit Montagu.

As a follow-up of his Ph.D. work, Santiago Bautista designed an abstract domain that can express relations between values that are built from scalar values, (non-recursive) algebraic data types and functional arrays [12]. This abstract domain can express *function summaries* for first-order functional programs that manipulate integers, algebraic data types and functional arrays, and can serve as a basis for the modular static analysis of such programs.

7.4 Verification of Functional Programs using Tree Automata and Shallow Horn Clauses

Participants: Théo Losekoot, Thomas Genet, Thomas Jensen.

We develop a specific theory and the related tools for analyzing functional programs manipulating algebraic data types. The domain and the co-domain of such functions are (generally) infinite set of terms. We use tree automata to finitely represent such infinite sets of terms. We have already shown how to exploit those informations using a *dedicated type system* associating regular language types to variables, expressions, etc. of a program. By automatically inferring such types we perform fully automatic verification of safety properties of tree-processing higher-order functional programs. Experiments are detailed here. Such regular abstractions are powerful but cannot represent relations between the input and the output of a function.

In [38], we used *convoluted tree automata* to finitely approximate the infinite input-output relation of first-order functions manipulating algebraic data types. This year, we designed a new formalism to represent relations: Shallow Horn Clauses (SHoCs). Shallow Horn clauses are a restriction of Horn clauses and can represent all the relations recognized by convoluted tree automata. They can also represent relations that are out of reach of convoluted tree automata. Interestingly, in spite of their improved expressivity, SHoCs are still closed by boolean operations and provide a more compact representation

of relations [34]. Using SHoCs makes the verification of relational properties more efficicient . This has published and presented at the SAS'24 conference [24].

7.5 Machine checked proof of an rBPF virtual machine

Participants: Frédéric Besson, Shenghao Yuan, Jean-Pierre Talpin.

The rBPF virtual machine adapts the eBPF (extended Berkeley Packet Filters) technology to resource constrained devices running the RIOT micro-kernel. Typically, eBPF programs are untrusted user-provided programs that are used to monitor the kernel behaviour.

As the VM runs with kernel privileges on micro-controllers which rarely feature hardware memory protection, isolation is an essential property that is needed to ensure system integrity against potentially malicious programs.

In previous works, we have shown how to derive, within the Coq proof assistant, the verified C implementation of an eBPF virtual machine from a Gallina specification [42]. We have augmented the virtual machine with a verified JIT compiler for straighline code [25]. One challenge is to augment the CompCert semantics so that the C code may call binary code that is dynamically generated in memory while still abiding to the calling conventions. The JIT compiler substantially improves the performance for arithmetic intensive benchmarks.

7.6 Hardware Support for Cryptographic Constant-time Programming

Participants: Frédéric Besson, Jean-Loup Hatchikian Houdot, Pierre Wilke, Guillaume Hiet.

Cryptographic constant-time is a programming discipline for protecting against timing attacks. This discipline forbids branching or performing memory accesses depending on secrets. Protecting memory accesses using software only countermeasures is error-prone and costly. We have proposed a new cache locking mechanism which ensures that locked addresses can be accessed in constant-time. To avoid cache misses, locked address cannot be evicted. To avoid any leakage of information, the meta-data of the cache (dirty-bit, LRU tag) are also protected. We have a formal proof that our cache locking is secure even in the presence of an attacker able to run arbitrary code at any moment and able to observe every memory access. Our benchmarks show that our cache locking enable constant-time code running with very low overheard [19].

7.7 Verified Compilation

Participants: Sandrine Blazy, Delphine Demange, Tony Law, Roméo La Spina.

In 2024, we continued our work on verified compilation using the Coq proof assistant, focusing on formalizing dataflow solvers, new intermediate representations for dataflow circuits and a domain specific language for packet filtering.

We formalized specific dataflow solvers inspired by the work of Bourdoncle, in which an iteration order is pre-computed, based on the structure of the control-flow graph of programs. Central to the proof of correctness is the general notion of a Weak Topological Ordering. Our correctness proofs are valid for any such ordering. The first solver implements an iterative strategy over the ordering, the second solver implements a recursive strategy. Our solvers are extractable to OCaml code. Our formalization is fully compatible with the interface of dataflow solvers within the verified, optimizing C CompCert compiler. We conducted practical experiments on the wide range of forward and backward analyses from

CompCert, demonstrating the practicality of our solvers in terms of efficiency and precision. A research article [20] was accepted for publication at ESOP 2025, that describes our formalization and experiments.

