

# 2025 Activity Report

RESEARCH CENTRE: Inria Centre at the University of Lille

IN PARTNERSHIP WITH: Université de Lille, CNRS

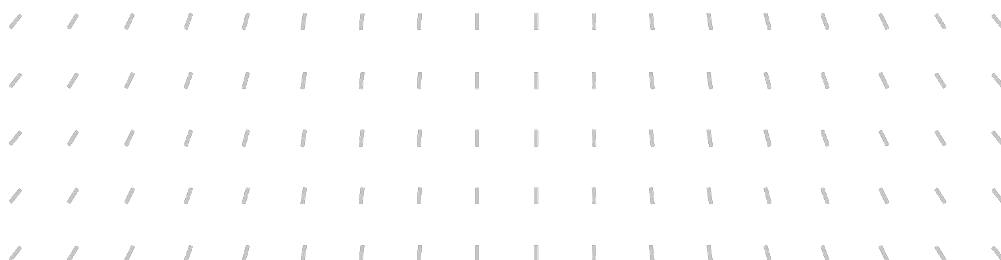
  
Project-Team

## SYCOMORES

Symbolic analysis and Component-based design for  
Modular Real-Time Embedded Systems



*In collaboration with* Centre de Recherche en Informatique, Signal et Automatique  
de Lille



## **Project-Team SYCOMORES**

*Creation of the Project-Team: 2021 October 01*

Each year, Inria research teams publish an Activity Report presenting their work and results over the reporting period. These reports follow a common structure, with some optional sections depending on the specific team. They typically begin by outlining the overall objectives and research programme, including the main research themes, goals, and methodological approaches. They also describe the application domains targeted by the team, highlighting the scientific or societal contexts in which their work is situated. The reports then present the highlights of the year, covering major scientific achievements, software developments, or teaching contributions. When relevant, they include sections on software, platforms, and open data, detailing the tools developed and how they are shared. A substantial part is dedicated to new results, where scientific contributions are described in detail, often with subsections specifying participants and associated keywords. Finally, the Activity Report addresses funding, contracts, partnerships, and collaborations at various levels, from industrial agreements to international cooperations. It also covers dissemination and teaching activities, such as participation in scientific events, outreach, and supervision. The document concludes with a presentation of scientific production, including major publications and those produced during the year.

## Keywords

### Computer sciences and digital sciences

- A2.1.9. – Synchronous languages
- A2.3.1. – Embedded systems
- A2.3.3. – Real-time systems
- A2.6.1. – Operating systems
- A4.5.1. – Static analysis
- A4.5.3. – Program proof
- A7.2. – Logic in Computer Science

### Other research topics and application domains

- B6.6. – Embedded systems

## Contents

<b>Project-Team SYCOMORES</b>	<b>1</b>
<b>1 Team members, visitors, external collaborators</b>	<b>5</b>
<b>2 Overall objectives</b>	<b>5</b>
<b>3 Research program</b>	<b>7</b>
3.1 Axis 1: Design and implementation of RT-components	7
3.1.1 Short term	7
3.1.2 Medium term	7
3.2 Axis 2: Formal methods	8
3.2.1 Short Term	8
3.2.2 Medium Term	9
3.3 Long term	10
<b>4 Application domains</b>	<b>11</b>
<b>5 Social and environmental responsibility</b>	<b>12</b>
<b>6 Highlights of the year</b>	<b>12</b>
<b>7 Latest software developments, platforms, open data</b>	<b>12</b>
7.1 Latest software developments	12
7.1.1 prelude	12
7.1.2 ptask	12
7.1.3 Catala	12
7.1.4 dates-calc	13
7.1.5 Mopsa	13
7.1.6 Haddock	13
7.1.7 Polymalys	13
7.1.8 WSymb	14
7.1.9 Seplog	14
7.1.10 RevState	14
<b>8 New results</b>	<b>14</b>
8.1 CUTEcat: Concolic Execution for Computational Law	14
8.2 Compositional Static Value Analysis for Higher-Order Numerical Programs	15
8.3 Stop treating code like an afterthought: record, share and value it	15
8.4 Easing Maintenance of Academic Static Analyzers	16
8.5 Mopsa at the Software Verification Competition	16
8.6 Implementing CNN on embedded real-time systems	16
8.7 A Kleene Algebra with Tests for Union Bound Reasoning About Probabilistic Programs	17
8.8 BiGKAT: an algebraic framework for relational verification of probabilistic programs	17
8.9 Session Types for the Concurrent Composition of Interactive Differential Privacy	17
8.10 A characterization of Basic Feasible Functionals through higher-order rewriting and tuple interpretations	18
8.11 Dependent Coeffects for Local Sensitivity Analysis	18
8.12 Towards determinism in PDL: relations and proof theory	18
8.13 A Domain-Theoretic Framework for Composing Effectful Programs and Their Equations	19
8.14 Pleasant Imperative Program Proofs with GallinaC	19
<b>9 Bilateral contracts and grants with industry</b>	<b>19</b>
9.1 Bilateral Grants with Industry	19

<b>10 Partnerships and cooperations</b>	<b>20</b>
10.1 International initiatives	20
10.1.1 Participation in other International Programs	20
10.2 International research visitors	20
10.2.1 Visits of international scientists	20
10.2.2 Visits to international teams	21
10.3 National initiatives	21
10.3.1 Inria Exploratory Action AVoCat	21
10.3.2 ANR JCJC RAISIN: Resource-Aware Conservative Static Analysis	22
10.3.3 ANR PRC HOPR: Higher-Order Probabilistic and Resource-aware Reasoning	22
10.4 Public policy support	22
<b>11 Dissemination</b>	<b>22</b>
11.1 Promoting scientific activities	22
11.1.1 Scientific events: selection	22
11.1.2 Journal	22
11.1.3 Invited talks	22
11.1.4 Leadership within the scientific community	23
11.1.5 Scientific expertise	23
11.1.6 Research administration	23
11.2 Teaching - Supervision - Juries - Educational and pedagogical outreach	23
11.2.1 Supervision	23
11.2.2 Juries	23
11.3 Popularization	23
11.3.1 Specific official responsibilities in science outreach structures	23
11.3.2 Participation in Live events	23
<b>12 Scientific production</b>	<b>23</b>
12.1 Major publications	23
12.2 Publications of the year	24
12.3 Cited publications	26

## 1 Team members, visitors, external collaborators

### Research Scientists

- Patrick Baillot [CNRS, Senior Researcher, HDR]
- Raphael Monat [INRIA, Researcher]
- Vlad Rusu [INRIA, Researcher, HDR]

### Faculty Members

- Giuseppe Lipari [Team leader, UNIV LILLE, Professor]
- Julien Forget [UNIV LILLE, Associate Professor]
- Leandro Moreira Gomes [UNIV LILLE, Associate Professor, from Sep 2025]

### Post-Doctoral Fellow

- Leandro Moreira Gomes [UNIV LILLE, until Aug 2025]

### PhD Students

- Chiara Daini [INRIA, until Sep 2025]
- Nordine Feddal [INRIA]
- Andrei Florea [LIX, until Sep 2025]
- Pierre Goutagny [INRIA]
- Victor Sannier [UNIV LILLE]
- Leonardo Speranzon [INRIA, from Sep 2025]
- Artur Szafarczyk [INRIA, from Sep 2025]
- Milla Valnet [LIP 6]

### Interns and Apprentices

- Tom Goalard [INRIA, Intern, from Sep 2025]

### Administrative Assistant

- Nathalie Bonte [INRIA]

## 2 Overall objectives

The SYCOMORES project-team aims at developing a **framework for the design and the analysis of embedded real-time systems** based on **symbolic analysis of parametric components**.

To reduce the complexity, we will use a modular approach to the design, development and analysis of ERTS. In particular, we will decline *modularity* along several directions (Figure 1):

- a *component-based design and development* methodology; a ERTS system is decomposed in generic, configurable and reusable components that can be combined and instantiated in the final system;
- *parametric specification* of inputs, configurations and properties; a component is characterised by its interface that will be specified using parameters whose concrete values are unknown at design time;

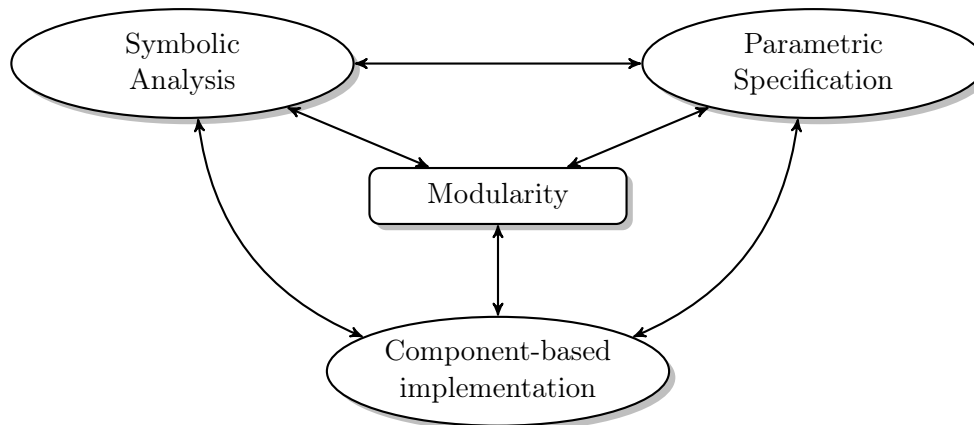


Figure 1: Ontology of terms in SYCOMORES.

