

# 2025 Activity Report

RESEARCH CENTRE: Inria Centre at Université Côte d'Azur  
IN PARTNERSHIP WITH: Université de Bologne (Italie)

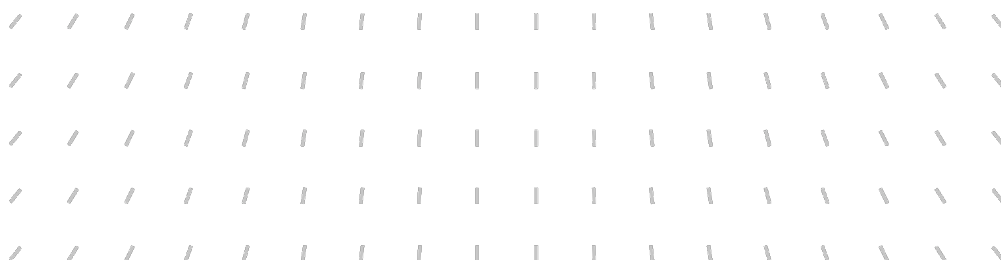
---

Project-Team

## OLAS

Operational, Logical, and Algebraic foundations for  
Software systems

---



## **Project-Team OLAS**

*Creation of the Project-Team: 2023 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

- A1.3. – Distributed Systems
- A2. – Software sciences
  - A2.1. – Programming Languages
    - A2.1.1. – Semantics of programming languages
    - A2.1.4. – Functional programming
    - A2.1.6. – Concurrent programming
    - A2.1.7. – Distributed programming
  - A4.5. – Formal method for verification, reliability, certification
    - A4.5.1. – Static analysis
    - A4.5.3. – Program proof
- A7. – Theory of computation
  - A7.2. – Logic in Computer Science

### Other research topics and application domains

- B6.1. – Software industry
- B6.3. – Network functions
- B6.4. – Internet of things
- B9.5.1. – Computer science

## Contents

<b>Project-Team OLAS</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>6</b>
3.1 Models	6
3.2 Behavioral equivalences and metrics	6
3.3 Types	6
3.4 Proof theory	7
<b>4 Application domains</b>	<b>7</b>
<b>5 Social and environmental responsibility</b>	<b>7</b>
<b>6 Highlights of the year</b>	<b>7</b>
6.1 Awards	7
<b>7 Latest software developments, platforms, open data</b>	<b>8</b>
7.1 Latest software developments	8
7.1.1 Corinne	8
7.1.2 CauDEr	8
7.1.3 QuRA	9
7.1.4 Ranflood	9
7.1.5 APP	9
7.1.6 JOLIE	10
7.1.7 FunLess	11
7.1.8 JoT	11
7.1.9 Choral	12
7.1.10 ChorEr	12
<b>8 New results</b>	<b>13</b>
8.1 Service Oriented Computing and Cloud Computing	13
8.1.1 Distributed systems and choreographies	13
8.1.2 Serverless computing	13
8.1.3 Microservices	14
8.1.4 Infrastructure as code for security	14
8.2 Reversible computing	15
8.3 Quantitative analysis	15
8.3.1 Deterministic program analysis	15
8.3.2 Probabilistic program analysis	16
8.3.3 Quantum program analysis	16
8.3.4 Applications to cryptography and security	17
8.4 Qualitative semantics	17
8.4.1 Model checking higher-order functions	17
8.4.2 Asynchronous session types	18
8.4.3 Coinductive proof techniques	18
8.4.4 Unifying semantics and comparisons among models	18
8.5 Miscellaneous	19
8.5.1 Cybersecurity: ransomware defense	19
8.5.2 Educational technology and social analysis	19
8.5.3 Programming language implementation	19

<b>9</b>	<b>Bilateral contracts and grants with industry</b>	<b>20</b>
9.1	Bilateral contracts with industry . . . . .	20
9.2	Bilateral Grants with Industry . . . . .	20
<b>10</b>	<b>Partnerships and cooperations</b>	<b>20</b>
10.1	International research visitors . . . . .	20
10.1.1	Visits of international scientists . . . . .	20
10.2	European initiatives . . . . .	22
10.2.1	Horizon Europe . . . . .	22
10.3	National initiatives . . . . .	22
10.3.1	HOPR . . . . .	22
10.3.2	SmartCloud . . . . .	22
<b>11</b>	<b>Dissemination</b>	<b>22</b>
11.1	Promoting scientific activities . . . . .	22
11.1.1	Scientific events: organization . . . . .	22
11.1.2	Journal . . . . .	24
11.1.3	Invited talks . . . . .	24
11.1.4	Leadership within the scientific community . . . . .	24
11.1.5	Research administration . . . . .	24
11.2	Teaching - Supervision - Juries - Educational and pedagogical outreach . . . . .	24
11.2.1	Juries . . . . .	25
<b>12</b>	<b>Scientific production</b>	<b>25</b>
12.1	Publications of the year . . . . .	25
12.2	Cited publications . . . . .	29

# 1 Team members, visitors, external collaborators

## Research Scientist

- Martin Avanzini [INRIA, Researcher]

## Faculty Members

- Davide Sangiorgi [Team leader, UNIV BOLOGNE, Professor, HDR]
- Ugo Dal Lago [UNIV BOLOGNE, Professor]
- Saverio Giallorenzo [UNIV BOLOGNE, Assistant Professor]
- Ivan Lanese [UNIV BOLOGNE, Associate Professor]
- Gianluigi Zavattaro [UNIV BOLOGNE, Professor]

## Post-Doctoral Fellows

- Vikraman Choudhury [UNIV. BOLOGNE, until Aug 2025]
- Zeinab Galal [UNIV BOLOGNE, Post-Doctoral Fellow, until Jun 2025]

## PhD Students

- Andrea Colledan [UNIV BOLOGNE]
- Giuseppe De Palma [UNIV BOLOGNE]
- Mehdi Golpayegani [INRIA, from Nov 2025]
- Matteo Trentin [UNIV BOLOGNE]

## Administrative Assistant

- Christine Claux [INRIA]

## External Collaborators

- Maurizio Gabbrielli [UNIV. BOLOGNE, Professor]
- Daniel Hirschhoff [ENS Lyon]
- Simone Martini [UNIV BOLOGNE, Professor]

# 2 Overall objectives

Software is more and more transforming our daily lives. However, it is also becoming more and more complex. This raises tremendous challenges when it comes to ensuring that software works correctly and efficiently. Correctness is about ensuring that a program satisfies expected requirements. Efficiency is about reducing the resource usage of the runs of a program without affecting the overall behavior.

In OLAS, we study models and techniques for reasoning about the correctness and efficiency of modern software systems. We focus on languages and formalisms that are higher-order, in that they allow, either syntactically or semantically, the representation of general functions, including functions that take other functions as arguments. A distinctive feature of higher-order languages is open-endedness: the visibility that a term has of its environment may change over time, because the interaction of the term with its environment will affect the future capabilities of interactions. Another feature is abstraction, both on data and on behavior.

Higher-order constructs are important in modern high-level programming languages. For instance, software is typically open-ended because it is connected to the Internet; and abstraction is important for writing concise code and for enhancing modularity. Indeed, modern mainstream programming languages normally include higher-order constructs.

While the higher-order languages that we investigate may present even strikingly different features (e.g., constructs for concurrency, distribution, probability), we follow a common and unifying methodology. The first and most important aspect of this methodology is given by the kind of techniques employed, namely *operational* techniques—whereby the meaning of systems is expressed in terms of the dynamics of programs, i.e., how the programs evolve in a stepwise fashion—complemented with mathematical *logic* (e.g., type systems) and *algebraic* reasoning. The other major unifying aspect of our methodology has to do with *models*: we first experiment on models, aiming at identifying the ones with solid logical and algebraic roots, and then move up towards languages and software systems.

We believe that such a methodology is well adapted to reason compositionally and under the ‘open world’ assumption of higher-order languages. Compositionality is a central principle in our approaches because of the complexity of software systems.

## 3 Research program

### 3.1 Models

The topic and objective of OLAS is the development of semantics, concepts, techniques, and possibly also linguistic constructs and tools, for specifying and reasoning about higher-order software systems. Fundamental to these activities is *modeling*. Therefore designing, developing and studying appropriate computational models is a central activity in OLAS. The models are used to formalize and verify important computational properties of the systems, as well as to propose new linguistic constructs. The models we study have their roots in *algebra* and *logic*, following the tradition stemming from the  $\lambda$ -calculus and process calculi. As such, they well address compositionality—a central property in our approach to problems. The use of foundational models inevitably leads to opportunities for developing the foundational models themselves, with particular interest for issues of expressiveness and for the transplant of concepts or techniques from a model to another one.

### 3.2 Behavioral equivalences and metrics

Behavioral equivalences equate processes that “behave in the same way” in all contexts, that is, under all environments in which they could possibly be used. Equivalence is particularly useful as a tool for justifying program transformations, (“we can validly replace  $P$  by  $Q$  because they have the same behavior in all contexts”), performed either by programmers during system development or by optimizing phases of compilers. Such transformations require equivalence relations to be congruences, i.e. preserved by all operators of the underlying languages, a property that is also fundamental to perform compositional reasoning on complex systems.

A useful refinement of behavioral equivalence is represented by metrics. Metrics allow one to be more informative about the comparison between two systems, so to reveal “how different” two programs are. Two systems may not be exactly behaviorally equal, but they may still be “similar”, and the difference between them may be acceptable. For instance, a program may approximate another one by performing less precise real number calculations but may thus consume less energy. The use of metrics, in place of ordinary behavioral equivalence, is particularly relevant for languages that capture quantitative aspects such as probabilities.

### 3.3 Types

The reason why we enhance our operational and algebraic techniques with logical formalisms such as types is that types appear to offer a good trade-off between expressiveness and amenability to efficient verification and validation techniques. By showing that a program has a certain type one may be capable of guaranteeing certain desirable behavioral properties, such as termination (the property that a program will not run forever).

Types may also provide formal descriptions of the interaction protocols (the dialogues) among the components of system.

### 3.4 Proof theory

Proof theory is a branch of mathematical logic which has been proved to have many applications to computer science. One paradigmatic example is Girard's Linear Logic: defined more than thirty years ago, it has been applied to several distinct domains in computer science, from programming language theory to security, from automatic theorem proving to computational complexity. From the perspective of OLAS, Linear Logic offers some elegant tools for resource-control, which we often use also as a mean for enhancing type systems. We use such techniques for expressing bounds on different kinds of resources, both spatial (having to do with the memory needs of a program) and temporal. The bounds may also formalize different kinds of analysis, including a worse case complexity, an average case complexity, as well as forms of tail probability (informally, measuring how far a certain complexity can spread from its mean, thus indicating the likelihood of occurrence of certain behavioral anomalies in a system).