New results on dataflow circuits propose a mechanized formal semantics: rather than following a static schedule predetermined at generation time, the execution of the components in a circuit is constrained solely by the availability of their input data. Circuit components are modeled as abstract computing units, asynchronously connected with each other through unidirectional, unbounded FIFO. We formalize sufficient conditions to achieve the determinacy of circuits executions: all possible schedules of such circuits lead to a unique observable behavior. Moreover, we provide two equivalent views for circuits. The first one is a direct and natural representation as graphs of components. The second is a core, structured term calculus, which enables constructing and reasoning about circuits in a inductive way. We prove that both representations are semantically equivalent. We experimentally validate its relevance by applying our general semantic framework to dataflow circuits generated with Dynamatic, a recent HLS tool exploiting dataflow circuits to generate dynamically scheduled, elastic circuits. A research article [21] was accepted for publication at OOPSLA 2025, that describes our formalization and experiments.

We have developed a CompCert backend for a domain specific language for packet filtering [18]. The compiler is using a BDD-based intermediate representation where the nodes are atomic formulae. The size (and the depth) are reduced by optimising the ordering of the BDD nodes while taking into account infeasible path due to incompatible arithmetic constraints. The experiements show that the optimisations may be costly but significantly improve the code size. They also show that, in term of throughput, the generated code outperforms the network packet filter (nft) and is competitive with the optimised implementation of nftset.

7.8 Static analysis of smart contracts

Participant: Thomas Jensen.

This work [15] concerns the design and implementation from scratch of MichelsonLiSA1, a static analyzer based on abstract interpretation for the verification of smart contracts executing on the Tezos blockchain. It shows how LiSA (Library for Static Analysis) facilitates this task, also for low-level languages such as Michelson.

Once deployed in blockchain, smart contracts become immutable: attackers can exploit bugs and vulnerabilities in their code, that cannot be replaced with a bug-free version. For this reason, the verification of smart contracts before they are deployed in blockchain is important. However, the development of verification tools is not easy, especially if one wants to obtain guarantees by using formal methods. This work describes the development, from scratch, of a static analyzer based on abstract interpretation for the verification of real-world Tezos smart contracts. The analyzer is generic with respect to the property under analysis. This paper shows taint analysis as a concrete instantiation of the analyzer, at different levels of precision, to detect untrusted cross-contract invocation.

Joint work with Univeristy of Venezia, University of Parma and University of Verona.

7.9 An information flow logic based on partial equivalence relations

Participant: Thomas Jensen.

Information flow control (IFC) is a key element for ensuring that programs do not leak confidential data, and a number of enforcement mechanisms (based on static analysis, run-time monitoring, or combinations thereof) have been proposed. The type-based approach to IFC initiated by Volpano *et al.* is traditionally presented in terms of statically dividing program variables (or, more generally, components of data structures, such as record fields) into *high and low security*, together with a program logic for proving *non-interference* However, for many purposes, just classifying data as secret or public is too

coarse to express and prove natural security policies that one would want to impose on code that inspects confidential data.

In [14] we present a relational program logic for reasoning about information flow properties formalised in an assertion language based on partial equivalence relations. We define and prove the soundness of the logic, a proof technique for precise, logic-based information flow properties. The logic extends Hoare logic and its unary state predicates to binary PER-based predicates for relating observationally equivalent states. A salient feature of the logic is that it is capable of reasoning about programs that test on secret data in a secure manner.

Joint work with the Andrzej Filinski and Ken Friis Larsen from University of Copenhagen.

7.10 Static detection of sensibility to the evaluation order

Participant: Benoit Montagu, Thomas Jensen, Sebastien Bonduelle.

Some programming language semantics do not specify a precise evaluation order of sub-expressions. This is the case, for instance, of the C programming language and of OCaml, in which the order of evaluation of the arguments passed to a function can be liberally chosen by the compiler. While this underspecified behaviour gives more freedom to compiler implementors, for example to implement advanced program optimisations, this might be seen as a difficulty for programmers: A program can have, indeed, several semantics. This issue is even more important for the designers of a static analyser for such programming languages: Because a static analyser must over-approximate all the possible executions of a program, the freedom in the order of evaluation either incurrs a cost in the analysis (to browse all the possible evaluation orders a compiler might choose), or asks the designers to choose a specific evaluation order (*e.g.*, by mimicking the choices made by a specific compiler).

The goal of the Master internship of Sébastien Bonduelle was to lay the foundations for a static analysis, that would detect which parts of a program might exhibit different observable behaviours when different evaluation orders are chosen. During the internship, he focused on a small imperative language, and designed an analysis that takes inspiration from information flow analyses. In his Ph.D. work, that started in September 2024, Sébastien Bonduelle will design and implement an effective analysis that detects such problematic cases. Starting from the tiny imperative language, the goal is then to extend the analysis to larger languages, that support algebraic data types, exceptions, dynamic allocation, and first-class functions. This work could lead, ultimately, to an analysis for OCaml programs, that could be integrated in the Salto static analyser.

7.11 Back to the trees: Identifying plants with Human intelligence

Participant: Simon Castellan, Aurore Alcolei.

