RESEARCH CENTRE

Inria Centre at Université Côte d'Azur

IN PARTNERSHIP WITH:

Université de Bologne (Italie)

2023

ACTIVITY REPORT

Team

FOCUS

Foundations of Component-based Ubiquitous Systems

Inria teams are typically groups of researchers working on the definition of a common project, and objectives, with the goal to arrive at the creation of a project-team. Such project-teams may include other partners (universities or research institutions)

IN COLLABORATION WITH: Dipartimento di Informatica - Scienza e Ingegneria (DISI), Universita' di Bologna

DOMAIN

Networks, Systems and Services, Distributed Computing

THEME

Distributed programming and Software engineering

Inria

Contents

Team FOCUS

Creation of the Team: 2022 June 22

Keywords

Computer sciences and digital sciences

A1. – Architectures, systems and networks

A1.3. – Distributed Systems

A1.4. – Ubiquitous Systems

A2.1.1. – Semantics of programming languages

A2.1.6. – Concurrent programming

A2.1.7. – Distributed programming

A2.4.3. – Proofs

Other research topics and application domains

B6.1. – Software industry

B6.3. – Network functions

B6.4. – Internet of things

B9.5.1. – Computer science

1 Team members, visitors, external collaborators

Research Scientist

- Martin Avanzini [INRIA, Researcher, until Sep 2023]

Faculty Members

- Davide Sangiorgi [Team leader, UNIV BOLOGNE, Professor, HDR]
- Mario Bravetti [UNIV BOLOGNE, Associate Professor, until Sep 2023]
- Ugo Dal Lago [UNIV BOLOGNE, Associate Professor, until Sep 2023]
- Maurizio Gabbrielli [UNIV BOLOGNE, Professor, until Sep 2023]
- Saverio Giallorenzo [UNIV BOLOGNE, Associate Professor]
- Ivan Lanese [UNIV BOLOGNE, Professor, until Sep 2023]
- Cosimo Laneve [UNIV BOLOGNE, Professor, until Sep 2023]
- Simone Martini [UNIV BOLOGNE, Professor, until Sep 2023, HDR]
- Fabio Zanasi [UNIV BOLOGNE, Professor, from Mar 2023 until Sep 2023]
- Gianluigi Zavattaro [UNIV BOLOGNE, Associate Professor, until Sep 2023]

Post-Doctoral Fellows

- Vikraman Choudhury [UNIV BOLOGNE, from Dec 2023]
- Michael Lodi [UNIV BOLOGNE, Post-Doctoral Fellow, until Sep 2023]
- Paolo Pistone [UNIV BOLOGNE, Post-Doctoral Fellow]
- Ken Sakayori [UNIV BOLOGNE, Post-Doctoral Fellow, until Oct 2023]
- Gabriele Vanoni [INRIA, Post-Doctoral Fellow, until Aug 2023]
- Stefano Zingaro [UNIV BOLOGNE, Post-Doctoral Fellow, until Sep 2023]

PhD Students

- Melissa Antonelli [UNIV BOLOGNE, until Jan 2023]
- Andrea Colledan [UNIV BOLOGNE, until Sep 2023]
- Pietro Lami [UNIV BOLOGNE, from Sep 2023]
- Sourabh Pal [UNIV BOLOGNE, from Oct 2023]
- Adele Veschetti [UNIV BOLOGNE, until Apr 2023]

External Collaborators

- Claudio Guidi [Italiana Software, until Sep 2023]
- Daniel Hirschkoff [ENS DE LYON, until Sep 2023]
- Fabrizio Montesi [UNIV Southern Denmark, until Sep 2023]

2 Overall objectives

Ubiquitous Computing refers to the situation in which computing facilities are embedded or integrated into everyday objects and activities. Networks are large-scale, including both hardware devices and software agents. The systems are highly mobile and dynamic: programs or devices may move and often execute in networks owned and operated by others; new devices or software pieces may be added; the operating environment or the software requirements may change. The systems are also heterogeneous and open: the pieces that form a system may be quite different from each other, built by different people or industries, even using different infrastructures or programming languages; the constituents of a system only have a partial knowledge of the overall system, and may only know, or be aware of, a subset of the entities that operate on the system.

A prominent recent phenomenon in Computer Science is the emergence of interaction and communication as key architectural and programming concepts. This is especially visible in ubiquitous systems. Complex distributed systems are being thought of and designed as structured composition of computational units, usually referred to as *components*. These components are supposed to interact with each other and such interactions are supposed to be orchestrated into conversations and dialogues. In the remainder, we will write *CBUS* for Component-Based Ubiquitous Systems.

In CBUS, the systems are complex. In the same way as for complex systems in other disciplines, such as physics, economics, biology, in CBUS theories are needed that allow us to understand the systems, to design or program them, and to analyze them.

Focus investigates the semantic foundations for CBUS. The foundations are intended as instrumental to formalizing and verifying important computational properties of the systems, as well as to proposing linguistic constructs for them. Prototypes are developed to test the implementability and usability of the models and the techniques. Throughout our work, 'interaction' and 'component' are central concepts.

The members of the project have a solid experience in algebraic and logical models of computation, and related techniques, and this is the basis for our study of ubiquitous systems. 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 Research program

3.1 Foundations 1: Models

The objective of Focus is to develop concepts, techniques, and possibly also tools, that may contribute to the analysis and synthesis of CBUS. Fundamental to these activities is *modeling*. Therefore designing, developing and studying computational models appropriate for CBUS is a central activity of the project. 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 are in the process calculi (e.g., the π-calculus) and λ-calculus tradition. Such models, with their emphasis on algebra, address properly compositionality—a central property in our approach to problems. Accordingly, the techniques we employ are mainly operational techniques based on notions of behavioral equivalence, and techniques based on algebra, mathematical logics, and type theory.

3.2 Foundations 2: Foundational calculi and interaction

Modern distributed systems have witnessed a clear shift towards interaction and conversations as basic building blocks for software architects and programmers. The systems are made by components, that are supposed to interact and carry out dialogues in order to achieve some predefined goal; Web services are a good example of this. Process calculi are models that have been designed precisely with the goal of understanding interaction and composition. The theory and tools that have been developed on top of process calculi can set a basis with which CBUS challenges can be tackled. Indeed, industrial proposals of languages for Web services such as BPEL (Business Process Execution Language) are strongly inspired by process calculi, notably the π-calculus.

3.3 Foundations 3: Type systems and logics

Type systems and logics for reasoning on computations are among the most successful outcomes in the history of the research in λ-calculus and (more recently) in process calculi. Type systems can also represent a powerful means of specifying dialogues among components of CBUS. For instance—again referring to Web services—current languages for specifying interactions only express basic connectivity, ignoring causality and timing aspects (e.g., an intended order on the messages), and the alternative is to use Turing Complete languages that are however undecidable. Types can come in handy here: they can express causality and order information on messages [36, 35, 37], while remaining decidable systems.

3.4 Foundations 4: Implicit computational complexity

A number of elegant and powerful results have been obtained in implicit computational complexity concerning the λ-calculus, where ideas from Linear Logics enable a fine-grained control over computations. This experience can be profitable when tackling issues of CBUS related to resource consumption, such as resource allocation, access to resources, certification of bounds on resource consumption (e.g., ensuring that a service will answer to a request in time polynomial with respect to the size of the input data).

4 Application domains

4.1 Ubiquitous Systems

The main application domain for Focus are ubiquitous systems, i.e. systems whose distinctive features are: mobility, high dynamicity, heterogeneity, variable availability (the availability of services offered by the constituent parts of a system may fluctuate, and similarly the guarantees offered by single components may not be the same all the time), open-endedness, complexity (the systems are made by a large number of components, with sophisticated architectural structures). In Focus we are particularly interested in the following aspects.