## 4 Application domains

OLAS targets models and techniques for reasoning about higher-order software systems. These systems are found in different application domains. In OLAS we are particularly interested in the following ones:

- *Concurrent and distributed systems*, including *service-oriented systems* (e.g., *microservices*, and *serverless architectures*);
- *Bayesian languages* (roughly, probabilistic programs that feature, besides sampling, also operations for conditioning via observations or scoring);
- *Quantum programming* (where resource control is of paramount importance);
- *Cryptographic systems*, in particular cryptographic proof assistants.

While these areas are very wide in scope, the aspects that are of interest in OLAS represent a relatively small part of the areas themselves. As an example, we study boundaries in probabilistic program verification with relevance to the verification of cryptographic systems, itself a narrow subfield of cryptography.

The unifying aspects of our approach – the focus on higher-order languages, notably reasoning techniques which stem from the common basis of operational semantics supported by algebras and logics — favor the transfer of techniques and ideas between languages developed for different application domains.

## 5 Social and environmental responsibility

Our research activities impact the environment negatively, primarily due to the need to attend scientific events. To limit our impact, we give preference to remote participation or environment friendly modes of transport, such as train, when feasible.

## 6 Highlights of the year

- The Marie-Curie Doctoral Network E-CoRe on Energy-efficient Computing via Reversibility coordinated by Ivan Lanese has been approved.

### 6.1 Awards

- The paper [33] received the distinguished paper award at the European Conference on Object-Oriented Programming (ECOOP) 2025.

## 7 Latest software developments, platforms, open data

### 7.1 Latest software developments

#### 7.1.1 Corinne

**Keywords:** Choreography automata, Communicating finite state machines

**Scientific Description:** Corinne relies on the theory of choreography automata, which is described in:

Franco Barbanera, Ivan Lanese, Emilio Tuosto: Choreography Automata. COORDINATION 2020: 86-106

Franco Barbanera, Ivan Lanese, Emilio Tuosto: Composition of choreography automata. CoRR abs/2107.06727 (2021)

**Functional Description:** Choreography automata (c-automata) are finite state automata whose transitions are labelled with interactions of the form  $A \rightarrow B : m$ , representing a communication in which participant A sends a message (of type) m to participant B, and participant B receives it. Corinne allows one to display c-automata represented in the dot format, and: (a) project them on communicating finite state machines representing the behavior of single participants, (b) compute a product c-automaton corresponding to the concurrent execution of two c-automata, (c) synchronize two participants of a c-automaton transforming them into coupled gateways, and (d) check well-formedness conditions ensuring that the system of participants obtained via projection behaves well.

**Release Contributions:** Version 4.0 of Corinne improves the usability of the tool and refines the check for connectedness, allowing the so called late join, when a participant participate to a choreography only in some branches, after having been contacted by some other participant.

**URL:** <https://github.com/lanese/corinne-3>

**Publication:** [hal-03468190](https://hal.archives-ouvertes.fr/hal-03468190)

**Contact:** Ivan Lanese

**Partner:** Gran Sasso Science Institute

#### 7.1.2 CauDEr

**Name:** Causal-consistent Debugger for Erlang

**Keywords:** Debug, Reversible computing

**Scientific Description:** The CauDEr reversible debugger for Erlang is based on the theory of causal-consistent reversibility, which states that any action can be undone provided that its consequences, if any, are undone beforehand. This theory relies on a causal semantics for the target language, and can be used even if different processes have different notions of time. Replay is based on causal-consistent replay, which allows one to replay any future action, together with all and only its causes.

**Functional Description:** CauDEr is a debugger allowing one to explore the execution of concurrent and distributed Erlang programs both forward and backward. Notably, when going backward, any action can be undone provided that its consequences, if any, are undone beforehand. The debugger also provides commands to automatically find relevant past actions (e.g., send of a given message) and undo them, including their consequences. Forward computation can be driven by a log taken from a computation in the standard Erlang/OTP environment. An action in the log can be selected and replayed together with all and only its causes. The debugger enables one to find a bug by following the causality links from the visible misbehavior to the bug.

**URL:** <https://github.com/mistupv/cauder>

**Publications:** [hal-03005383v1](https://hal.archives-ouvertes.fr/hal-03005383v1), [hal-01912894v1](https://hal.archives-ouvertes.fr/hal-01912894v1), [hal-02313745v1](https://hal.archives-ouvertes.fr/hal-02313745v1)

**Contact:** Ivan Lanese

**Participant:** Ivan Lanese

**Partner:** Universitat Politècnica de València

### 7.1.3 QuRA

**Name:** Quipper Resource Analysis

**Keywords:** Static analysis, Quantum programming, Programming language

**Functional Description:** QuRA is a static analysis tool for the verification of the resource consumption of quantum circuit description programs. QuRA takes as input a program written in a variant of Quipper called PQ and outputs two things: a type for the program and an upper bound to the size of the circuit it will build. QuRA is capable of estimating global size metrics of circuits, such as their width and gate count, as well as size metrics that are local to individual wires, such as depth. It does so in an almost fully automatic fashion, with few annotations, thanks to the use of SMT solvers.

**URL:** <https://github.com/andreacolledan/qura>

**Contact:** Andrea Colledan

**Participants:** Andrea Colledan, Ugo Dal Lago, Niki Vazou

**Partner:** The IMDEA Software Institute

### 7.1.4 Ranflood

**Keywords:** Cybersecurity, Ransomware

**Functional Description:** Ranflood is an anti-crypto-ransomware tool that counteracts the encryption phase by flooding specific folders (e.g., the attacked location, the user's folders) with decoy files and helps users recover their files after an attack.

This action has a twofold effect.

First, it confounds the genuine files of the user with decoy files, causing the attacking ransomware to waste time on sacrificial data rather than on the victim's genuine files.

Second, the file-flooding IO-intensive activity contends with the ransomware to access the victim's computing resources, further slowing down the attack of the malware.

**Release Contributions:** Ranflood 0.7-beta includes a) new code utilities and features, two new Ranflood flooding strategies based on Shamir's Secret Sharing, and a related restore routine for the FileChecker, and b) an HTTP and a WebSocket server that clients can use to connect to the Ranflood daemon (useful also to support the connection with the new Ranflood webclient and additional functionalities for a more fine-grained control of the daemon's processes).

**URL:** <https://ranflood.netlify.app/>

**Contact:** Saverio Giallorenzo

### 7.1.5 APP

**Name:** Allocation Priority Policies

**Keywords:** Serverless, Scheduling, Cloud computing, Optimisation

**Scientific Description:** APP addresses the problem of function execution scheduling, i.e., how to schedule the execution of Serverless functions to optimise their performance against some user-defined goals, by specifying policies that inform the scheduling of function execution.

**Functional Description:** Serverless computing is a Cloud development paradigm where developers write and compose stateless functions, abstracting from their deployment and scaling.

APP is a declarative language of Allocation Priority Policies to specify policies that inform the scheduling of Serverless function execution to optimise their performance against some user-defined goals.

APP is currently implemented as a prototype extension of the Serverless Apache OpenWhisk platform.

**Release Contributions:** 0.1: APP first release introduced the APP declarative language used to write scheduling policies in serverless platforms. The first release also introduced support for the OpenWhisk platform with an alternative Load Balancer for APP scripts.

0.1-tapp: This release introduces an extension of APP, named tAPP (topology-aware Allocation Priority Policies), that adds the capability to declare topological constraints on function-scheduling. An implementation on top of the OpenWhisk platform is also provided.

0.1-aapp: Another extension, dubbed aAPP (affinity-aware Allocation Priority Policies), that adds the capability to define affinity and anti-affinity constraints on 2 or more functions, together with an updated implementation on OpenWhisk.

**URL:** <https://github.com/giusdp/openwhisk>

**Contact:** Saverio Giallorenzo

**Partner:** University of Southern Denmark

### 7.1.6 JOLIE

**Name:** Jolie

**Keyword:** Microservices

**Scientific Description:** Jolie enforces a strict separation of concerns between behaviour, describing the logic of the application, and deployment, describing the communication capabilities. The behaviour is defined using the typical constructs of structured sequential programming, communication primitives, and operators to deal with concurrency (parallel composition and input choice). Jolie communication primitives comprise two modalities of interaction typical of Service-Oriented Architectures (SOAs), namely one-way (sends an asynchronous message) and request-response (sends a message and waits for an answer). A main feature of the Jolie language is that it allows one to switch among many communication media and data protocols in a simple, uniform way. Since it targets the field of SOAs, Jolie supports the main communication media (TCP/IP sockets, Bluetooth L2CAP, Java RMI, and Unix local sockets) and data protocols (HTTP, JSON-RPC, XML-RPC, SOAP and their respective SSL versions) from this area.

**Functional Description:** Jolie is a language for programming service-oriented and microservice applications. It directly supports service-oriented abstractions such as service, port, and session. Jolie allows one to program a service behaviour, possibly obtained by composing existing services, and supports the main communication protocols and data formats used in service-oriented architectures. Differently from other service-oriented programming languages such as WS-BPEL, Jolie is based on a user-friendly Java-like syntax (more readable than the verbose XML syntax of WS-BPEL). Moreover, the kernel of Jolie is equipped with a formal operational semantics. Jolie is used to provide proof-of-concept implementations around OLAS activities.

**Release Contributions:** Version 1.13 includes 4 minor releases. This version introduces several major enhancements. The most significant change was the upgrade to Java 21 as the language requirement. Other key features include adding location data and line numbers to faults, support for minimum values in number ranges, and a complete JAP module system refactoring query. The release also introduced a new Jolie2Java implementation, YAML support, JSON schema enhancements, jolie2openapi improvements, and refactored joliedoc and joliemock tools to support the new syntax.

**URL:** <http://www.jolie-lang.org/>

**Contact:** Saverio Giallorenzo

**Participants:** Claudio Guidi, Fabrizio Montesi, Maurizio Gabbrielli, Saverio Giallorenzo, Ivan Lanese, Stefano Zingaro

### 7.1.7 FunLess

**Keywords:** Serverless, WebAssembly

**Functional Description:** FunLess is a Function-as-a-Service (FaaS) platform that, unlike traditional FaaS solutions that rely on resource-heavy containerisation technologies, employs WebAssembly (Wasm) as its runtime environment. Using Wasm ensures lightweight execution, reduced cold starts, and enhanced portability, enabling functions to run seamlessly on diverse hardware architectures (particularly useful in cloud-edge environments). The platform supports a consistent function development and deployment process, allowing developers to write functions in multiple supported languages (Rust, Go, and JavaScript), compile them into Wasm binaries, and execute them across heterogeneous devices without requiring additional runtime adjustments. FunLess allows flexible deployments, supporting both bare-metal installations and optional integration with container orchestration tools like Kubernetes.