Plant descriptions have been recorded by botanists since a few thousands years. Identifying the characteristic criteria of a species, (invariants under all the individuals of that species), as well as describing precise morphology with words, proved to be a difficult endeavour. As a result, each botanist tend to use their own vocabulary and way of describing species. Moreover descriptions tend not to be uniform. We show how to use basic tools from type theory and probabilistic programming can help account for morphological diversity in a way that can be understood by a machine.

By providing a compositional metalanguage to describe traits as types, we empower botanists to work together to create a probabilistic representation of plants that accounts for diverse form of polymorphism in plants. Each description becomes a probabilistic element of that type. This data can then be turned automatically into identification keys to help people learn plant identification.

New results showing how to use probabilistic programming to convert between different type, to turn a scientific description into an informal one are being investigated.

7.12 Non-Deterministic Abstract Machines as Semantic Models

Participant: Małgorzata Biernacka, Dariusz Biernacki, Sergueï Lenget, Alan Schmitt.

In complement to skeletal semantics, we explore the definition and formalization of the semantics of programming languages through the specification of non-deterministic abstract machines and the study of their properties.

In 2024, we finalized our work on a fully abstract encoding of the λ-calculus in a process-calculus using non-deterministic abstract machines in a journal paper [13].

We also showed that non-deterministic abstract machines can express the full semantics of the λ-calculus, where computation is allowed anywhere. The abstract machine can then be restricted to recover well-known machines corresponding to reductions strategies [17, 30].

Finally, we showed that complex features of process calculi, such as name restriction or join patterns, can be faithfully described by non-deterministic abstract machines. To this end, we extended our previous work on zipper semantics with a new derivation strategy [22, 32].

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

Salto: static analyses for OCaml programs

Participants: Thomas Genet, Thomas Jensen, Pierre Lermusiaux, Benoît Montagu.

Title: Salto

Industrial partner: Nomadic Labs

Date/Duration: two years (Nov 2022 – Oct 2024)

Participants: Thomas Genet, Thomas Jensen, Pierre Lermusiaux, Benoît Montagu

Additional infos/keywords: As part of the Inria-Nomadic Labs partnership, the EPICURE research team is working on the development of Salto, a static analyzer for OCaml programs. The goal of this analyzer is to help the Nomadic Labs engineers improve the trust on their OCaml code-base, that implements the runtime system for the Tezos blockchain. The Salto static analyzer builds upon abstract interpretation techniques and recent work on control-flow analyses [40] and regular tree languages [37] that are developed in our research team. The aim of the Salto static analyzer is to detect whether an OCaml program might violate some safety properties, such as: May a program raise some uncaught exception? May a program violate some user-defined assertion or invariant? May a program access some data outside the bounds of an array? May a program perform some undesired arithmetic overflow?

Development of Salto

Participants: Pierre Lermusiaux, Benoît Montagu.

Title: Salto

Industrial partner: OCaml Software Foundation

Date/Duration: one year (Nov 2024 – Oct 2025)

Participants: Pierre Lermusiaux, Benoît Montagu

Additional info/keywords: The OCaml Software Foundation supported the development of the Salto static analyser thanks to a one year grant, to fund a research engineer. The goal is to broaden the scope of the analyser, improve its precision and its efficiency, and experimenting with new ideas, so that the analyser can be released to the OCaml community in the near future.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Visits to international teams

Research stays abroad

Sandrine Blazy

Visited institution: Newton Institute

Country: United Kingdom

Dates: 1-11 October

Context of the visit: invited to the Big Specification Prorgamme

Mobility program/type of mobility: research stay, invited talk

Sandrine Blazy, Delphine Demange, Tony Law

Visited institution: EPFL

Country: Switzerland

Dates: 25-29 February

Context of the visit: collaboration with Clément Pit-Claudel

Mobility program/type of mobility: research stay, invited talk

9.2 National initiatives

9.2.1 PEPR Cybersécurité Secureval

Participants: Thomas Jensen, Frederic Besson, Sandrine Blazy, Benjamin Farinier, Alexandre Drewery, Clement Chavanon.

The Secureval project concerns the assessment of the security of digital systems. Digital system security assessment relies on compliance and vulnerability analyses to provide recognized cybersecurity assurances. Innovative tools will be designed around new digital technologies in order to verify the absence of hardware and software vulnerabilities, and to carry out the required proof of conformity. EPICURE contributes with research on advanced static analysis and verified compilation techniques.

10 Dissemination

Participants: Alan Schmitt, Sandrine Blazy, Benoît Montagu, Thomas Jensen, Frédéric Besson, Delphine Demange, Simon Castellan, Thomas Genet.

