IN PARTNERSHIP WITH:
Université de Bologne (Italie)

Activity Report 2017

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
B9.4.1. - Computer science

1. Personnel

Faculty Members

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

Post-Doctoral Fellows

Charles Grellois [Inria, until Aug 2017]
Saverio Giallorenzo [University Bologna]

PhD Students

Raphaelle Crubillé [Univ Denis Diderot and University Bologna]
Adrien Durier [ENS Lyon and University Bologna]
Francesco Gavazzo [University Bologna]
Liu Tong [University Bologna]
Michael Lodi [University Bologna, from Jun 2017]
Vincenzo Mastandrea [Univ de Nice - Sophia Antipolis]
Marco Solieri [CNRS, until Apr 2017]
Valeria Vignudelli [University Bologna, until Jan 2017]
Stefano Pio Zingaro [University Bologna]

Administrative Assistants

Christine Claux [Inria]
Christine Riehl [Inria]

External Collaborators

Martin Avanzini [University of Innsbruck]
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. 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.

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 of 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: for example, 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

- Fabrizio Montesi, external collaborator in Focus, has been awarded the "Innovation Award 2017" from his university (University of Southern Denmark), for his work and contributions in the language Jolie.

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 is a service-oriented programming language. Jolie can be used to program services that interact over the Internet using different communication protocols.

Differently from other Web Services programming languages such as WS-BPEL, Jolie is based on a user-friendly C/Java-like syntax (more readable than the verbose XML syntax of WS-BPEL) and, moreover, the language is equipped with a formal operational semantics. This language is used for the *proof of concepts* developed around Focus activities. For instance, contract theories can be exploited for checking the conformance of a Jolie program with respect to a given contract.
FUNCTIONAL DESCRIPTION: Developments in 2017: 2017 has seen many efforts around the language to increase its usage in industry. These include:

- Organisation of two events. One in Italy, called Meeting on Microservices, organised by italianaSoftware and Monrif SpA in December 2016. The second one in Denmark, organised by Southern Denmark University and Università di Bologna in October 2017. Common aim of both events was presenting the language from a practical, industrial point of view, to illustrate with real-world cases how its abstractions can increase productivity of companies. Both venues contributed in growing the community of companies that have adopted the language or plan to adopt it in the near future.

- Revision of the language documentation, migrating it to GitBook. In this way, Jolie users can access its documentation as HTML pages, as a PDF, and as an eBook. The choice of GitBook has been guided by the need to give a proper tool to users to collaborate, discuss, and request fixes and extensions on the documentation.

- Development of several tools, frameworks, and libraries to ease the management of architectures of microservices. The main ones are:

- the publication of libraries to interact with and orchestrate the Docker containerisation technology. This work, called Jocker, has been the fulcrum of other projects that streamline the creation and management of container-based microservice architectures,

- the publication of a fundamental companion for any industrial-grade language: a packing system. The project, called jpm, automatises the process of publishing, installing, upgrading, configuring, and removing libraries in Jolie software projects,

- the inclusion in the language interpreter of hooks for modular, distributed tracing, a renowned problem of microservices and distributed systems. Developed to output program traces in JSON, this work maintained an open perspective on both output formats and logging deployment, which can be extended in a modular way. The project also includes a visualiser of several distributed traces for debugging purposes,

- the publication of a unit testing framework for microservices, a fundamental building block for continuous integration processes. This framework includes also functionalities to automatically test microservices within a distributed, sandboxed environment, thanks to its integration with Jocker. The framework is also the first step towards a more comprehensive suite to test complete microservice architectures,

- the creation of a deployment framework that automatises the deployment of microservice architectures. This is an important issue in microservice and distributed system deployment, where correctly installing programs on execution nodes and making sure they are properly linked to each other is a daunting and time-consuming task. The framework, given a deployment schema, i) automatises the creation of containers where one or more microservices coexist, ii) deploys the containers into assigned machines, and iii) binds the deployed containers so that microservices within different containers can communicate,

- the creation of the Jiot project, aimed at integrating IoT-related technologies into the Jolie language. The final goal is to provide easy-to-use and flexible communication abstractions to interconnect and make interact disparate IoT islands. Work in 2017 comprised the inclusion of the CoAP/UDP and MQTT/TCP protocols among the communication technologies supported by the language.

