IN PARTNERSHIP WITH:
Université de Bologne (Italie)

Activity Report 2019

Project-Team FOCUS

Foundations of Component-based Ubiquitous Systems

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

RESEARCH CENTER
Sophia Antipolis - Méditerranée

THEME
Distributed programming and Software engineering

Table of contents

Project-Team FOCUS

Creation of the Project-Team: 2010 January 01

Keywords:

Computer Science and Digital Science:

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, Visitors, External Collaborators

Research Scientist

Martin Avanzini [Inria, Researcher]

Faculty Members

Mario Bravetti [University Bologna, Associate Professor]
Ugo Dal Lago [University Bologna, Professor]
Maurizio Gabbrielli [University Bologna, Professor]
Ivan Lanese [University Bologna, Associate Professor]
Cosimo Laneve [University Bologna, Professor]
Simone Martini [University Bologna, Professor]
Davide Sangiorgi [Team leader, University Bologna, Professor, HDR]
Gianluigi Zavattaro [University Bologna, Professor]

Post-Doctoral Fellows

Francesco Gavazzo [University Bologna, Post-Doctoral Fellow, from Nov 2019]
Guillaume Geoffroy [University Bologna, Post-Doctoral Fellow, from Dec 2019]
Thomas Nicolas Georges Leventis [Inria, Post-Doctoral Fellow, from Apr 2019 until Aug 2019]
Michael Lodi [University Bologna, Post-Doctoral Fellow, from Oct 2019]
Doriana Medic [Inria, Post-Doctoral Fellow, from Aug 2019]
Paolo Pistone [University Bologna, Post-Doctoral Fellow, from Dec 2019]
Akira Yoshimizu [Inria, Post-Doctoral Fellow, until Jan 2019]
Stefano Pio Zingaro [University Bologna, Post-Doctoral Fellow, from Nov 2019]

PhD Students

Melissa Antonelli [University Bologna, PhD Student, from Dec 2019]
Raphaelle Crubillé [Univ Denis Diderot, PhD Student]
Adrien Durier [ENS Lyon and University Bologna, PhD Student]
Tong Liu [University Bologna, PhD Student, until Oct 2019]

Michael Lodi [University Bologna, PhD Student, until Oct 2019]
Gabriele Vanoni [University Bologna, PhD Student]
Stefano Pio Zingaro [University Bologna, PhD Student, until Oct 2019]

Administrative Assistant

Christine Claux [Inria, Administrative Assistant]

Visiting Scientist

Emma Kerinec [University Bologna, until Mar 2019]

External Collaborators

Saverio Giallorenzo [University of Southern Denmark]
Claudio Guidi [Italiana Software]
Daniel Hirschkoff [Ecole Normale Supérieure Lyon]
Fabrizio Montesi [University of Southern Denmark]

2. Overall Objectives

2.1. 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 emerging 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, so in CBUS theories are needed that allow us to understand the systems, design or program them, 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 formalise 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, well address compositionality—a central property in our approach to problems. Accordingly, the techniques we employ are mainly operational techniques based on notions of behavioural 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 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 at hand here: they can express causality and order information on messages [53], [49], [54], while remaining decidable systems.

3.4. Foundations 4: Implicit computational complexity

A number of elegant and powerful results have been recently obtained in implicit computational complexity in the λ-calculus in which 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 resources 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, broadly 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 behaviour 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

5.1. Highlights of the Year

5.1.1. Awards

- Ugo Dal Lago has been awarded an ERC CoG for his project “Differential Program Semantics” (DIAPASoN), which started on March 1st, 2019.
- Francesco Gavazzo has received the award for “Best Italian PhD Thesis in Theoretical Computer Science”, by the Italian Chapter of EATCS (European Association for Theoretical Computer Science)
- Raphaëlle Crubillé has been awarded the “prix de thèse Gilles Kahn 2019 (Societé Informatique de France and Académie des Sciences)

6. New Software and Platforms

6.1. HoCA

Higher-Order Complexity Analysis

KEYWORDS: Ocaml - Verification - Runtime Complexity Analysis

SCIENTIFIC DESCRIPTION: Over the last decade, various tools for the static analysis of resource properties of programs have emerged. In particular, the rewriting community has recently developed several tools for the time complexity analysis of term rewrite systems. These tools have matured and are nowadays able to treat non-trivial programs, in a fully automatic setting. However, none of these automatic complexity analysers can deal with higher-order functions, a pervasive feature of functional programs. HoCA (Higher-Order Complexity Analyser) overcomes this limitation by translating higher-order programs – in the form of side-effect free OCaml programs - into equivalent first-order rewrite systems. At the heart of our tool lies Reynold’s defunctionalization technique. Defunctionalization however is not enough. Resulting programs have a recursive structure too complicated to be analysed automatically in all but trivial cases. To overcome this issue, HoCA integrates a handful of well established program transformation techniques, noteworthy dead-code elimination, inlining, instantiation and uncurrying. A complexity bound on the resulting first-order program can be relayed back reliably to the higher-order program of interest. A detailed description of HoCA is available on http://arxiv.org/abs/1506.05043.

FUNCTIONAL DESCRIPTION: HoCA is an abbreviation for Higher-Order Complexity Analysis, and is meant as a laboratory for the automated complexity analysis of higher-order functional programs. Currently, HoCA consists of one executable pcf2trs which translates a pure subset of OCaml to term rewrite systems, in a complexity reflecting manner. As a first step, HoCA desugars the given program to a variation of Plotkin's PCF with data-constructors. Via Reynold's defunctionalization, the PCF program is turned into an applicative term rewrite system (ATRS for short), call-by-value reductions of the PCF program are simulated by the ATRS step-by-step, on the ATRS, and various complexity reflecting transformations are performed: inlining, dead-code-elminiation, instantiation of higher-order variables through a call-flow-analysis and finally uncurrying. This results finally in a first-order rewrite system, whose runtime-complexity reflects the complexity of the initial program, asymptotically.

- Participants: Martin Avanzini and Ugo Dal Lago
- Contact: Ugo Dal Lago
- URL: http://cbr.uibk.ac.at/tools/hoca/

6.2. JOLIE

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 FUNCTIONAL DESCRIPTION: There are many fixes to the HTTP extension, improvements to the embedding engine for Javascript programs, and improvements to the support tools jolie2java and wsdl2jolie.

NEWS OF THE YEAR: During 2019 the Jolie project saw three major actions.

The first action regards the build system used for the development of the language, which has been transitioned to Maven, the main build automation tool used for Java projects. The move to Maven is dictated by two needs. The first is to streamline the development and release processes of Jolie, as Maven greatly helps in obtaining, updating, and managing library dependencies. The second necessity addressed by Maven is helping in partitioning the many sub-projects that constitute the Jolie codebase, reducing development and testing times. Having Jolie as a Maven project also helps in providing Jolie sub-components (as Maven libraries) to other projects. Finally, the move to Maven is set within a larger effort to expedite the inclusion in the main Jolie development branch of contributions by new members of its growing community.

The second action regards the transition to Netty as a common framework to support communication protocols and data formats in Jolie. Netty is a widely-adopted Java framework for the development of network applications, and it was used in 2018 to successfully support several IoT communication protocols and data formats in a Jolie spin-off project, called JIoT. The work in 2019 integrated into the Jolie codebase the protocols and data format developed within the JIoT project and pushed towards the integration of the Netty development branch into the main branch of the Jolie project (i.e., re-implementing using Netty the many protocol and data-formats already supported by Jolie). The Netty development branch is currently in a beta phase and it is subject to thorough in-production tests, to ensure consistent behaviour with the previous implementation.

The third action regards the development and support for a new official IDE for Jolie. Hence, along with the ones already existing for the Atom and Sublime Text text editors, Jolie developers can use the Jolie plugin (based on the Language Server Protocol) for the Visual Studio Code text editor to obtain syntax highlighting, documentation aids, file navigation, syntax checking, semantic checking, and quick-run shortcuts for their Jolie programs.

In addition to the above actions, in 2019 Jolie transitioned through three minor releases and a major one, from 1.7.1 to 1.8.2. The minor releases mainly fixed bugs, improved performance, and included new protocol/data-format functionalities. The major release included a slim-down of the notation for the composition of statements, types definitions, and tree structures, for a terser codebase. Upgrades to 1.8.2 also introduced: timeouts for solicit-response invocations to handle the interruption of long-standing requests, more user-friendly messages from the Jolie interpreter, including easier-to-parse errors and the pretty-printing of data structures, for a more effective development and debugging experience.