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

- Alan Schmitt: Steering Committee of JFLA
- Sandrine Blazy: Steering Committee of ACM SIGPLAN CPP, steering committee of ACM SIGPLAN POPL

Member of the organizing committees

- Benoît Montagu: Artifact Evaluation co-Chair for ICFP 2024

10.1.2 Scientific events: selection

Chair of conference program committees

- Sandrine Blazy: PC co-chair of ACM SIGPLAN CPP 2024 and 2025

Member of the conference program committees

- Sandrine Blazy: program committee member of ProLaLa 2024, FMTea 2024 and PriSC 2025
- Benoît Montagu: program committee member of IFL 2024
- Thomas Jensen: program committee member of ACM ICFP 2024
- Frédéric Besson: programm committee member of ACM CCS 2024, WEB conference

Reviewer

- Benoît Montagu: external reviewer for ICFP 2024
- Benoît Montagu: program committee member for the GT-MFS (Méthodes Formelles pour la Sécurité)

10.1.3 Journal

Member of the editorial boards

- Sandrine Blazy: member of the editorial board of the LMCS journal

10.1.4 Invited talks

- "From operational semantics to verified compilation", Sandrine Blazy, ETAPS unifying speaker, Luxembourg, April 2024
- "30 years as an academic", Sandrine Blazy, ETAPS mentoring workshop, April 2024
- "Compilation vérifiée : vers du logiciel zéro défaut", Sandrine Blazy, Colloquium d'informatique, Sorbonne Université, Paris, November 2024
- "Program analysis for software security", Thomas Jensen, 1st International Summer School on Abstract Interpretation, Lipari, Italy, September 2024.

10.1.5 Leadership within the scientific community

Thomas Jensen is director of the Laboratoire d'Excellence CominLabs.

10.1.6 Scientific expertise

- Alan Schmitt, Member of Conseil Scientifique of LMF, Formal Methods Laboratory, Paris Saclay
- Sandrine Blazy, Member of the International Scientific Advisory Board (ISAB) of the Flanders Strategic Research Program in Cybersecurity,

10.1.7 Research administration

- Sandrine Blazy is deputy director of the IRISA CNRS laboratory.

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Licence: Alan Schmitt, L3 INFO, 8h, ENS Rennes, France
- Master: Alan Schmitt, Advanced Semantics, 32h, M2, ENS Rennes, France
- Master: Alan Schmitt, Preparation of Agregation exam, 50h, M2, ENS Rennes, France
- Licence : Benoît Montagu, Programmation de Confiance, 36h, L3, Université Rennes, France
- Master : Benoît Montagu, Analyse et Conception Formelles, 24h, M1, Université Rennes, France
- Master : Benoît Montagu, Software Security, 6.5h, M2, ENS Rennes, France
- Licence : Frédéric Besson, Programmation Fonctionnelle, 28h, L3, Insa, France
- Master : Frédéric Besson, Programmation et Se'curite', 21h, M2, CentraleSupele, France
- Licence : Delphine Demange, Programmation Impérative, 55h, L1, Université de Rennes, France
- Licence : Delphine Demange, Algorithmique et Complexité, 40h, L1, Université de Rennes, France
- Licence : Sandrine Blazy, Programmation de Confiance, 55h, L3, Université Rennes, France
- Master : Sandrine Blazy, Mechanized Semantics, 32h, M1, Université Rennes, France
- Doctorate : Sandrine Blazy, Compiler Verification, MOVEP (MOdeling and VErification of Parallel processes) summer school, 3h, Rennes, France
- Master : Thomas Jensen, Software Security, 20 h, University of Rennes
- Master : Thoams Jensen, Software security, 24h, University of Copenhagen
- Master : Simon Castellan, Sobriété numérique, 20h, ENS Rennes
- Licence : Thomas Genet, Algorithmique et complexité, 40h, L1, Université de Rennes, France
- Licence : Thomas Genet, Sécurité des logiciels et des protocoles, 30h, L3, Université de Rennes, France
- Master : Thomas Genet, Analyse et Conception Formelles, 38h, M1, Université Rennes, France
- Master : Thomas Genet, Blockchains, 12h, M2, Université Rennes, France
- Master : Thomas Genet, Blockchains, 9h, M2, IMT Atlantique, France