**URL:** <https://funless.dev/>

**Publication:** [hal-04826377v1](#)

**Contact:** Matteo Trentin

### 7.1.8 JoT

**Name:** Jolie Testing

**Keywords:** Microservices, Software testing

**Functional Description:** JoT is a testing framework for Microservice Architectures based on technology agnosticism, a core principle of microservices.

The main advantage of JoT is that it reduces the amount of work for a) testing microservices that use different technology stacks, b) writing tests that involve multiple services, and c) reusing tests under different deployment configurations or after changing some of its components (e.g., when, for performance, one reimplements a service with a different technology).

In JoT, tests are orchestrators that can both consume or offer operations from/to the microservice architecture under test. The language for writing JoT tests is Jolie, which provides constructs that support technology agnosticism and the definition of terse test behaviors.

**Release Contributions:** In version 0.0.27, JoT underwent some minor updates to fix some issues and integrate its functionalities with container technologies (Docker) and continuous integration platforms (GitHub Actions).

**URL:** <https://github.com/jolie/jot>

**Contact:** Saverio Giallorenzo

**Partner:** University of Southern Denmark

### 7.1.9 Choral

**Keywords:** Choreographic Programming, Compilation, Modularity, Distributed programming

**Scientific Description:** In essence, Choral developers program a choreography with the simplicity of a sequential program. Then, through the Choral compiler, they obtain a set of programs that implement the roles acting in the distributed system. The generated programs coordinate in a decentralised way and they faithfully follow the specification from their source choreography, avoiding possible incompatibilities arising from discordant manual implementations. Programmers can use or distribute the single implementations of each role to their customers with a higher level of confidence in their reliability. Moreover, they can reliably compose different Choral(-compiled) programs, to mix different protocols and build the topology that they need.

Choral currently interoperates with Java (and it is planned to support also other programming languages) at three levels: 1) its syntax is a direct extension of Java (if you know Java, Choral is just a step away), 2) Choral code can reuse Java libraries, 3) the libraries generated by Choral are in pure Java with APIs that the programmer controls, and that can be used inside of other Java projects directly.

**Functional Description:** Choral is a language for the programming of choreographies. A choreography is a multiparty protocol that defines how some roles (the proverbial Alice, Bob, etc.) should coordinate with each other to do something together.

Choral is designed to help developers program distributed authentication protocols, cryptographic protocols, business processes, parallel algorithms, or any other protocol for concurrent and distributed systems. At the press of a button, the Choral compiler translates a choreography into a library for each role. Developers can use the generated libraries to make sure that their programs (like a client, or a service) follow the choreography correctly. Choral makes sure that the generated libraries are compliant implementations of the source choreography.

**Release Contributions:** In 2025, Choral had 3 minor releases. The releases focused primarily on bug fixes and minor improvements to the compiler and standard library. These releases include fixing header bugs, resolving problems with the building system, and addressing compilation and compatibility issues with the standard library.

**URL:** <https://www.choral-lang.org/>

**Contact:** Saverio Giallorenzo

**Participants:** Saverio Giallorenzo, Fabrizio Montesi

**Partner:** University of Southern Denmark

### 7.1.10 ChorEr

**Name:** Choreographies from Erlang

**Keywords:** Choreography automata, Erlang, Choreography extraction

**Scientific Description:** ChorEr reconstructs the possible message-passing interactions between Erlang processes and represents them using Local Views, which are then combined into a single Global View. The extraction follows a bottom-up, over-approximating approach: any communication that might occur is included. This ensures that potential issues, including subtle concurrency bugs, are highlighted without being missed. Deadlocks are highlighted directly in the Global View graph: they appear as red states.

This tool is an early prototype and still under development,

**Functional Description:** ChorEr is an analysis tool that examines Erlang programs and extracts Choreography Automata describing their communication behaviour. Its purpose is to help developers uncover communication bugs in distributed and actor-based systems, specifically those related to communication behaviour (races, deadlocks, ...).

**Release Contributions:** Deployment as web application

**URL:** <https://chorer.cappuccino.ovh/>

**Contact:** Ivan Lanese

**Partner:** I3S

## 8 New results

### 8.1 Service Oriented Computing and Cloud Computing

**Participants:** Saverio Giallorenzo, Ivan Lanese, Matteo Trentin, Giuseppe De Palma, Gianluigi Zavattaro.

Modern distributed systems face challenges in coordination, resource management, and maintaining consistency across components while ensuring security, performance, and reliability. OLAS contributions focus on developing and refining new programming and architectural paradigms, tools, and platforms to address these challenges, particularly in the context of serverless computing and service orchestration.

#### 8.1.1 Distributed systems and choreographies

Considering failures and disruptions, we worked on a behavioral theory for distributed systems with weak recovery [7], introducing a process calculus that models dynamic nodes/links, crash failures, imperfect knowledge, and weak recovery with incarnation numbers. The work provides a fully abstract coinductive characterization of weak barbed congruence by means of a labeled transition system semantics and its associated weak bisimilarity.

For specifying and analyzing complex distributed coordination patterns, we advanced choreography-based approaches with two complementary contributions: (1) pomsets for process management [34] applied to healthcare authorization/accreditation protocols, featuring efficient realisability checking algorithms for both synchronous and asynchronous communication; and (2) choreography extraction for program understanding [45], developing bottom-up techniques that automatically generate global choreography automata from Erlang message-passing code to reveal communication patterns, deadlocks, and unexpected behaviors even in ill-formed systems [27].

Focusing on cloud-native systems, we presented two complementary contributions addressing architecture adaptation. First, we introduced Adaptable TeaStore [20], an extension of the TeaStore benchmark [47] that incorporates adaptability as a first-class design concern. The architecture distinguishes between mandatory services and optional ones, supporting multiple component variants (called flavors) with varying resource requirements. Key innovations include outsourceable functionalities that can be provided internally or by external providers, and local cache mechanisms ensuring resilience when external dependencies fail. This contribution also includes a catalogue of varied adaptation scenarios (e.g., database unavailability, cyberattacks, cloud provider outages) and an open-source implementation for metrics collection and adaptation triggers. The second contribution [35] is an implementation of the Adaptable TeaStore architecture using AIOCJ [46], a choreographic language that ensures correctness properties such as deadlock freedom before, during, and after adaptation. The approach uses adaptation scopes to specify code sections that may change, while adaptation rules – which one can add at runtime – define the execution of new distributed behaviors based on triggering conditions and on code annotations, environment variables, and local state. The implementation revealed both strengths of the choreographic approach and areas for improvement, including better integration with service controllers and more structured patterns for multi-scope adaptations.

#### 8.1.2 Serverless computing

We addressed various aspects of serverless computing architectures. Continuing our development of the APP (Allocation Priority Policies) language, we presented aAPP [36] an APP extension that supports affinity-aware serverless function scheduling, allowing function allocation decisions to depend on the presence or absence

of other functions on execution nodes. This capability addresses both performance requirements (e.g., co-locating functions to reduce latency) and security concerns (e.g., preventing co-location of trusted and untrusted functions). We presented an implementation on Apache OpenWhisk to demonstrate improved performance in affinity-aware scenarios with negligible overhead in non-affinity contexts. We also formally analyzed APP and aAPP expressiveness [38, 37], investigated from the point of view of computational complexity of reachability analysis for APP and aAPP scripts – reachability essentially answers the question “can a given function run on a specific node?”. While reachability analysis has linear time complexity in APP, the addition of affinity constraints in aAPP makes the problem PSPACE-complete. Recognizing the correspondence between aAPP reachability problems and automated planning problems, we also developed a static analyzer that leverages PDDL (Planning Domain Definition Language) planners to verify scheduling properties. An open problem in the usage of APP is defining the “right” scheduling policy, often requiring rounds of refinement involving knowledge of the underlying infrastructure, guesswork, and empirical testing. We proposed a cost-aware variant of APP [13] that lightens the burden on the shoulders of users by deriving cost information from the functions, via static analysis.

Remaining within the serverless domain, we presented Fenrir [42], a framework that facilitates transitioning from monolithic programming to serverless architectures. In Fenrir, developers write applications in a monolithic style and use annotations to specify which components should become separate serverless functions. Then, Fenrir generates a deployable serverless codebase, ensuring alignment of execution semantics between monolithic and serverless code while supporting rapid development and testing cycles.

We further explored the application of the serverless computing paradigm in different areas than the cloud. Specifically, we explored the application of the paradigm in highly dynamic ad-hoc drone networks [12]. To support the routing of functions in a decentralized way over moving drones, we proposed the adoption of a two-layer network overlay architecture that combines gossip-based topology management with distributed function scheduling. To allow users to control the deployment of functions over the drones, we introduced another APP variant, called AHAPP (Ad-Hoc APP), that enables function deployment based on resource constraints and operational needs. The approach addresses network volatility through execution semantics for both stable and dynamic topologies, function offloading, and resilience to network disruptions. These distributed systems must cope with various types of failures, including crash failures with recovery.

### 8.1.3 Microservices

In the area of microservices, we introduced an approach to microservice scaling [1] based on a proactive-reactive global scaling algorithm that leverages knowledge of functional dependencies between microservices. Unlike traditional local scaling approaches, which adjust individual services independently, global scaling performs coordinated architecture-level reconfigurations to reach a target computational load. Based on this idea, we developed an integrated architectural modeling and execution language based on the ABS language, demonstrating that proactive-reactive global scaling outperforms purely reactive approaches while optimizing both performance and resource consumption. On the same thread of global scaling, we presented a constraint optimization approach [9] for deploying multi-flavored applications (where the same components have different versions) on Cloud-Edge infrastructure topologies. We demonstrate the practical feasibility of the approach through experiments on a variety of realistic Cloud-Edge infrastructural topologies and component architectures.

The application of AI techniques to microservices lifecycle management represents an emerging and rapidly growing field. In this context, we conducted an exhaustive systematic mapping study of AI techniques in the microservices’ lifecycle [11], analyzing 269 peer-reviewed papers (2017–2023) to identify 16 research themes connecting specific AI domains, DevOps phases, and quality attributes. The study reveals performance efficiency as the dominant focus (80%) and highlights emerging trends in AIOps and Industry 4.0 applications.