Jolie also transitioned from version 1.6 to 1.6.2, which are minor releases, however they contain many performance optimisations and bug fixes.

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.

- Participants: Claudio Guidi, Fabrizio Montesi, Maurizio Gabbrielli and Saverio Giallorenzo
- 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

SCIENTIFIC 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 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 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.
FUNCTIONAL DESCRIPTION: AIOCJ is an open-source choreography programming language for developing adaptive systems.

- 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 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. This 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.

- 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

7. New Results

7.1. Service-Oriented Computing

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

7.1.1. Microservices

Microservices represent an architectural style inspired by service-oriented computing that has recently started gaining popularity. As we have discussed in [37], one of the main advantages of the microservices approach is that it improves scalability of the developed applications. In [43] we have analyzed the impact of microservices on the overall line of research on software architectures, by pointing out specific open problems and future challenges. One of the challenges is concerned with programming languages because microservice systems are currently developed using general-purpose programming languages that do not provide dedicated abstractions for service composition. In [46] we have discussed the limitations of the current practices and we have proposed a novel language-based approach to the engineering of microservices based on the Jolie programming language.

7.1.2. Orchestrations and choreographies

The practice of programming distributed systems is extremely error-prone, due to the complexity in correctly implementing separate components that, put together, enact an agreed protocol. Theoretical and applied research is, therefore, fundamental, to explore new tools to assist the development of such systems. In particular, usage of so-called session types in orchestration languages guarantees correct communication by means of corresponding type system theories. In this context, we carried out studies about: foundations of the classical theory of session types, by providing an encoding into pi-calculus typing [17]; and subtyping in the context of asynchronous communication, showing it to be an undecidable problem [15]. Choreographies are also an important specification tool in that they can be compiled to obtain projected orchestrations that enjoy deadlock freedom by construction. Moreover they allow one to express dynamic behaviours at the level of the whole system, which then reflect on each involved orchestration. In this context, in [16] we studied the theory and implementation of dynamic choreographies and in [45] we showed how to use them for programming microservice-based applications. Finally, we considered applications in the context of Mobility-as-a-Service (MaaS) scenarios, where solutions of different transportation providers are dynamically composed into a single, consistent interface. We devised the prototype of an enabling software platform for MaaS [27] and we studied MaaS security issues [28].

7.2. Models for Reliability

Participant: Ivan Lanese.

7.2.1. Reversibility

We have continued the study of reversibility started in the past years. In particular, in [19], we thoroughly studied causal-consistent reversibility in the coordination language μKlaim [50], a distributed version of Linda [49]. More specifically, we gave an abstract specification of a causal-consistent rollback operator and showed that our semantics satisfies it. The main novelty of μKlaim w.r.t. process calculi studied in past work is that it includes a primitive to read a datum without consuming it, that, from the causality point of view, creates asymmetric dependencies. The same technique could be used to reverse languages with shared memory.

In [24] we studied how to exploit reversibility to improve client-server interactions. In particular, we defined retractable contracts, namely contracts including the possibility of undoing past agreements, which are more expressive than standard session contracts for binary interactions [48], yet preserve their nice properties: compliance and the subcontract relation are both decidable in polynomial time, the dual of a contract always exists and has a simple syntactic characterization. Furthermore we showed that the same contracts can also describe speculative interactions.

7.3. Probabilistic Systems and Resource Control

Participants: Martin Avanzini, Raphaelle Crubillé, Ugo Dal Lago, Francesco Gavazzo, Charles Grellois, Davide Sangiorgi, Valeria Vignudelli.

7.3.1. Probabilistic termination

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. One of the most basic (but nevertheless desirable) properties of programs is certainly termination. When probabilistic choice comes into play, termination can be defined in more than one way. As an example, one can stipulate that a probabilistic program terminates if and only if its probability of convergence is 1, this way being *almost surely* terminating. Alternatively, a probabilistic program can be 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 presence of probabilistic choice. Actually, it becomes provably harder, being strictly higher in the arithmetical hierarchy. Probabilistic termination has received quite some attention in recent years, but most contributions are concerned either with its abstract nature, or with verification methodologies for imperative programs. Along 2017, we have initiated the study of probabilistic termination in probabilistic higher-order functional languages. Our contribution in this direction is twofold. On the one hand, we have analysed the impact of endowing a strongly normalising typed lambda calculus, namely Godel's **T**, with various forms of probabilistic choice operators [26]. Unsurprisingly, the obtained systems are all almost surely terminating, but interestingly, only *some* of them are positively so. In particular, binary probabilistic choice and the geometric distribution can have dramatically different effects. Another line of work has to do with types, and in particular with sized types, which we have generalised to a higher-order functional language with higher order recursion and binary probabilistic programs. We showed how the obtained system is sound for almost sure termination [35], but also that it captures interesting examples like various forms of random walks.