10.2.2 Supervision

- L3 internship, Yu-Hui Leana Chiang: Alan Schmitt
- PhD defended, Vincent Rébiscoul: “Analyses Statiques pour Sémantiques Squelettiques”, Thomas Jensen and Alan Schmitt
- L3 internship, Tom Goalard: Benoit Montagu
- M2 internship, Sébastien Bonduelle: Benoît Montagu and Thomas Jensen
- PhD (ongoing), Sébastien Bonduelle, "Analyses statiques de flux d’information par interprétation abstraite": Benoît Montagu and Thomas Jensen
- PhD defended: Solène Mirliaz, "Static relational cost analysis for superscalar architectures", 12/2024.
- PhD in progress: Clément Chavanon, "Refinement of formal specifications for secure environements" since September 2023, Sandrine Blazy and Frédéric Besson.
- PhD in progress: Alexandre Drewery, "Analyse statique incrémentale pour la sécurité logicielle", Thomas Jensen
- PhD in progress: Romeo La Spina, "Analyse de flot de données et de dépendances pour la compilation optimisante vérifiée", Sandrine Blazy and Delphine Demange.
- PhD in progress: Tony Law, "Formally verified high-level synthesis", Sandrine Blazy and Delphine Demange.
- PhD in progress: Alain Delaët-Tixieuil, "Verified compilation for a language describing the law", Sandrine Blazy and Denis Merigoux.
- PhD defended, Théo Losekoot: “Automatic Program Verification by Inference of Relational Models”, Thomas Genet and Thomas Jensen.
- Phd defended, Jean-Loup Hatchkian-Houdot: "Mécanisme de sécurité contre les attaques temporelles via une coopération entre logiciel et matériel embarqué", Frédéric Besson, Pierre Wilke, Guillaume Hier.
- PhD in progress, Malo Revel: “Proving regular theorems on functional programs”, Thomas Genet and Thomas Jensen.

10.2.3 Juries

- Sandrine Blazy: jury member (president) for the PhD defense of Matthieu Baty, CentraleSupelec, December 2024.
- Sandrine Blazy: jury member (president) for the PhD defense of Thaïs Baudon, ENS Lyon, October 2024.
- Sandrine Blazy: jury member for the PhD defense of Nathanaëlle Courant, University Paris cité, September 2024.
- Sandrine Blazy: jury member (president) for the PhD defense of Henrik Plate, Rennes University, September 2024.
- Sandrine Blazy: jury member (president) for the PhD defense of Quentin Le Dilavrec, Rennes University, Februaryc2024.
- Alan Schmitt, jury member (reviewer and president) for the HDR defense of Cinzia di Giusto, March 2024, Université de Nice Côte d’Azur

- Alan Schmitt, jury member (reviewer) for the PhD defense of Mickaël Laurent, June 2024, Université Paris Cité
- Alan Schmitt, jury member (reviewer) for the PhD defense of Giovanni Fabbretti, October 2024, Université Grenoble Alpes
- Alan Schmitt, jury member (reviewer) for the PhD defense of Colin Gonzáles Duburc, November 2024, Université Paris Cité
- Alan Schmitt, jury member (reviewer) for the PhD defense of Nicolas Chappe, November 2024, École Normale Supérieure de Lyon
- Alan Schmitt, jury member (reviewer) for the PhD defense of Loïc Sylvestre, November 2024, Sorbonne Université
- Alan Schmitt, jury member (reviewer) for the PhD defense of Houda Mouhcine, December 2024, Université Paris-Saclay
- Thomas Jensen, jury member (reviewer) for the PhD defense of Francesco Parolini, June 2024, Sorbonne Université.
- Thomas Jensen, jury member (reviewer) for the PhD defense of Denis Mazzucato, December 2024, ENS Rennes.

10.3 Popularization

10.3.1 Productions (articles, videos, podcasts, serious games, ...)

- Article in ERCIM News on the Salto static analyser [36]
- Article in ERCIM News on the SCRATCHS project [35]

10.3.2 Others science outreach relevant activities

- Alan Schmitt, presentation of the work of a researcher, classe de première NSI, Lycée Assomption, May 2024
- Thomas Genet: "Bug, virus, pirates. So many threats and no solution? Yes, mathematics.", given in 5 High Schools close to Rennes.

11 Scientific production

11.1 Major publications

[1] O. Andreescu, T. Jensen, S. Lescuyer and B. Montagu. 'Inferring frame conditions with static correlation analysis'. In: *Proceedings of the ACM on Programming Languages* 3.POPL (2nd Jan. 2019), pp. 1–29. DOI: 10.1145/3290360. URL: https://hal.inria.fr/hal-02413262.

[2] A. Barrière, S. Blazy, O. Flückiger, D. Pichardie and J. Vitek. 'Formally verified speculation and deoptimization in a JIT compiler'. In: *Proceedings of the ACM on Programming Languages* 5.POPL (4th Jan. 2021), p. 26. DOI: 10.1145/3434327. URL: https://hal.science/hal-03185848.

[3] A. Barrière, S. Blazy and D. Pichardie. 'Formally Verified Native Code Generation in an Effectful JIT - or: Turning the CompCert Backend into a Formally Verified JIT Compiler'. In: *Proceedings of the ACM on Programming Languages* (Jan. 2023). DOI: 10.1145/3571202. URL: https://hal.inria.fr/hal-03882598.