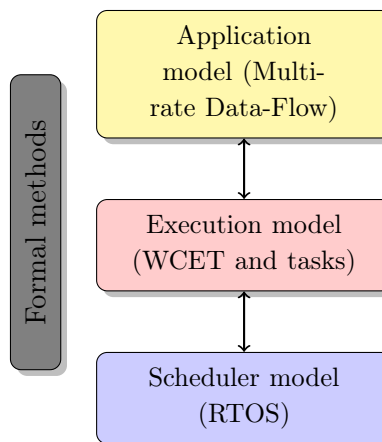


Figure 2: Abstraction Layers.

- *symbolic analysis* of components; by using symbolic techniques, the properties of a component will be proved correct at design time even when parameters have unknown values.

Finally, we will propose *composition operators* to integrate a set of components into a larger component, or into a complete system. The operators will allow to (partially or totally) instantiate parameters and connect component interfaces guaranteeing their compatibility and preserving their properties.

To this end, we will investigate several challenging problems at three different levels of abstraction, as shown in Figure 2. At the application level, we will focus on *multi-rate data-flow* programming languages like SCADE [28], SIMULINK [41], or in our case PRELUDE [42], as they are widely used in safety-critical domains like avionics and automotive. We want to tackle the notions of component, parametric interfaces and contracts in the ERTS context. We will investigate **modular code generation** (compilation) of a synchronous component (as opposed to a synchronous program) with real-time constraints. In the long term, we would like to **prove the semantic preservation of properties** of the program during compilation by using interactive proof systems.

The compiler will produce the component implementation as a set of concurrent real-time tasks with their scheduling parameters. At this level, we will work on **parametric and modular worst-case execution time (WCET) analysis** of tasks' code, and **symbolic schedulability analysis** of the components' tasks. We will also work on analysis of the correctness of system integration, and **safe dynamic replacement/upgrade** of components.

At the lowest level, we will work on the scheduler implementation in the Operating System. We will propose a **hierarchical scheduling architecture**, in order to provide temporal isolation to real-time

components. We will formally prove the scheduler properties by using **modular proof techniques** in the Coq proof assistant [47], and we will provide a **correct-by construction implementation of the scheduler** by using code generation techniques from a Coq program specification.

As a cross-cutting activity, since safety will be a prime concern in our project, we will rely on *formal methods* throughout the project: synchronous programming languages, abstract interpretation, symbolic analyses and proof techniques. Our research objectives will be tightly coupled to ensure the safety of the final framework.

For instance, our tools and techniques will share proof facts (e.g. timing analysis is correct only if schedulability analysis and WCET analysis are both correct), models (e.g. the schedulability analysis, programming language and WCET analysis must share the same task model), and results (e.g. the WCET and schedulability results will impact the program design).

### 3 Research program

We detail the objectives by categorising them according to 2 major research axes. Within each axis, we distinguish Short Term (1-3 years) and Medium Term (2-4 years) objectives, and we describe how we will achieve them. We also detail how the objectives are related to one another. Finally, we present our Long Term (4+ years) objectives.

#### 3.1 Axis 1: Design and implementation of RT-components

In this research axis we will focus on the programming of RT-components. We will cover programming aspects at different levels of the development process, from high-level design, to low-level implementation on distributed heterogeneous embedded platforms.

##### 3.1.1 Short term

**(A1.S1) Synchronous languages** We continue working on synchronous programming languages for generating correct-by-construction code. In particular, one of the objectives of the ANR projet CORTEVA (which was coordinated by Giuseppe Lipari, started in Jan. 2018, and lasted 48 months) was to extend the synchronous language PRELUDE [42] to address systems with extremely variable execution time.

The language has been extended (its semantics and its compilation) with constructs to dynamically change the behaviour of a subset of the system based on rare events, like the occurrence of a large execution time of a task. The proposed extensions guarantee that the system respects correctness properties even in the presence of such rare events.

*Dependencies:* We rely on Parametric WCET to detect the occurrence of large execution times (A2.S1).

**(A1.S2) Scheduling of complex task models** Modern hardware platforms consist of heterogeneous processors with a large degree of parallelism. For example, the NVIDIA Jetson AGX Xavier chip that is used in modern ADAS systems in the automotive domain, comprises 2 DLAs (Deep Learning Accelerators), 1 integrated GPU (Graphical Processing Unit), and 8 ARM 57 processing cores. Efficiently programming ERTS for such hardware is not easy, as the complexity of the interaction between software and hardware resources makes it difficult to predict performance. Building predictable real-time applications on these platforms requires appropriate programming models. In the recent literature, many graph-based task models have been proposed to exploit the parallelism of such architectures [46, 31]. Such models are neither based on components, nor are they parametric. It is our intention to investigate the possibility to apply component-based techniques to such complex task models.

##### 3.1.2 Medium term

**(A1.M1) Parametric scheduling analysis of Real-Time components.** Analyzing the schedulability of a system under variations of the task parameters is a complex problem: the complexity of the analysis grows exponentially with the size of the system, and it explodes for medium-to-large sized systems.

The problem can be decomposed by analysing the schedulability of smaller real-time components under variation of some parameters. The idea is to follow a Resource Reservation strategy, where components are assigned and guaranteed a fraction of the system resources. In this way, the temporal behaviour of one component is *isolated* from the interference of other components.

First, we will introduce the notion of *parametric interface* for a component modelled as a multi-rate data-flow program. The interface will specify the temporal properties and constraints on the input/outputs of a component: periodicity and deadline constraints, dimension of the buffer for communicating with other components, and communication protocols.

To analyse the schedulability of a component under the hypothesis of variation of its parameters (the ones specified in the interface as well as the WCET of the tasks), we will extend the *sensitivity analysis* techniques, first proposed by Bini et al. [33, 32] to more complex task models and to hierarchical scheduling systems. The objective is to find the range of the parameters for which the component is schedulable.

**(A1.M2) Predictable communication costs.** ERTS are increasingly being built using multicore hardware. This transition is however still significantly delayed by the difficulty to provide *safe* and *tight* timing guarantees. Indeed, even though multicore architectures enable the simultaneous parallel execution of programs on different cores, these programs are not completely independent. They have to be synchronized at some point to exchange data and they also share some common resources, such as peripherals, communication bus and (parts of the) memory. Synchronizations and shared resource contentions cause hard-to-predict interferences and timing overheads, which significantly increase the complexity of timing analyzes (WCET and scheduling).

One solution for mitigating these overheads, and making them more predictable, consists in relying on multi-phase task models that decouple communication phases from computation phases [44, 36]. This greatly simplifies WCET analysis of computation phases, and also makes it possible to schedule communication phases so as to reduce their interference.

Memory architectures based on local addressable memories (scratchpads), instead of caches, are also being proposed [37, 43], to avoid the hard-to-predict timing effects of cache consistency mechanisms (which are used to speed-up access to the main shared memory).

We plan to integrate these two approaches in our project, since predictability is a major concern. The PRELUDE compiler will be extended to provide multi-phase task code generation, to be executed on scratchpad-based memory architectures. We will also develop the corresponding schedulability analysis method. Our objective is to devise our development framework such that the programmer abstracts from the implementation of communication mechanisms related on the target OS and hardware. This will enable the programmer to seamlessly transition between different architectures (unicore/multicore, cache-based/scratchpad-based).

## 3.2 Axis 2: Formal methods

In this research axis, we will focus on compositional techniques for proving compositional system properties. On one hand, we will design static program analysis techniques for RT components at the binary level. While these techniques will mainly focus on WCET analysis, some results will also be reused to analyze security properties instead. On the other hand, we will focus on compositional techniques for proving RT components of operating systems. Our plan is to focus mainly on RT schedulers.

### 3.2.1 Short Term

**(A2.S1) Symbolic WCET analysis.** We will work towards improving static analysis techniques for Worst-Case Execution Time Analysis. In particular, we will extend the Parametric WCET method [30] with more powerful symbolic capabilities. Currently, Parametric WCET can represent the WCET of a function as a formula in a number of parameters (e.g. input data). It does however suffer from two important limitations. First, it cannot represent relations between distinct parameters. We will extend the Parametric WCET by abstract interpretation of binary code to detect linear relations between distinct parameters in the code, by using techniques similar to [29]. Second, it struggles to represent program properties that relate instructions of the program that are not close to one another, such as for instance infeasible execution paths [45]. We are currently extending this work to enable the representation of properties related to such *global execution contexts*.

**(A2.S2) A proof methodology for a standard scheduling policy.** We shall start with the standard scheduling policy EDF (Earliest Deadline First) and develop a proof methodology for it, which we shall later adapt to more advanced policies. The methodology incorporates a formal notion *refinement* as a means to master complexity and to smoothly descend from abstract definitions down to executable schedulers.