- *Linguistic primitives* for programming dialogues among components.
- *Contracts* expressing the functionalities offered by components.
- *Adaptability and evolvability* of the behavior of components.
- *Verification* of properties of component systems.
- Bounds on component *resource consumption* (e.g., time and space consumed).

4.2 Service Oriented Computing and Cloud Computing

Today the component-based methodology often refers to Service Oriented Computing. This is a specialized form of component-based approach. According to W3C, a service-oriented architecture is “a set of components which can be invoked, and whose interface descriptions can be published and discovered”. In the early days of Service Oriented Computing, the term “services” was strictly related to that of Web Services. Nowadays, it has a much broader meaning as exemplified by the XaaS (everything as a service) paradigm: based on modern virtualization technologies, Cloud computing offers the possibility to build sophisticated service systems on virtualized infrastructures accessible from everywhere and from any kind of computing device. Such infrastructures are usually examples of sophisticated service oriented architectures that, differently from traditional service systems, should also be capable to elastically adapt on demand to the user requests.

5 Highlights of the year

- The Focus project-team arrived at the end of its life time, and gave rise to the new OLAS project-team (which started on October 1st, 2023).

- The Marie-Curie postdoc fellowship ReGraDe-CS of Vikraman Choudhury (supervised by Ivan Lanese) has been accepted and started this year.
- From February 24th 2023, Michael Lodi is Junior assistant professor (fixed-term), fully financed by the Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High Performance Computing, Big Data and Quantum Computing. Next Generation EU (PNRR – M4C2 – I1.4 – CN00000013 – CUP J33C22001170001).

5.1 Awards

- *Best artifact award* at COORDINATION 2023 with the paper "Giallorenzo, S., Montesi, F., Peressotti, M., Rademacher, F., & Unwerawattana, N. (2023). JoT: A Jolie Framework for Testing Microservices. In S.-S. Jongmans & A. Lopes (Eds.), Coordination Models and Languages".
- *Best student award* at Microservices 2023 with the paper "De Palma, G., Giallorenzo, S., Trentin, M., & Vjerdha, G. (2023). A Framework for Bridging the Gap between Monolithic and Serverless Programming".
- *Distinguished paper award* at LICS (Logic in Computer Science) 2023 with the paper "Sakayori, K., & Sangiorgi, D. (2023). Extensional and Non-extensional Functions as Processes".
- Gabriele Vanoni won with his PhD thesis the *Ackermann Award for Outstanding Dissertation in Logic in Computer Science 2023* (awarded by EACSL – the European Association for Computer Science Logic), the *E.W. Beth Dissertation Prize 2023* (awarded by FoLLI – the Association for Logic, Language and Information) and the *best Italian PhD thesis in theoretical computer science 2023 award* (awarded by the Italian Chapter of EATCS – the European Association for Theoretical Computer Science).

6 New software, platforms, open data

6.1 New software

6.1.1 JOLIE

Name: Java Orchestration Language Interpreter Engine

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 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 concepts around Focus activities.

Release Contributions: Release 1.11 includes almost 500 commits and 14 different contributors and it is one of the biggest releases so far!

The main changes include: - New call-expressions. You can now call solicit-response operations in expressions, for example (in the condition of a conditional): ‘if(trim@stringUtils("Hello ") == "Hello") println@console("Hooray!")()‘ - New if-expressions, for example: ‘x = if(condition) expr1 else expr2.‘ - More modern alternative throw syntax: ‘throw fault(data)‘. - New engine for error reporting. For example, when you misspell a keyword or a type name, Jolie now looks at the context and tries to guess which word you were trying to type by using edit distance. - New templating engine for HTTP. - Improved management of open channels. Jolie services with many interactions with the same clients/services now consume noticeably fewer connections and RAM. - New standard library operations, including a library for assertions, a library for vector manipulation, operations for URL encoding and decoding, an operation for string interpolation, and support for reading lines synchronously from the terminal. - Support for Mustache templates (see packages/mustache.ol). - Improved tracing messages for protocol internals. - Many bugfixes to HTTP, SOAP, XML, and JSON data handling. - is_defined now correctly returns true for defined variables of type void.

URL: http://www.jolie-lang.org/

Contact: Saverio Giallorenzo

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

6.1.2 NightSplitter

Keyword: Constraint-based programming

Functional Description: Nightsplitter deals with the group preference optimization problem. We propose to split users into subgroups trying to optimize members' satisfaction as much as possible. In a large city with a huge volume of activity information, designing subgroup activities and avoiding time conflict is a challenging task. Currently, the Demo is available only for restaurant and movie activities in the city of Paris.

URL: http://cs.unibo.it/t.liu/nightsplitter/

Contact: Maurizio Gabbrielli

6.1.3 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 misbehaviour to the bug.

URL: https://github.com/mistupv/cauder

Publications: hal-03005383v1, hal-01912894v1, hal-02313745v1

Contact: Ivan Lanese

Participant: Ivan Lanese

Partner: Universitat Politècnica de València

6.1.4 SUNNY-AS

Name: SUNNY FOR ALGORITHM SELECTION

Keywords: Optimisation, Machine learning

Functional Description: SUNNY-AS is a portfolio solver derived from SUNNY-CP for Algorithm Selection Problems (ASLIB). The goal of SUNNY-AS is to provide a flexible, configurable, and usable portfolio solver that can be set up and executed just like a regular individual solver.

URL: https://github.com/lteu/oasc

Contact: Maurizio Gabbrielli

6.1.5 eco-imp

Name: Expected Cost Analysis for Imperative Programs

Keywords: Software Verification, Automation, Runtime Complexity Analysis, Randomized algorithms

Functional Description: Eco-imp was originally envisaged as a cost analyzer for probabilistic and non-deterministic imperative programs. Particularly, it features dedicated support for sampling from distributions, and can thereby accurately reason about the average case complexity of randomized algorithms, in a fully automatic fashion. The tool is based on an adaptation of the ert-calculus of Kaminski et al., extended to the more general setting of cost analysis where the programmer is free to specify a (non-uniform) cost measure on programs. The main distinctive feature of eco-imp, though, is the combination of this calculus with an expected value analysis. This provides the glue to analyze program components in complete independence, that is, the analysis is modular and thus scalable.

In its most recent version, the tool has been completely overhauled. On the one hand, it is capable of reasoning about a broader set of events, through the analysis of expected outcomes. On the other hand, the new version also supports recursive procedures.

URL: http://www-sop.inria.fr/members/Martin.Avanzini/software/eco-imp/

Publications: hal-04345663v1, hal-03013544

Contact: Martin Avanzini

6.1.6 PRISM+

Keyword: Stochastic process

Functional Description: PRISM is a probabilistic model checker, a tool for formal modelling and analysis of systems that exhibit random or probabilistic behaviour. We extend the language in order to model the Bitcoin system. The tool now supports three dynamic data types: block, ledger and list. As a consequence, it is now possible to perform simulations and analyze transient probabilities, i.e. probabilities that are dependent on time, for the Bitcoin protocol. It has been used to understand how the system changes during the execution and to analyze the probabilities of reaching an inconsistent state in different settings.

URL: https://github.com/adeleveschetti/bitcoin-analysis

Contact: Adele Veschetti

6.1.7 Tquery

Keywords: Ephemeral Data, Microservices, Big data, Querying

Scientific Description: The adoption of edge/fog systems and the introduction of privacy-preserving regulations compel the usage of tools for expressing complex data queries in an ephemeral way—ensuring the queried data does not persist.