In 2019 Jolie also saw the development of a new Jolie library, called TQuery, which 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 sound variant of the Aggregation Framework, the query language of the most popular document-oriented database: MongoDB. Usage scenarios for Tquery are (but not limited to) eHealth, the Internet-of-Things, and Edge Computing, where data should be handled in an ephemeral way, i.e., in a real-time manner but with the constraint that data shall not persist in the system.

- Participants: Claudio Guidi, Fabrizio Montesi, Maurizio Gabbrielli, Saverio Giallorenzo and Ivan Lanese
- Contact: Fabrizio Montesi
- URL: http://www.jolie-lang.org/

6.3. 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.

- Contact: Tong Liu
- URL: http://cs.unibo.it/t.liu/nightsplitter/

6.4. AIOCJ

Adaptive Interaction-Oriented Choreographies in Jolie

KEYWORD: Dynamic adaptation

SCIENTIFIC DESCRIPTION: AIOCJ is an open-source choreographic programming language for developing adaptive systems. It allows one to describe a full distributed system as a unique choreographic program and to generate code for each role avoiding by construction errors such as deadlocks. Furthermore, it supports dynamic adaptation of the distributed system via adaptation rules.

FUNCTIONAL DESCRIPTION: AIOCJ is a framework for programming adaptive distributed systems based on message passing. AIOCJ comes as a plugin for Eclipse, AIOCJ-ecl, allowing to edit descriptions of distributed systems written as adaptive interaction-oriented choreographies (AIOC). From interaction-oriented choreographies the description of single participants can be automatically derived. Adaptation is specified by rules allowing one to replace predetermined parts of the AIOC with a new behaviour. A suitable protocol ensures that all the participants are updated in a coordinated way. As a result, the distributed system follows the specification given by the AIOC under all changing sets of adaptation rules and environment conditions. In particular, the system is always deadlock free. AIOCJ can interact with external services, seen as functions, by specifying their URL and the protocol they support (HTTP, SOAP, ...). Deadlock-freedom guarantees of the application are preserved provided that those services do not block.

NEWS OF THE YEAR: In 2019 we performed a major upgrade to AIOCJ: the possibility to introduce new roles, absent from a running choreography, within a given adaptation rule. The inclusion of new roles is supported by a slight, incremental change in the AIOCJ syntax and by a new component of the AIOCJ runtime environment.

- Participants: Ivan Lanese, Jacopo Mauro, Maurizio Gabbrielli, Mila Dalla Preda and Saverio Giallorenzo
- Contact: Saverio Giallorenzo
- URL: http://www.cs.unibo.it/projects/jolie/aiocj.html

6.5. CauDEr

Causal-consistent Debugger for Erlang

KEYWORDS: Debug - Reversible computing

SCIENTIFIC DESCRIPTION: The CauDEr reversible debugger 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 semantic for the target language, and can be used even if different processes have different notions of time

FUNCTIONAL DESCRIPTION: CauDEr is a debugger allowing one to explore the execution of concurrent 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. The debugger takes an Erlang program but debugging is done on its translation into Core Erlang.

NEWS OF THE YEAR: Work in 2019 consisted in maintenance, bug fixing and some minor refinements, in particular on the logging part.

- Partner: Universitat Politècnica de València
- Contact: Ivan Lanese
- URL: https://github.com/mistupv/cauder

6.6. SUNNY-AS

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.

- Contact: Tong Liu
- URL: https://github.com/lteu/oasc

6.7. eco-imp

Expected Cost Analysis for Imperative Programs

KEYWORDS: Software Verification - Automation - Runtime Complexity Analysis - Randomized algorithms

FUNCTIONAL DESCRIPTION: Eco-imp is a cost analyser 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 adaption 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 analyse program components in complete independence, that is, the analysis is modular and thus scalable. As a consequence, confirmed by our experiments, eco-imp runs on average orders of magnitude faster than comparable tools: execution times of several seconds become milliseconds.

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

7. New Results

7.1. Service-Oriented and Cloud Computing

Participants: Mario Bravetti, Maurizio Gabbrielli, Saverio Giallorenzo, Claudio Guidi, Ivan Lanese, Cosimo Laneve, Fabrizio Montesi, Gianluigi Zavattaro, Stefano Pio Zingaro.

7.1.1. Service-Oriented Computing and Internet of Things

Session types, i.e. types for structuring service communication, are recently being integrated into mainstream programming languages. In practice, a very important notion for dealing with such types is that of subtyping, since it allows for typing larger classes of system, where a program has not precisely the expected behavior but a similar one. We recently showed that, when asynchronous communication is considered, unfortunately, such a subtyping relation is undecidable. In [27] we present an algorithm (the first one that does not restrict type syntax or limit communication) and a tool for checking asynchronous subtyping which is sound, but not complete: in some cases it terminates without returning a final verdict. In [29] we discuss the relationship between session types and service behavioural contracts and we show the existence of a fully abstract interpretation of session types into a fragment of contracts, mapping subtyping into binary compliance-preserving contract refinement. This also yields an original undecidability result for asynchronous contract refinement.

In [43] we elaborate on our previous work on choreographies, which specify in a single artefact the expected behaviour of all the participants in a service oriented system. In particular, we extend dynamic choreographies, which model system updates at runtime, with the feature of dynamic inclusion of new unforeseen participants. In [30] we propose, in the context of platooning (a freight organization system where a group of vehicles follows a predefined trajectory maintaining a desired spatial pattern), a two layered, composable technical solution for federated platooning: a decentralized overlay network that regulates the interactions among the stakeholders, useful to mitigate issues linked to data safety and trustworthiness; and a dynamic federation platform, needed to monitor and interrupt deviant behaviors of federated members.

Finally, in [44] we focused on the use of our service-oriented language Jolie in an Internet of Things (IoT) setting. Technically, a key feature of Jolie is that it supports uniform linguistic abstractions to exploit heterogeneous communication stacks, i.e. for service oriented computing, protocols such as TCP/IP, Bluetooth, and RMI at transport level, and HTTP and SOAP at application level. We extend Jolie in order to support, uniformly as well, also the two most adopted protocols for IoT communication, i.e. CoAP and MQTT, and we report our experience on a case study on home automation.

7.1.2. Cloud Computing

In [18] we investigate the problem of modeling the optimal and automatic deployment of cloud applications and we experiment such an approach by applying it to the Abstract Behavioural Specification language ABS. In [28] we show that automated deployment, proven undecidable in the general case, is, instead, algorithmically treatable for the specific case of microservices: we implement an automatic optimal deployment tool and compute deployment plans for a realistic microservice architecture. In [35] we propose a core formal programming model (combining features from λ-calculus and π-calculus) for serverless computing, also known as Functions-as-a-Service: a recent paradigm aimed at simplifying the programming of cloud applications. The idea is that developers design applications in terms of functions and the infrastructure deals automatically with cloud deployment in terms of distribution and scaling.

7.2. Models for Reliability

Participants: Ivan Lanese, Doriana Medic.

7.2.1. Reversibility

We have continued the study of reversibility started in the past years. First, we continued to study reversibility in the context of the Erlang programming language. In particular, we devised a technique to record a program execution and replay it [37] inside the causal-consistent reversible debugger for Erlang we developed in the last years. More precisely, we may not replay the exact same execution, but any execution which is causal-consistent to it. We proved that this is enough to replay misbehaviours, hence to look for the bugs causing them. Second, we compared [48] various approaches to causal-consistent reversibility in CCS and π-calculus. In CCS, we showed that the two main approaches for causal-consistent reversibility, namely the ones of RCCS [51] and of CCSk [55] give rise to isomorphic LTSs (up to some structural rules). In π-calculus, we showed that one can define a causal semantics for π-calculus parametric on the data structure used to track extruded names, and that different instances capture causal semantics from the literature. All such semantics can be used to define (different) causal-consistent reversible semantics. As a final contribution, we studied reversibility in the context of Petri nets [41]. There, we do not considered causal-consistent reversibility, but a notion of local reversibility typical of Petri nets. In particular, we say that a transition is reversible if one can add a set of effect-reverses (an effect-reverse, if it can trigger, undoes the effect of the transition) to undo it in each marking reachable by it, without changing the set of reachable markings. We showed that, contrarily to what happens in bounded nets, transition reversibility is not decidable in general unbounded nets. It is however decidable in some significant subclasses of Petri nets, in particular all transitions of cyclic nets (nets where the initial marking is reachable from any state) are reversible. Finally, we show how to restructure nets by adding new places so to make their transitions reversible without altering their behaviour.