First, we are planning to model EDF at an abstract level in Coq. We shall formally prove at this level the *schedulability* property: under adequate hypotheses any given set of periodic hard real-time tasks can be scheduled, such that each task completes before its deadline. Since in the short term we shall only deal with strictly periodic, hard real-time tasks (reading data from sensors, performing some computation and sending the results to actuators), one only needs to consider the schedulability on finite executions, whose length equals the hyper-period: the least common multiplier of the task's periods. As a result, we expect that *induction*, well supported by Coq, will be the appropriate proof technique for this stage of the project.

Then, we shall refine the abstract EDF scheduling policy into a scheduling *algorithm* written in Coq's input language *Gallina*, a purely functional language, and shall prove again by induction that the algorithm preserves the already established properties of the policy.

Next, our plan is to further refine these *functional* algorithms into *imperative* Gallina programs, in order to get a step closer to executable code<sup>1</sup>.

Imperative programs in purely functional languages such as Gallina are traditionally written using *monads* e.g., *state* monads, which enable variable assignments in functional code. For doing this we shall benefit from the experience of our colleagues in the 2XS team, our closest collaborators within the CRISAL laboratory. They have been developing a minimalistic OS kernel called Pip [48] in imperative Gallina using the state-monad technology and have directly proved properties of the kernel in Coq. Unlike them, we do not prove properties directly on the monadic code, but shall prove a refinement step (from functional to imperative) ensuring that schedulability holds on the imperative scheduler.

The final step is translating the imperative Gallina scheduler to executable code. For this we shall use a translator also developed by the 2XS team, which essentially maps word-for-word imperative Gallina programs to C programs with appropriate primitives that, after compilation of the C code, turn our schedulers into executable programs within Pip.

### 3.2.2 Medium Term

**(A2.M1) Modular WCET analysis.** Traditional WCET analysis is performed on a whole program. This approach is limited by two factors: first, performance concerns (analysis time tends to grow faster than the analyzed program size), and second, the analysis requires access to the complete program (including libraries, etc.). A modular and composable WCET analysis approach would reduce the analysis time, and enable the integration of third-party library or system calls in the analysis process.

To this end, we want to extend our polyhedra-based abstract interpretation [29] to support inter-procedural analysis, based on function *summaries* [35], describing relations between the inputs and outputs of the function. This will enable us to compute WCET-related functional properties, such as e.g. loop bounds, in a composable way.

This work on composable analysis will lead to a full composable WCET analysis, by integrating it in our symbolic WCET computation framework [30].

*Dependencies:* To achieve this objective, we will need the advanced parameter handling techniques of A2.S1.

**(A2.M2) Security analysis of binary code.** Program analysis of ERTS has historically focused mainly on *safety*, i.e. ensuring the absence of system failures. However, in recent years ERTS have become increasingly more connected to other computer systems through communication networks. This makes ERTS more vulnerable to external attacks, as illustrated by various reports of hacks of highly computerized modern cars. This results in an increased need for analyses that focus on ensuring the *security* of ERTS, i.e. ensuring there is no vulnerability to external malevolent computer attacks.

Our work on abstract interpretation of binary code [29] enables to detect linear relations between the data-locations accessed by a binary program (registers, memory addresses and their contents). While this

<sup>1</sup>Going directly from functional to executable code is not an option, since that would require a complex compilation process with a high risk of losing the schedulability properties already proved on the functional code.

work was initially targeted for WCET analysis, we plan to apply the developed techniques to the security domain as well. In particular, we will analyze security properties specific to the binary structure (e.g. unauthorized accesses to specific memory sections) and study the discovery of potential security threats in closed-source software (to detect malwares).

**(A2.M3) More advanced schedulers and other RT components.** Once the validation of the proof/refinement methodology on the EDF example is complete, we are planning to progressively generalise it to other schedulers. The next likely candidate is the *Grub* scheduler, developed by Giuseppe Lipari [27, 39], which generalises EDF and makes it possible to mix periodic tasks (hard real-time) with sporadic tasks (soft real-time). The algorithm guarantees that the deadlines of the hard real-time tasks are met, while for the soft real-time ones a certain quality of service is guaranteed.

We are also planning to formally verify existing resource reservation hierarchical schedulers [40] by extending the proof/refinement approach with compositional features that exploit the component-based nature of the considered applications.

Depending on the availability of human resources, we would also like to formally verify other RT components, such as interruption multiplexers, memory managers, or synchronisation mechanisms.

**(A2.M4) More advanced proof techniques.** We expect that different verification techniques will be necessary to prove these more advanced schedulers. The EDF scheduler can be proved correct by considering its behaviour over a limited period of time, which as explained above can be dealt with using induction. However, *Grub* also has to schedule sporadic tasks that, by definition, do not repeat themselves periodically. Without a periodic repetition the behaviour cannot be studied over a finite interval, infinite executions have to be considered. Hence, we are planning to use coinduction, the natural technique for defining and reasoning about infinite objects. As stated in the Preliminaries we already have experience with coinduction, especially, with improving the way it is dealt with in Coq to make it better suited for use in nontrivial applications. We are planning to continue doing this both for corecursive program definitions (e.g., reactive systems, including schedulers, are such programs) and for coinductive reasoning techniques.

### 3.3 Long term

In the long term, we will integrate the elements studied during the medium term phase, in order to provide a seamless framework that goes from high-level design to final implementation. Translations will be automated and proved correct. These objectives are transverse by nature, so all members of the project-team will participate. Since these objectives focus on integration of our previous results, they are related to all of our short and medium term objectives.

**(L1) Integrated framework.** During the medium term phase, we will develop bricks that contribute to the design and analysis of ERTS. In the long term phase, we will integrate these bricks into a complete framework. This will raise several research topics.

First, we will have to evaluate the impact of scheduling on WCET analysis. Though this topic has already been studied before, our symbolic approach to both problems will raise new opportunities and challenges.

Second, we will evaluate the scalability of the approach with respect to realistic and complex real-time programs. In particular, the advent of powerful heterogeneous hardware platform permits to exploit their large scale parallelism. It is then important to check the suitability and the expressiveness of our framework with respect to these new powerful platforms.

**(L2) Proving semantics preservation.** In our framework, an ERTS is first specified with a high-level data-flow semantics (in *PRELUDE*). Then, it undergoes translations to produce the final program (in *C*) embedded on the hardware platform. One of our long term objectives is to prove that the complete translation process, from *PRELUDE* to *C*, is semantics-preserving.

There exists previous work and techniques for the formal verification of compilers such as *COMP CERT* [38] (from *C* code to binary code) and the *LUSTRE* verified compiler [34]. However, these works focus on the preservation of the *functional semantics* (computing the correct values). A major novelty of our work will be to focus on the preservation of the *temporal semantics* (computing values at the correct time) as well. As far

as we know, proving the preservation of temporal semantics was previously explored in theoretical models (e.g. timed automata, or Petri nets), but not in programming languages and compilers. It is an ambitious challenge, because it connects compilation with scheduling and WCET analysis.

**(L3) Corecursion-preserving compilation and coinductive verification techniques.** An alternative view of reactive programs (such as PRELUDE programs) is that of corecursive transformers from infinite flows of inputs to infinite flows of outputs. We envisage an enhanced compilation chain that, in addition to what is planned in L2, enables the tracing of corecursion down to the executable code. Since corecursive calls typically compile to low-level instructions such as loops or jumps, such a tracing mechanism would enable recognising the corecursive calls at the low-level. This, in turn, would enable the formal reasoning about low-level corecursive programs, using coinductive techniques that we shall develop in A2. We note that schedulers can also be expressed as corecursive programs, hence the verification boils down to compositionally verifying compositions of corecursive programs. It is, again, an ambitious challenge at every step, since corecursive programs are notoriously difficult to define, compose, and verify. This envisaged works depends on progress in essentially all the objectives enumerated above.

**(L4) Dynamic update of components.** ERTS may evolve over time, in particular, one component may be upgraded while the system is operational. In the traditional development process of critical software, once the system has been developed, tested and *certified*, it is not possible to change it anymore, with the only exception of correcting a critical bug. Dynamic updates are not possible since they would require to re-certify the whole system.

In the long term, we would like to apply our component based development process to the problem of guaranteeing the correctness of dynamic updates. This means that the system must be able to evolve over time, during operation, without jeopardizing the guarantee on existing properties.

Given the large complexity of the formal methods we will use for off-line analysis, it is unthinkable to apply the same type of analysis on-line. One possibility is to separate the analysis into two distinct parts: an off-line part which may be complex and does most of the work; and an on-line part which is much simpler but relies on the off-line computation. Alternatively, it is possible to off-load part of the analysis to the cloud, where there is abundance of computational resources.

## 4 Application domains

The long term research we propose to pursue in this project will advance the current development practice for embedded real-time critical systems.

First, it will impact the design and development of critical software for domains like avionics, automotive, train, etc. It is well known that developing safety-critical software is a long and costly process, where each error could endanger human life. It has been estimated that the cost to certify 30K lines of DAL A code is around 2M\$ if the code has been developed by experienced programmers, and it jumps to 8M\$ if the code has been produced by non-experienced ones.