### 8.1.4 Infrastructure as code for security

Following the recent trend of Infrastructure as Code automatization, we proposed applications that showcase the advantages of the “as-code” paradigm to security. First, we presented SAFARI [23], an open-source framework for safe and efficient security investigation, focused on ransomware analysis. The framework emphasizes scalability, air-gapped security, and automation, leveraging virtualization, Infrastructure-as-Code, and OS-agnostic task automation to create isolated environments for reproducible and controlled ransomware

execution. We demonstrated SAFARI’s capabilities through case studies analyzing notorious ransomware strains including WannaCry and LockBit, as well as evaluating countermeasure tools. In the subsequent work [3], we extended SAFARI for supporting Operational Technology (OT) environments, enabling Security-Investigation-as-Code for industrial systems. This adaptation addresses the challenges of validating security in complex OT architectures where in-production testing is impractical. Integrating technologies based on Infrastructure-as-Code and Software Defined Networks with breach-and-simulation platforms (MITRE Caldera), the framework provides scalable, reproducible multi-node security assessment capabilities while maintaining complete air-gapping.

## 8.2 Reversible computing

**Participants:** Ivan Lanese, Vikraman Choudhury.

In reversible computing, computation can proceed not only in the standard, forward direction, but also backwards, recovering past states. We have continued the study of reversible computing started in the past years, focusing on causal-consistent reversibility (suitable for concurrent systems). The axiomatic approach to causal-consistent reversibility developed in previous work allows one to prove relevant properties of concurrent reversible formalisms by checking a few simple axioms. The approach works on Labeled Transition Systems equipped with a notion of Independence (LTSIs). Even if the axioms are quite simple, verifying them on non-trivial LTSIs is time consuming and involves a few subtleties. We have created Tallulah [17], a tool which allows one to automatically verify various axioms on concrete LTSIs and suggests how to patch the LTSI when some axiom does not hold.

Also, we studied [41] the interplay between reversibility and irreversibility. In particular, we extended roll- $\pi$ , a higher-order  $\pi$ -calculus with rollbacks, with two mechanisms to limit reversibility. First, a commit operation, which disallows rollbacks beyond some past action. Second, we decided that rollback decisions are propagated along communications only if both the sender and the receiver agree to do so. We show that these extensions enable one to model a speculative consensus algorithm.

## 8.3 Quantitative analysis

**Participants:** Martin Avanzini, Andrea Colledan, Ugo Dal Lago, Zeinab Galal, Ivan Lanese.

In OLAS, we are interested in studying *quantitative aspects* of higher-order programs, such as resource consumption, not necessarily only in a pure setting but also when placed in an interactive scenario. One key interest of OLAS are quantitative properties of probabilistic programs, such as the expected runtime, or the probability that some event occurs. This is motivated, in parts, by the role that randomization plays in cryptography, or by the use of randomization as a mean to make algorithms more efficient (on average). Further, due to the rise of quantum models and the prospect of quantum machines, our focus extends also to the analysis of quantitative properties of quantum languages.

Below we describe the results obtained by OLAS this year, categorized by application areas.

### 8.3.1 Deterministic program analysis

Deterministic programs, following the traditional model of computation, can exhibit quantitative effects and in particular give rise to time and space costs, which should for very good reasons be kept under control. Moreover, the quantitative relational analysis of such programs through metrics is also of interest. In OLAS, we study how this is possible and we thus deal with semantics and verification of deterministic programs.

One class of deterministic programs which are of interest for OLAS are certainly the higher-order ones, for which notions of complexity, called type-two complexity classes, are well known. In [2], we investigate the class of type-two basic feasible functionals, which serves as the higher-order analogue of polynomial-time computable functions, and we study how this class can be captured using higher-order term rewriting. We

model type-two computation by viewing higher-order rewriting systems as mechanisms for computing functionals that take first-order functions as inputs. Building on existing approaches that use interpretations to analyze termination and complexity in first-order rewriting, we focus on a recently proposed framework of cost-size interpretations adapted to the higher-order setting. We show that when higher-order terms admit polynomially bounded cost-size interpretations, they represent exactly the class of basic feasible functionals. This result establishes a precise correspondence between second-order polynomial-time computation defined via oracle Turing machines and a syntactic, rewriting-based characterization grounded in higher-order term rewriting theory.

In [25], we study the metric nature of differential logical relations as a way to perform a quantitative analysis of deterministic programs, measuring how program outputs vary in response to variations in their inputs. Unlike standard metric-based approaches, which assign a single numeric distance between programs, we emphasize that differences between higher-order programs should themselves be functions. These functions relate input errors to output errors, yielding finer-grained and context-sensitive quantitative information about program behavior. Our goal is to clarify the metric nature of these relations. Although prior work shows that differential logical relations do not generally induce (quasi-)metric or partial metric spaces, we show that the resulting distance functions, what we call quasi-quasi-metrics, can be systematically related to both quasi-metrics and partial metrics.

### 8.3.2 Probabilistic program analysis

A class of programs and processes that by their very nature exhibit quantitative effects and are therefore suitable to quantitative analysis are certainly probabilistic programs. In our team, we are interested in studying probabilistic programs both from a foundational and from a more applicative viewpoint.

In [8], we extend intersection types to a computational  $\lambda$ -calculus with algebraic operations in the sense of Plotkin and Power, an example of which is the probabilistic  $\lambda$ -calculus. We do so by introducing monadic intersections, allowing computational effects to be reflected not only in the operational semantics but also directly in the type system. Because termination is no longer the sole property of interest in an effectful setting, we focus on analyzing the interactive behavior of typed programs with their environment. The resulting type system characterizes a natural notion of observation, covering both finitary and infitary behaviors. We then extend the system with subtyping to account for a broader range of effects, using monads on preorders rather than on sets, which enables in particular the modeling of non-determinism. The core technical contribution is a novel combination of syntactic methods with abstract relational reasoning.

Abstract reduction systems provide a foundational framework for operational semantics but are primarily limited to qualitative reasoning. In [19], we introduce *weighted rewrite systems*, which enrich reductions with quantitative information and enable the study of quantitative program properties. We develop a theory of boundedness, a refinement of termination, together with sound and complete proof methods. Importantly, we demonstrate that weighted rewrite systems form an abstract framework in which, in particular, probabilistic program properties can be studied.

### 8.3.3 Quantum program analysis

Quantum computation is a promising and emerging computational paradigm which can efficiently solve problems considered to be intractable on classical computers. However, the unintuitive nature of quantum mechanics poses interesting and challenging questions for the design and analysis of quantum programming languages, with functional correctness and complexity analysis being of prime importance.

We study quantum program analysis with the goal of statically reasoning about the size of the quantum circuits generated by high-level programs, this being a crucial parameter for the realizability of the so-called quantum advantage. In particular, we focus on circuit description languages, where classical programs produce quantum circuits that may far exceed the capabilities of current hardware. We develop type-based techniques to derive parametric upper bounds on circuit resources such as width, depth, and gate count, as well as refined variants that focus on specific kinds of wires or gates [5, 4]. Our approach relies on expressive type-and-effect systems combining linearity, refinement types, and indexed effects, where indices are arithmetic expressions whose interpretation depends on the chosen resource metric. This design allows us to flexibly adapt the analysis to different notions of quantum cost. We apply these ideas to Quipper, a circuit description language, proving standard type safety results and showing that the inferred resource bounds are

sound with respect to a big-step operational semantics. Finally, we implement our techniques in the QuRA tool and demonstrate empirically that our framework can automatically infer tight and meaningful bounds for realistic quantum algorithms, including the Quantum Fourier Transform and Grover’s algorithm.

In [32], we also extend a minimal system of Multiparty Session Types with quantum data and quantum operations, obtaining Quantum Multiparty Session Types as a formal framework for specifying and analyzing quantum communication protocols. Our approach supports protocol descriptions at both the global level, where we specify the overall communication structure and dependencies, and the local level, where we describe the expected behavior of each participant. In addition to guaranteeing standard communication properties such as deadlock freedom, our system enforces single ownership of qubits at any point in time, thereby capturing fundamental quantum principles such as no-cloning and no-deleting. We validate the expressiveness and usefulness of our framework by verifying several representative quantum protocols from the literature, including Teleportation, Secret Sharing, Bit-Commitment, and Key Distribution.

### 8.3.4 Applications to cryptography and security

We have also become increasingly interested in topics of *cryptography* and *security*.

Relational program logics are a fundamental tool in game-based cryptographic proofs, where security arguments rely on relating the behavior of pairs of probabilistic programs. In [18], we introduce eRHL, a quantitative relational program logic whose assertions take values in the extended non-negative reals, enabling reasoning about relational expectation properties. By moving to quantitative assertions, eRHL overcomes the randomness-alignment restrictions inherent in earlier logics. As a result, eRHL is the first relational probabilistic program logic to admit non-trivial soundness and completeness results for all almost surely terminating programs. Noteworthy, the logic is sound and complete with respect to program equivalence, statistical distance, and differential privacy. The practicality of this logic has been demonstrated on several examples stemming from cryptography and privacy, including the verification of randomized response and privacy amplification mechanisms that ensure differential privacy, and a proof of the PRP/PRF switching lemma showing that pseudorandom permutations (PRPs) and functions (PRFs) are computationally indistinguishable.

In [6], we have been re-investigating the role of memory layout randomization (e.g. Kernel Layout randomization as employed in the Linux kernel) as an (in)effective mitigation strategy against attacks in the Spectre era. This work encompasses a revised, formal thread model based on the usual multi-tier memory model employed in modern kernels as well as standard defense mechanisms (such as DEP, SMAP/SMEP, etc), but where an attacker can exploit side-channel information leaks and influence speculative execution. We have proven that attacks comprising memory safety such as the blindsight and related attacks are not only possible due to practical limitations of implementations, but are indeed inherent to the attack vectors underlying modern computer architectures. To remedy, we establish *speculative memory safety* through an adaptation of the notion of constant time, a well established implementation technique to mitigate side-channel attacks. We also investigate program transformations to bridge the gap between memory safety in speculative and classical scenarios.

## 8.4 Qualitative semantics

**Participants:** Vikraman Choudhury, Ugo Dal Lago, Daniel Hirschhoff, Davide Sangiorgi, Gianluigi Zavattaro.

In this area, during the past year, our efforts have gone into the following directions, that have to do with (i) model checking higher-order functions, (ii) proof techniques for coinduction and induction, (iii) asynchronous session types, (iv) coinductive proof techniques, and (v) unifying semantics and comparisons among models.

### 8.4.1 Model checking higher-order functions

Model checking is a powerful technique for verifying systems and programs which, since the pioneering results by Knapik et al. [48], Ong [50], and Kobayashi [49], is known to be applicable to functional programs

with higher-order types against properties expressed by formulas of monadic second-order logic. Recent undecidability results of ours had shown that applying higher-order model checking techniques to programs that use effect handlers is a major challenge. This is the challenge addressed in [15], using answer-type modifications, the use of a monomorphic version of which allows to recover decidability. However, the absence of polymorphism leads to a loss of modularity, reusability, and even expressiveness. Hence we also study, and propose a solution for, the problem of defining a calculus that, on the one hand, supports answer-type polymorphism and subtyping but, on the other hand, ensures the underlying model checking problem to remain decidable. We have also implemented a proof-of-concept model checker.