7.3. Probabilistic Systems and Resource Control

Participants: Martin Avanzini, Mario Bravetti, Raphaelle Crubillé, Ugo Dal Lago, Francesco Gavazzo, Gabriele Vanoni, Akira Yoshimizu.

7.3.1. Probabilistic Programming and Static Analysis

In FoCUS, we are interested in studying probabilistic higher-order programming languages and, more generally, the fundamental properties of probabilistic computation when placed in an interactive scenario, for instance concurrency. One of the most basic but nevertheless desirable properties of programs is of course termination. Termination can be seen as a minimal guarantee about the time complexity of the underlying program. When probabilistic choice comes into play, termination can be defined by stipulating that a program is terminating if its probability of convergence is 1, this way giving rise to the notion of *almost sure termination*. Alternatively, a probabilistic program is said to be *positively* almost surely terminating if its average runtime is finite. The latter condition easily implies the former. Termination, already undecidable for deterministic (universal) programming languages, remains so in the presence of probabilistic choice, even becoming provably harder.

The FoCUS team has been the first in advocating the use of types to guarantee probabilistic termination, in the form of a monadic sized-type system [17]. Developed in collaboration with Grellois by Dal Lago, this system substantially generalises usual sized-types, and allows this way to capture probabilistic, higher-order programs which terminate almost surely. Complementary, in collaboration with Ghyselen, Avanzini and Dal Lago have recently defined a formal system for reasoning about the *expected runtime* of higher-order probabilistic programs, through a *refinement type system* capable of *modeling probabilistic effects* with exceptional accuracy [26]. To the best of our knowledge, this provides the first formal methodology for *average case complexity analysis* of higher-order programs. Remarkably, the system is also *extensionally complete*.

In 2018, we have started to investigate the foundations for *probabilistic abstract reduction systems* (*probabilistic ARSs*), which constitute a general framework to study fundamental properties of probabilistic computations, such as termination or confluence. In 2019, we have significantly revised this initial development [11]. Particularly, we have refined Lyapunov ranking functions by conceiving them as *probabilistic embeddings*. The ramifications of this work are two-fold. First, we obtain a sound and complete method for reasoning about strong positive almost sure termination. Second, this method has been instantiated in the setting of (first-order) *probabilistic rewrite systems*, giving rise to the notion of *barycentric algebras*, generalising the well-known interpretation method. Barycentric algebras have been integrated in the termination prover *NaTT* [1], confirming the feasibility of the approach.

We have also worked on higher-order model checking as a way to prove termination of probabilsitic variations on higher-order recursion schemes [36], obtaining encouraging results. More specifically, an algorithm for approximating the probability of convergence of any such scheme has been designed and proved sound, although the problem of precisely computing the probability of convergence is shown to be undecidable at order 2 or higher. Finally, we have published a new version of a contribution we wrote in 2017 about how implicit computational complexity could help in proving that certain cryptographic constructions have the desired complexity-theoretic properties [12].

7.3.2. Higher-Order end Effectful Programs: Relational Reasoning

In FoCUS, we are also interested in relational reasoning about programs written in higher-order programming languages. In the recent years, this research has been directed to effectful programs, namely programs whose behaviour is not purely functional. Moreover, there has recently been a shift in our interests, driven by the projects REPAS and DIAPASoN, towards quantitative kinds of relational reasoning, in which programs are not necessarily dubbed equivalent (or not), but rather put at a certain distance.

The first contribution we had in this direction is due to Dal Lago and Gavazzo [31], who generalized the so-called open normal-form bisimilarity technique to higher-order programs exhibiting any kind of monadic effect. The key ingredient here is that of a relator, and allows to lift relations on a set to relations on monadic extensions to the same set. This allows to define open normal-form bisimilarity, and to prove it correct. This, together, with other contributions, have also appeared in Gavazzo's PhD Thesis, which has been successfully defended in April 2019 [10], and which has been awarder the Prize for the Best PhD Thesis in Theoretical Computer Science by the Italian Chapter of the EATCS.

We have also given the notion of differential logical relations [33], a generalization of Plotkin's logical relations in which programs are dubbed being at a certain *distance* rather than being just *equivalent*. Noticeably, this distance is not necessarily numeric, but is itself functional if the compared programs have a non-ground type. This allows to evaluate the distance between programs taking into account the possible actions the environment can make on the compared programs.

7.3.3. Alternative Probabilistic Models

We are also interested in exploring probabilistic models going beyond the usual ones, in which determinisitic programming languages are endowed with discrete probabilistic choice.

[1] See https://www.trs.css.i.nagoya-u.ac.jp/NaTT/.

We have first of all studied bayesian λ-calculi, namely λ-calculi in which not only an operator for probabilistic choice is available, but also one for *scoring*, which serves as the basis to model conditioning in probabilistic programming. We give a geometry of interaction model for such a typed λ-calculus [34], namely a paradigmatic calculus for higher-order Bayesian programming in the style of PCF. The model is based on the category of measurable spaces and partial measurable functions, and is proved adequate with respect to both a distribution-based and a sampling-based operational semantics.

We have also introduced a probabilistic extension of a framework to specify and analyze software product lines [15]. We define a syntax of the language including probabilistic operators and define operational and denotational semantics for it. We prove that the expected equivalence between these two semantic frameworks holds. Our probabilistic framework is supported by a set of scripts to show the model behavior.

7.4. Verification Techniques

Participants: Ugo Dal Lago, Adrien Durier, Daniel Hirschkoff, Ivan Lanese, Cosimo Laneve, Davide Sangiorgi, Akira Yoshimizu, Gianluigi Zavattaro.

Extensional properties are those properties that constrain the behavioural descriptions of a system (i.e., how a system looks like from the outside). Examples of such properties include classical functional correctness, deadlock freedom and resource usage.

In the last year of the Focus project, we have worked on three main topics: (*i*) *name mobility and coinductive techniques*, (*ii*) *deadlock analysis*, and (*iii*) *cost analysis of properties* of languages for actors and for smart contracts.

7.4.1. Name Mobility and Coinductive Techniques

In [19], we propose proof techniques for bisimilarity based on unique solution of equations. The results essentially state that an equation (or a system of equations) whose infinite unfolding never produces a divergence has the unique-solution property. We distinguishing between different forms of divergence; derive an abstract formulation of the theorems, on generic LTSs; adapt the theorems to other equivalences such as trace equivalence, and to preorders such as trace inclusion; we compare the resulting techniques to enhancements of the bisimulation proof method (the 'up-to techniques'). In [20], we study how to adapt such techniques to higher-order languages. In such languages proving behavioural equivalences is known to be hard, because interactions involve complex values, namely terms of the language. The soundness of proof techniques is usually delicate and difficult to establish. The language considered is the Higher-Order π-calculus.

The contribution [42] studies the representation of the call-by-need λ-calculus in the pure message-passing concurrency of the π-calculus, precisely the Local Asynchronous π-calculus, that has sharper semantic properties than the ordinary π-calculus. We exploit such properties to study the validity of of β-reduction (meaning that the source and target terms of a beta-reduction are mapped onto behaviourally equivalent processes). Nearly all results presented fail in the ordinary π-calculus.

In [45], we investigate basic properties of the Erlang concurrency model. This model is based on asynchronous communication through mailboxes accessed via pattern matching. In particular, we consider Core Erlang (which is an intermediate step in Erlang compilation) and we define, on top of its operational semantics, an observational semantics following the approach used to define asynchronous bisimulation for the π-calculus. Our work allows us to shed some light on the management of process identifiers in Erlang, different from the various forms of name mobility already studied in the literature. In fact, we need to modify standard definitions to cope with such specific features of Erlang.

The paper [25] reviews the origins and the history of enhancements of the bisimulation and coinduction proof methods.