7.3.2. Automating complexity analysis of higher-order functional programs

Complexity analysis of higher-order functional programs has been one of the core research directions within Focus since its inception. Progressively, however, our interest has shifted from foundations to automation. The latter is indeed the main research direction we have pursued in 2017. More specifically, we have been trying to overcome the main shortcoming of our software tool HoCA, namely the fact that most analysis techniques it implements are not modular, and are thus bound not to scale. We have looked at sized type systems as a way to do complexity analysis of functional programs by performing type inference on a so-called ticking-transformed version of them [13]. The obtained design methodology has been proved to allow the analysis of programs which could not be handled by HoCA.

7.3.3. Relational reasoning about effectful and concurrent programs

Building on our knowledge on semantic and coinductive techniques for reasoning about higher-order programs, we have studied how to reason relationally when the programs at hand exhibit some form of effect including probabilistic choice, but also algebraic effects. We have first of all concluded our investigation about metric reasoning about terms in a probabilistic lambda calculus. We discovered that in the general case of a fully-fledged probabilistic lambda calculus, any reasonable metric is bound to trivialise to an equivalence [31]. This negative result convinced us that a richer and more refined notion of comparison is needed, on which we are currently investigating. We also looked at how Abramsky's applicative bisimilarity can be generalised to a language with algebraic effects. Since the notion of algebraic effect is abstract, this is best done by injecting concepts from category theory, and in particular those of a monad and of a relator, into the playground. Mild conditions on the latter allow one to generalise the classic proof of congruence for applicative bisimilarity, due to Howe [33], [34]. This way, conductive proof techniques for equivalence can be shown sound with respect to context equivalence for various forms of algebraic effects including probabilistic choice, global state, exceptions, and combinations. One last line of work we have pursued in 2017 has to do with geometry of interaction, a dynamic semantic framework which is known to faithfully model higher-order computation. We have this year managed to show that multitoken machines, a generalisation of geometry of interaction we introduced three years ago, can faithfully model quantum lambda calculi [32], but also process algebras like the π-calculus, through multiport interaction combinators [14].

7.4. Verification Techniques

Participants: Mario Bravetti, Adrien Durier, Daniel Hirschkoff, Ivan Lanese, Cosimo Laneve, Davide Sangiorgi.

We analyze sensible properties of concurrent systems, including deadlock freedom and resource usages, and proof techniques for deriving behavioural equalities and preorders on processes.

7.4.1. Deadlock detection and cloud elasticity

In order to verify sensible properties of concurrent programs we use a technique consisting of (1) extracting information by means of behavioural type systems and (2) analyzing types by means of ad-hoc tools.

In [20] we study deadlock detection for value-passing CCS (and for π calculus). In this paper we analyze complex programs that create networks with arbitrary numbers of nodes. To enable the analysis of such programs, (1) we define an algorithm for detecting deadlocks of a basic model featuring recursion and fresh name generation, and (2) we design a type system that returns behavioural types. We show the soundness of the type system, and develop a type inference algorithm for it.

In [39] we apply the above technique to a language for stateful active objects. This is challenging because active objects use futures to refer to results of pending asynchronous invocations and because these futures can be stored in object fields, passed as method parameters, or returned by invocations. The type system traces the access to object fields by means of effects. For this reason, it is possible to compute behavioural types that express synchronisation patterns in a precise way. The behavioural types are thereafter analysed by a solver that discovers potential deadlocks. The PhD thesis of Vincenzo Mastandrea [11] addresses deadlock detection of stateful active objects.