#### 8.4.2 Asynchronous session types

Concerning session types, our efforts have gone into studying a theory of asynchronous sessions ensuring that well-typed processes terminate under a suitable fairness assumption [33]. Fair termination ensures that all messages are eventually consumed, despite the asynchrony assumption. The type system is also the first of its kind that is firmly rooted in linear logic; e.g., asynchronous communication is modeled through commuting conversions and cut reductions in linear logic proofs.

#### 8.4.3 Coinductive proof techniques

‘Up-to techniques’ represent enhancements of the coinduction proof method and are widely used on coinductive behavioral relations such as bisimilarity. In recent years, we have shown that these techniques are intimately related to the proof technique of *unique-solution of equations*, originally proposed by Milner in the 1980s. In [26] we first review two improvements of the unique-solution technique that we had proposed in recent years. The first one builds on a new behavioral preorder, called contraction. The second one relaxes the conditions under which unique-solution can be applied by focusing on divergence in processes. We then compare the two techniques and present applications of these techniques to reason about higher-order languages.

Abstract formulations of the up-to techniques exist, using fixed-points or category theory. In a recent proposal, we had studied how to transport such techniques onto the concrete realms of inductive behavioral relations, i.e., relations defined from inductive observables, e.g., traces or enriched forms of traces. The abstract meaning of these ‘inductive enhancements’, however, remained unclear. We propose an abstract account in [40], using fixed-point theory in complete lattices.

#### 8.4.4 Unifying semantics and comparisons among models

A common theme in OLAS is the comparison between models. On this, we have continued and refined work started in previous years aimed at comparing the  $\lambda$ -calculus, as a pure model of functions, and the  $\pi$ -calculus, as a pure model of processes [14]. There has been a lot of work in the literature on the representation of the  $\lambda$ -calculus into the  $\pi$ -calculus, since the early 1990s, where the process representations always yield (at best) non-extensional lambda-theories (i.e., the beta rule holds, whereas eta does not). Our goal here was to see how to obtain extensional representations. In particular we have showed that the representation of functions can be made parametric on certain abstract components called wires (intuitively, processes whose task is to connect two end-point channels). When a few algebraic properties of wires hold, the representation yields a lambda-theory. Further, by varying the shape of wires (exploiting symmetries and dualities of the calculus), we can obtain well-known lambda-theories, both extensional and non-extensional (intuitively, those of Böhm trees with infinite eta, of Levy-Longo trees, and of Böhm trees).

Comparing models also offers us the opportunity for transferring ideas and techniques between conceptually different models. In [30], we take ideas from Game Semantics for  $\lambda$ -calculi and transport them onto the  $\pi$ -calculus. Precisely, we focus on the Game Semantics concept of *visibility*. Following the concept of visibility, we first design a type system for the  $\pi$ -calculus in which processes may only store, or remember, first-order values. We then show that visibility has a relevant impact on process behaviors, and propose proof techniques to reason about such behaviors.

In the semantics of higher-order languages, a useful concept is that of duality; it may be regarded as a unifying concept, as it allows one to relate seemingly different objects. Examples are: values and continuations; call-by-value and call-by-name, programs and evaluation contexts; clients and servers in session types, strict and lazy evaluation; product and sum types, effects (monads) and coeffects (comonads). In [22],

we develop a different perspective: a duality of abstraction – of currying and cocurrying. Abstraction is at the heart of functional programming. Just as higher-order functions give exponentials, higher-order continuations give coexponentials. From this, we design a language that combines exponentials and coexponentials, producing a duality of lambda abstraction. We develop the semantics of this language using the axiomatic structure of continuations, yielding a complete axiomatization of control effects. We further develop duals of first-order arrow languages using coexponentials, and discuss the implementation of this duality as control operators in programming.

## 8.5 Miscellaneous

We published results spanning areas beyond the core focus of OLAS. We report them in this section. They span interdisciplinary topics across cybersecurity, educational technology, social computing, and programming language implementation.

### 8.5.1 Cybersecurity: ransomware defense

Two complementary works advanced the Data Flooding Against Ransomware (DFaR) approach [44], which contrasts ransomware with resource contention and confusion by overwhelming attackers with decoy files that slow down malicious operations and provide time for intervention. The first contribution [29] introduces a watermarking-based technique that embeds imperceptible markers in randomly-generated decoy files, enabling reliable identification and removal during system restoration without requiring pre-attack file lists. This approach addresses a critical limitation of existing random-content DFaR strategies while maintaining the deceptive properties essential for confusing ransomware. The second work [24] introduces a new copy-based flooding modality based on Shamir’s Secret Sharing (SSS), providing protection against both crypto-ransomware and (for the first time) exfiltration attacks. The proposal works by splitting user files into shards. Since users can parametrize the number of shards needed to reconstruct the original files, the technique can achieve high resilience when contrasting crypto-ransomware – allowing users to recover data from few shards – while maintaining secrecy against exfiltration – requiring attackers to obtain many shards to reconstruct victims’ data.

### 8.5.2 Educational technology and social analysis

We presented two distinct contributions at the intersection of technology and education, addressing different challenges in educational contexts. The first [39] investigates the affordability and effectiveness of using Large Language Models (LLMs) to answer multiple-choice questions in undergraduate Programming Languages courses. By fine-tuning smaller LLaMA-2 models (7B and 13B parameters) with course textbook material, this work demonstrates that resource-efficient fine-tuned models outperform larger generic ones, making LLM deployment more accessible for educational institutions with limited computational resources. The second work [10] presents a framework for quantitatively measuring and improving social inclusion in educational settings using affordable Internet of Things (IoT) devices for contact tracing combined with network analysis techniques. Employing PageRank and Betweenness centrality measures, the framework identifies socially isolated students and proposes sustainable interventions aligned with natural interaction patterns. A real-world case study in a primary school validates the methodology’s effectiveness in providing actionable insights for social integration without requiring substantial infrastructure investment.

### 8.5.3 Programming language implementation

Within the broader scope of systems programming, we introduced SILB-Recycler, a novel garbage collection algorithm [28] based on a reference counting logic capable of collecting reference cycles. The algorithm features resilience to errors during tracing, support for object finalization, no need for supplementary heap memory during collection, and a fast breadth-first tracing approach that avoids stack overflows. Implemented as a Rust library, the garbage collector provides smart pointers with APIs compatible with Rust’s standard library. Benchmarks show that the algorithm performs comparably to popular Rust alternatives and outperforms them when dealing with garbage cycles.

## 9 Bilateral contracts and grants with industry

### 9.1 Bilateral contracts with industry

**Ranflood** Giallorenzo co-leads a three-year project collaboration, called “Ranflood”, between the “Regional Environmental Protection and Energy Agency” of Emilia-Romagna (ARPAE Emilia-Romagna) and the “Department of Computer Science and Engineering” (DISI) at the University of Bologna. The collaboration regards the development of techniques and software to combat the spread of malware by exploiting resource contention.

**Participants:** Saverio Giallorenzo.

### 9.2 Bilateral Grants with Industry

**Enhancing Situational Awareness: Real-Time Monitoring and Replanning with Collaborative Unmanned Vehicles and Onboard Sensing** Zavattaro and Giallorenzo participate in a two-year project collaboration, called “Enhancing Situational Awareness: Real-Time Monitoring and Replanning with Collaborative Unmanned Vehicles and Onboard Sensing”, started in December 2023, between Leonardo Company S.p.A., Thales Alenia Space Italia S.p.A., and the University of Bologna. The collaboration regards the analysis of techniques for managing the development and deployment of software applications in the context of multi-drone systems in disaster scenarios.

**Participants:** Gianluigi Zavattaro, Saverio Giallorenzo.

## 10 Partnerships and cooperations

### 10.1 International research visitors

#### 10.1.1 Visits of international scientists

**Guilhem Jaber**

**Status** MdC

**Institution of origin:** Inria Nantes and Univ. Nantes

**Country:** France

**Dates:** 3 weeks overall, various periods

**Context of the visit:** Game Semantics and Processes

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

**Ken Sakayori**

**Status** Researcher

**Institution of origin:** Univ. Tokyo

**Country:** Japan

**Dates:** May 4-10, 2025

**Context of the visit:** Denotational semantics of concurrent processes, quantum languages

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

**Alain Girault**

**Status** DR INRIA

**Institution of origin:** Inria Grenoble

**Country:** France

**Dates:** June 17-20, 2025

**Context of the visit:** Various subjects on programming languages

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

**Naohiko Hoshino**

**Status** Associate Professor

**Institution of origin:** Sojo University

**Country:** Japan

**Dates:** July 9-13, 2025

**Context of the visit:** compilation of higher-order quantum programming languages

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

**Neea Rusch**

**Status** PhD

**Institution of origin:** Augusta University

**Country:** USA

**Dates:** May 20 – May 23, 2025

**Context of the visit:** Implicit Computational Complexity

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

**Davide Davoli**

**Status** Postdoctoral fellow

**Institution of origin:** MPI Bochum

**Country:** Germany

**Dates:** December 09 – December 12, 2025

**Context of the visit:** Higher-order probabilistic program logics

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

**Laura Bocchi**

**Status** Reader

**Institution of origin:** University of Kent

**Country:** UK

**Dates:** 1 or 2 days visits, multiple times

**Context of the visit:** concurrent reversible systems

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

## 10.2 European initiatives

### 10.2.1 Horizon Europe

**ReGraDe-CS (Reversible Gray Debugging of Concurrent Systems)** is a Marie-Curie Postdoc Fellowship started in December 2023 and with a 2 years duration. The fellow is Vikraman Choudhury, supervised by Ivan Lanese. The project tackles gray debugging of concurrent systems. Debugging concurrent systems is notoriously hard. Reversible causal-consistent debugging and replay allows one to log a faulty execution in production environment and replay it in the debugger. There, it can be explored backwards and forwards following causality links from the visible misbehavior to the bug causing it. ReGraDe-CS will extend the approach to gray debugging, namely debugging of systems where only part of the source code is accessible (e.g., the system invokes external services such as Google Maps).

**Participants:** Vikraman Choudhury, Ivan Lanese.

## 10.3 National initiatives

### 10.3.1 HOPR

HOPR (Higher-Order Probabilistic and resource-aware Reasoning) aims at investigating logical and semantical frameworks to reason on computation including higher-order, probabilistic and resource-bounded aspects. It explores applications to cryptography and to differential privacy. This project started in January 2025, for a duration of 4 years.