7.4.2. Deadlock Analysis

The contributions [22] and [50] address deadlock analysis of `Java`-like programs. The two papers respectively cover two relevant features of these languages: (*i*) multi-threading and reentrant locks and (*ii*) co-ordination primitives (`wait`, `notify` and `notifyAll`). In both cases, we define a behavioral type system that associates abstract models to programs (lams and Petri Nets with inhibitor arcs) and define an algorithm for detecting deadlocks. The two systems are consistent and our technique is intended to be an effective tool for the deadlock analysis of programming languages.

The paper [16] addresses the π-calculus. It defines a type system for guaranteing that typable processes never produce a run-time error and, even if they may diverge, there is always a chance for them to finish their work, i.e., to reduce to an idle process (a stronger property than deadlock freedom). The type system uses so-called *non-idempotent intersections* and, therefore, applies to a large class of processes. Indeed, despite the fact that the underlying property is $\prod_2^0$-complete, there is a way to show that the system is complete, i.e., that any well-behaved process is typable, although for obvious reasons infinitely many derivations need to be considered.

7.4.3. Static Analysis of Properties of Concurrent Programs

We have analyzed the computational time of actor programs, following a technique similar to [52], and we have begun a new research direction that deals with the analysis of `Solidity` smart contracts.

In [23], we propose a technique for estimating the computational time of programs in an actor model. To this aim, we define a compositional translation function returning cost equations, which are fed to an automatic off-the-shelf solver for obtaining the time bounds. Our approach is based on so-called *synchronization sets* that capture possible difficult synchronization patterns between actors and helps make the analysis efficient and precise. The approach is proven to correctly over-approximate the worst computational time of an actor model of concurrent programs. The technique is complemented by a prototype analyzer that returns upper bound of costs for the actor model.

In [38], we analyze the bahaviour of smart contracts, namely programs stored on some blockchain that control the transfer of assets between parties under certain conditions. In particular, we focus on the interactions of smart contracts and external actors (usually, humans) in order to maximize objective functions. 5 To this aim, we define a core language of programs, which is reminiscent of `Solidity`, with a minimal set of smart contract primitives and we describe the whole system as a parallel composition of smart contracts and users. We therefore express the system behaviour as a first order logic formula in Presburger arithmetics and study the maximum profit for each actor by solving arithmetic constraints.

7.5. Computer Science Education

Participants: Michael Lodi, Simone Martini.

We study why and how to teach computer science principles (nowadays often referred to as “computational thinking”, CT), in the context of K-12 education. We are interested in philosophical, sociological, and historical motivations to teach computer science. Furthermore, we study what concepts and skills related to computer science are not only technical abilities, but have a general value for all students. Finally, we try to find/produce/evaluate suitable materials (tools, languages, lesson plans...) to teach these concepts, taking into account: difficulties in learning CS concepts (particularly programming); stereotypes about computer science (teachers’ and students’ mindset); teacher training (both non-specialist and disciplinary teachers); innovative teaching methodologies (primarily based on constructivist and constructionist learning theories).

7.5.1. Computational Thinking, Unplugged Activities, and Constructionism

We reviewed some relevant literature related to learning CS and, more specifically, programming in a constructivist and constructionist light. We investigated some cognitive aspects, for example, the notional machine and its role in understanding, misunderstanding, and difficulties of learning to program. We reviewed programming languages for learning to program, with particular focus on educational characteristics of block-based languages [24].

We analyzed the widespread but debated pedagogical approach of “unplugged activities”: activities without a computer, like physical games, used to teach CS concepts. We explicitly connect computational thinking to the “CS Unplugged” pedagogical approach, by analyzing a representative sample of CS Unplugged activities in light of CT. We found the activities map well onto commonly accepted CT concepts, although caution must be taken not to regard CS Unplugged as being a complete approach to CT education [14].

Moreover, we found similarities (e.g., kinesthetic activities) and differences (e.g., structured vs. creative activities) between Unplugged and constructivism or constructionism. We argue there is a tension between the constructivist need to link the CS concepts to actual implementations and the challenge of teaching CS principles without computers, to undermine the misconceptions of CS as “the science of computers” [13].

7.5.2. CS in Primary School

We designed, produced and implemented in a primary school some “unplugged + plugged” teaching materials and lesson plans [47]. The unplugged activities are structured as an incremental discovery, scaffolded by the instructors, of the fundamental concepts of structured programming (e.g., sequence, conditionals, loops, variables) but also complexity in terms of computational steps and generalization of algorithms. The plugged activities follow the creative learning approach, using Scratch as the primary tool, both for free creative expression and for learning other disciplines (e.g., drawing regular polygons).

7.5.3. Growth Mindset and Transfer

Every person holds an idea (mindset) about intelligence: someone thinks it is a fixed trait, like eye colour (fixed mindset), while others believe it can grow like muscles (growth mindset). The latter is beneficial for students to have better results, particularly in STEM disciplines, and to not being influenced by stereotypes. Computer science is a subject that can be affected by fixed ideas (“geek gene”), and some (small) studies showed it can induce fixed ideas. By contrast, some claims stating that studying CS can foster a GM have emerged. However, educational research shows that the transfer of competences is hard. We measured [40] some indicators (e.g., mindset, computer science mindset) at the beginning and the end of a high school year in different classes, both CS and non-CS oriented. At the end of the year, none of the classes showed a statistically significant change in their mindset. Interestingly, non-CS oriented classes showed a significant decrease in their computer science growth mindset, which is not desirable.

7.6. Constraint Programming

Participants: Maurizio Gabbrielli, Liu Tong.

In Focus, we sometimes make use of constraint solvers (e.g., cloud computing, service-oriented computing). Since a few years we have thus began to develop tools based on constraints and constraint solvers.

In [39] we have used constraints in the setting of Service Function Chaining (SFC) deployment. SFCs represent sequences of Virtual Network Functions that compose a service. They are found within Network Function Virtualization (NFV) and Software Defined Networking (SDN) technologies, that recently acquired a great momentum thanks to their promise of being a flexible and cost-effective solution for replacing hardware-based, vendor-dependent network middleboxes with software appliances running on general purpose hardware in the cloud.

We employ constraint programming to solve the SFC design problem. Indeed we argue that constraint programming can be effectively used to address this kind of problems because it provides expressive and flexible modeling languages which come with powerful solvers, thus providing efficient and scalable performance.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

In 2019 we have started the Innovation Lab on Blockchain and New Technologies (https://site.unibo.it/blockchain-and-newtechnologies/en). The Lab is a new joint laboratory of the Computer Science and Engineering Department of the University of Bologna and KPMG Advisory S.p.A. that is committed to scientific research and technology transfer of systems based on blockchain and new technologies. The laboratory joins the efforts of several researchers of the Department and uses the experience in technology transfer of KPMG Advisory S.p.A.

The Lab has received a grant of 10KE from KPMG and a grant of 10KE from CIRFOOD, one of the biggest Italian companies in organised commercial and collective catering.

9. Partnerships and Cooperations

9.1. National Initiatives

- DCore (Causal debugging for concurrent systems) is a 4-years ANR project that started on March 2019. 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. Main persons involved: Lanese, Medic.
- REPAS (Reliable and Privacy-Aware Software Systems via Bisimulation Metrics) is an ANR Project that started on October 2016 and that will finish on October 2020. The project aims at investigating quantitative notions and tools for proving program correctness and protecting privacy. In particular, the focus will be put on bisimulation metrics, which are the natural extension of bisimulation to quantitative systems. As a key application, we will develop a mechanism to protect the privacy of users when their location traces are collected. Main persons involved: Dal Lago, Gavazzo, Sangiorgi.
- COCAHOLA (Cost models for Complexity Analyses of Higher-Order Languages) is an ANR Project that started on October 2016 and that finished on October 2019. The project aims at developing complexity analyses of higher-order computations. The focus is not on analyzing fixed programs, but whole programming languages. The aim is the identification of adequate units of measurement for time and space, i.e. what are called *reasonable* cost models. Main persons involved: Dal Lago, Martini.
- PROGRAMme ("What is a program? Historical and philosophical perspectives"), is an ANR project started on October 2017 and that will finish on October 2022; PI: Liesbeth De Mol (CNRS/Université de Lille3). The aim of this project is to develop a coherent analysis and pluralistic understanding of "computer program" and its implications to theory and practice. Main person involved: Martini.