Database engines partially address this need, as they provide domain-specific languages for querying data. Unfortunately, using a database in an ephemeral setting has inessential issues related to throughput bottlenecks, scalability, dependency management, and security (e.g., query injection). Moreover, databases can impose specific data structures and data formats, which can hinder the development of microservice architectures that integrate heterogeneous systems and handle semi-structured data.

Tquery is the first query framework designed for ephemeral data handling in microservices. Tquery joins the benefits of a technology-agnostic, microservice-oriented programming language, Jolie, and of one of the most widely-used query languages for semi-structured data in microservices, the MongoDB aggregation framework. With Tquery, users express in a terse syntax how to collect data from heterogeneous sources and how to query it in local memory, defining pipelines of high-level operators. The development of Tquery follows a "cleanroom software engineering process", based on the definition of a theory for querying semi-structured data compatible with Jolie and inspired by a consistent variant of the key operators of the MongoDB aggregation framework.

Functional Description: Tquery is a query framework integrated into the Jolie language for the data handling/querying of Jolie trees.

Tquery is based on a tree-based instantiation (language and semantics) of MQuery, a formalisation of a sound fragment of the Aggregation Framework, the query language of the most popular document-oriented database: MongoDB.

Tree-shaped documents are the main format in which data flows within modern digital systems - e.g., eHealth, the Internet-of-Things, and Edge Computing. Tquery is particularly suited to develop real-time, ephemeral scenarios, where data shall not persist in the system.

Release Contributions: Stable release of Tquery, which includes the following operators: Match, Unwind, Project, Group, Join, and Pipeline.

In 2023 there were no updates.

URL: https://github.com/jolie/tquery

Contact: Saverio Giallorenzo

Partner: University of Southern Denmark

6.1.8 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 platform. 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 contraints 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

6.1.9 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: First release

URL: https://www.choral-lang.org/

Contact: Saverio Giallorenzo

Participants: Saverio Giallorenzo, Fabrizio Montesi

Partner: University of Southern Denmark

6.1.10 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 -> 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 couple gateways, and (d) check well-formedness conditions ensuring that the system of participants obtained via projection behaves well.

URL: https://github.com/lanese/corinne-3

Publication: hal-03468190

Contact: Ivan Lanese

Partner: Gran Sasso Science Institute

6.1.11 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: First release of Ranflood.

At the moment, Ranflood supports three types of flooding strategies:

- random: based on the flooding of a given location with randomly generated files.

- on-the-fly: a copy-based strategy, where the generation of files uses copies of actual files found at a flooding location. File replication adds a layer of defence as it helps to increase the likelihood of preserving the users' files by generating additional, valid copies that might escape the ransomware. This strategy introduces a new snapshoot action that precedes the flooding one, which saves a list of the valid files with their digest signature (e.g., MD5), so that the flooding operations can use the signature as an integrity verification to skip the encrypted files.

- shadow: another copy-based strategy that increases the efficiency of the on-the-fly strategy by preserving archival copies of the files of the user.

URL: https://ranflood.netlify.app/

Contact: Saverio Giallorenzo

7 New results

7.1 Service-oriented and Cloud Computing

Participants: Mario Bravetti, Saverio Giallorenzo, Ivan Lanese, Gianluigi Zavattaro, Stefano Pio Zingaro.

In Service-Oriented Computing (SOC), programmers specify systems made of interacting and collaborating components by describing the "local behavior" of each of them (i.e., the way one component may interact with the others), and/or the "global behavior" of the system (i.e., the expected interaction protocols that should take place within the system).

From the language perspective, orchestration and choreography respectively address the programming of "local" and "global" behaviors. Regarding applications, where usually SOC meets Cloud Computing, we find two state-of-the-art architectural styles. Microservices are a revisitation of service-oriented architectures where fine-grained, loosely coupled, independent services help developers assemble reusable, flexible, and scalable architectures. Serverless is a programming style and deployment technique where users program Cloud applications in terms of stateless functions, which execute and scale in proportion to inbound requests.

7.1.1 Foundations and Implementations for Orchestration and Choreography

Communication is an essential element of modern software, yet the programming and analysis of communicating systems are difficult tasks. A reason for this difficulty is the lack of compositional mechanisms that preserve relevant communication properties.

In [12], we present a comprehensive treatment of the case of synchronous communications. We consider both symmetric synchronous communications and asymmetric synchronous communications (where senders decide independently which message should be exchanged). The composition mechanism preserves different properties under different conditions depending on the considered type of synchronous communication. We show here that the preservation of lock freedom requires an additional condition on gateways for asymmetric communication. Such a condition is also needed for the preservation of deadlock freedom, lock freedom or strong lock freedom for symmetric communications. This is not needed, instead, for the preservation of either deadlock freedom or strong lock freedom with asymmetric interactions.

We also worked on formalizations and implementations of choreographic languages.

In [11], we introduce a meta-model based on formal languages, dubbed formal choreographic languages, to study message-passing systems. Our framework allows us to generalize standard constructions from the literature and to compare them. In particular, we consider notions such as global view, local view, and projections from the former to the latter. The correctness of local views projected from global views is characterized in terms of a closure property. We consider a number of communication properties – such as (dead)lock-freedom – and give conditions on formal choreographic languages to guarantee them. Finally, we show how formal choreographic languages can capture existing formalisms; specifically we consider communicating finite-state machines, choreography automata, and multiparty session types. Notably, formal choreographic languages, differently from most approaches in the literature, can naturally model systems exhibiting non-regular behavior.

In the paradigm of choreographic programming, choreographies are programs that can be compiled into executable implementations. Choreographic Programming originated primarily in the context of process calculi, with preliminary work done to establish its foundations and experiment with implementations.

In [16], we present Choral, the first choreographic programming language based on mainstream abstractions. The key idea in Choral is a new notion of data type able to express data distribution over

different participants. We use this idea to reconstruct the paradigm of choreographic programming through object-oriented abstractions. Choreographies are classes, and instances of choreographies are objects with states and behaviors implemented collaboratively by the participants. Choral comes with a compiler that, given a choreography, generates an implementation for each of its roles. These implementations are libraries in pure Java, whose types are under the control of the Choral programmer. Developers can then modularly compose these libraries in their programs, to participate correctly in choreographies. Choral is the first incarnation of choreographic programming offering such modularity, which finally connects more than a decade of research on the paradigm to practical software development. The integration of choreographic and object-oriented programming yields other powerful advantages, where the features of one paradigm benefit the other in ways that go beyond the sum of the parts. On the one hand, the high-level abstractions and static checks from the world of choreographies can be used to write concurrent and distributed object-oriented software more concisely and correctly. On the other hand, we obtain a much more expressive choreographic language from object-oriented abstractions than in previous work. This expressiveness supports writing more reusable and flexible choreographies. For example, object passing makes Choral the first higher-order choreographic programming language, whereby one can parametrize choreographies over other choreographies without the need for central coordination. We also extend method overloading to a new dimension: specialization based on data location. The integration of overloading, together with subtyping and generics, allows Choral to elegantly support user-defined communication mechanisms and middleware.

7.1.2 Microservices, Serverless, and Cloud Deployment

The service-oriented programming language Jolie is a long-standing project within Focus.

In [17] we introduce LEMMA2Jolie, a tool for translating domain models of microservice architectures given in LEMMA into concrete APIs of microservices in the Jolie programming language. The tool combines the state of the art for the design and implementation of microservices: developers can use Domain-Driven Design (DDD) for the construction of the domain models of a microservice architecture, and then automatically transition to a service-oriented programming language that provides native linguistic support for implementing the behavior of each microservice. In [28] we formally define and integrate into the LEMMA2Jolie tool a translation of domain and service models. As a result, LEMMA2Jolie now supports a software development process whereby one can design microservice architectures in collaboration with domain experts in LEMMA, and then automatically translate the design into programmable Jolie APIs.