The avionic industry is slowly adopting formal methods to reduce the cost by reducing (or, in certain cases, eliminating altogether) testing and improve confidence in the methodology. Our integrated framework (L1) will greatly reduce the development cost of safety-critical software because it will automatise many of the steps in the design and development methodology. For example, the use of semantic preserving transformations (L2, L3) will enhance the confidence in the correctness of the generated code, reducing the need for extensive testing, thus further reducing cost.

Other critical domains, like automotive, have not yet fully adopted safety-critical standard methodologies like in the avionic domain, mainly for cost reasons. Our framework will lower the cost of developing safety critical software, thus improving the safety of critical software in a wider range of domains.

Finally, thanks to the research in (L4), it will be possible to update a software component on-line without performing a complete analysis of the system, *while the system is operational*. An example of futuristic application will be the possibility to update one software subsystem of an autonomous vehicle while the vehicle is running, without compromising its functionality.

## 5 Social and environmental responsibility

Raphaël Monat takes part in the gender-equality commission of the CRISStAL laboratory. There are no team-specific actions to promote diversity and gender balance. However, we adhere to the general policies of the CRISStAL laboratory and the Inria center of the University of Lille.

## 6 Highlights of the year

The paper "CUTECat: Concolic Execution for Computational Law" has received a "Distinguished Artifact Award" at ESOP'25.

## 7 Latest software developments, platforms, open data

### 7.1 Latest software developments

#### 7.1.1 prelude

**Keywords:** Synchronous language, Real time

**Functional Description:** Prelude is a synchronous data-flow language with real-time constraints. Prelude programs are compiled into multi-thread C code. The language and its compiler are developed in collaboration with ONERA Toulouse.

**URL:** <https://www.cristal.univ-lille.fr/~forget/prelude.html>

**Publications:** [hal-00802695](#), [tel-01942421](#), [hal-00688490](#), [inria-00638936](#), [inria-00618587](#), [hal-03817684](#)

**Contact:** Julien Forget

**Partner:** Onera

#### 7.1.2 ptask

**Keywords:** Real time, Library

**Functional Description:** PTASK is a library for programming real-time systems in Linux. It serves as the target RTOS API in the SYCOMORES project. The PTASK library is authored by Giuseppe Lipari. It has been initially developed to support teaching real-time systems at the Scuola Sant'Anna. It has later been extended with additional capabilities and it is being used as target for design tools such as CPAL19 from RTaW20 and by other companies.

**URL:** <https://github.com/glipari/ptask>

**Contact:** Giuseppe Lipari

**Partner:** Scuola Superiore Sant'Anna

#### 7.1.3 Catala

**Keywords:** Domain specific, Programming language, Law

**Functional Description:** Catala is a domain-specific programming language designed for deriving correct-by-construction implementations from legislative texts. Its specificity is that it allows direct translation from the text of the law using a literate programming style, that aims to foster interdisciplinary dialogue between lawyers and software developers. By enjoying a formal specification and a proof-oriented design, Catala also opens the way for formal verification of programs implementing legislative specifications.

**URL:** <https://catala-lang.org/en>

**Publications:** [hal-04391612](#), [hal-03712130](#), [hal-03781578](#), [hal-04907935](#), [hal-03128248](#), [hal-03159939](#), [hal-02936606](#), [hal-03869335](#)

**Contact:** Denis Merigoux

**Participants:** Vincent Botbol, Romain Primet, Denis Merigoux, Louis Gesbert, Aymeric Fromherz, Alain Delaet-Tixeuil, Raphael Monat

**Partner:** Université Panthéon-Sorbonne

#### 7.1.4 dates-calc

**Keywords:** Law, Programming language

**Functional Description:** A date calculation library with a well-defined semantics

**URL:** <https://github.com/CatalaLang/dates-calc>

**Publication:** [hal-04536403](#)

**Contact:** Raphael Monat

#### 7.1.5 Mopsa

**Keywords:** Formal methods, Static analysis, Abstraction

**Functional Description:** Mopsa is an open-source static analysis platform relying on abstract interpretation. It provides a novel way to combine abstract domains, in order to offer extensibility and cooperation between them, which is especially beneficial when relational numerical domains are used. It is able to analyze C, Python, and programs mixing these two languages. Mopsa was originally developed at LIP6, Sorbonne Université following an ERC Consolidator Grant award to Antoine Miné. It is now partially developed at Inria.

**URL:** <https://gitlab.com/mopsa/mopsa-analyzer/>

**Contact:** Antoine Mine

**Partner:** Sorbonne Université

#### 7.1.6 Haddock

**Keywords:** Partial function, (Co)Recursive Function, Coq

**Functional Description:** A Coq library for defining and reasoning about partial (co)recursive functions.

**URL:** <https://github.com/vladmgrusu/haddock>

**Contact:** Vlad Rusu

**Partners:** Université de Lille, CNRS, Université de Bucarest

#### 7.1.7 Polymalys

**Keywords:** Abstract interpretation, Polyhedra

**Functional Description:** Polymalys is a tool for static analysis of binary code. It discovers linear relations between data locations (i.e. memory locations as well as registers) of the code. The analysis relies on abstract interpretation using a polyhedra-based abstract domain. The current main application is Worst-Case Execution Time analysis in combination with the WSymb tool.

**Publications:** [hal-01939659](#), [hal-03794951](#)

**Contact:** Julien Forget

### 7.1.8 WSymb

**Name:** Symbolic Worst-case execution time computation

**Keywords:** WCET, Real time

**Functional Description:** WSymb is a WCET analysis tool. Its main specificity is that, instead of a constant WCET, it computes a WCET formula, where symbols (or parameters) can correspond to various values unknown at analysis time. The formula can later be instantiated, when parameter values are known.

**URL:** <https://gitlab.cristal.univ-lille.fr/otawa-plugins/WSymb>

**Publications:** [hal-01665076](#), [hal-04118213](#)

**Contact:** Julien Forget

### 7.1.9 Seplog

**Keywords:** Separation Logic, Coq

**Functional Description:** Seplog is a separation logic for GallinaC, a shallow embedding of an imperative, pointer-manipulating language in the Coq proof assistant.

**URL:** <https://gitlab.inria.fr/rusu/seplog>

**Contact:** Vlad Rusu

**Partners:** CNRS, Université de Lille

### 7.1.10 RevState

**Name:** The Reverse State Monad in Rocq

**Keywords:** Reverse causality, Monad, Rocq

**Scientific Description:** Based on Haddock (<https://inria.hal.science/hal-04360660>)

**Functional Description:** This is a Rocq library. It is built on our existing library for domain theory (Haddock). It includes definitions and proofs for monads in the category of CPOs, including the continuation monad transformer (to encode the reverse state monad transformer) as well as monads for identity, non-determinism and state enriched with value recursion operators.

**URL:** <https://gitlab.inria.fr/haddock/revstate>

**Contact:** Vlad Rusu

## 8 New results

### 8.1 CUTECat: Concolic Execution for Computational Law

Many legal computations, including the amount of tax owed by a citizen, whether they are eligible to social benefits, or the wages due to civil state servants, are specified by computational laws. Their application, however, is performed by expert computer programs intended to faithfully transcribe the law into computer code. Bugs in these programs can lead to dramatic societal impact, e.g., paying employees incorrect amounts, or not awarding benefits to families in need. To address this issue, we consider concolic unit testing, a combination of concrete execution with SMT-based symbolic execution, and propose CUTECat, a concolic execution tool targeting implementations of computational laws. Such laws typically follow a pattern where a base case is later refined by many exceptions in following law articles, a pattern that can be formally modeled using default logic. We show how to handle default logic inside a concolic execution tool, and implement our approach in the context of Catala, a recent domain-specific language tailored to implement computational

laws. We evaluate CUTEcAT on several programs, including the Catala implementation of the French housing benefits and Section 132 of the US tax code. We show that CUTEcAT can successfully generate hundreds of thousands of testcases covering all branches of these bodies of law. Through several heuristics, we improve CUTEcAT's scalability and usability, making the testcases understandable by lawyers and programmers alike. We believe CUTEcAT thus paves the way for the use of formal methods during legislative processes.

This work [20] has been presented at ESOP'25, and received a distinguished artifact award.

**Participants:** Pierre Goutagny, Raphael Monat.

## 8.2 Compositional Static Value Analysis for Higher-Order Numerical Programs

Static analyzers have been successfully developed to detect runtime errors in many languages. However, the automatic analysis of functional languages remains a challenge due to their recursive functions, recursive algebraic data types, and higher-order functions. Classic type systems provide compositional methods that are in general not precise enough to prove the absence of runtime errors such as assertion failures. At the other end of the spectrum, deductive methods are more expressive but may require user guidance to prove invariants. Our work describes a static value analysis by abstract interpretation for a higher-order pure functional language. This analysis provides a sound and automatic approach to discover invariants and prevent assertion and match failures. We have designed a compositional analysis: functions are analyzed only once, at their definition site, generating a summary of their behavior. The summaries can be viewed as input-output relations expressed with relational abstract domains. We present two new abstract domains. A first abstract domain summarizes recursive algebraic data types. A second abstract domain lifts existing disjunctive relational summaries to higher-order by formalizing them as domains able to abstract higher-order functions. Both abstractions are parameterized by the abstractions of basic types (strings, integers, . . . ). Thanks to this parametric nature, both domains can be combined, allowing the analysis of higher-order functions manipulating algebraic data types and, conversely, algebraic data types using functions as first-class values. We have implemented this analysis in the open-source MOPSA platform. Preliminary evaluation confirms the precision of our approach on a set of 40 handwritten toy programs as well as 20 programs from the state-of-the-art Salto analyzer benchmark.