9.2. European Initiatives

9.2.1. FP7 & H2020 Projects

- BEHAPI (Behavioural Application Program Interfaces) is an European Project H2020-MSCA-RISE-2017, running in the period March 2018 - February 2022. The topic of the project is behavioural types, as a suite of technologies that formalise 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 behavioural 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.

- ICT COST Action IC1405 (Reversible computation - extending horizons of computing). Initiated at the end of April 2015 and with a 4-year duration, this COST Action studies reversible computation and its potential applications, which include circuits, low-power computing, simulation, biological modeling, reliability and debugging. Reversible computation is an emerging paradigm that extends the standard forwards-only mode of computation with the ability to execute in reverse, so that computation can run backwards as naturally as it can go forwards.

  Main persons involved: Lanese (vice-chair of the action).

9.2.2. Collaborations with Major European Organizations

We list here the cooperations and contacts with other groups, without repeating those already listed in previous sections.

- ENS Lyon (on concurrency models and resource control). Contact person(s) in Focus: Dal Lago, Martini, Sangiorgi. Some visit exchanges during the year, in both directions. A joint PhD (Adrien Durier).
- University of Innsbruck (on termination and complexity analysis of probabilistic programs). Contact person(s) in Focus: Avanzini. Some short visits during the year.
- University of Southern Denmark (on service-oriented computing). Contact person(s) in Focus: Gabbrielli, Lanese, Zavattaro.
- Universitat Politecnica de Valencia, Spain (on reversibility for Erlang). Contact person(s) in Focus: Lanese. Some visit exchanges during the year, in both directions.
- Laboratoire d'Informatique, Université Paris Nord, Villetaneuse (on implicit computational complexity). Contact person(s) in Focus: Dal Lago, Martini.
- Institut de Mathématiques de Luminy, Marseille (on lambda-calculi, linear logic and semantics). Contact person(s) in Focus: Dal Lago, Martini.
- Team PPS, IRIF Lab, University of Paris-Diderot Paris 7 (on logics for processes, resource control). Contact person(s) in Focus: Dal Lago, Martini, Sangiorgi. Some short visits in both directions during the year.
- IRILL Lab, Paris (on models for the representation of dependencies in distributed package based software distributions). Contact person(s) in Focus: Gabbrielli, Zavattaro. Some short visits in both directions during the year.
- IMDEA Software, Madrid (G. Barthe) (on implicit computational complexity for cryptography). Contact person(s) in Focus: Dal Lago. Some visits during the year.

9.3. International Initiatives

9.3.1. Inria Associate Teams Not Involved in an Inria International Lab

9.3.1.1. CRECOGI

Title: Concurrent, Resourceful and Effectful Computation by Geometry of Interaction

International Partner (Institution - Laboratory - Researcher):

Kyoto (Japan) - Research Institute for Mathematical Sciences - Naohiko Hoshino

Start year: 2018

See also: http://crecogi.cs.unibo.it

The field of denotational semantics has successfully produced useful compositional reasoning principles for program correctness, such as program logics, fixed-point induction, logical relations, etc. The limit of denotational semantics was however that it applies only to high-level languages and to extensional properties. The situation has changed after the introduction of game semantics and the geometry of interaction (GoI), in which the meaning of programs is formalized in terms of movements of tokens, through which programs "talk to" or "play against" each other, thus having an operational flavour which renders them suitable as target language for compilers. The majority of the literature on GoI and games only considers sequential functional languages. Moreover, computational effects (e.g. state or I/O) are rarely taken into account, meaning that they are far from being applicable to an industrial scenario. This project's objective is to develop a semantic framework for concurrent, resourceful, and effectful computation, with particular emphasis on probabilistic and quantum effects. This is justified by the greater and greater interest which is spreading around these two computation paradigms, motivated by applications to AI and by the efficiency quantum parallelism induces.

9.3.2. Participation in Other International Programs

Focus has taken part in the creation of the Microservices Community (http://microservices.sdu.dk/), an international community interested in the software paradigm of Microservices. Main aims of the community are: i) sharing knowledge and fostering collaborations about microservices among research institutions, private companies, universities, and public organisations (like municipalities); ii) discussing open issues and solutions from different points of view, to create foundations for both innovation and basic research.

U. Dal Lago is “Partner Investigator” in the project “Verification and analysis of quantum programs”, whose Chief Investigator is Prof Yuan Feng, University of Technology Sydney. The project is funded by the Australian Research Council.

9.4. International Research Visitors

9.4.1. Visits of International Scientists

The following researchers have visited Focus for short periods; we list them together with the title of the talk they have given during their stay, or the topic discussed during their stay.

- Ornela Dardha (University of Glasgow) and Laura Bocchi (University of Kent): collaboration within BehAPI RISE H2020 project, September 2019.
- Guilhem Jaber (University of Nantes): “Game semantics for higher-order functions with state”, December 2019.
- Naohiko Hoshino, April 2019 and October 2019.
- Gilles Barthe, May 2019.
- Boaz Barak, July 2019.
- Francesco Dagnino, “Generalizing Inference Systems by Corules", November 2019.

9.4.1.1. Sabbatical programme

Simone Martini has been Fellow at the Collegium - Lyon Institute for Advanced Studies, since September 2018 and until June 2019 https://collegium.universite-lyon.fr.

9.4.1.2. Research Stays Abroad

- Ugo Dal Lago has spent overall a few weeks in Japan: RIMS (Kyoto) and NII (Tokyo), as part of ongoing collaborations with Naohiko Hoshino and Shin-ya Katsumata.
- Ivan Lanese has visited Xibis Limited and University of Leicester, UK (in particular Irek Ulidowski and Emilio Tuosto) from 3/7/2019 to 2/8/2019, to work on choreographies, and the University of Torun, Poland (in particular Lukasz Mikulski and Kamila Barylska), from 13/8/2019 to 29/8/2019, to work on reversible Petri nets.
- Cosimo Laneve and Gianluigi Zavattaro have spent overall a few weeks in Malta visit to Prof. Adrian Francalanza at the University of Malta within the BehAPI RISE H2020 project.
- Michael Lodi has visited Prof. Tim Bell and the Computer Science Education Research Group at the Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, from 26th of October 2018 to 17th of April 2019, as part of his Ph.D. course.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Organisation

10.1.1.1. General Chair, Scientific Chair

Mario Bravetti has been scientfiic co-organizer for Int. PhD School and Bootcamp on Behavioural Application Program Interfaces (BehAPI 2019), Leicester, UK.

Ugo Dal Lago has organized the Second Workshop on Probabilistic Interactive and Higher-Order Computation (6-7 February 2019) http://pihoc2019.cs.unibo.it/.

10.1.1.2. Member of the Organizing Committees

Local Organising Committee:

Michael Lodi has been a member of the Local Organising Committee of the 13th Conference of European Science Education Research Association (ESERA '19). Bologna, 26th - 30th August 2019

Steering Committee membership:

I. Lanese: Conference on Reversible Computation (RC); IFIP Int. Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE); Interaction and Concurrency Experience (ICE)

D. Sangiorgi: Int. Conference on Concurrency Theory (CONCUR)

10.1.2. Scientific Events: Selection

10.1.2.1. Member of the Conference Program Committees

M. Bravetti: 22nd International Conference on Fundamental Approaches to Software Engineering (FASE/ETAPS 2019); IEEE International Conference on Big Data (BigData 2019); 19th IEEE International Conference on Software Quality, Reliability, and Security (QRS 2019); 8th IPM International Conference on Fundamentals of Software Engineering (FSEN 2019)

U. Dal Lago: Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019; 47th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2020); 28th European Symposium on Programming (ESOP 2019); 4th International Conference on Formal Structures for Computation and Deduction (FSCD 2019)

I. Lanese: 11th Conference on Reversible Computation (RC 2019); 12th Interaction and Concurrency Experience (ICE 2019); 16th International Conference on Formal Aspects of Component Software (FACS 2019); 12th IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2019); First Workshop on Artificial Intelligence and fOrmal VERification, Logic, Automata, and sYnthesis (OVERLAY 2019); 4th Workshop on Formal Reasoning about Causation, Responsibility, and Explanations in Science and Technology (CREST 2019); 2nd International Conference on Microservices (MICROSERVICES 2019); 12th Innovations in Software Engineering Conference (ISEC 2019)

S. Martini: Workshop on History of Formal Methods (HFM2019); Fifth International Conference on History and Philosophy of Computing (HAPOC 5).