Another work related to Jolie is JoT [29], a testing framework for Microservice Architectures (MSAs) 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 for MSAs whose services use different technology stacks, b) writing tests that involve multiple services, and c) reusing tests of the same MSA 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 MSA under test. The language for writing JoT tests is Jolie, which provides constructs that support technology agnosticism and the definition of terse test behaviors.

7.1.3 Other Important Results Regarding the Programming of Serverless Scheduling Policies.

APP (and variants) is a platform-agnostic declarative language developed within Focus that allows serverless platforms to support multiple scheduling logics. Indeed, proprietary and open-source serverless platforms follow opinionated, hardcoded scheduling policies to deploy the functions to be executed over the available workers. Such policies may decrease the performance and the security of the application due to locality issues (e.g., functions executed by workers far from the databases to be accessed). APP helps in overcoming these limitations. However, defining the “right” scheduling policy in APP is a non-trivial task since it often requires rounds of refinement involving knowledge of the underlying infrastructure, guesswork, and empirical testing. In [32] we present a gentle introduction to APP through an illustrative application developed over several incremental steps to help developers identify and specify relevant properties of their serverless architectures. In [31], we start investigating how one can use static analysis to inform APP scheduling function policies for selecting the best-performing workers at function allocation.

We substantiate our proposal by presenting a pipeline able to extract cost equations from functions' code, synthesizing cost expressions through the usage of off-the-shelf solvers, and extending APP allocation policies to consider this information.

7.1.4 Counteracting Ransomeware Aattacks

Ransomware is one of the most infamous kinds of malware, particularly the "crypto" subclass, which encrypts users' files, asking for some monetary ransom in exchange for the decryption key. Recently, crypto-ransomware grew into a scourge for enterprises and governmental institutions. The most recent and impactful cases include an oil company in the US, an international Danish shipping company, and many hospitals and health departments in Europe. Attacks result in production lockdowns, shipping delays, and even risks to human lives. To contrast ransomware attacks (crypto, in particular), in [14] we propose a family of solutions, called Data Flooding against Ransomware (DFaR), tackling the main phases of detection, mitigation, and restoration, based on a mix of honeypots, resource contention, and moving target defence. These solutions hinge on detecting and contrasting the action of ransomware by flooding specific locations (e.g., the attack location, sensitive folders, etc.) of the victim's disk with files. Building on the DFaR approach, in [15] we present an open-source tool, called Ranflood, that implements the mitigation and restoration phases. In particular, Ranflood supports three flooding strategies, apt for different attack scenarios. At its core, Ranflood buys time for the user to counteract the attack, e.g., to access an unresponsive, attacked server and shut it down manually.

7.2 Reversibility

Participants: Ivan Lanese, Pietro Lami, Vikraman Choudhury.

We have continued the study of reversibility started in the past years. We have provided a first attempt of a taxonomy to categorize approaches to reversible computing [30], focusing on the intrinsic features of the reversibility mechanism, and abstracting away from the different underlying models and application areas. We hope that such a work will shed light on the relation among the various approaches. We then concentrated on two specific approaches, causal-consistent reversibility, as used in concurrent systems, and time reversibility, as used in Markov Chains. In the former, any action can be undone provided that its consequences, if any, are undone beforehand. The latter instead stipulates that the stochastic behavior of a system remains the same when the direction of time is reversed, which supports efficient performance evaluation. As main result, we show that causal-consistent reversibility is a sufficient condition for time reversibility [21].

7.3 Quantitative Analysis

Participants: Melissa Antonelli, Martin Avanzini, Andrea Colledan, Ugo Dal Lago, Gabriele Vanoni, Paolo Pistone.

In Focus, 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, for instance the one of concurrent systems. Motivated by the use of randomization as a mean to make algorithms more efficient (on average), by the relative recent advent of Bayesian languages, and the significance development of quantum models of computation, our focus extended towards probabilistic and quantum languages.

In addition to the analysis of complexity properties, which can be seen as a property of individual programs, Focus has also been interested, for some years now, in the study of relational properties of programs. More specifically, we are interested in how to evaluate the differences between behaviors of distinct programs, going beyond the concept of program equivalence, but also beyond that of metrics. In

this way, approximate correct program transformations can be justified, while it also becomes possible to give a measure of how close a program is to a certain specification.

These trends extended to 2023. Below we describe the results obtained by Focus this year, dividing them into several strands.

7.3.1 Static Complexity Analysis

In Focus, we are interested in studying notions of termination and resource analysis for non-standard computing paradigms, like those induced by the presence of randomized and quantum effects. For instance, over the last years we have developed the cost analyser `eco-imp` 6.1.5, which can derive bounds on the average runtime of probabilistic imperative programs, in a fully automated manner. This year, we have extended the tool along two axes. First, we have extended the capabilities of the language, for instance, the new version is capable of analyzing recursively defined functions. Second, the tool has been extended from reasoning about expected runtime to a broader set of events, through the analyzed notion of expected outcomes. All of this required significant extensions to the underlying inference machinery, described in [10].

7.3.2 Quantitative Relational Reasoning

Another very important research axis within Focus is that about the study of metrics and quantitative reasoning on programs, supported by tools such as relational logic and program metrics. This year there have been a number of contributions in this direction.

We first of all looked at Mardare et al.'s quantitative algebras, and at whether they can be adapted to the structures which naturally emerge from Combinatory Logic and the λ-calculus [25]. First of all, we show that the framework is indeed applicable to those structures, and give soundness and completeness results. Then, we prove some negative results, clearly delineating to which extent categories of metric spaces can be models of such theories. We conclude by giving several examples of non-trivial higher-order quantitative algebras.

We then introduced *contextual behavioral metrics* (CBMs) as a novel way of measuring the discrepancy in behavior between processes in a concurrent scenario, taking into account both quantitative aspects and contextual information. This way, process distances by construction take the environment into account: two (non-equivalent) processes may still exhibit very similar behavior in some contexts, e.g., when certain actions are never performed. We first show how CBMs capture many well-known notions of equivalence and metric, including Larsen's environmental parametrized bisimulation. We then study compositional properties of CBMs with respect to some common process algebraic operators, namely prefixing, restriction, non-deterministic sum, parallel composition and replication [26].

Finally, we also studied higher-order logic and its role in quantitative reasoning. More specifically, we introduce a variation on Barthe et al.'s higher-order logic in which formulas are interpreted as predicates over open rather than closed objects. This way, concepts which have an intrinsically functional nature, like continuity, differentiability, or monotonicity, can be expressed and reasoned about in a very natural way, following the structure of the underlying program. We give open higher-order logic in distinct flavors, and in particular in its relational and local versions, the latter being tailored for situations in which properties hold only in part of the underlying function's domain of definition [24].

7.3.3 Abstract Machines and their Performances

Another topic of interest in Focus is the study of abstract machines for the implementation of high-level languages and in particular the analysis of their performance. In the last years, we have been concerned with the study of abstract machines derived from Girard's Geometry of Interaction. In a joint work between Dal Lago and Vanoni, we study one of the two formulations of the interaction abstract machine, namely that obtained from the so-called "boring" translation of intuitionistic logic into linear logic. We prove the correctness of the resulting call-by-name machine, at the same time establishing an improved bisimulation with Krivine's abstract machine. The proof makes essential use of the definition of a novel relational property linking configurations of the two machines [27]. This turned out to be a surprising fact, because the "boring" translation is well-known to be related to call-by-value evaluation.