This work [24] has been published at ECOOP 2025.

**Participants:** Raphael Monat.

## 8.3 Stop treating code like an afterthought: record, share and value it

From short scripts to vast simulations of Earth's climate, protein structures or even the cosmos, it is hard to imagine scientific research without software. Scientists use software code in myriad ways -to plan experiments; to record, organize, analyse, visualize and archive data; to control scientific instruments, and more. But software evolves. Most open-source software used in research is refined both iteratively and collectively, and has no published 'version of record'. Updates can target various versions and releases, meaning that each aspect of the software -the project as a whole, a specific version or a single file -can require a different way to refer to it. This creates confusion. And so software comes with a double bind: like data, it supports the findings of a study and should be preserved and published. Yet it should also remain available and supported, and possibly be improved, over time. Scholars, librarians, research institutions and funding agencies are wrestling with how to reconcile these two requirements.

This work has been published as a commentary in Nature [14].

**Participants:** Raphael Monat.

## 8.4 Easing Maintenance of Academic Static Analyzers

Academic research in static analysis produces software implementations. These implementations are time-consuming to develop and some need to be maintained in order to enable building further research upon the implementation. While necessary, these processes can be quickly challenging. This article documents the tools and techniques we have come up with to simplify the maintenance of Mopsa since 2017. Mopsa is a static analysis platform that aims at being sound. First, we describe an automated way to measure precision that does not require any baseline of true bugs obtained by manually inspecting the results. Further, it improves transparency of the analysis, and helps discovering regressions during continuous integration. Second, we have taken inspiration from standard tools observing the concrete execution of a program to design custom tools observing the abstract execution of the analyzed program itself, such as abstract debuggers and profilers. Finally, we report on some cases of automated testcase reduction.

This work has been published in a special issue of the STTT journal [15].

**Participants:** Raphael Monat.

## 8.5 Mopsa at the Software Verification Competition

Mopsa is a multilanguage static analysis platform relying on abstract interpretation. It is able to analyze C, Python, and programs mixing these two languages.

In Fall 2025, Raphaël Monat participated to the [Software Verification Competition \(SV-Comp\)](#) by submitting the Mopsa static analyzer he is co-developing. Mopsa earned a [silver medal in the SoftwareSystems category](#); this category aims at "representing verification tasks from real software systems". There were 22 competing tools.

This third participation is described in [21].

**Participants:** Raphaël Monat.

## 8.6 Implementing CNN on embedded real-time systems

Many Artificial Intelligence algorithms (e.g. Convolutional Neural Networks -CNNs) can be modeled as a collection of functions which communicate with each other according to a directed acyclic graph. The main goal of [16] is to optimize the execution of CNNs on real-time embedded systems based on a multicore architecture with scratchpad memory.

In a typical multicore platform, cores share a complex memory hierarchy with one or more levels of cache memories, leading to potential interference and contention on the shared communication buses. To reduce contention, it is possible to use architectures based on scratchpads, where every processor has a dedicated programmable fast memory to perform its local computations, and data is moved between the main memory and the local memories according to a timed schedule. We studied the problem of allocating CNN inference functions to processor cores, and scheduling the execution and memory communications. We proposed an Integer Linear Programming (ILP) model that accounts for both the cost of copying data to and from scratchpad memories and the parallel computation costs on the cores, with the goal of meeting real-time temporal constraints.

This work is part of the PhD thesis of Chiara Daini [25].

**Participants:** Giuseppe Lipari, Chiara Daini.

## 8.7 A Kleene Algebra with Tests for Union Bound Reasoning About Probabilistic Programs

Kleene algebras with tests (KAT) are an algebraic approach for reasoning on imperative programs with equational proofs, which has been shown by Kozen to be as expressive as propositional Hoare logic. We have proposed in [18] an extension of KAT for reasoning on imperative programs with probabilistic primitives. Our system can prove that a probabilistic program satisfies a given postcondition except with a given probability. We have proved that it is as expressive as a probabilistic variant of propositional Hoare logic from the literature (Union bound program logic). These results have been obtained as part of Leandro Gomes' postdoc and presented at the conference CSL2025 [18].

**Participants:** Patrick Baillot, Leandro Gomes.

## 8.8 BiGKAT: an algebraic framework for relational verification of probabilistic programs

The paper [19] is devoted to formal reasoning on relational properties of probabilistic imperative programs. Relational properties are properties which relate the execution of two programs (possibly the same one) on two initial memories. We aim at extending the algebraic approach of Kleene Algebras with Tests (KAT) to relational properties of probabilistic programs. For that we consider the approach of Guarded Kleene Algebras with Tests (GKAT), which can be used for representing probabilistic programs, and define a relational version of it, called Bi-guarded Kleene Algebras with Tests (BiGKAT) together with a semantics. We show that the setting of BiGKAT is expressive enough to encode a finitary version of probabilistic Relational Hoare Logic (pRHL) (without the While rule), a program logic that has been introduced in the literature for the verification of relational properties of probabilistic programs. We also discuss the additional expressivity brought by BiGKAT.

This work has been carried out as part of Leandro Gomes' postdoc and presented at the conference FoSSaCS 2025 [19].

**Participants:** Patrick Baillot, Leandro Gomes.

## 8.9 Session Types for the Concurrent Composition of Interactive Differential Privacy

Differential privacy (DP) is a statistical definition of privacy which ensures that the outcome of a computation by an analyst only depends in a negligible way on the presence of a single record in the dataset. This framework has been extended first to the interactive setting where the analyst can ask an adaptive sequence of queries, and then to the concurrent interactive setting where the adaptive queries can be performed concurrently to the same database. An important advantage of these frameworks is the presence of composition theorems, which enable data curators to combine multiple differentially private algorithms, resulting in a new algorithm that still satisfies differential privacy. Deriving composition theorems within the concurrent interactive framework, as well as for advanced notions of DP, is a complex task, for which some progress has been made recently. On the other hand a variety of tools have been proposed for certifying that some given algorithms are differentially private. Among them, the typing approach embodied by the Fuzz language consists in using a functional programming language endowed with a type system ensuring that well-typed programs can automatically be rendered differentially private. However this setting does not allow to represent concurrent interactive systems. In [23] we therefore propose to extend it by using a process calculus similar to the  $\pi$ -calculus as the language. This calculus is equipped with operational semantics that enable us to express the DP property as a form of approximate trace equivalence. Moreover, we introduce a type system in the form of session types and prove a soundness result stating that if a system of processes is well-typed, then it is differentially private.

This work is part of the ongoing thesis of Victor Sannier, advised by Patrick Baillot and co-advised by Marco Gaboardi (Boston University). It has been presented at the conference Computer Security Foundations (CSF 2025) [23].

**Participants:** Patrick Baillot, Victor Sannier.

### 8.10 A characterization of Basic Feasible Functionals through higher-order rewriting and tuple interpretations

The class of type-two basic feasible functionals (BFF2) is the analogue of FP (polynomial time functions) for type-2 functionals, that is, functionals that can take (first-order) functions as arguments. BFF2 can be defined through Oracle Turing Machines with running time bounded by second-order polynomials. On the other hand, higher-order term rewriting provides an elegant formalism for expressing higher-order computation. In [11] we address the problem of characterizing BFF2 by higher-order term rewriting. Various kinds of interpretations for first-order term rewriting have been introduced in the literature for proving termination and characterizing first-order complexity classes. In this paper, we consider a recently introduced notion of cost-size tuple interpretations for higher-order term rewriting and see second order rewriting as ways of computing type-2 functionals. We then prove that the class of functionals represented by higher-order terms admitting polynomially bounded cost-size interpretations exactly corresponds to BFF2.

This work has been published in the journal Logical Methods in Computer Science [11].

**Participants:** Patrick Baillot.

### 8.11 Dependent Coeffects for Local Sensitivity Analysis

Differential privacy is a formal definition of privacy that bounds the maximum acceptable information leakage when a query is performed on sensitive data. To ensure this property, a key technique involves bounding the query's sensitivity (how much input variations affect the output) and adding noise to the result according to this quantity. While prior work like the Fuzz type system focuses on global sensitivity, many useful queries have infinite global sensitivity, restricting the scope of such approaches. This limitation can be addressed by considering a more fine-grained measure: local sensitivity, which quantifies output change for inputs adjacent to a specific dataset. In [22] we introduce Local Fuzz, a type system with dependent coeffects designed to bound the local sensitivity of programs written in a simple functional language. We provide a denotational semantics for this system in the category of extended premetric spaces, leveraging the recently introduced construction of a dependently graded comonad. Finally, we illustrate how Local Fuzz can lead to better differential privacy guarantees than Fuzz, both for mechanisms that rely on global sensitivity and for those that leverage local sensitivity, such as the Propose-Test-Release framework.