D. Sangiorgi: 22nd International Conference on Foundations of Software Science and Computation Structures (FOSSACS); 44th International Symposium on Mathematical Foundations of Computer Science (MFCS); 12th A.P. Ershov Informatics Conference (PSI); Workshop on History of Formal Methods (HFM2019); 15th International Conference on Software Technologies (ICSOFT)

G. Zavattaro: 17th International Conference on Service Oriented Computing (ICSOC'19); International Conference TOOLS 50+1: Technology of Object-Oriented Languages and Systems (TOOLS'19); 30th International Conference on Concurrency Theory (CONCUR'19); 34th ACM/SIGAPP Symposium On Applied Computing – track on Microservices, DevOps, and Service-Oriented Architecture (ACM-SAC 2019)

10.1.3. Journals

10.1.3.1. Member of the Editorial Boards

M. Bravetti: Journal of Universal Computer Science.

U. Dal Lago: Logical Methods in Computer Science; Mathematical Structures in Computer Science; Acta Informatica.

M. Gabbrielli: Int. Journal Theory and Practice of Logic Programming.

C. Laneve: Frontiers in ICT (Section Formal Methods).

I. Lanese: Editor in chief of the Open Journal of Communications and Software (Scientific Online).

D. Sangiorgi: Acta Informatica, Distributed Computing, RAIRO Theoretical Informatics and Applications.

10.1.4. Invited Talks

U. Dal Lago: Bellairs Workshop on Higher-Order Probabilistic Computation; 1st Computer Science Workshop; “Mission 10000 Conference: Quantum Science and Technologies”

D. Sangiorgi: International Conference TOOLS 50+1: Technology of Object-Oriented Languages and Systems (TOOLS’19)

Schools:

M. Avanzini: International School on Rewriting, Paris, France, 1–6 July, 2019

U. Dal Lago: 1st “Caleidoscope” Summer School

G. Zavattaro: BehAPI Summer School: Behavioural Approaches for API-Economy with Applications, Leicester, UK, 8–12 July, 2019

10.1.5. Leadership within the Scientific Community

U. Dal Lago has been elected member of the Scientific Council of the Italian Chapter IC-EATCS (November 2017).

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, 2017-2021.

G. Zavattaro is member of the scientific committee of GRIN (GRuppo INformatici), Italy.

10.1.6. Administration duties

M. Gabbrielli is Deputy Head of the Department of Computer Science and Engineering, University of Bologna, since May 2018.

D. Sangiorgi is coordinator of postgraduate studies at the Department of Computer Science and Engineering, University of Bologna.

G. Zavattaro is coordinator of undergraduate studies at the Department of Computer Science and Engineering, University of Bologna.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

- Mario Bravetti

  Master: “Linguaggi, Compilatori e Modelli Computazionali”, 120 hours, 1st year, University of Bologna, Italy.
- Ugo Dal Lago

  Undergraduate: “Algorithms and Data Structures for Biology”, 60 hours, 2nd year, University of Bologna, Italy. 20 hours, 1st year, University of Bologna, Italy.

  Undergraduate: “Optimization”, 36 hours, 2nd year, University of Bologna, Italy.

  Master: “Foundations of Logic for Computer Science”, 24 Hours, 2nd year. University of Bologna, Italy.

  Master: “Cryptography”, 40 Hours, 2nd year, University of Bologna, Italy’.

  Master: “Languages and Algorithms for AI: Machine Learning Theory”, 32 Hours, 1st year, University of Bologna, Italy’.
- Maurizio Gabbrielli

  Undergraduate: “Programming languages”, 40 hours, 2nd year, University of Bologna, Italy.

  Master: “Artificial Intelligence”, 60 hours, 2nd year, University of Bologna, Italy.
- Francesco Gavazzo

  Undergraduate: “Programming languages”, 30 hours, 2nd year, University of Bologna, Italy.

  Undergraduate: “Basic Computer Skills”. 30 hours, BSc Medical Chemistry and Pharmaceutical Technology, BSc Biology, University of Bologna.
- Ivan Lanese

  Undergraduate: “Architettura degli Elaboratori”, 66 hours, 1st year, University of Bologna, Italy.

  Master: “Ingegneria del Software Orientata ai Servizi”, 22 hours, 2nd year, University of Bologna, Italy.

  Master: “Algorithms and data structures for computational biology”, 36 hours, 1at year, University of Bologna, Italy.

  Master: “Programming for bioinformatics”, 30 hours, 1st year, University of Bologna, Italy.
- Cosimo Laneve

  Undergraduate: “Programmazione”, 70 hours, 1st year, University of Bologna, Italy.

  Master: “Analisi di Programmi”, 42 hours, 1st year, University of Bologna, Italy.
- Simone Martini

  Master: “Introduction to Algorithms and Programming”, 32 hours, 1st year, University of Bologna, Italy.
- Davide Sangiorgi

  Undergraduate: “Operating Systems”, 110 hours, 2nd year, University of Bologna, Italy.

  Undergraduate: “Computer abilities for biologists”, 8 hours, 1st year, University of Bologna, Italy.
- Gianluigi Zavattaro

Master: “Scalable and Cloud Programming”, 50 hours, 2nd year, University of Bologna, Italy.

Undergraduate: “Algoritmi e strutture dati”, 60 hours, 2nd year, University of Bologna, Italy.

Master: “Languages and Algorithms for Artificial Intelligence”, 32 hours, 1st year, University of Bologna, Italy (Master in Artificial Intelligence).

10.2.2. Supervision

Below are the details on the PhD students in Focus: starting date, topic or provisional title of the thesis, supervisor(s).

- Melissa Antonellii, November 2019. “Probabilistic Arithmetic and Almost-sure Termination”. Supervisor Ugo Dal Lago.
- Adrien Durier, September 2016, "Proving behavioural properties of higher-order concurrent languages", ENS de Lyon and University of Bologna. Supervisors: Daniel Hirschkoff and Davide Sangiorgi.
- Michael Lodi, January 2017, “Introducing Computational Thinking in K-12 Education: Historical, Epistemological, Cognitive and Affective Aspects”. Supervisor: S. Martini.
- Gabriele Vanoni, November 2018. “Optimal Reduction, Geometry of Interaction, and the Space-Time Tradeoff”. Supervisor Ugo Dal Lago.
- Stefano Pio Zingaro, November 2016, “High level languages for Internet of Things applications”. Supervisor: Maurizio Gabbrielli.

PhD thesis completed in 2018:

- Raphaelle Crubillé, October 2015, “Bisimulation Metrics and Probabilistic Lambda Calculi”, Université Denis Diderot and University of Bologna. Supervisors Thomas Ehrhard and Ugo Dal Lago.
- Francesco Gavazzo, October 2015, “Coinductive Techniques for Effectful Lambda Calculi”. Supervisor U. Dal Lago.
- Tong Liu, November 2015, “Constraint based languages for Software Defined Networks”. Supervisor: Maurizio Gabbrielli.

10.2.3. Juries

G. Zavattaro has been member of the PhD evaluation committee of Doriana Medic, supervisor Claudio Antares Mezzina, IMT Lucca, Italy.

10.3. Popularization

Michael Lodi and Simone Martini have carried out extended work of scientific popularization, including the following.