In [44] we apply the same technique to Java byte-code. In particular [44] gives a practical presentation of JaDA, a static deadlock analyzer for Java that extracts behavioral types and analyzes these types by means of a fixpoint algorithm that reports potential deadlocks in the original Java code. We also present some of the features for customising the analysis: while the main strength of JaDA is to run in a fully automatic way, user interaction is possible and may enhance the accuracy of the results. The whole theory behind JaDa is fully developed in the PhD thesis of Abel Garcia Celestrin [10].

In [18] we address a concurrent language with explicit acquire and release operations on virtual machines. In our language it is possible to delegate other (ad-hoc or third party) concurrent code to release virtual machines (by passing them as arguments of invocations). In this case, we define (i) a type system associating programs with behavioural types that record relevant information for resource usage (creations, releases, and concurrent operations), (ii) a translation function that takes behavioural types and returns cost equations, and (iii) an automatic off-the-shelf solver for the cost equations. A soundness proof of the type system establishes the correctness of our technique with respect to the cost equations. We have experimentally evaluated our technique using a cost analysis solver. The experiments show that our analysis allows us to derive bounds for programs that are better than other techniques, such as those based on amortized analysis.

7.4.2. Most general property-preserving updates

Systems need to be updated to last for a long time in a dynamic environment, and to cope with changing requirements. It is important for updates to preserve the desirable properties of the system under update, while possibly enforcing new ones. We consider a simple yet general update mechanism [25] that replaces a component of the system with a new one. The context, i.e., the rest of the system, remains unchanged. We define contexts and components as Constraint Automata interacting via either asynchronous or synchronous communication, and we express properties using Constraint Automata too. Then we build most general updates which preserve specific properties, considering both a single property and all the properties satisfied by the original system, in a given context or in all possible contexts.

7.4.3. Proof techniques based on unique solutions

In [22], we study bisimilarity, a behavioural equivalence whose success is much due to the associated bisimulation proof method. In particular, we discuss a different proof method, based on unique solution of special forms of inequations called contractions, and inspired by Milner's theorem on unique solution of equations. The method is as powerful as the bisimulation proof method and its up-to context enhancements. The definition of contraction can be transferred onto other behavioural equivalences, possibly contextual and non-coinductive. This enables a coinductive reasoning style on such equivalences, either by applying the method based on unique solution of contractions, or by injecting appropriate contraction preorders into the bisimulation game.

In [38] we develop the above proof method in a different direction: rather than introducing contractions, we remain within equations, and we investigate conditions that guarantee unique solutions, for bisimilarity as well as for other behavioural equivalences such as trace equivalence. We also consider preorders such as trace inclusion. We finally develop abstract formulations of the theorems, on generic Labeled Transition Systems.

7.4.4. Fuzzy logics

In [14] we introduce a framework for detecting anomalies in the clocks of the different components of a network of sensor stations connected with a central server for measuring air quality. We propose a novel approach, supported by a formal representation of the network using fuzzy-timed automata, to precisely represent the expected behaviour of each component of the network. Using fuzzy logic concepts, we can specify admissible mismatches between the clocks.

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 particular in the context of K-12 education (students aged approximately from 5 to 18). We study philosophical, sociological and historical motivations to teach computer science at all school levels. Furthermore, we study what concepts and skills related to computer science are not barely 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 (particularly gender related issues); teacher training (particularly non specialist teachers).

7.5.1. Computational thinking definition

There is no accepted definition of computational thinking. From one hand we tried to find out the main common elements in the most important proposed definitions, and investigate, in a large sample of K-12 teachers, if they have a correct idea [30] about CT. We found the vast majority of them held misconceptions or partial views about it. We argued these may be consequences of a massive use of the term in school context [23]; we made clear "computational thinking" is not a new subject, but just a name to indicate computer science principles that should be taught to all students [21].

7.5.2. Evaluation of popularization initiatives

We analyzed [29] the sentiment of a large sample of teachers participating in the national project "Programma il Futuro" (Program the Future) - an italian version of Code.org with support materials. The sentiment was largely positive. Among other results, we note reported interest is equally distributed between male and female students in primary school, and shifts towards a higher male interest only from secondary school, suggesting a social influence.

7.5.3. Growth mindset and teacher training

Every person holds an idea (mindset) about intelligence: someone thinks it is a fixed trait, like eye color (fixed mindset), while others think 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. Teachers' mindset directly affects students' one. We propose [42] a line of research to investigate mindset of pre-service primary school teachers before and after a "creative computing course", to analyze and, in perspective, to change their specific "computer science mindset".