**Participants:** Martin Avanzini, Ugo Dal Lago.

### 10.3.2 SmartCloud

SmartCloud (Smart dynamic adaptivity for cloud computing systems) is an ANR project that started on January 2024 and that will end in July 2027.

It aims at studying techniques for cloud computing systems which can be updated in a smart, dynamic and coordinated way to cope with changing requirements and environment conditions. The project will define a new framework for adaptation combining a model of both platform and application aspects and heuristics for online optimization.

**Participants:** Saverio Giallorenzo, Ivan Lanese, Gianluigi Zavattaro.

## 11 Dissemination

### 11.1 Promoting scientific activities

#### 11.1.1 Scientific events: organization

##### Chair of conference program committees

- 1st Workshop of Adaptable Cloud Architectures, WACA 2025 (Saverio Giallorenzo)

**Member of the conference program committees**

- 41th Object-Oriented Programming, Systems, Languages & Application, OOPSLA 2026 (Martin Avanzini)
- 45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2025 (Ugo Dal Lago)
- 30th ACM SIGPLAN International Conference on Functional Programming, ICFP 2025 (Ugo Dal Lago)
- 26th Italian Conference on Theoretical Computer Science, ICTCS 2025 (Ugo Dal Lago)
- 28th International Conference on Foundations of Software Science and Computation Structures, FoSSaCS 2025 (Ugo Dal Lago)
- 33rd EACSL Annual Conference on Computer Science Logic, CSL 2025 (Ugo Dal Lago)
- 17th Conference on Reversible Computation, RC 2025 (Ivan Lanese)
- 27th International Symposium on Practical Aspects of Declarative Languages, PADL 2025 (Ivan Lanese)
- Workshop on Components Operationally: Reversibility and System Engineering, CORSE 2025 (Ivan Lanese)
- Workshop on Adaptable Cloud Architectures, WACA 2025 (Ivan Lanese)
- 11th International Workshop on Rewriting Techniques for Program Transformations and Evaluation, WPTE 2025 (Ivan Lanese)
- 33rd EACSL Annual Conference on Computer Science Logic, CSL 25 (Davide Sangiorgi)
- FSEN 2025 - Fundamentals of Software Engineering (Davide Sangiorgi)
- COORDINATION 2025 - International Conference on Coordination Models and Languages (Gianluigi Zavattaro)
- ICSOC 2025 - International Conference on Service-Oriented Computing (Gianluigi Zavattaro)
- ICWS 2025 - IEEE International Conference on Web Services (Gianluigi Zavattaro)
- ESOC 2025 - European Conference On Service-Oriented And Cloud Computing (Gianluigi Zavattaro)

**Member of conference steering committees**

- International Conference on Logic in Computer Science, LICS (Ugo Dal Lago)
- International Conference on Formal Structures for Computation and Deduction, FSCD (Ugo Dal Lago)
- International Conference on Reversible Computation, RC (Ivan Lanese)
- International Federated Conference on Distributed Computing Techniques, DisCoTec (Ivan Lanese, till June)
- International Conference on Concurrency Theory (Davide Sangiorgi)
- International Conference on Coordination Models and Languages (Gianluigi Zavattaro)

### 11.1.2 Journal

#### Member of the editorial boards

- Logical Methods in Computer Science (Ugo Dal Lago)
- Mathematical Structures in Computer Science (Ugo Dal Lago)
- Acta Informatica (Ugo Dal Lago and Davide Sangiorgi)
- TheoretiCS (Ugo Dal Lago)
- ACM Transactions on Computational Logic (Ugo Dal Lago)
- Distributed Computing (Davide Sangiorgi)
- Foundations and Trends in Programming Languages (Davide Sangiorgi)
- SN Computer Science (Davide Sangiorgi)

### 11.1.3 Invited talks

- 21st Conference on Computability in Europe, CiE 2025 (Ugo Dal Lago)
- 27th International Symposium on Principles and Practice of Declarative Programming, PPDP 2025 (Ugo Dal Lago)
- 7th International Workshop on Asynchronous Programming Models (Davide Sangiorgi)

### 11.1.4 Leadership within the scientific community

- The Microservices Community is a European-based, international, non-profit organization purposed to promote the development of microservices by bridging research, education, and innovation within and between businesses, universities, organizations and individuals. Members of OLAS have played active roles in the Community since its inception in 2019. The organization includes members from the Innopolis University (Russia), the Dortmund University of Applied Sciences and Arts (Germany), SINTEF and the University of Oslo (Norway), the University of Pisa and the University of Sassari (Italy), WSO2 (U.S.A.), and the Zurich University of Applied Sciences (Switzerland). Lanese and Giallorenzo are respectively part of the research and communication Community groups.
- Ivan Lanese has been chair of the IFIP (International Federation for Information Processing) WG6.1 on Architectures and Protocols for Distributed Systems till June.

### 11.1.5 Research administration

- Martin Avanzini is member of the "comité NICE" for welcoming external researchers (post-docs, "delegation").

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

- Saverio Giallorenzo
  - “Linguaggi di Programmazione”, 30 hours, 2nd year Bachelor, University of Bologna, Italy.
  - “Network Analysis”, 30 hours, 2nd year Master, University of Bologna, Italy.
- Ugo Dal Lago
  - “Introduction to Computability and Complexity”, 34 hours, 1st year Master, University of Bologna, Italy.
  - “Cryptography”, 40 hours, 2nd year Master, University of Bologna, Italy.

- “Introduction to Quantum Computing”, 20 hours, 1st year Master, University of Bologna, Italy.
- Ivan Lanese
  - “Architettura degli Elaboratori”, 34 hours, 1st year Bachelor, University of Bologna, Italy.
  - “Computational Methods for Bioinformatics”, 58 hours, 1st year Master, University of Bologna, Italy.
  - “Architetture Software a Microservizi”, 22 hours, 1st year Master, University of Bologna, Italy.
  - “Introduction to Quantum Computing”, 20 hours, 1st year Master, University of Bologna, Italy.
- Davide Sangiorgi
  - ‘Operating Systems’, 110 hours, 2nd year undergraduate program, University of Bologna, Italy.
  - ‘Computer abilities’, 16 hours, 2nd year Master in Medicine, University of Bologna, Italy.
- Gianluigi Zavattaro
  - “Algoritmi e Strutture di Dati”, 70 hours, 1st year Bachelor, University of Bologna, Italy.
  - “Scalable and Cloud Programming”, 50 hours, 2nd year Master, University of Bologna, Italy.

### 11.2.1 Juries

- Martin Avanzini has been member of the evaluation committee of the PhD thesis of Neea Rusch at Augusta University, Georgia, USA
- Ugo Dal Lago has been member of the evaluation committee of the PhD thesis of Adrienne Lancelot at IRIF, Paris, France
- Ivan Lanese has been member of the evaluation committee of the PhD thesis of Aurélie Kong Win Chang at INRIA Grenoble, France
- Ivan Lanese has been member of the evaluation committee of the PhD thesis of Andrea Esposito at Università di Urbino Carlo Bo, Italy

## 12 Scientific production

### 12.1 Publications of the year

#### International journals

- [1] L. Bacchiani, M. Bravetti, S. Giallorenzo, M. Gabbrielli, G. Zavattaro and S. P. Zingaro. ‘Proactive–reactive microservice architecture global scaling’. In: *Journal of Systems and Software* 220 (Feb. 2025), p. 112262. DOI: [10.1016/J.JSS.2024.112262](https://doi.org/10.1016/J.JSS.2024.112262). URL: <https://hal.science/hal-04887557> (cit. on p. 14).
- [2] 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. 15).
- [3] F. Callegati, S. Giallorenzo, A. Melis, S. Melloni, M. Prandini and A. Vannini. ‘Investigating Operational Technology Attacks as Code’. In: *Empirical Software Engineering* 30.6 (2nd Sept. 2025), p. 158. DOI: [10.1007/s10664-025-10713-2](https://doi.org/10.1007/s10664-025-10713-2). URL: <https://inria.hal.science/hal-05370961> (cit. on p. 15).
- [4] A. Colledan and U. Dal Lago. ‘Flexible Type-Based Resource Estimation in Quantum Circuit Description Languages’. In: *Proceedings of the ACM on Programming Languages* 9.POPL, Article 47 (9th Jan. 2025), pp. 1386–1416. DOI: [10.1145/3704883](https://doi.org/10.1145/3704883). URL: <https://hal.science/hal-05429267> (cit. on p. 16).