- They are members of the technical committee of Olimpiadi del Problem Solving (at Italian Ministry of Education), http://www.olimpiadiproblemsolving.com; this involves preparation of material and supervision and jury during the finals.
- Simone Martini has given the following talks, among others:
  - De la création d’une “théorie mathématique du calcul” à la “pensée informatique”. Quatrième journée académique sur l’enseignement de l’informatique, Marseille (avril 2019)
  - To code or not to code: the school curriculum facing the digital revolution. Collegium—Lyon Institute for Advanced Studies (Juin 2019)
  - Logic and computing in Italy at the birth of the Italian Computer Science, Roma Tre, September 2019

11. Bibliography

Major publications by the team in recent years

[1] M. BRAVETTI, G. ZAVATTARO. *A Foundational Theory of Contracts for Multi-party Service Composition*, in "Fundam. Inform.", 2008, vol. 89, n^{o} 4, pp. 451-478

[2] N. BUSI, M. GABBRIELLI, G. ZAVATTARO. *On the expressive power of recursion, replication and iteration in process calculi*, in "Mathematical Structures in Computer Science", 2009, vol. 19, n^{o} 6, pp. 1191-1222

[3] P. COPPOLA, S. MARTINI. *Optimizing optimal reduction: A type inference algorithm for elementary affine logic*, in "ACM Trans. Comput. Log.", 2006, vol. 7, n^{o} 2, pp. 219-260

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

[5] D. HIRSCHKOFF, É. LOZES, D. SANGIORGI. *On the Expressiveness of the Ambient Logic*, in "Logical Methods in Computer Science", 2006, vol. 2, n^{o} 2

[6] U. D. LAGO, 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, J. STEFANI. *Reversibility in the higher-order π-calculus*, in "Theor. Comput. Sci.", 2016, vol. 625, pp. 25–84, https://doi.org/10.1016/j.tcs.2016.02.019

[8] F. MONTESI, C. GUIDI, 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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[10] F. GAVAZZO. *Coinductive Equivalences and Metrics for Higher-order Languages with Algebraic Effects*, Alma Mater Studiorum Università di Bologna, April 2019, https://hal.inria.fr/tel-02386201

Articles in International Peer-Reviewed Journals

[11] M. AVANZINI, U. DAL LAGO, A. YAMADA. *On probabilistic term rewriting*, in "Science of Computer Programming", January 2020, vol. 185, 102338 p. [*DOI :* 10.1016/J.SCICO.2019.102338], https://hal.inria.fr/hal-02381877

[12] P. BAILLOT, G. BARTHE, U. DAL LAGO. *Implicit Computational Complexity of Subrecursive Definitions and Applications to Cryptographic Proofs*, in "Journal of Automated Reasoning", December 2019, vol. 63, n^{o} 4, pp. 813-855 [*DOI :* 10.1007/978-3-662-48899-7_15], https://hal.archives-ouvertes.fr/hal-01197456

[13] T. BELL, M. LODI. *Authors' Response: Keeping the "Computation" in "Computational Thinking" Through Unplugged Activities*, in "Constructivist foundations", July 2019, vol. 14, n^{o} 3, pp. 357-359, https://hal.inria.fr/hal-02378782

[14] T. BELL, M. LODI. *Constructing Computational Thinking Without Using Computers*, in "Constructivist foundations", July 2019, vol. 14, n^{o} 3, pp. 342-351, https://hal.inria.fr/hal-02378761

[15] C. CAMACHO, L. LLANA, A. NÚÑEZ, M. BRAVETTI. *Probabilistic Software product lines*, in "Journal of Logical and Algebraic Methods in Programming", October 2019 [*DOI :* 10.1016/J.JLAMP.2019.05.007], https://hal.inria.fr/hal-02387462

[16] U. DAL LAGO, M. DE VISME, D. MAZZA, A. YOSHIMIZU. *Intersection Types and Runtime Errors in the Pi-Calculus*, in "Proceedings of the ACM on Programming Languages", January 2019, vol. 3, n^{o} POPL, pp. 1-29 [*DOI :* 10.1145/3290320], https://hal.archives-ouvertes.fr/hal-02399565

[17] U. DAL LAGO, C. GRELLOIS. *Probabilistic Termination by Monadic Affine Sized Typing*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", June 2019, vol. 41, n^{o} 2, pp. 1-65 [*DOI :* 10.1145/3293605], https://hal.archives-ouvertes.fr/hal-02399423

[18] S. DE GOUW, J. MAURO, G. ZAVATTARO. *On the modeling of optimal and automatized cloud application deployment*, in "Journal of Logical and Algebraic Methods in Programming", October 2019, vol. 107, pp. 108-135 [*DOI :* 10.1016/J.JLAMP.2019.06.001], https://hal.inria.fr/hal-02401380

[19] A. DURIER, D. HIRSCHKOFF, D. SANGIORGI. *Divergence and unique solution of equations*, in "Logical Methods in Computer Science", August 2019, https://arxiv.org/abs/1806.11354 - This is an extended version of the paper with the same title published in the proceedings of CONCUR'17 [*DOI :* 10.23638/LMCS-15(3:12)2019], https://hal.archives-ouvertes.fr/hal-02376814

[20] A. DURIER, D. HIRSCHKOFF, D. SANGIORGI. *Towards 'up to context' reasoning about higher-order processes*, in "Theoretical Computer Science", 2019, forthcoming [*DOI :* 10.1016/J.TCS.2019.09.036], https://hal.archives-ouvertes.fr/hal-01857391

[21] M. FALASCHI, M. GABBRIELLI, C. OLARTE, C. PALAMIDESSI. *Dynamic slicing for Concurrent Constraint Languages*, in "Fundamenta Informaticae", 2019, forthcoming, https://hal.archives-ouvertes.fr/hal-02423973

[22] C. LANEVE. *A lightweight deadlock analysis for programs with threads and reentrant locks*, in "Science of Computer Programming", 2019, vol. 181, pp. 64 - 81 [*DOI :* 10.1016/J.SCICO.2019.06.002], https://hal.inria.fr/hal-02392938

[23] C. LANEVE, M. LIENHARDT, K. I. PUN, G. ROMÁN-DÍEZ. *Time analysis of actor programs*, in "Journal of Logical and Algebraic Methods in Programming", 2019, vol. 105, pp. 1 - 27 [*DOI :* 10.1016/J.JLAMP.2019.02.007], https://hal.inria.fr/hal-02392909

[24] M. LODI, D. MALCHIODI, M. MONGA, A. MORPURGO, B. SPIELER. *Constructionist Attempts at Supporting the Learning of Computer Programming: A Survey*, in "Olympiads in Informatics: An International Journal", July 2019, vol. 13, pp. 99-121 [*DOI :* 10.15388/IOI.2019.07], https://hal.inria.fr/hal-02379084

[25] D. POUS, D. SANGIORGI. *Bisimulation and Coinduction Enhancements: A Historical Perspective*, in "Formal Aspects of Computing", December 2019, vol. 31, n^{o} 6, pp. 733-749 [*DOI :* 10.1007/S00165-019-00497-W], https://hal.archives-ouvertes.fr/hal-02393949

International Conferences with Proceedings

[26] M. AVANZINI, U. DAL LAGO, A. GHYSELEN. *Type-Based Complexity Analysis of Probabilistic Functional Programs*, in "2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)", Vancouver, Canada, IEEE, June 2019, pp. 1-13 [*DOI :* 10.1109/LICS.2019.8785725], https://hal.inria.fr/hal-02381829

[27] M. BRAVETTI, M. CARBONE, J. LANGE, N. YOSHIDA, G. ZAVATTARO. *A Sound Algorithm for Asynchronous Session Subtyping*, in "CONCUR 2019 - 30th International Conference on Concurrency Theory", Amsterdam, Netherlands, August 2019 [*DOI :* 10.4230/LIPICS.CONCUR.2019.38], https://hal.inria.fr/hal-02387473

[28] M. BRAVETTI, S. GIALLORENZO, J. MAURO, I. TALEVI, G. ZAVATTARO. *Optimal and Automated Deployment for Microservices*, in "Fundamental Approaches to Software Engineering - 22nd International Conference, FASE 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings", Prague, Czech Republic, April 2019 [*DOI :* 10.1007/978-3-030-16722-6_21], https://hal.inria.fr/hal-02387483

[29] M. BRAVETTI, G. ZAVATTARO. *Relating Session Types and Behavioural Contracts: the Asynchronous Case*, in "Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings", Oslo, Norway, September 2019, https://hal.inria.fr/hal-02387456

[30] F. CALLEGATI, M. GABBRIELLI, S. GIALLORENZO, A. MELIS, M. PRANDINI. *Federated Platooning: Insider Threats and Mitigations*, in "Hawaii International Conference on System Sciences", Grand Wailea, Maui, Hawaii, USA,, United States, January 2019 [*DOI :* 10.24251/HICSS.2019.389], https://hal.inria.fr/hal-02400010

[31] U. DAL LAGO, F. GAVAZZO. *Effectful Normal Form Bisimulation*, in "European Symposium on Programming", Prague, Czech Republic, April 2019, https://hal.inria.fr/hal-02386004