This work is part of the ongoing thesis of Victor Sannier, advised by Patrick Baillot and co-advised by Marco Gaboardi (Boston University). It has been presented at the conference Principles of Programming Languages (POPL 2026) [22].

**Participants:** Patrick Baillot, Victor Sannier.

### 8.12 Towards determinism in PDL: relations and proof theory

Propositional dynamic logic (PDL) has been presented as a very powerful modal logic to reason about imperative programming languages, allowing in particular to express nondeterminism. It is known that deciding program equivalence in full PDL is EXPTIMEcomplete. In [12] we took a first step to optimize the complexity of dynamic logic equivalence: we have presented a fragment of PDL that allows only if-then-else

and while statements, equipped with a semantics over relational models and a sound axiomatisation. Although this fragment is less expressive than full PDL, it is sufficient to encode standard imperative programming constructions. We then presented a Natural Deduction system for this logic, proving its soundness and completeness with respect to the axiomatisation.

This work has been published in the Journal of Logic and Computation [12].

**Participants:** Leandro Gomes.

### 8.13 A Domain-Theoretic Framework for Composing Effectful Programs and Their Equations

Interaction trees are a uniform compositional framework for representing effectful and potentially non-terminating computations. In [26], David Nowak (2XS team) and myself have developed a domain-theoretic formalization of interaction trees in the Rocq prover. Domain theory enables us to handle (co)recursion naturally, without being held back by the frailty of Rocq’s builtin (co)recursion mechanisms. We show that coproducts of interaction trees exist, allowing the modular combination of effect signatures, and that the equations defining each individual effect are seamlessly integrated through this structure. This enables modular equational reasoning for programs combining multiple effects. To illustrate the expressiveness of our approach, we verify the correctness of a stateful probabilistic computation and prove the equivalence between two programs whose termination depends on a mathematical conjecture.

**Participant:** Vlad Rusu.

### 8.14 Pleasant Imperative Program Proofs with GallinaC

In [17], we present GallinaC, a shallow embedding of a Turing-complete imperative language directly inside the functional programming language of the Rocq proof assistant, Gallina. In particular, it features a truly generic and unbounded while loop. Having a functional core means proofs about GallinaC programs may use the same tactics as proofs about pure functional ones. Compilation from GallinaC to binary is possible through the CompCert certified compiler. A chain of forward simulations guarantees that compilation passes preserve semantics. Work on GallinaC is still under progress, but we present first promising results. A prototype implementation has shown the viability of GallinaC with the correctness proof of a list reversal procedure for linked-lists of unknown size. We currently focus on the forward simulation between the GallinaC intermediate representation (IR) and Cminor, the entry language of the CompCert back-end. The GallinaC sources are available [here](#).

**Participant:** Vlad Rusu.

## 9 Bilateral contracts and grants with industry

### 9.1 Bilateral Grants with Industry

Raphael Monat has been awarded an [Amazon Research Award](#) on Resource-Aware Conservative Static Analysis.

**Participant:** Raphael Monat.

## 10 Partnerships and cooperations

### 10.1 International initiatives

#### 10.1.1 Participation in other International Programs

International Emerging Action (IEA) CNRS: Formal methods for probabilistic programs and differential privacy (FMPDP)

This is an exchange project with Boston University (MA, US) funded by CNRS for 2025 and 2026 (11k€), led by Patrick Baillot, on the topic of formal methods for differential privacy, including type systems, program logics and algebraic methods. It will fund three visits of members of the team to Boston University and three visits of researchers or PhD student of this University to Lille.

In 2025 Victor Sannier visited Boston University for 9 weeks as part of this project to work with his PhD co-adviser, Marco Gaboardi.

### 10.2 International research visitors

#### 10.2.1 Visits of international scientists

##### Other international visits to the team

###### **Pietro Ferrara**

**Status** Associate professor

**Institution of origin** University of Ca'Foscari in Venice

**Country** Italy

**Dates** 20/03/2025

**Context of the visit** Prospective

###### **APR team**

**Institution of origin:** LIP6 laboratory in Paris

**Country:** France

**Dates:** 2-3 juin 2025

**Context of the visit:** The internal Annual Symposium of APR team was held in Lille, at the premises of the CRISAL Laboratory. The members of the SYCOMORES team were invited to participate with scientific presentations and useful exchanges.

###### **Renato Mancuso**

**Status** associate professor

**Institution of origin:** Boston University

**Country:** USA

**Dates:** 30/06/2025-07/07/2025

**Context of the visit:** Prospective

**Mobility program/type of mobility:** IUF Giuseppe Lipari

**Shin-Ya Katsumata**

**Status** Professor

**Institution of origin:** Kyoto Sangyo University

**Country:** Japan

**Dates:** 10/02/2025-11/02/2025

**Context of the visit:** Prospective

**Ugo Dal Lago**

**Status** Professor

**Institution of origin:** Università di Bologna

**Country:** Italie

**Dates:** 10/03/2025-11/03/2025

**Context of the visit:** ANR HOPR meeting

**Clément Aubert**

**Status** Associate Professor

**Institution of origin:** Augusta University

**Country:** USA

**Dates:** 19/06/2025

**Context of the visit:** Prospective

**10.2.2 Visits to international teams****Research stays abroad****Victor Sannier**

**Visited institution:** Boston University

**Country:** USA

**Dates:** 20/4/2025-10/7/2025

**Context of the visit:** IEA CNRS FMPDP project, to work with his PhD co-adviser, Marco Gaboardi

**Mobility program/type of mobility:** research stay

**10.3 National initiatives****10.3.1 Inria Exploratory Action AVoCat**

Raphaël Monat is co-PI of an Inria Exploratory Action called AVoCat, aiming at exploring Automatic Verification of Catala programs. The project is shared with Aymeric Fromherz at Inria Paris.

**Participants:** Raphaël Monat.

### 10.3.2 ANR JCJC RAISIN: Resource-Aware Conservative Static Analysis

Raphaël Monat has been granted an ANR JCJC, starting January 2025. Additional members: Julien Forget, and Sophie Cerf (researcher at Inria Lille). This grant is funded by the National Research Agency, through a call open to early-career permanent researchers. It will fund a PhD student and a postdoc to work on Resource-Aware Conservative Static Analysis.

**Participants:** Raphaël Monat, Julien Forget.

### 10.3.3 ANR PRC HOPR: Higher-Order Probabilistic and Resource-aware Reasoning

(01/01/2025-31/12/2029). Coordinator: Patrick Baillot. Inria Lille/CRIStAL; Inria Paris; Inria Sophia-Antipolis; IRISA Rennes. Total funding: 479k€. This project deals with logical frameworks for reasoning on cryptographic constructions and on privacy protection. It aims at improving proof-assistants for these purposes by extending logical frameworks with the possibility of handling higher-order computation, probabilistic reasoning and complexity-bounded computation. These progresses will be used in cryptography to improve the proof-assistants Squirrel and EasyCrypt. In privacy the project will develop new program logics for reasoning on differential privacy and combining program logic with typing.

**Participants:** Patrick Baillot, Leandro Gomes, Victor Sannier, Artur Szafarczyk.

## 10.4 Public policy support

Raphaël Monat, Pierre Goutagny participate to AEx AVoCat, which aims at bringing state-of-the-art verification techniques within the Catala language, currently trialed to be used in some public administrations.

## 11 Dissemination

### 11.1 Promoting scientific activities

#### 11.1.1 Scientific events: selection

**Member of the conference program committees** Raphael Monat, SAS 2025

Patrick Baillot, CSL 2026

Giuseppe Lipari, ERTS 2026

**Steering committee** Patrick Baillot has been steering committee chair of the FSCD conference since 2024.

#### 11.1.2 Journal

**Member of the editorial boards** Giuseppe Lipari is member of the editorial board of the Real-Time Systems Journal (RTSJ) and of the Journal of Systems Architectures (JSA).

**Reviewer - reviewing activities** Julien Forget, JSA, JSS.

#### 11.1.3 Invited talks

Julien Forget, "Model-based end-to-end analysis of synchronous task chains", [Workshop on End-to-End Analysis and Optimization of Task Chains](#), Turin, Nov 3-4.

#### 11.1.4 Leadership within the scientific community

Julien Forget is the head of the challenge **PAE** and of the group **PARTS** of the new CNRS GDR **Scilog**.

#### 11.1.5 Scientific expertise

Raphael Monat attended two invite-only Dagstuhl seminars "**Testing Program Analyzers and Verifiers**" and "**Sound Static Program Analysis in Modern Software Engineering**".

#### 11.1.6 Research administration

Patrick Baillot is a scientific advisor (Délégué scientifique) at CNRS Sciences Informatiques, for section 02, since 9/2022 (40% of his working time).

### 11.2 Teaching - Supervision - Juries - Educational and pedagogical outreach

#### 11.2.1 Supervision

Julien Forget supervised the M1 internship of Rémi Boursault, on "Scratchpad memory management for real-time applications".

Raphael Monat and Julien Forget supervised a master internship of **Tom Goalard** around automated divergence pinpointing in static analyzers.

Raphael Monat supervised an M2 project of Aymane Ismail around a web interface for Mopsa.