[4] F. Besson, S. Blazy, A. Dang, T. Jensen and P. Wilke. 'Compiling Sandboxes: Formally Verified Software Fault Isolation'. In: ESOP 2019 - 28th European Symposium on Programming. Vol. 11423. LNCS. Prague, Czech Republic: Springer, 6th Apr. 2019, pp. 499–524. DOI: 10.1007/978-3-030-17184-1_18. URL: https://hal.inria.fr/hal-02316189.

[5] F. Besson, A. Dang and T. Jensen. 'Information-Flow Preservation in Compiler Optimisations'. In: CSF 2019 - 32nd IEEE Computer Security Foundations Symposium. Hoboken, United States: IEEE, 25th June 2019, pp. 1–13. URL: https://hal.inria.fr/hal-02180303.

[6] M. Bodin, P. Gardner, T. Jensen and A. Schmitt. 'Skeletal Semantics and their Interpretations'. In: *Proceedings of the ACM on Programming Languages* 44 (2019), pp. 1–31. DOI: 10.1145/3290357. URL: https://hal.inria.fr/hal-01881863.

[7] S. Castellan and P. Clairambault. 'The Geometry of Causality: Multi-Token Geometry of Interaction and its Causal Unfolding'. In: *Proceedings of the ACM on Programming Languages* (Jan. 2023). URL: https://hal.science/hal-03286443.

[8] T. Haudebourg, T. Genet and T. Jensen. 'Regular Language Type Inference with Term Rewriting - extended version'. In: *Proceedings of the ACM on Programming Languages*. International Conference on Functional Programming (ICFP) 4.ICFP (2020), pp. 1–29. DOI: 10.1145/3408994. URL: https://hal.inria.fr/hal-02795484.

[9] P. Lermusiaux and B. Montagu. 'Detection of Uncaught Exceptions in Functional Programs by Abstract Interpretation'. In: *Programming Languages and Systems, 33rd European Symposium on Programming, ESOP 2024, Lecture Notes in Computer Science*. ESOP 2024 - 33rd European Symposium on Programming. Vol. 14577. Lecture Notes in Computer Science. Luxembourg, Luxembourg, 5th Apr. 2024, pp. 391–420. DOI: 10.1007/978-3-031-57267-8_15. URL: https://hal.science/hal-04547480.

[10] B. Montagu and T. Jensen. 'Stable relations and abstract interpretation of higher-order programs'. In: *Proceedings of the ACM on Programming Languages* 4.ICFP (2nd Aug. 2020), pp. 1–30. DOI: 10.1145/3409001. URL: https://hal.inria.fr/hal-02916996.

[11] B. Montagu and T. Jensen. 'Trace-Based Control-Flow Analysis'. In: PLDI 2021 - 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. Virtual, Canada: ACM, 20th June 2021, pp. 1–15. DOI: 10.1145/3453483.3454057. URL: https://hal.inria.fr/hal-03266981.

11.2 Publications of the year

International journals

[12] S. Bautista, T. Jensen and B. Montagu. 'An input–output relational domain for algebraic data types and functional arrays'. In: *Formal Methods in System Design* (13th June 2024). DOI: 10.1007/s10703-024-00456-z. URL: https://hal.science/hal-04612474 (cit. on p. 8).

[13] M. Biernacka, D. Biernacki, S. Lenglet, P. Polesiuk, D. Pous and A. Schmitt. 'Fully Abstract Encodings of Lambda-Calculus in HOcore through Abstract Machines'. In: *Logical Methods in Computer Science* 20.3 (3rd July 2024), pp. 1–45. DOI: 10.46298/lmcs-20(3:3)2024. URL: https://inria.hal.science/hal-04638249 (cit. on p. 12).

[14] A. Filinski, K. F. Larsen and T. Jensen. 'Axiomatising an information flow logic based on partial equivalence relations'. In: *International Journal on Software Tools for Technology Transfer* 26.4 (25th June 2024), pp. 445–461. DOI: 10.1007/s10009-024-00756-z. URL: https://inria.hal.science/hal-04827704 (cit. on p. 11).

[15] L. Olivieri, L. Negrini, V. Arceri, T. Jensen and F. Spoto. 'Design and Implementation of Static Analyses for Tezos Smart Contracts'. In: *Distributed Ledger Technologies: Research and Practice* (29th Jan. 2024), pp. 1–23. DOI: 10.1145/3643567. URL: https://inria.hal.science/hal-04827693 (cit. on p. 10).

Invited conferences

[16] S. Blazy. 'From Mechanized Semantics to Verified Compilation: the Clight Semantics of CompCert'. In: *Lecture Notes in Computer Science*. FASE 2024 - 27th International Conference on Fundamental Approaches to Software Engineering. Vol. 14573. Lecture Notes in Computer Science. Luxembourg, Luxembourg: Springer Nature Switzerland, 6th Apr. 2024, pp. 1–21. DOI: 10.1007/978-3-031-57259-3_1. URL: https://inria.hal.science/hal-04553834.