- [5] A. Colledan, U. D. Lago and N. Vazou. ‘Circuit Width Estimation via Effect Typing and Linear Dependency’. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 47.3 (15th Sept. 2025), pp. 1–35. DOI: [10.1145/3737282](https://doi.org/10.1145/3737282). URL: <https://hal.science/hal-05435683> (cit. on p. 16).
- [6] D. Davoli, M. Avanzini and T. Rezk. ‘Comprehensive Kernel Safety in the Spectre Era: Mitigations and Performance Evaluation’. In: *ACM Transactions on Privacy and Security* (12th June 2025). DOI: [10.1145/3743678](https://doi.org/10.1145/3743678). URL: <https://hal.science/hal-05173527> (cit. on p. 17).
- [7] G. Fabbretti, I. Lanese and J.-B. Stefani. ‘A Behavioral Theory for Distributed Systems with Weak Recovery’. In: *Logical Methods in Computer Science* 21.3 (1st July 2025). DOI: [10.46298/lmcs-21\(3:1\)2025](https://doi.org/10.46298/lmcs-21(3:1)2025). URL: <https://inria.hal.science/hal-05272017> (cit. on p. 13).
- [8] Z. Galal, F. Gavazzo, R. Treglia and G. Vanoni. ‘Monadic Intersection Types, Relationally and Ordered’. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 47.4 (Dec. 2025), pp. 1–51. DOI: [10.1145/377748](https://doi.org/10.1145/377748). URL: <https://hal.science/hal-05442856> (cit. on p. 16).
- [9] S. Gazza, R. Amadini, A. Brogi, A. d’Iapico, S. Forti, S. Giallorenzo, P. Plebani, F. Ponce, J. Soldani, M. Vitali and G. Zavattaro. ‘A Constraint-Based Approach to Optimise QoS- and Energy-Aware Cloud-Edge Application Deployments’. In: *ACM Transactions on Internet Technology* (30th July 2025). DOI: [10.1145/3757061](https://doi.org/10.1145/3757061). URL: <https://inria.hal.science/hal-05370970> (cit. on p. 14).
- [10] S. Giallorenzo, E. Sinagra, A. Trotta, L. Vergolini and A. Zanellati. ‘A Framework for Improving Social Inclusion Using Network Analysis and IoT-Based Contact Tracing’. In: *IEEE Transactions on Network Science and Engineering* (2025), pp. 1–22. DOI: [10.1109/TNSE.2025.3633148](https://doi.org/10.1109/TNSE.2025.3633148). URL: <https://inria.hal.science/hal-05370976> (cit. on p. 19).
- [11] S. Moreschini, S. Pour, I. Lanese, D. Balouek, J. Bogner, X. Li, F. Pecorelli, J. Soldani, E. Truyen and D. Taibi. ‘AI Techniques in the Microservices Life-Cycle: a Systematic Mapping Study’. In: *Computing* 107.4 (29th Mar. 2025), p. 100. DOI: [10.1007/s00607-025-01432-z](https://doi.org/10.1007/s00607-025-01432-z). URL: <https://hal.science/hal-05196358> (cit. on p. 14).
- [12] G. de Palma, S. Giallorenzo, A. Heideker, M. Trentin, A. Trotta and G. Zavattaro. ‘Distributed serverless function scheduling in ad-hoc drone networks’. In: *Ad Hoc Networks* 178 (Nov. 2025), p. 103951. DOI: [10.1016/j.adhoc.2025.103951](https://doi.org/10.1016/j.adhoc.2025.103951). URL: <https://inria.hal.science/hal-05370954> (cit. on p. 14).
- [13] G. de Palma, S. Giallorenzo, C. Laneve, J. Mauro, M. Trentin and G. Zavattaro. ‘Leveraging static analysis for cost-aware serverless scheduling policies’. In: *International Journal on Software Tools for Technology Transfer* (2nd Jan. 2025). DOI: [10.1007/s10009-024-00776-9](https://doi.org/10.1007/s10009-024-00776-9). URL: <https://hal.science/hal-04887650> (cit. on p. 14).
- [14] K. Sakayori and D. Sangiorgi. ‘Extensional and Non-extensional Functions as Processes’. In: *Logical Methods in Computer Science* Volume 21, Issue 3 (5th Sept. 2025). DOI: [10.46298/LMCS-21\(3:25\)2025](https://doi.org/10.46298/LMCS-21(3:25)2025). URL: <https://hal.science/hal-05371670> (cit. on p. 18).
- [15] T. Sekiyama, U. Dal Lago and H. Unno. ‘On Higher-Order Model Checking of Effectful Answer-Type-Polymorphic Programs’. In: *Proceedings of the ACM on Programming Languages* 9.OOPSLA2 (9th Oct. 2025), pp. 3726–3754. DOI: [10.1145/3763184](https://doi.org/10.1145/3763184). URL: <https://hal.science/hal-05431381> (cit. on p. 18).

#### Invited conferences

- [16] U. Dal Lago. ‘Counting Qubits and Gates: Resource Analysis in Quantum Programming Languages’. In: *PPDP 2025 - 27th International Symposium on Principles and Practice of Declarative Programming*. PPDP ’25: Proceedings of the 27th International Symposium on Principles and Practice of Declarative Programming 1. Rende, Italy: ACM, 10th Sept. 2025, pp. 1–3. DOI: [10.1145/3756907.3756908](https://doi.org/10.1145/3756907.3756908). URL: <https://hal.science/hal-05431383>.

**International peer-reviewed conferences**

- [17] W. Arnone and I. Lanese. ‘Tallulah, a Tool to Support the Axiomatic Approach to Causal-Consistent Reversibility’. In: *RC 2025 - 17th Conference on Reversible Computation*. Vol. 15716. Lecture Notes in Computer Science. Odense, Denmark: Springer Nature Switzerland, 22nd June 2025, pp. 1–8. DOI: [10.1007/978-3-031-97063-4\\_1](https://doi.org/10.1007/978-3-031-97063-4_1). URL: <https://inria.hal.science/hal-05399407> (cit. on p. 15).
- [18] M. Avanzini, G. Barthe, D. Davoli and B. Grégoire. ‘A Quantitative Probabilistic Relational Hoare Logic’. In: *ACM Digital Library*. POPL 2025 - 52nd ACM SIGPLAN Symposium on Principles of Programming Languages. Vol. 9. Proceedings of the ACM on Programming Languages POLP 2025. Denver (Colorado), United States, 20th Jan. 2025. DOI: [10.1145/3704876](https://doi.org/10.1145/3704876). URL: <https://inria.hal.science/hal-04834149> (cit. on p. 17).
- [19] M. Avanzini and A. Yamada. ‘Weighted Rewriting’. In: *Lecture Notes in Computer Science. Lecture Notes in Artificial Intelligence*. FroCoS 2025 - 15th International Symposium Frontiers of Combining Systems. Frontiers of Combining Systems 15th International Symposium, FroCoS 2025, Reykjavik, Iceland, September 29 – October 1, 2025, Proceedings LNCS-LNAI-15979. Reykjavik, Iceland: Springer Nature Switzerland, 15th Sept. 2026, pp. 191–208. DOI: [10.1007/978-3-032-04167-8\\_11](https://doi.org/10.1007/978-3-032-04167-8_11). URL: <https://inria.hal.science/hal-05380979> (cit. on p. 16).
- [20] S. Bliudze, G. de Palma, S. Giallorenzo, I. Lanese, G. Zavattaro and B. A. Zemtsop Ndadji. ‘Adaptable TeaStore’. In: *Electronic Proceedings in Theoretical Computer Science*. WACA 2025 - Workshop on Adaptable Cloud Architectures. Vol. 438. Lille, France, 29th Dec. 2025, pp. 1–14. DOI: [10.4204/EPTCS.438.1](https://doi.org/10.4204/EPTCS.438.1). URL: <https://inria.hal.science/hal-05446548> (cit. on p. 13).
- [21] M. Bravetti, L. Padovani and G. Zavattaro. ‘A Sound and Complete Characterization of Fair Asynchronous Session Subtyping’. In: *CONCUR 2025 - 36th International Conference on Concurrency Theory*. Aarhus, Denmark: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. DOI: [10.4230/LIPIcs.CONCUR.2025.11](https://doi.org/10.4230/LIPIcs.CONCUR.2025.11). URL: <https://inria.hal.science/hal-05437580>.
- [22] V. Choudhury and S. J. Gay. ‘The Duality of  $\lambda$ -Abstraction’. In: *ACM Digital Library*. POPL 2025 - 52nd ACM SIGPLAN Symposium on Principles of Programming Languages. Proceedings of the ACM on Programming Languages. Denver (Colorado), United States, 19th Jan. 2025. DOI: [10.1145/3704848](https://doi.org/10.1145/3704848). URL: <https://hal.science/hal-04838618> (cit. on p. 18).
- [23] T. Compagnucci, F. Callegati, S. Giallorenzo, A. Melis, S. Melloni and A. Vannini. ‘SAFARI: A Scalable Air-Gapped Framework for Automated Ransomware Investigation’. In: *IFIP Advances in Information and Communication Technology (Springer)*. IFIP SEC 2025 - ICT 40th IFIP International Information Security and Privacy Conference. Vol. AICT-745. ICT Systems Security and Privacy Protection : 40th IFIP International Conference, SEC 2025, Maribor, Slovenia, May 21–23, 2025, Proceedings, Part I. Maribor, Slovenia: Springer Nature, 16th May 2025, pp. 210–223. DOI: [10.1007/978-3-031-92882-6\\_15](https://doi.org/10.1007/978-3-031-92882-6_15). URL: <https://inria.hal.science/hal-05370994> (cit. on p. 14).
- [24] D. d’Ugo, S. Giallorenzo and S. Melloni. ‘Contrasting Crypto and Exfiltration Ransomware with Shamir’s Secret Sharing Data Flooding’. In: *IEEE Xplore*. TrustCom 2025 - 24th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. Guiyang, China, 14th Nov. 2025. URL: <https://inria.hal.science/hal-05371004> (cit. on p. 19).
- [25] U. Dal Lago, N. Hoshino and P. Pistone. ‘On the Metric Nature of (Differential) Logical Relations’. In: *FSCD 2025 - 10th International Conference on Formal Structures for Computation and Deduction*. Vol. 337. 15. Birmigham, United Kingdom: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. DOI: [10.4230/LIPIcs.FSCD.2025.15](https://doi.org/10.4230/LIPIcs.FSCD.2025.15). URL: <https://hal.science/hal-05431373> (cit. on p. 16).
- [26] A. Durier, D. Hirschhoff and D. Sangiorgi. ‘Proof Techniques for Behavioural Relations Based on Unique-Solutions of Equations and Inequations’. In: *Lecture Notes in Computer Science*. Rebeca for Actor Analysis in Action - Essays Dedicated to Marjan Sirjani. Lecture Notes in Computer Science LNCS-15560. Västerås, Sweden: Springer Nature Switzerland, 21st Mar. 2025, pp. 425–439. DOI: [10.1007/978-3-031-85134-6\\_19](https://doi.org/10.1007/978-3-031-85134-6_19). URL: <https://hal.science/hal-05371648> (cit. on p. 18).