Giuseppe Lipari supervised the internship of Justin Bauer, a master student of the École Centrale, on the implementation of a EDF scheduler in FreeRTOS; of Kevan Touron, a 4th year student of Polytech Lille, on hierarchical real-time scheduling in FreeRTOS; of Louis Becue, a L3 student of the Licence en Informatique, on a parser for the Sched\_analyzer library; of Adil Zouar, a L3 student of the licence en Informatique, on the implementation of a web server on a Raspberry Pico; of Gabriel Warde, a master student in Data Science on a ML model for WCET estimation on GPUs.

#### 11.2.2 Juries

Raphael Monat was part of the hiring committee for Inria Researchers in Lille.

Julien Forget was part of the hiring committee for an Associate Professor position at the University of Lille (252419).

### 11.3 Popularization

#### 11.3.1 Specific official responsibilities in science outreach structures

Raphael Monat co-organized, with Clémentine Maurice, a visit to the CRISAL and Inria laboratories for L3 and M1 students at ENS Rennes (January 2025). This **three-day visit** enabled 44 students to interact with around 40 colleagues (postdocs, research engineers, C and E/C) attached to CRISAL, Inria and IRCICA.

#### 11.3.2 Participation in Live events

Raphael Monat was invited to give a talk around static program analysis at the InCyber International CyberSecurity Forum 2025.

## 12 Scientific production

### 12.1 Major publications

- [1] C. Ballabriga, J. Forget, S. Grebant and G. Lipari. 'New challenges in adaptive real-time systems with parametric WCET'. In: RTSOPS 2023 - 12th International Real-Time Scheduling Open Problems Seminar. Vienne, Austria, 11th July 2023. URL: <https://hal.science/hal-04197411>.

- [2] C. Ballabriga, J. Forget and J. Ruiz. ‘Relational abstract interpretation of arrays in assembly code’. In: *Formal Methods in System Design* (2nd Oct. 2022). DOI: [10.1007/s10703-022-00399-3](https://doi.org/10.1007/s10703-022-00399-3). URL: <https://hal.inria.fr/hal-03794951>.
- [3] F. Bouquillon, S. Niar and G. Lipari. ‘Reducing the fault vulnerability of hard real-time systems’. In: *Journal of Systems Architecture* 133 (Dec. 2022), p. 102758. DOI: [10.1016/j.sysarc.2022.102758](https://doi.org/10.1016/j.sysarc.2022.102758). URL: <https://hal.archives-ouvertes.fr/hal-03842393>.
- [4] H. Cheval, D. Nowak and V. Rusu. ‘Formal Definitions and Proofs for Partial (Co)Recursive Functions’. In: *Journal of Logic and Algebraic Methods in Programming* 141 (Oct. 2024), p. 27. DOI: [10.1016/j.jlamp.2024.100999](https://doi.org/10.1016/j.jlamp.2024.100999). URL: <https://inria.hal.science/hal-04360660>.
- [5] S. Grebant, C. Ballabriga, J. Forget and G. Lipari. ‘WCET analysis with procedure arguments as parameters’. In: *RTNS 2023: The 31st International Conference on Real-Time Networks and Systems*. Dortmund, Germany: ACM, 2023, pp. 11–22. DOI: [10.1145/3575757.3593655](https://doi.org/10.1145/3575757.3593655). URL: <https://hal.science/hal-04118213>.
- [6] R. Monat, A. Fromherz and D. Merigoux. ‘Formalizing Date Arithmetic and Statically Detecting Ambiguities for the Law’. In: *Lecture Notes in Computer Science*. ESOP 2024 - 33rd European Symposium on Programming. Vol. 14577. Lecture Notes in Computer Science. Luxembourg City, Luxembourg: Springer Nature Switzerland, 5th Apr. 2024, pp. 421–450. DOI: [10.1007/978-3-031-57267-8\\_16](https://doi.org/10.1007/978-3-031-57267-8_16). URL: <https://hal.science/hal-04536403>.
- [7] R. Monat, A. Ouadjaout and A. Miné. ‘Mopsa-C: Modular Domains and Relational Abstract Interpretation for C Programs (Competition Contribution)’. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2023)*. Vol. 13994. Lecture Notes in Computer Science. Paris, France: Springer, Cham, 20th Apr. 2023, pp. 565–570. DOI: [10.1007/978-3-031-30820-8\\_37](https://doi.org/10.1007/978-3-031-30820-8_37). URL: <https://inria.hal.science/hal-04077678>.
- [8] D. Nowak and V. Rusu. ‘While Loops in Coq’. In: *Electronic Proceedings in Theoretical Computer Science*. 7th Symposium on Working Formal Methods (FROM 2023). Vol. 389. Bucarest, Romania, 22nd Sept. 2023, pp. 96–109. DOI: [10.4204/eptcs.389.8](https://doi.org/10.4204/eptcs.389.8). URL: <https://inria.hal.science/hal-04254872>.
- [9] F. Vanhems, V. Rusu, D. Nowak and G. Grimaud. ‘A Formal Correctness Proof for an EDF Scheduler Implementation’. In: *Proc. 28th IEEE Real-Time and Embedded Technology and Applications Symposium*. RTAS 2022: 28th IEEE Real-Time and Embedded Technology and Applications Symposium. Milan, Italy, 4th May 2022. DOI: [10.1109/RTAS54340.2022.00030](https://doi.org/10.1109/RTAS54340.2022.00030). URL: <https://hal.inria.fr/hal-03671598>.
- [10] J. Wunder, A. A. D. Amorim, P. Baillot and M. Gaboardi. ‘Bunched Fuzz: Sensitivity for Vector Metrics’. In: *ESOP 2023 - European Symposium on Programming*. Proceedings of ESOP 2023 (European Symposium on Programming). Paris, France: Springer, 24th Apr. 2023. DOI: [10.48550/arXiv.2202.01901](https://doi.org/10.48550/arXiv.2202.01901). URL: <https://hal.science/hal-03870966>.

## 12.2 Publications of the year

### International journals

- [11] P. Baillot, U. Dal Lago, C. Kop and D. Vale. ‘A Characterization of Basic Feasible Functionals Through Higher-Order Rewriting and Tuple Interpretations’. In: *Logical Methods in Computer Science* (2025). URL: <https://hal.science/hal-05343191>. In press (cit. on p. 18).
- [12] M. Benevides, L. Gomes and B. Lopes. ‘Towards determinism in PDL: relations and proof theory’. In: *Journal of Logic and Computation* 35.5 (1st July 2025). DOI: [10.1093/logcom/exae022](https://doi.org/10.1093/logcom/exae022). URL: <https://hal.science/hal-05023810> (cit. on pp. 18, 19).
- [13] C. Daini, M. Laura Delle Monache, P. Goatin and A. Ferrara. ‘Traffic Control via Fleets of Connected and Automated Vehicles’. In: *IEEE Transactions on Intelligent Transportation Systems* 26.2 (2025), p. 1. DOI: [10.1109/TITS.2024.3506703](https://doi.org/10.1109/TITS.2024.3506703). URL: <https://hal.science/hal-04366870>.

- [14] R. Di Cosmo, S. Granger, K. Hinsén, N. Jullien, D. Le Berre, V. Louvet, C. Maumet, C. Maurice, R. Monat and N. Rougier. ‘Stop treating code like an afterthought: record, share and value it: Scientists, research institutions, funders, libraries and publishers must all improve software practices’. In: *Nature* 646.8084 (7th Oct. 2025), pp. 284–286. DOI: [10.1038/d41586-025-03196-0](https://doi.org/10.1038/d41586-025-03196-0). URL: <https://inria.hal.science/hal-05306427> (cit. on p. 15).
- [15] R. Monat, A. Ouadjaout and A. Miné. ‘Easing Maintenance of Academic Static Analyzers’. In: *International Journal on Software Tools for Technology Transfer* CSV 2024 Special Issue (14th Jan. 2025). DOI: [10.1007/s10009-024-00770-1](https://doi.org/10.1007/s10009-024-00770-1). URL: <https://inria.hal.science/hal-04652657> (cit. on p. 16).