7.3.4 Quantum Programming Languages

In the realm of quantum computing, circuit description languages represent a valid alternative to traditional QRAM-style languages. They indeed allow for finer control over the output circuit, without sacrificing flexibility nor modularity. We introduce in [23] a generalization of the paradigmatic lambda-calculus Proto-Quipper-M, which models the core features of the quantum circuit description language Quipper. The extension, called Proto-Quipper-K, is meant to capture a very general form of dynamic lifting. This is made possible by the introduction of a rich type and effect system in which not only computations, but also the very types are effectful. The main results we give for the introduced language are the classic type soundness results, namely subject reduction and progress.

7.4 Qualitative semantics

Participants: Daniel Hirschkoff, Ken Sakayori, Davide Sangiorgi.

7.4.1 Unifying semantics

In the area of qualitative semantics, during the past year our efforts have gone mainly in the direction of "unifying semantics", which has been our major direction in the past few years.

In particular we have deepened the study of the comparison between the the λ-calculus, as a pure model of functions, and the π-calculus, as a pure model of processes [33]. There has been a lot of work in the literature on the representation of λ-calculus into π-calculus, since early 1990s, where the process representations always yield (at best) non-extensional lambda-theories (i.e., beta rule holds, whereas eta does not). We have studied how to obtain extensional representations. In particular we have showed how 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 π-calculus), we can obtain well-known lambda-theories, both extensional and non-extensional (intuitively, those of Bohm trees with infinite eta, of Levy-Longo trees, and of Bohm trees).

7.4.2 Session Types

To celebrate the 30th edition of EXPRESS (Expressiveness in Concurrency) and the 20th edition of SOS (Structural Operational Semantics) we have produced an overview on how session types can be expressed in a type theory for the standard π-calculus by means of a suitable encoding [18]. The encoding allows one to reuse results about the π-calculus in the context of session-based communications, thus deepening the understanding of sessions and reducing redundancies in their theoretical foundations. We have also reviewed the practical implications of these results (e.g., refined forms of deadlock analysis, type inference).

7.5 Computer Science Education

Participants: Maurizio Gabbrielli, Michael Lodi, Simone Martini, Stefano Pio Zingaro.

7.5.1 Cryptography Education

We designed and tested an interdisciplinary training module on cryptography [13] for prospective STEM teachers that leveraged some “boundary objects” between Math and CS (e.g., adjacency matrices, graphs, computational complexity, factoring) in an important social context (the debate on the benefits and risks of end-to-end cryptography). The module proved useful in making students mobilize concepts, methods,

and practices of the two disciplines and making them move between semiotic representations of the interdisciplinary objects involved.

7.5.2 Primary students education

We co-designed with teachers [22] a learning module to teach iteration to second graders using a visual programming environment and following the Use-Modify-Create methodology. The co-designed learning module was piloted with three second-grade classes. Sharing the different perspectives of researchers and teachers improved the quality of the resulting learning module and constituted a very significant professional development opportunity for both teachers and researchers.

7.5.3 Large Language Models in Education

We studied [20] the problem-solving ability of GPT-3 in solving tasks proposed in the *Bebras* challenge. GPT-3 was able to answer with a majority of correct answers in about one-third of the Bebras tasks. It provided explanations that sounded correct but often logically wrong. The system was good in applying procedures, but quite bad at synthesis or logically complex tasks.

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

Ranflood Giallorenzo co-leads a three-year project collaboration, called “Ranflood”, started in July 2021, 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 Partnerships and cooperations

9.1 International initiatives

9.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

TCPro3

Title: Termination and Complexity Properties of Probabilistic Programs

Duration: 2019 — ongoing

Coordinator: Romain Péchoux (Inria project team Mocqua)

Partners: Inria project team Mocqua, Inria Nancy Grand-Est; University of Innsbruck (Austria)

Inria contact: Romain Péchoux

Summary: Probabilistic languages consist in higher-order functional, imperative languages, and reduction systems with sampling and conditioning primitive instructions. While deep theoretical results have been established on the semantic properties of such languages, applications of termination and complexity analysis are restricted to academic examples so far. The associate team TCPro3 has the aim to contribute to the field by developing methods for reasoning on quantitative properties of probabilistic programs and models. Extensions of these methods to quantum programs will be studied.

9.2 International research visitors

9.2.1 Visits of international scientists

Other international visits to the team

Laura Bocchi

Status Reader

Institution of origin: University of Kent

Country: UK

Dates: July 4-12, 2023 and October 15-22, 2023

Context of the visit: behavioral APIs, reversible timed systems

Mobility program/type of mobility: secondment of BehAPI RISE project

Annah Gommerstadt

Status researcher

Institution of origin: Vassar College

Country: USA

Dates: March 20-25, 2023

Context of the visit: session typing

Mobility program/type of mobility: ECCO bilateral

Guilhem Jaber

Status researcher (maitre de conference)

Institution of origin: Université de Nantes

Country: France

Dates: 3 weeks spread over the year

Context of the visit: collaboration on semantics of higher-order languages

Mobility program/type of mobility: research visit

Irek Ulidowski

Status Associate professor

Institution of origin: University of Leicester

Country: UK

Dates: June 30 - July 6, 2023

Context of the visit: reversible computing

Mobility program/type of mobility: research visit

9.2.2 Visits to international teams

Research stays abroad

Michael Lodi

Visited institution: Universidad de La Laguna (Tenerife)

Country: Spain

Dates: February 2-March 3, 2023

Context of the visit: Giving two seminars about computational thinking and about its introduction in schools (as part of "Latest Advances in Computer Science" seminar series), and to discuss and work with on the topic with the researchers of the Dpto. Ingeniería Informática y de Sistemas.

Mobility program/type of mobility: Seminar and research stay

Ugo Dal Lago

Visited institution: University of Victoria

Country: Canada

Dates: April 13-21, 2023

Context of the visit: Research visit to Bruce Kapron. Collaboration on cryptographic probabilistic separation logic.

Mobility program/type of mobility: ERC DIAPASON

Ugo Dal Lago

Visited institution: University of Kumamoto

Country: Japan

Dates: November 7-12, 2023

Context of the visit: Research visit to Naohiko Hoshino. Collaboration on geometry of interaction and quantum computation.

Mobility program/type of mobility: ERC DIAPASON

Gianluigi Zavattaro

Visited institution: Faculty of Information and Communication Technology, University of Malta

Country: Malta

Dates: July 5-19, 2023

Context of the visit: Collaboration in the context of the BehAPI (Behavioral Application Program Interfaces) EU H2020 RISE project

Mobility program/type of mobility: Research visits

9.3 European initiatives

9.3.1 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.

9.3.2 H2020 projects

BEHAPI (Behavioral Application Program Interfaces) is a European Project H2020-MSCA-RISE-2017, running in the period March 2018 – February 2024. The topic of the project is behavioral types, as a suite of technologies that formalize the intended usage of API interfaces. Indeed, currently APIs are typically flat structures, i.e. sets of service/method signatures specifying the expected service parameters and the kind of results one should expect in return. However, correct API usage also requires the individual services to be invoked in a specific order. Despite its importance, the latter information is either often omitted, or stated informally via textual descriptions. The expected benefits of behavioral types include guarantees such as service compliance, deadlock freedom, dynamic adaptation in the presence of failure, load balancing etc. The project aims to bring the existing prototype tools based on these technologies to mainstream programming languages and development frameworks used in industry.

Participants: Mario Bravetti, Maurizio Gabbrielli, Ivan Lanese, Cosimo Laneve, Davide Sangiorgi, Gianluigi Zavattaro, Stefano Zingaro.

9.4 National initiatives

9.4.1 FREEDA

FREEDA (Failure-Resilient, Energy-aware, and Explainable Deployment of microservice-based Applications over Cloud-IoT infrastructures) is a PRIN PE6 24-month project started on October 2023.