- [27] G. Genovese, I. Lanese, C. Di Giusto, E. Tuosto and G. Vidal. ‘Choreographies for Program Understanding’. In: *Lecture notes in computer science. LNCS. FORTE 2025 - 45th International Conference on Formal Techniques for Distributed Objects, Components, and Systems. Formal Techniques for Distributed Objects, Components, and Systems LNCS-15732*. Lille, France: Springer Nature Switzerland, 11th June 2025, pp. 173–181. doi: [10.1007/978-3-031-95497-9\\_10](https://doi.org/10.1007/978-3-031-95497-9_10). URL: <https://inria.hal.science/hal-05399397> (cit. on p. 13).
- [28] S. Giallorenzo and F. Gorette. ‘Breadth-first Cycle Collection Reference Counting: Theory and a Rust Smart Pointer Implementation’. In: *ACM digital library. 40th ACM/SIGAPP Symposium on Applied Computing, SAC 2025*. Catania (IT), Italy, 31st Mar. 2025. doi: [10.1145/3672608.3707785](https://doi.org/10.1145/3672608.3707785). URL: <https://hal.science/hal-04887627> (cit. on p. 19).
- [29] S. Giallorenzo, S. Melloni and P. Sami. ‘Reliable and Robust Watermarking for Data Flooding against Ransomware Random Techniques’. In: *IEEE Xplore. TrustCom 2025 - 24th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. Guiyang, China, 14th Nov. 2025. URL: <https://inria.hal.science/hal-05371011> (cit. on p. 19).
- [30] D. Hirschhoff, I. Quémerais and D. Sangiorgi. ‘First-order store and visibility in name-passing calculi’. In: *36th International Conference on Concurrency Theory (CONCUR 2025)*. Aarhus (Denmark), Denmark: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi: [10.4230/LIPIcs](https://doi.org/10.4230/LIPIcs). URL: <https://hal.science/hal-05033251> (cit. on p. 18).
- [31] U. D. Lago, Z. Galal and G. Giusti. ‘On Computational Indistinguishability and Logical Relations’. In: *Lecture notes in computer science. APLAS 2024 - 22nd Asian Symposium on Programming Languages and Systems. Vol. LNCS-15194. Programming Languages and Systems : 22nd Asian Symposium, APLAS 2024*, Kyoto, Japan, October 22-24, 2024, Proceedings. Kyoto, Japan: Springer Nature Singapore, 28th Oct. 2025, pp. 241–263. doi: [10.1007/978-981-97-8943-6\\_12](https://doi.org/10.1007/978-981-97-8943-6_12). URL: <https://inria.hal.science/hal-04834943>.
- [32] I. Lanese, U. Dal Lago and V. Choudhury. ‘Towards Quantum Multiparty Session Types’. In: *PLanQC 2025 - Fifth International Workshop on Programming Languages for Quantum Computing*. Denver (Colorado), United States, 25th Jan. 2025. URL: <https://hal.science/hal-04838281> (cit. on p. 17).
- [33] L. Padovani and G. Zavattaro. ‘Fair Termination of Asynchronous Binary Sessions’. In: *Leibniz International Proceedings in Informatics (LIPIcs). ECOOP 2025 - European Conference on Object-Oriented Programming. 39th European Conference on Object-Oriented Programming (ECOOP 2025) LIPIcs-39*. Bergen, Norway: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi: [10.4230/LIPIcs.ECOOP.2025.24](https://doi.org/10.4230/LIPIcs.ECOOP.2025.24). URL: <https://inria.hal.science/hal-05437581> (cit. on pp. 7, 18).
- [34] S. Pal, R. Guanciale, I. Lanese, E. Tuosto and M. Clo. ‘Pomsets for Process Management: A Healthcare Case Study’. In: *ICTAC 2025 - International Colloquium on Theoretical Aspects of Computing. Vol. 16237. Lecture Notes in Computer Science*. Marrakech, Morocco: Springer Nature Switzerland, 23rd Nov. 2026, pp. 378–395. doi: [10.1007/978-3-032-11176-0\\_22](https://doi.org/10.1007/978-3-032-11176-0_22). URL: <https://inria.hal.science/hal-05435263> (cit. on p. 13).
- [35] G. de Palma, S. Giallorenzo, I. Lanese and G. Zavattaro. ‘Adaptable TeaStore: A Choreographic Approach’. In: *WACA 2025 - Workshop on Adaptable Cloud Architectures. Vol. 438*. Lille, France, 29th Dec. 2025, pp. 79–99. doi: [10.4204/EPTCS.438.5](https://doi.org/10.4204/EPTCS.438.5). URL: <https://inria.hal.science/hal-05447341> (cit. on p. 13).
- [36] G. de Palma, S. Giallorenzo, J. Mauro, M. Trentin and G. Zavattaro. ‘Affinity-aware Serverless Function Scheduling’. In: *IEEE Xplore. 22nd IEEE International Conference on Software Architecture*. Odense, Denmark, 31st Mar. 2025. URL: <https://hal.science/hal-04887637> (cit. on p. 13).
- [37] G. de Palma, S. Giallorenzo, J. Mauro, M. Trentin and G. Zavattaro. ‘Reachability Analysis of Function-as-a-Service Scheduling Policies’. In: *Lecture notes in computer science. LNCS. IFM 2025 - 20th International Conference on Integrated Formal Methods. Integrated Formal Methods : 20th International Conference, iFM 2025*, Paris, France, November 19–21, 2025, Proceedings LNCS-16194. Paris, France, 17th Nov. 2025. doi: [10.1007/978-3-032-10794-7\\_12](https://doi.org/10.1007/978-3-032-10794-7_12). URL: <https://inria.hal.science/hal-05370998> (cit. on p. 14).

- [38] G. D. Palma, S. Giallorenzo, J. Mauro, M. Trentin and G. Zavattaro. ‘Function-as-a-Service Allocation Policies Made Formal’. In: *Lecture notes in computer science*. ISO/FA 2024 - 12th International Symposium ON Leveraging Applications of Formal Methods, Verification and Validation. REoCAS Colloquium in Honor of Rocco De Nicola. Vol. LNCS-15219. Leveraging Applications of Formal Methods, Verification and Validation. REoCAS Colloquium in Honor of Rocco De Nicola. Crete, France: Springer Nature Switzerland, 9th Oct. 2025, pp. 306–321. doi: [10.1007/978-3-031-73709-1\\_19](https://doi.org/10.1007/978-3-031-73709-1_19). URL: <https://inria.hal.science/hal-04826399> (cit. on p. 14).
- [39] B. Raimondi, S. Giallorenzo and M. Gabbrielli. ‘Affordably Fine-tuned LLMs Provide Better Answers to Course-specific MCQs’. In: *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, SAC 2025, Catania, Italy, March 31-April 4, 2025*. 40th ACM/SIGAPP Symposium on Applied Computing, SAC 2025. Catania (IT), Italy, 31st Mar. 2025. doi: [10.1145/3672608.3707732](https://doi.org/10.1145/3672608.3707732). URL: <https://hal.science/hal-04887593> (cit. on p. 19).
- [40] D. Sangiorgi. ‘An Abstract Account of Up-to Techniques for Inductive Behavioural Relations’. In: *Lecture notes in computer science*. ISO/FA 2024 - 12th International Symposium Leveraging Applications of Formal Methods Verification and Validation. REoCAS Colloquium in Honor of Rocco De Nicola. Vol. LNCS-15219. Leveraging Applications of Formal Methods, Verification and Validation. REoCAS Colloquium in Honor of Rocco De Nicola. Crete Island, Greece: Springer Nature Switzerland, 9th Oct. 2025, pp. 62–74. doi: [10.1007/978-3-031-73709-1\\_5](https://doi.org/10.1007/978-3-031-73709-1_5). URL: <https://hal.science/hal-04827142> (cit. on p. 18).

### Scientific book chapters

- [41] I. Lanese, C. A. Mezzina and M. Vassor. ‘Bounded Reversibility in HO $\pi$ ’. In: *Components Operationally: Reversibility and System Engineering. Essays Dedicated to Jean-Bernard Stefani on the Occasion of His 65th Birthday*. Vol. 16065. Lecture Notes in Computer Science. Springer Nature Switzerland, 19th Oct. 2025, pp. 24–45. doi: [10.1007/978-3-031-99717-4\\_2](https://doi.org/10.1007/978-3-031-99717-4_2). URL: <https://hal.science/hal-05332969> (cit. on p. 15).
- [42] G. de Palma, S. Giallorenzo, J. Mauro, M. Trentin and G. Vjerdha. ‘Towards a Framework for Transitioning from Monolith to Serverless’. In: *The Combined Power of Research, Education, and Dissemination: Essays Dedicated to Tiziana Margaria on the Occasion of Her 60th Birthday*. Vol. LNCS-15240. Lecture Notes in Computer Science. 1st Jan. 2025, pp. 167–182. doi: [10.1007/978-3-031-73887-6\\_13](https://doi.org/10.1007/978-3-031-73887-6_13). URL: <https://hal.science/hal-04887566> (cit. on p. 14).

### Reports & preprints

- [43] V. Choudhury and W. Wong. *Symmetries in Sorting*. 2025. URL: <https://hal.science/hal-05437636>.

## 12.2 Cited publications

- [44] D. Berardi, S. Giallorenzo, A. Melis, S. Melloni, L. Onori and M. Prandini. ‘Data Flooding against Ransomware: Concepts and Implementations’. In: *Computers & Security* 131 (Aug. 2023), p. 103295. doi: [10.1016/j.cose.2023.103295](https://doi.org/10.1016/j.cose.2023.103295). URL: <https://hal.science/hal-04316302> (cit. on p. 19).
- [45] T. J. Biggerstaff, B. G. Mitbender and D. E. Webster. ‘The Concept Assignment Problem in Program Understanding’. In: *Proceedings of the 15th International Conference on Software Engineering, Baltimore, Maryland, USA, May 17-21, 1993*. Ed. by V. R. Basili, R. A. DeMillo and T. Katayama. IEEE Computer Society / ACM Press, 1993, pp. 482–498 (cit. on p. 13).
- [46] M. Dalla Preda, M. Gabbrielli, S. Giallorenzo, I. Lanese and J. Mauro. ‘Dynamic Choreographies: Theory And Implementation’. In: *Log. Methods Comput. Sci.* 13.2 (2017). doi: [10.23638/LMCS-13\(2:1\)2017](https://doi.org/10.23638/LMCS-13(2:1)2017) (cit. on p. 13).

- [47] J. von Kistowski, S. Eismann, N. Schmitt, A. Bauer, J. Grohmann and S. Kounev. ‘TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research’. In: *26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, MASCOTS 2018, Milwaukee, WI, USA, September 25-28, 2018*. IEEE Computer Society, 2018, pp. 223–236. DOI: [10.1109/MASCOTS.2018.00030](https://doi.org/10.1109/MASCOTS.2018.00030). URL: <https://doi.org/10.1109/MASCOTS.2018.00030> (cit. on p. 13).
- [48] T. Knapik, D. Niwinski and P. Urzyczyn. ‘Higher-Order Pushdown Trees Are Easy’. In: *Foundations of Software Science and Computation Structures, 5th International Conference, FOSSACS 2002. Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002 Grenoble, France, April 8-12, 2002, Proceedings*. Ed. by M. Nielsen and U. Engberg. Vol. 2303. Lecture Notes in Computer Science. Springer, 2002, pp. 205–222. DOI: [10.1007/3-540-45931-6\\_15](https://doi.org/10.1007/3-540-45931-6_15). URL: [https://doi.org/10.1007/3-540-45931-6\\_15](https://doi.org/10.1007/3-540-45931-6_15) (cit. on p. 17).
- [49] N. Kobayashi. ‘Types and higher-order recursion schemes for verification of higher-order programs’. In: *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*. Ed. by Z. Shao and B. C. Pierce. ACM, 2009, pp. 416–428. DOI: [10.1145/1480881.1480933](https://doi.org/10.1145/1480881.1480933). URL: <https://doi.org/10.1145/1480881.1480933> (cit. on p. 17).
- [50] C. L. Ong. ‘On Model-Checking Trees Generated by Higher-Order Recursion Schemes’. In: *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*. IEEE Computer Society, 2006, pp. 81–90. DOI: [10.1109/LICS.2006.38](https://doi.org/10.1109/LICS.2006.38). URL: <https://doi.org/10.1109/LICS.2006.38> (cit. on p. 17).