International peer-reviewed conferences

[17] M. Biernacka, D. Biernacki, S. Lenglet and A. Schmitt. 'Optimizing a Non-Deterministic Abstract Machine with Environments'. In: *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*. FSCD 2024 - 9th International Conference on Formal Structures for Computation and Deduction. Tallinn, Estonia: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, pp. 1–22. DOI: 10.4230/LIPIcs.FSCD.2024.11. URL: https://inria.hal.science/hal-04643294 (cit. on p. 12).

[18] C. Chavanon, F. Besson and T. Ninet. 'PfComp: A Verified Compiler for Packet Filtering Leveraging Binary Decision Diagrams'. In: 13th ACM SIGPLAN International Conference on Certified Programs and Proofs - CPP 2024. London, United Kingdom: ACM, 9th Jan. 2024, pp. 89–102. DOI: 10.1145/3636501.3636954. URL: https://inria.hal.science/hal-04893360 (cit. on p. 10).

[19] J.-L. Hatchikian-Houdot, P. Wilke, F. Besson and G. Hiet. 'Formal Hardware/Software Models for Cache Locking Enabling Fast and Secure Code'. In: *ESORICS 2024, 29th European Symposium on Research in Computer Security, Bydgoszcz, Poland, September 16–20, 2024, Proceedings, Part III*. ESORICS 2024 - 29th European Symposium on Research in Computer Security. Vol. 14984. Lecture Notes in Computer Science. Bydgoszcz, Poland: Springer Nature Switzerland, 6th Sept. 2024, pp. 153–173. DOI: 10.1007/978-3-031-70896-1_8. URL: https://hal.science/hal-04804914 (cit. on p. 9).

[20] R. La Spina, D. Demange and S. Blazy. 'Formal Verification of WTO-based Dataflow Solvers'. In: 34th European Symposium on Programming. Hamilton, Canada, 3rd May 2025. URL: https://hal.science/hal-04851724 (cit. on p. 10).

[21] T. Law, D. Demange and S. Blazy. 'A Mechanized Semantics for Dataflow Circuits'. In: *Proceedings of the ACM on Programming Languages, Issue OOPSLA 1*. ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity. Singapore, Singapore, 12th Oct. 2025. URL: https://hal.science/hal-04851772 (cit. on p. 10).

[22] S. Lenglet and A. Schmitt. 'Leaf-First Zipper Semantics'. In: *FORTE 2024, LNCS*. FORTE 2024 - 44th International Conference on Formal Techniques for Distributed Objects, Components, and Systems. Groningen, Netherlands, 2024. DOI: 10.1007/978-3-031-62645-6_7. URL: https://inria.hal.science/hal-04571340 (cit. on p. 12).

[23] P. Lermusiaux and B. Montagu. 'Detection of Uncaught Exceptions in Functional Programs by Abstract Interpretation'. In: *Programming Languages and Systems, 33rd European Symposium on Programming, ESOP 2024, Lecture Notes in Computer Science*. ESOP 2024 - 33rd European Symposium on Programming. Vol. 14577. Lecture Notes in Computer Science. Luxembourg, Luxembourg, 5th Apr. 2024, pp. 391–420. DOI: 10.1007/978-3-031-57267-8_15. URL: https://hal.science/hal-04547480 (cit. on p. 8).

[24] T. Losekoot, T. Genet and T. Jensen. 'Verification of programs with ADTs using Shallow Horn Clauses'. In: *Lecture Notes in Computer Science*. Static Analysis Symposium. Vol. 31st Static Analysis Symposium. 14995. Pasadena (CA), United States: Springer, 2024. URL: https://inria.hal.science/hal-04820358 (cit. on p. 9).

[25] S. Yuan, F. Besson and J.-P. Talpin. 'End-to-End Mechanized Proof of a JIT-Accelerated eBPF Virtual Machine for IoT'. In: CAV 2024 - 36th International Conference on Computer Aided Verification. Vol. 14681. Lecture Notes in Computer Science. Montreal, Canada: Springer Nature Switzerland, 26th July 2024, pp. 325–347. DOI: 10.1007/978-3-031-65627-9_16. URL: https://inria.hal.science/hal-04762503 (cit. on p. 9).

National peer-reviewed Conferences

[26] M. Andrieux and A. Schmitt. 'Skeletal Semantics of a Fragment of Python'. In: 35es Journées Francophones des Langages Applicatifs (JFLA 2024). Saint-Jacut-de-la-Mer, France, 2024. URL: https://inria.hal.science/hal-04406392 (cit. on p. 7).

Edition (books, proceedings, special issue of a journal)

[27] D. Demange and A. Guatto, eds. *JFLA 2024 - 35es Journées Francophones des Langages Applicatifs*. 35es Journées Francophones des Langages Applicatifs (JFLA 2024). Jan. 2024, pp. 1–328. URL: https://inria.hal.science/hal-04407194.