[32] U. DAL LAGO, F. GAVAZZO. *On Bisimilarity in Lambda Calculi with Continuous Probabilistic Choice*, in "Mathematical Foundations of Programming Semantics XXXV", London, United Kingdom, June 2019, https://hal.inria.fr/hal-02386083

[33] U. DAL LAGO, F. GAVAZZO, A. YOSHIMIZU. *Differential Logical Relations Part I: The Simply-Typed Case*, in "46th International Colloquium on Automata, Languages and Programming", Patras, Greece, July 2019 [*DOI :* 10.4230/LIPICS.ICALP.2019.XXX], https://hal.inria.fr/hal-02386110

[34] U. DAL LAGO, N. HOSHINO. *The Geometry of Bayesian Programming*, in "2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)", Vancouver, Canada, IEEE, June 2019, pp. 1-13 [*DOI :* 10.1109/LICS.2019.8785663], https://hal.archives-ouvertes.fr/hal-02399343

[35] M. GABBRIELLI, S. GIALLORENZO, I. LANESE, F. MONTESI, M. PERESSOTTI, S. P. ZINGARO. *No More, No Less - A Formal Model for Serverless Computing*, in "21th International Conference on Coordination Languages and Models (COORDINATION)", Kongens Lyngby, Denmark, H. R. NIELSON, E. TUOSTO (editors), Coordination Models and Languages, Springer International Publishing, 2019, vol. LNCS-11533,

pp. 148-157, Part 3: Exploring New Frontiers [*DOI :* 10.1007/978-3-030-22397-7_9], https://hal.inria.fr/hal-02365509

[36] N. KOBAYASHI, U. DAL LAGO, C. GRELLOIS. *On the Termination Problem for Probabilistic Higher-Order Recursive Programs*, in "2019 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)", Vancouver, France, IEEE, June 2019, pp. 1-14 [*DOI :* 10.1109/LICS.2019.8785679], https://hal.archives-ouvertes.fr/hal-02399361

[37] I. LANESE, A. PALACIOS, G. VIDAL. *Causal-Consistent Replay Debugging for Message Passing Programs*, in "FORTE 2019 - 39th International Conference on Formal Techniques for Distributed Objects, Components, and Systems", Copenhagen, Denmark, J. A. PÉREZ, N. YOSHIDA (editors), Formal Techniques for Distributed Objects, Components, and Systems, Springer International Publishing, 2019, vol. LNCS-11535, pp. 167-184, Part 1: Full Papers [*DOI :* 10.1007/978-3-030-21759-4_10], https://hal.inria.fr/hal-02313745

[38] C. LANEVE, C. S. COEN, A. VESCHETTI. *On the Prediction of Smart Contracts' Behaviours*, in "SG65 -Colloquium in Honour of Stefania Gnesi", Porto, Portugal, M. H. TER BEEK, A. FANTECHI, L. SEMINI (editors), Software Engineering to Formal Methods and Tools, and Back - Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday, October 2019, vol. 11865, pp. 397–415 [*DOI :* 10.1007/978-3-030-30985-5_23], https://hal.inria.fr/hal-02392997

[39] T. LIU, F. CALLEGATI, W. CERRONI, C. CONTOLI, M. GABBRIELLI, S. GIALLORENZO. *Constraint programming for flexible Service Function Chaining deployment*, in "HICS 2019 - 52nd Hawaii International Conference on System Sciences", Maui, United States, Proceedings of the 52nd Hawaii International Conference on System Sciences, January 2019, https://hal.inria.fr/hal-02395208

[40] M. LODI. *Does Studying CS Automatically Foster a Growth Mindset?*, in "ITiCSE '19 Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education", Aberdeen, United Kingdom, ACM Press, July 2019, vol. 7, pp. 147-153 [*DOI :* 10.1145/3304221.3319750], https://hal.inria.fr/hal-02379130

[41] L. MIKULSKI, I. LANESE. *Reversing Unbounded Petri Nets*, in "PETRI NETS 2019", Aachen, Germany, S. DONATELLI, S. HAAR (editors), June 2019 [*DOI :* 10.1007/978-3-030-21571-2_13], https://hal.inria.fr/hal-02376158

[42] D. SANGIORGI. *Asynchronous pi-calculus at Work: The Call-by-Need Strategy*, in "The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy", Paris, France, Lecture Notes in Computer Science, November 2019, vol. 11760, pp. 33-49 [*DOI :* 10.1007/978-3-030-31175-9_3], https://hal.inria.fr/hal-02399695

Scientific Books (or Scientific Book chapters)

[43] M. GABBRIELLI, S. GIALLORENZO, I. LANESE, J. MAURO. *Guess Who's Coming: Runtime Inclusion of Participants in Choreographies*, in "The Art of Modelling Computational Systems: A Journey from Logic and Concurrency to Security and Privacy", 2019 [*DOI :* 10.1007/978-3-030-31175-9_8], https://hal.inria.fr/hal-02376243

[44] M. GABBRIELLI, S. GIALLORENZO, I. LANESE, S. P. ZINGARO. *Linguistic Abstractions for Interoperability of IoT Platforms*, in "Towards Integrated Web, Mobile, and IoT Technology", August 2019, pp. 83-114 [*DOI :* 10.1007/978-3-030-28430-5_5], https://hal.inria.fr/hal-02383918

[45] I. LANESE, D. SANGIORGI, G. ZAVATTARO. *Playing with Bisimulation in Erlang*, in "Models, Languages, and Tools for Concurrent and Distributed Programming", 2019 [*DOI :* 10.1007/978-3-030-21485-2_6], https://hal.inria.fr/hal-02376217

Research Reports

[46] M. AVANZINI, U. DAL LAGO, A. GHYSELEN. *Type-Based Complexity Analysis of Probabilistic Functional Programs (Technical Report)*, Inria Sophia Antipolis ; University of Bologna ; ENS Lyon, April 2019, https://hal.archives-ouvertes.fr/hal-02103943

Scientific Popularization

[47] M. LODI, R. DAVOLI, R. MONTANARI, S. MARTINI. *Informatica senza e con computer nella Scuola Primaria*, in "Coding e oltre: Informatica nella scuola", E. NARDELLI (editor), 2019, forthcoming, https://hal.inria.fr/hal-02379212

Other Publications

[48] D. MEDIC. *Relative expressiveness of calculi for reversible concurrency*, October 2019, The Concurrency Column of EATCS Bulletin, No 129, https://hal.inria.fr/hal-02376279

References in notes

[49] M. CARBONE, K. HONDA, N. YOSHIDA. *A Calculus of Global Interaction based on Session Types*, in "Electr. Notes Theor. Comput. Sci.", 2007, vol. 171, n^{o} 3, pp. 127–151

[50] L. COSIMO, L. PADOVANI. *Deadlock Analysis of Wait-Notify Coordination*, University of Bologna ; University of Torino, 2019, To appear in LNCS 11760, Springer, 2019, https://hal.inria.fr/hal-02166082

[51] V. DANOS, J. KRIVINE. *Reversible Communicating Systems*, in "CONCUR 2004", P. GARDNER, N. YOSHIDA (editors), Lecture Notes in Computer Science, Springer, 2004, vol. 3170, pp. 292–307

[52] A. GARCIA, C. LANEVE, M. LIENHARDT. *Static analysis of cloud elasticity*, in "17th International Symposium on Principles and Practice of Declarative Programming", Siena, Italy, Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming, Moreno Falaschi and Elvira Albert, July 2015, 12 p. [*DOI :* 10.1145/2790449.2790524], https://hal.inria.fr/hal-01229424

[53] A. IGARASHI, N. KOBAYASHI. *Resource usage analysis*, in "POPL conference", ACM Press, 2002, pp. 331–342

[54] N. KOBAYASHI, D. SANGIORGI. *A hybrid type system for lock-freedom of mobile processes*, in "ACM Trans. Program. Lang. Syst.", 2010, vol. 32, n^{o} 5

[55] I. C. C. PHILLIPS, I. ULIDOWSKI. *Reversing algebraic process calculi*, in "J. Log. Algebr. Program.", 2007, vol. 73, n^{o} 1-2, pp. 70–96, https://doi.org/10.1016/j.jlap.2006.11.002