The evolution of Cloud computing, driven by the demands of smart connected devices, calls for a transition to pervasive distributed environments at the network's Edge. The increasing use of Microservice-based Applications (MSAs) in enterprise settings, along with the expansion of Cloud-IoT infrastructures, needs careful deployment planning. FREEDA aims to facilitate comprehensive MSA deployment over Cloud-IoT infrastructures by analyzing deployment requirements, considering factors like MSA complexity, failure resilience, energy consumption, and compliance with sustainable IT standards. The system utilizes constraint reasoning to effectively balance conflicting deployment requirements and employs continuous reasoning to adapt quickly to changes in the deployed MSA and Cloud-IoT infrastructure.

Participants: Maurizio Gabbrielli, Saverio Giallorenzo, Gianluigi Zavattaro.

9.4.2 DCore

DCore (Causal debugging for concurrent systems) is an ANR project that started on March 2019 and that will end in March 2024.

The overall objective of the project is to develop a semantically well-founded, novel form of concurrent debugging, which we call “causal debugging”. Causal debugging will comprise and integrate two main engines: (i) a reversible execution engine that allows programmers to backtrack and replay a concurrent or distributed program execution and (ii) a causal analysis engine that allows programmers to analyze concurrent executions to understand why some desired program properties could be violated.

Participants: Ivan Lanese.

9.4.3 PROGRAMme

PROGRAMme (What is a program? Historical and philosophical perspectives) is an ANR project started on October 2017 and that has finished on October 2023 (extension of one year granted);

The aim of this project is to develop a coherent analysis and pluralistic understanding of “computer program” and its implications to theory and practice.

Participants: Simone Martini.

9.4.4 PPS

PPS (Probabilistic Programming Semantics) is an ANR PCR project that started on January 2020 and that will finish on December 2024.

Probabilities are essential in Computer Science. Many algorithms use probabilistic choices for efficiency or convenience and probabilistic algorithms are crucial in communicating systems. Recently, probabilistic programming, and more specifically, functional probabilistic programming, has become crucial in various works in Bayesian inference and Machine Learning. Motivated by the rising impact of such probabilistic languages, the aim of this project is to develop formal methods for probabilistic computing (semantics, type systems, logical frameworks for program verification, abstract machines etc.) to systematize the analysis and certification of functional probabilistic programs.

Participants: Martin Avanzini, Ugo Dal Lago, Davide Sangiorgi, Gabriele Vanoni.

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

General chair, scientific chair

- IFIP Int. Conference on Formal Techniques for Distributed Objects, Components and Systems (I. Lanese, chair of steering committee)

Member of the steering/organizing committees

- Conference on Reversible Computation (I. Lanese)
- Interaction and Concurrency Experience (I. Lanese)
- International Federated Conference on Distributed Computing Techniques (I. Lanese)
- Agility with Microservice Programming workshop (S. Giallorenzo)

- Concurrency Theory (D. Sangiorgi)
- International Conference on Foundations of Software Science and Computation Structures (U. Dal Lago)
- Symposium on Logic in Computer Science (U. Dal Lago)

10.1.2 Scientific events: selection

Member of the conference program committees

- 8th International Conference on Formal Structures for Computation and Deduction (FSCD) 2023 (M. Avanzini)
- 15th Conference on Reversible Computation (RC) 2023 (I. Lanese)
- 19th International Conference on Formal Aspects of Component Software (FACS) 2023 (I. Lanese)
- International Conference on Microservices (MICROSERVICES) 2023 (I. Lanese)
- 50th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL) 2023 (U. Dal Lago)
- 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) 2023 (U. Dal Lago)
- Foundations of Software Technology and Theoretical Computer Science (FSTTCS) 2023 (U. Dal Lago)
- 24th Italian Conference on Theoretical Computer Science (ICTCS) 2023 (U. Dal Lago)
- 26th International Conference on Foundations of Software Science and Computation Structures (FOSSACS) 2023 (U. Dal Lago)
- 13th International Workshop on Developments in Computational Models (DCM) 2023 (U. Dal Lago)

10.1.3 Journal

Member of the editorial boards

- Journal of Universal Computer Science (M. Bravetti)
- Electronics Journal, section Computer Science and Engineering (M. Bravetti)
- Logical Methods in Computer Science (U. Dal Lago)
- Mathematical Structures in Computer Science (U. Dal Lago)
- Acta Informatica (U. Dal Lago)
- Acta Informatica (D. Sangiorgi)
- Distributed Computing (D. Sangiorgi)
- RAIRO – Theoretical Informatics and Applications (D. Sangiorgi)
- Foundations and Trends in Programming Languages (D. Sangiorgi)
- SN Computer Science (D. Sangiorgi)

10.1.4 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 Focus 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 (Swiss). Montesi is the president of the organization, Guidi is a board member, Lanese and Giallorenzo are respectively part of the research and communication Community groups.
- I. Lanese is chair of the IFIP (International Federation for Information Processing) WG6.1 on Architectures and Protocols for Distributed Systems.
- U. Dal Lago is a member of the scientific councils of the Italian Chapter of the EATCS, and of the Italian Association on Logic and its Applications.
- U. Dal Lago is involved since September 1st in an Italian National initiative called CN HPC, namely a new research center about high-performance computing. U. Dal Lago is responsible for topics related to quantum computing inside the University of Bologna.
- S. Martini is a member of the Council of the Commission on History and Philosophy of Computing, an organism of the International Union for History and Philosophy of Science, 2017-2023.

10.1.5 Research administration

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

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Martin Avanzini
  - “Advanced Logic”, 12 hours, Université Côte d’Azur, France.
- Ugo Dal Lago
  - “Optimization”, 36 hours, 2nd year, University of Bologna, Italy.
  - “Cryptography”, 40 hours, 2nd year Master, University of Bologna, Italy.
  - “Languages and Algorithms for AI: Machine Learning Theory”, 32 hours, 1st year Master, University of Bologna, Italy.
  - “On Randomization in (Higher-order) Programming”, 3 hours, Escuela de Ciencias Informáticas (ECI), Universidad de Buenos Aires, Argentina.
- Saverio Giallorenzo
  - “Programming Languages”, 30 hours, 2nd year Bachelor, University of Bologna, Italy.
  - “Network Analysis”, 30 hours, 2nd year Master, University of Bologna, Italy.
- Ivan Lanese
  - “Architettura degli Elaboratori”, 34 hours, 1st year, University of Bologna, Italy.
  - “Computational Methods for Bioinformatics”, 58 hours, 1st year Master, University of Bologna, Italy.

  - "Ingegneria del Software Orientata ai Servizi", 44 hours, 1st year master, University of Bologna, Italy.

- Michael Lodi
  - "Computer Science Education", 26 hours, 2nd year Master in CS, University of Bologna, Italy.
  - "Programmazione", 16 hours, 1st year Bachelor in Math, University of Bologna, Italy.
- Simone Martini
  - "Programmazione", 66 hours, 1st year, University of Bologna, Italy.
  - "Languages and Algorithms for Artificial Intelligence, module 1", 25 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, Bachelor in Computer Science, University of Bologna, Italy.
  - "Scalable and Cloud Programming", 50 hours, Master in Computer Science, University of Bologna, Italy.
  - "Architectures and Platforms for Artificial Intelligence", 24 hours, Master in Artificial Intelligence, University of Bologna, Italy.

10.2.2 Supervision

- Davide Davoli, October 2022, On Kernel Address Space Layout Randomization. Supervisors: Tamara Rezk, Ugo Dal Lago, Martin Avanzini.
- Pietro Lami, October 2021, Reversible Semantics of Programming Languages and Transactions. Supervisor: Jean-Bernard Stefani (SPADES, INRIA Grenoble), co-supervisor: Ivan Lanese.
- Sourabh Pal, November 2022, Choreographies for Healthcare Management. Supervisors: Ivan Lanese and Luca Cisbani.