Doctoral dissertations and habilitation theses

[28] V. Rébiscoul. 'Analyses statiques pour sémantiques squelettiques'. Université de Rennes, 14th May 2024. URL: https://theses.hal.science/tel-04849514 (cit. on p. 7).

Reports & preprints

[29] E. Bannier, S. Castellan, S. Derrien, F. Galassi, L. Garnier, L. Hoyet, A. l'Azou, N. Lahaye, M. J.-M. Macé, O. Martineau, A. Masson, T. Maugey, B. Ninassi, E. Rohou, M. Simonin and F. Taïani. *Reducing GHG emissions from business travel: A collaborative approach at IRISA/Inria*. Groupe de travail « missions » IRISA / Centre Inria de l'Université de Rennes, Mar. 2024, pp. 1–16. URL: https://univ-rennes.hal.science/hal-04506138.

[30] M. Biernacka, D. Biernacki, S. Lenglet and A. Schmitt. *Optimizing a Non-Deterministic Abstract Machine with Environments*. May 2024. URL: https://inria.hal.science/hal-04568253 (cit. on p. 12).

[31] S. Castellan and P. Clairambault. *Disentangling Parallelism and Interference in Game Semantics*. 2024. URL: https://hal.science/hal-03182043.

[32] S. Lenglet and A. Schmitt. *Leaf-First Zipper Semantics*. Apr. 2024. URL: https://inria.hal.science/hal-04537440 (cit. on p. 12).

[33] P. Lermusiaux and B. Montagu. *Detection of Uncaught Exceptions in Functional Programs by Abstract Interpretation (Extended Version)*. RR-9536. Inria, 8th Apr. 2024, p. 38. URL: https://inria.hal.science/hal-04410771.

[34] T. Losekoot, T. Genet and T. Jensen. *Verification of programs with ADTs using Shallow Horn Clauses – extended version*. 9th Aug. 2024. URL: https://inria.hal.science/hal-04669706 (cit. on p. 9).

Scientific popularization

[35] F. Besson, C. Le Du and P. Wilke. 'Side-Channel Resistant Applications through Co-designed Hardware/ Software: the SCRATCHS Project'. In: *ERCIM News*. Special theme: Software Security October 2024.139 (2024). URL: https://inria.hal.science/hal-04894615 (cit. on p. 17).

[36] P. Lermusiaux and B. Montagu. 'The Salto Project: Static Analysis of OCaml Programs by Abstract Interpretation'. In: *ERCIM News*. Special theme: Software Security October 2024.139 (1st Oct. 2024), p. 24. URL: https://hal.science/hal-04769799 (cit. on pp. 8, 17).

11.3 Cited publications

[37] T. Haudebourg, T. Genet and T. Jensen. 'Regular Language Type Inference with Term Rewriting - extended version'. In: *Proceedings of the ACM on Programming Languages*. International Conference on Functional Programming (ICFP) 4.ICFP (2020), pp. 1–29. DOI: 10.1145/3408994. URL: https://inria.hal.science/hal-02795484 (cit. on p. 12).

[38] T. Losekoot, T. Genet and T. Jensen. 'Automata-Based Verification of Relational Properties of Functions over Algebraic Data Structures'. In: *Lipics*. Rome, Italy: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, June 2023, pp. 1–21. DOI: 10.4230/LIPIcs.FSCD.2023.7. URL: https://inria.hal.science/hal-04216680 (cit. on p. 8).

[39] R. Monat. 'Static Type and Value Analysis by Abstract Interpretation of Python Programs with Native C Libraries'. PhD Thesis. Sorbonne Université, 2021 (cit. on p. 7).

[40] B. Montagu and T. Jensen. 'Trace-Based Control-Flow Analysis'. In: *PLDI 2021 - 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*. Virtual, Canada: ACM, June 2021, pp. 1–15. DOI: 10.1145/3453483.3454057. URL: https://inria.hal.science/hal-03266981 (cit. on p. 12).

[41] L. Noizet and A. Schmitt. 'Semantics in Skel and Necro'. In: *ICTCS 2022 - Italian Conference on Theoretical Computer Science*. CEUR Workshop Proceedings. Rome, Italy, Sept. 2022, pp. 1–17. URL: https://inria.hal.science/hal-03784478 (cit. on p. 7).

[42] S. Yuan, F. Besson, J.-P. Talpin, S. Hym, K. Zandberg and E. Baccelli. 'End-to-end Mechanized Proof of an eBPF Virtual Machine for Micro-controllers'. In: *CAV 2022 - 34th International Conference on Computer Aided Verification*. Haifa, Israel, Aug. 2022, pp. 1–23. URL: https://inria.hal.science/hal-03888082 (cit. on p. 9).