#### International peer-reviewed conferences

- [16] C. Daini, G. Lipari, H. Zahaf and P.-E. Hladik. ‘Optimizing CNN Inference on Multicore Scratchpad Architectures’. In: *Proceedings IEEE ISORC 2025*. IEEE 28th International Symposium on Real-Time Distributed Computing. Toulouse, France, 26th May 2025. URL: <https://hal.science/hal-05107897> (cit. on p. 16).
- [17] F. Fort, D. Nowak and V. Rusu. ‘Pleasant Imperative Program Proofs with GallinaC’. In: *Electronic Proceedings in Theoretical Computer Science (EPTCS)*. Working Formal Methods Symposium (FROM 2025). Vol. 427. Iași, Romania, 16th Sept. 2025, pp. 24–32. DOI: [10.4204/EPTCS.427.2](https://doi.org/10.4204/EPTCS.427.2). URL: <https://hal.science/hal-05405959> (cit. on p. 19).
- [18] L. Gomes, P. Baillot and M. Gaboardi. ‘A Kleene algebra with tests for union bound reasoning about probabilistic programs’. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. 33rd EACSL Annual Conference on Computer Science Logic (CSL 2025). Vol. 326. LIPIcs. Amsterdam, Netherlands: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 3rd Feb. 2025, 35:1–35:19. DOI: [10.4230/LIPIcs.CSL.2025.35](https://doi.org/10.4230/LIPIcs.CSL.2025.35). URL: <https://hal.science/hal-04196675> (cit. on p. 17).
- [19] L. Gomes, P. Baillot and M. Gaboardi. ‘BiGKAT: an algebraic framework for relational verification of probabilistic programs’. In: *LNCS*. Foundations of Software Science and Computation Structures - 28th International Conference, FoSSaCS 2025. Hamilton, Ontario, Canada, Canada, 5th May 2025. URL: <https://hal.science/hal-04017128> (cit. on p. 17).
- [20] P. Goutagny, A. Fromherz and R. Monat. ‘CUTECat: Concolic Execution for Computational Law’. In: *ESOP 2025 - 34th European Symposium on Programming*. Hamilton, ON, Canada, 3rd May 2025. URL: <https://inria.hal.science/hal-04907935> (cit. on p. 15).
- [21] R. Monat, A. Ouadjaout and A. Miné. ‘Mopsa-C with Trace Partitioning and Autosuggestions (Competition Contribution)’. In: *Tools and Algorithms for the Construction and Analysis of Systems*. TACAS 2025. Hamilton, Ontario, Canada, 3rd May 2025. URL: <https://inria.hal.science/hal-04968769> (cit. on p. 16).
- [22] V. Sannier and P. Baillot. ‘Dependent Coeffects for Local Sensitivity Analysis’. In: *Proceedings of POPL 2026*. ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2026). Rennes, France, 14th Jan. 2026. DOI: [10.1145/3776670](https://doi.org/10.1145/3776670). URL: <https://hal.science/hal-05191122> (cit. on p. 18).
- [23] V. Sannier, P. Baillot and M. Gaboardi. ‘Session Types for the Concurrent Composition of Interactive Differential Privacy’. In: *CSF 2025 – 38th IEEE Computer Security Foundations Symposium*. Santa Cruz, CA, United States, 31st Jan. 2025. URL: <https://hal.science/hal-04719333> (cit. on pp. 17, 18).
- [24] M. Valnet, R. Monat and A. Miné. ‘Compositional Static Value Analysis for Higher-Order Numerical Programs’. In: *Leibniz International Proceedings in Informatics*. 39th European Conference on Object-Oriented Programming (ECOOP 2025). Vol. 333. Bergen, Norway: Dagstuhl Publishing, 30th June 2025, p. 15. DOI: [10.4230/LIPIcs.ECOOP.2025.15](https://doi.org/10.4230/LIPIcs.ECOOP.2025.15). URL: <https://hal.science/hal-05047369> (cit. on p. 15).

### Doctoral dissertations and habilitation theses

- [25] C. Daini. ‘Predictable CNN Inference and Real-Time Scheduling on Multicore Embedded Platforms’. Université de Lille, 18th Nov. 2025. URL: <https://theses.hal.science/tel-05536127> (cit. on p. 16).

### Reports & preprints

- [26] D. Nowak and V. Rusu. *A Domain-Theoretic Framework for Composing Effectful Programs and Their Equations*. Nov. 2025. URL: <https://inria.hal.science/hal-05387384> (cit. on p. 19).

## 12.3 Cited publications

- [27] L. Abeni, L. Palopoli, C. Scordino and G. Lipari. ‘Resource Reservations for General Purpose Applications’. In: *IEEE Trans. Ind. Informatics* 5.1 (2009), pp. 12–21. DOI: [10.1109/TII.2009.2013633](https://doi.org/10.1109/TII.2009.2013633). URL: <https://doi.org/10.1109/TII.2009.2013633> (cit. on p. 10).
- [28] ANSYS. *SCADE Suite*. <http://www.esterel-technologies.com/products/scade-suite/>. 2018 (cit. on p. 6).
- [29] C. Ballabriga, J. Forget, L. Gonnord, G. Lipari and J. Ruiz. ‘Static Analysis Of Binary Code With Memory Indirections Using Polyhedra’. In: *International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI’19)*. Lisbon, Portugal, Jan. 2019. URL: <https://hal.archives-ouvertes.fr/hal-01939659> (cit. on pp. 8, 9).
- [30] C. Ballabriga, J. Forget and G. Lipari. ‘Symbolic WCET Computation’. In: *ACM Transactions on Embedded Computing Systems (TECS)* 17.2 (Dec. 2017), pp. 1–26. DOI: [10.1145/3147413](https://hal.archives-ouvertes.fr/hal-01665076). URL: <https://hal.archives-ouvertes.fr/hal-01665076> (cit. on pp. 8, 9).
- [31] S. Baruah. ‘Resource-Efficient Execution of Conditional Parallel Real-Time Tasks’. In: *Euro-Par 2018: Parallel Processing*. Cham: Springer International Publishing, 2018, pp. 218–231 (cit. on p. 7).
- [32] E. Bini and G. Buttazzo. ‘The space of EDF deadlines: the exact region and a convex approximation’. In: *Real-Time Systems* 41.1 (2009), pp. 27–51 (cit. on p. 8).
- [33] E. Bini, M. Di Natale and G. Buttazzo. ‘Sensitivity analysis for fixed-priority real-time systems’. In: *Real-Time Systems* 39.1-3 (2008), pp. 5–30 (cit. on p. 8).
- [34] T. Bourke, L. Brun, P.-É. Dagand, X. Leroy, M. Pouzet and L. Rieg. ‘A formally verified compiler for Lustre’. In: *ACM SIGPLAN Notices*. Vol. 52. 6. ACM. 2017, pp. 586–601 (cit. on p. 10).
- [35] R. Boutonnet and N. Halbwachs. ‘Disjunctive relational abstract interpretation for interprocedural program analysis’. In: *International Conference on Verification, Model Checking, and Abstract Interpretation*. Springer. 2019, pp. 136–159 (cit. on p. 9).
- [36] G. Durrieu, M. Faugere, S. Girbal, D. G. Pérez, C. Pagetti and W. Puffitsch. ‘Predictable flight management system implementation on a multicore processor’. In: *Embedded Real Time Software (ERTS’14)*. 2014 (cit. on p. 8).
- [37] A. Hamann, D. Dasari, S. Kramer, M. Pressler and F. Wurst. ‘Communication Centric Design in Complex Automotive Embedded Systems’. In: *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Dubrovnik, Croatia, 2017 (cit. on p. 8).
- [38] X. Leroy. *The CompCert C verified compiler*. <http://compcert.inria.fr>. 2018 (cit. on p. 10).
- [39] G. Lipari and S. K. Baruah. ‘Greedy reclamation of unused bandwidth in constant-bandwidth servers’. In: *12th Euromicro Conference on Real-Time Systems (ECRTS 2000), 19-21 June 2000, Stockholm, Sweden, Proceedings*. IEEE Computer Society, 2000, pp. 193–200. DOI: [10.1109/EMRTS.2000.854007](https://doi.org/10.1109/EMRTS.2000.854007). URL: <https://doi.org/10.1109/EMRTS.2000.854007> (cit. on p. 10).
- [40] G. Lipari and E. Bini. ‘Resource Partitioning among Real-Time Applications’. In: *Proc. 15th Euromicro Conf. Real-Time Systems*. IEEE Computer Society, 2003, pp. 151–158. DOI: [10.1109/EMRTS.2003.1212738](https://doi.org/10.1109/EMRTS.2003.1212738) (cit. on p. 10).

- [41] MathWorks. *Simulink*. <https://www.mathworks.com/products/simulink.html>. 2018 (cit. on p. 6).
- [42] C. Pagetti, J. Forget, F. Boniol, M. Cordovilla and D. Lesens. ‘Multi-task implementation of multi-periodic synchronous programs’. In: *Discrete Event Dynamic Systems* 21.3 (2011), pp. 307–338 (cit. on pp. 6, 7).
- [43] C. Pagetti, J. Forget, H. Falk, D. Oehlert and A. Luppold. ‘Automated generation of time-predictable executables on multi-core’. In: *RTNS 2018*. Oct. 2018 (cit. on p. 8).
- [44] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo and R. Kegley. ‘A predictable execution model for COTS-based embedded systems’. In: *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. 2011 (cit. on p. 8).
- [45] P. Raymond. ‘A general approach for expressing infeasibility in implicit path enumeration technique’. In: *2014 International Conference on Embedded Software (EMSOFT)*. IEEE. 2014, pp. 1–9 (cit. on p. 8).
- [46] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu and C. D. Gill. ‘Parallel Real-Time Scheduling of DAGs’. In: *IEEE Transactions on Parallel and Distributed Systems* 25.12 (Dec. 2014), pp. 3242–3252. doi: [10.1109/TPDS.2013.2297919](https://doi.org/10.1109/TPDS.2013.2297919) (cit. on p. 7).
- [47] *The Coq proof assistant reference manual*. <http://coq.inria.fr>. 2021 (cit. on p. 7).
- [48] *The Pip protokernel*. <http://pip.univ-lille1.fr/>. 2018 (cit. on p. 9).