10.2.3 Juries

- Fabio Zanasi was external examiner for the following PhD VIVAs:
  - Cole Comfort, October 31, 2023, University of Oxford; and
  - Guillaume Boisseau, July 5, 2023, University of Oxford.
- Martin Avanzini was member of the CSD (Commission de suivi des doctorants – doctoral student monitoring committee) of Guillaume Combette, June 19, 2023, UniCA.

10.3 Popularization

10.3.1 Education

- Michael Lodi and Simone Martini are members of the technical committee of Olimpiadi del Problem Solving (at Italian Ministry of Education); this involves preparation of material and supervision and jury during the finals.
- Michael Lodi gave three lectures titled "Informatics is (not) magic" and participated in the "graduation cerimony" of UniJunior, an outreach initiative for kids between 8 and 13 years old.

- Michael Lodi has been the scientific responsible for the project "Ragazze digitali" (a two week long informatics summer camp for high school girls, funded by Emilia Romagna Region) for Bologna and Imola, and the educational manager and teacher for the Bologna course.
- Maurizio Gabbrielli, Michael Lodi, Simone Martini and Stefano Zingaro organized and taught a week-long summer school for Bologna Business School, teaching informatics, creative computing and artificial intelligence principles to kids from 8 to 12 years old.

11 Scientific production

11.1 Major publications

[1] M. Bravetti and G. Zavattaro. 'A Foundational Theory of Contracts for Multi-party Service Composition'. In: *Fundam. Inform.* 89.4 (2008), pp. 451–478.

[2] N. Busi, M. Gabbrielli and G. Zavattaro. 'On the expressive power of recursion, replication and iteration in process calculi'. In: *Mathematical Structures in Computer Science* 19.6 (2009), pp. 1191–1222.

[3] P. Coppola and S. Martini. 'Optimizing optimal reduction: A type inference algorithm for elementary affine logic'. In: *ACM Trans. Comput. Log.* 7.2 (2006), pp. 219–260.

[4] M. Gabbrielli and S. Martini. *Programming Languages: Principles and Paradigms*. Springer, 2010.

[5] D. Hirschkoff, É. Lozes and D. Sangiorgi. 'On the Expressiveness of the Ambient Logic'. In: *Logical Methods in Computer Science* 2.2 (2006).

[6] U. D. Lago and M. Gaboardi. 'Linear Dependent Types and Relative Completeness'. In: *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011*. IEEE Computer Society, 2011, pp. 133–142.

[7] I. Lanese, C. A. Mezzina and J.-B. Stefani. 'Reversibility in the higher-order π-calculus'. In: *Theor. Comput. Sci.* 625 (2016), pp. 25–84. DOI: 10.1016/j.tcs.2016.02.019. URL: https://doi.org/10.1016/j.tcs.2016.02.019.

[8] F. Montesi, C. Guidi and G. Zavattaro. 'Composing Services with JOLIE'. In: *Fifth IEEE European Conference on Web Services (ECOWS 2007)*. 2007, pp. 13–22.

[9] D. Sangiorgi. *An introduction to Bisimulation and Coinduction*. Cambridge University Press, 2012.

11.2 Publications of the year

International journals

[10] M. Avanzini, G. Moser and M. Schaper. 'Automated Expected Value Analysis of Recursive Programs'. In: *Proceedings of the ACM on Programming Languages* 7.PLDI (6th June 2023), pp. 1050–1072. DOI: 10.1145/3591263. URL: https://inria.hal.science/hal-04345663.

[11] F. Barbanera, I. Lanese and E. Tuosto. 'A Theory of Formal Choreographic Languages'. In: *Logical Methods in Computer Science* 19.3 (2nd Aug. 2023). DOI: 10.46298/lmcs-19(3:9)2023. URL: https://inria.hal.science/hal-04343514.

[12] F. Barbanera, I. Lanese and E. Tuosto. 'Composition of synchronous communicating systems'. In: *Journal of Logical and Algebraic Methods in Programming* 135 (Oct. 2023), p. 100890. DOI: 10.1016/j.jlamp.2023.100890. URL: https://inria.hal.science/hal-04343521.

[13] E.-I. Bartzia, M. Lodi, M. Sbaraglia, S. Modeste, V. Durand-Guerrier and S. Martini. 'An Unplugged Didactical Situation on Cryptography between Informatics and Mathematics'. In: *Informatics in Education* (24th July 2023). DOI: 10.15388/infedu.2024.06. URL: https://hal.science/hal-04184262.

[14] D. Berardi, S. Giallorenzo, A. Melis, S. Melloni, L. Onori and M. Prandini. 'Data Flooding against Ransomware: Concepts and Implementations'. In: *Computers and Security* 131 (Aug. 2023), p. 103295. DOI: 10.1016/j.cose.2023.103295. URL: https://hal.science/hal-04316302.

[15] D. Berardi, S. Giallorenzo, A. Melis, S. Melloni and M. Prandini. 'Ranflood: A mitigation tool based on the principles of data flooding against ransomware'. In: *SoftwareX* 25 (Feb. 2024), p. 101605. DOI: 10.1016/j.softx.2023.101605. URL: https://inria.hal.science/hal-04362711.

[16] S. Giallorenzo, F. Montesi and M. Peressotti. 'Choral: Object-Oriented Choreographic Programming'. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* (22nd Nov. 2023). DOI: 10.1145/3632398. URL: https://hal.science/hal-04316324.

[17] S. Giallorenzo, F. Montesi, M. Peressotti and F. Rademacher. 'LEMMA2Jolie: A tool to generate microservice APIs from domain models'. In: *Science of Computer Programming* 228 (June 2023), p. 102956. DOI: 10.1016/j.scico.2023.102956. URL: https://hal.science/hal-04316295.

Invited conferences

[18] I. Castellani, O. Dardha, L. Padovani and D. Sangiorgi. 'EXPRESSing Session Types'. In: *Electronic Proceedings in Theoretical Computer Science*. EXPRESS / SOS 2023 - Combined 30th International Workshop on Expressiveness in Concurrency and 20th Workshop on Structural Operational Semantics. Vol. EPTCS-387. Proceedings Combined 30th International Workshop on Expressiveness in Concurrency and 20th Workshop on Structural Operational Semantics. Anvers (Antwerpen), Belgium, 14th Sept. 2023, pp. 8–25. DOI: 10.4204/EPTCS.387.2. URL: https://inria.hal.science/hal-04349502.

International peer-reviewed conferences

[19] P. Baillot, U. Dal Lago, C. Kop and D. Vale. 'On Basic Feasible Functionals and the Interpretation Method'. In: *Lecture Notes in Computer Science*. FoSSaCS 2024 - International Conference on Foundations of Software Science and Computation Structures. Luxembourg City, Luxembourg: Springer, 8th Apr. 2024. URL: https://hal.science/hal-04376613.

[20] C. Bellettini, M. Lodi, V. Lonati, M. Monga and A. Morpurgo. 'Davinci Goes to Bebras: A Study on the Problem Solving Ability of GPT-3'. In: *SCITEPRESS digital library*. CSEDU 2023 - 15th International Conference on Computer Supported Education. Vol. 2. Proceedings of the 15th International Conference on Computer Supported Education - Volume 2: CSEDU. Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 21st Apr. 2023, pp. 59–69. DOI: 10.5220/0012007500003470. URL: https://inria.hal.science/hal-04343267.

[21] M. Bernardo, I. Lanese, A. Marin, C. Mezzina, S. Rossi and C. Sacerdoti Coen. 'Causal Reversibility Implies Time Reversibility'. In: *Lecture notes in computer science*. QEST 2023 - International Conference on Quantitative Evaluation of Systems. Vol. LNCS-14287. Quantitative Evaluation of Systems 20th International Conference, QEST 2023, Antwerp, Belgium, September 20–22, 2023, Proceedings. Antwerp, Belgium: Springer Nature Switzerland, 15th Sept. 2023, pp. 270–287. DOI: 10.1007/978-3-031-43835-6_19. URL: https://inria.hal.science/hal-04343506.

[22] S. Capecchi, M. Lodi, V. Lonati and M. Sbaraglia. 'Castle and Stairs to Learn Iteration: Co-designing a UMC Learning Module with Teachers'. In: *ACM digital library*. ITiCSE 2023: Innovation and Technology in Computer Science Education. ITiCSE 2023: Innovation and Technology in Computer Science Education. Turku, Finland: ACM, 29th June 2023, pp. 222–228. DOI: 10.1145/3587102.3588793. URL: https://inria.hal.science/hal-04343281.

[23] A. Colledan and U. Dal Lago. 'On Dynamic Lifting and Effect Typing in Circuit Description Languages'. In: *Leibniz International Proceedings in Informatics (LIPIcs). International Conference on Types for Proofs and Programs (TYPES)*. TYPES 2022 - 28th International Conference on Types for Proofs and Programs. Vol. LIPIcs-269. 28th International Conference on Types for Proofs and Programs (TYPES 2022). Nantes, France: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28th July 2023. DOI: 10.4230/LIPIcs.TYPES.2022.3. URL: https://inria.hal.science/hal-04350852.

[24] U. Dal Lago, F. Gavazzo and A. Ghyselen. 'Open Higher-Order Logic'. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. CSL 2023 - 31st EACSL Annual Conference on Computer Science Logic. Vol. LIPIcs-252. 31st EACSL Annual Conference on Computer Science Logic (CSL 2023). Warsaw, Poland: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.CSL.2023.17. URL: https://inria.hal.science/hal-04356990.

[25] U. Dal Lago, N. Hoshino and P. Pistone. 'On the Lattice of Program Metrics'. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. FSCD 2023 - 8th International Conference on Formal Structures for Computation and Deduction. Vol. LIPIcs-260. 8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023). Rome, Italy: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.FSCD.2023.20. URL: https://inria.hal.science/hal-04356985.

[26] U. Dal Lago and M. Murgia. 'Contextual Behavioural Metrics'. In: *Leibniz International Proceedings in Informatics (LIPIcs)*. CONCUR 2023 - 34th International Conference on Concurrency Theory. Vol. LIPIcs-279. 34th International Conference on Concurrency Theory (CONCUR 2023). Antwerp, Belgium: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/LIPIcs.CONCUR.2023.38. URL: https://inria.hal.science/hal-04356984.

[27] U. Dal Lago and G. Vanoni. '(Not So) Boring Abstract Machines'. In: *CEUR workshop proceedings*. ICTCS 2023 - Italian Conference on Theoretical Computer Science 2023. Vol. CEUR-3587. Proceedings of the 24th Italian Conference on Theoretical Computer Science. Palermo, Italy, 2023. URL: https://inria.hal.science/hal-04356993.

[28] S. Giallorenzo, F. Montesi, M. Peressotti and F. Rademacher. 'Model-Driven Code Generation for Microservices: Service Models'. In: *Open Access Series in Informatics (OASIcs)*. Microservices 2020/2022 - Joint Post-proceedings of the Third and Fourth International Conference on Microservices. Vol. OASIcs-111. Joint Post-proceedings of the Third and Fourth International Conference on Microservices (Microservices 2020/2022). Paris, France: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/OASIcs.Microservices.2020-2022.6. URL: https://inria.hal.science/hal-04362712.

[29] S. Giallorenzo, F. Montesi, M. Peressotti, F. Rademacher and N. Unwerawattana. 'JoT: A Jolie Framework for Testing Microservices'. In: *Lecture notes in computer science*. COORDINATION 2023 - 25th International Conference on Coordination Models and Languages. Vol. LNCS-13908. Coordination Models and Languages 25th IFIP WG 6.1 International Conference, COORDINATION 2023, Held as Part of the 18th International Federated Conference on Distributed Computing Techniques, DisCoTec 2023, Lisbon, Portugal, June 19–23, 2023, Proceedings. Lisbon, Portugal: Springer Nature Switzerland, 15th June 2023, pp. 172–191. DOI: 10.1007/978-3-031-35361-1_10. URL: https://hal.science/hal-04316287.

[30] R. Glück, I. Lanese, C. A. Mezzina, J. A. Miszczak, I. Phillips, I. Ulidowski and G. Vidal. 'Towards a Taxonomy for Reversible Computation Approaches'. In: *15th International Conference, RC 2023, Giessen, Germany, July 18–19, 2023, Proceedings*. Reversible Computation - RC 2023. Vol. LNCS-13960. Lecture Notes in Computer Science. Giessen, Germany: Springer Nature Switzerland, 12th July 2023, pp. 24–39. DOI: 10.1007/978-3-031-38100-3_3. URL: https://inria.hal.science/hal-04343408.

[31] G. de Palma, S. Giallorenzo, C. Laneve, J. Mauro, M. Trentin and G. Zavattaro. 'Serverless Scheduling Policies based on Cost Analysis'. In: *Electronic Proceedings in Theoretical Computer Science*. TiCSA 2023 - First Workshop on Trends in Configurable Systems Analysis - TiCSA@ETAPS 2023. Vol. EPTCS-392. Paris, France, 31st Oct. 2023, pp. 40–52. DOI: 10.4204/EPTCS.392.3. URL: https://hal.science/hal-04316320.

[32] G. de Palma, S. Giallorenzo, J. Mauro, M. Trentin and G. Zavattaro. 'Custom Serverless Function Scheduling Policies: An APP Tutorial'. In: *Open Access Series in Informatics (OASIcs)*. Joint Post-proceedings of the Third and Fourth International Conference on Microservices (Microservices 2020/2022). Vol. OASIcs-111. Joint Post-proceedings of the Third and Fourth International Conference on Microservices (Microservices 2020/2022). Paris, France: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. DOI: 10.4230/OASIcs.Microservices.2020-2022.5. URL: https://inria.hal.science/hal-04362714.

[33] K. Sakayori and D. Sangiorgi. 'Extensional and Non-extensional Functions as Processes'. In: 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). Boston, United States, 26th June 2023. DOI: 10.1109/LICS56636.2023.10175686. URL: https://hal.science/hal-04081885.

Conferences without proceedings

[34] M. Antonelli. 'Two Remarks on Counting Propositional Logic'. In: *CEUR workshop proceedings*. BEWARE 2022 - 1st Workshop on Bias, Ethical AI, Explainability and the Role of Logic and Logic Programming. Vol. CEUR-3319. Udine, Italy, 2023. URL: https://inria.hal.science/hal-03921654.

11.3 Cited publications

[35] M. Carbone, K. Honda and N. Yoshida. 'A Calculus of Global Interaction based on Session Types'. In: *Electr. Notes Theor. Comput. Sci.* 171.3 (2007), pp. 127–151.

[36] A. Igarashi and N. Kobayashi. 'Resource usage analysis'. In: *POPL conference*. ACM Press, 2002, pp. 331–342.

[37] N. Kobayashi and D. Sangiorgi. 'A hybrid type system for lock-freedom of mobile processes'. In: *ACM Trans. Program. Lang. Syst.* 32.5 (2010).