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. This year, besides refining the work on SUNNY (described elsewhere, see also [40]) we have developed a new tool, NightSplitter, a scheduling tool to optimize (sub)group activities [41]. Humans are social animals and usually organize activities in groups. However, they are often willing to split temporarily a bigger group in subgroups to enhance their preferences. NightSplitter is an on-line tool that is able to plan movie and dinner activities for a group of users, possibly splitting them in subgroups to optimally satisfy their preferences. We first have modeled and proved that this problem is NP-complete. We have then used Constraint Programming (CP) or alternatively Simulated Annealing (SA) to solve it. Empirical results show the feasibility of the approach even for big cities where hundreds of users can select among hundreds of movies and thousands of restaurants. (More information on NightSplitter is found in the section on tools.)

8. Partnerships and Cooperations

8.1. National Initiatives

- ELICA (Expanding Logical Ideas for Complexity Analysis) is an ANR project that started on October 2014 and that will finish on September 2018. ELICA focuses on methodologies for the static analysis of programs and their resource consumption. The project's aim is to further improve on logical methodologies for complexity analysis (type systems, rewriting, etc.). More specifically, one would like to have more powerful techniques with less false negatives, being able at the same time to deal with nonstandard programming paradigms (concurrent, probabilistic, etc.). Main persons involved: Avanzini, Dal Lago, Hirschkoff, Martini, Sangiorgi.
- 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 will finish 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.

8.2. European Initiatives

8.2.1. Collaborations in European Programs, Except FP7 & H2020

- 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).
- ICT COST Action IC1402 ARVI (Runtime Verification beyond Monitoring). Initiated in December 2014 and with a 4-year duration, this COST Action studies runtime verification, a computing analysis paradigm based on observing a system at runtime to check its expected behaviour.

  Main persons involved: Bravetti, Lanese.

8.2.2. 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, Vignudelli. Some visit exchanges during the year, in both directions. A joint PhD started in September 2016 (Adrien Durier).
- Inria EPI Spades (on models and languages for components, reversibility). Contact person(s) in Focus: Lanese.
- 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.
- LMU Munich (M. Hofmann) (on implicit computational complexity and IntML). Contact person(s) in Focus: Dal Lago.
- IMDEA Software, Madrid (G. Barthe) (on implicit computational complexity for cryptography). Contact person(s) in Focus: Dal Lago,Sangiorgi. Some visits during the year.
- Facultad de Informatica, Universidad Complutense de Madrid (on web services). Contact person(s) in Focus: Bravetti. Bravetti is an external collaborator in the project "Desarrollo y Análisis formal de sistemas complejos en contextos DistribuidOS: fundamentos, herramientas y aplicaciones (DArDOS)" (Development and formal analysis of complex systems in distributed contexts: foundations, tools and applications) January 2016 - December 2018, funded by the Spanish Ministerio de Economia y Competitividad.

8.3. International Initiatives

8.3.1. Inria Associate Teams Not Involved in an Inria International Labs

8.3.1.1. CRECOGI

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

International Partner (Institution - Laboratory - Researcher):

Tokyo (Japan) - Department of Computer Science, Graduate School of Information Science and Technology - Ichiro HASUO

Start year: 2015

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

Game semantics and geometry of interaction (GoI) are two closely related frameworks whose strength is to have the characters of both a denotational and an operational semantics. They offer a high-level, mathematical (denotational) interpretation, but are interactive in nature. The formalization in terms of movements of tokens through which programs communicate with each other can actually be seen as a low-level program. The current limit of GoI is that the vast majority of the literature and of the software tools designed around it have a pure, sequential functional language as their source language. This project aims at investigating the application of GoI to concurrent, resourceful, and effectful computation, thus paving the way to the deployment of GoI-based correct-by-construction compilers in real-world software developments in fields like (massively parallel) high-performance computing, embedded and cyberphysical systems, and big data. The presence of both the japanese GoI community (whose skills are centered around effects and coalgebras) and the french GoI community (more focused on linear logic and complexity analysis) will bring essential, complementary, ingredients.

8.3.2. 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.

8.4. International Research Visitors

8.4.1. 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.

- German Vidal and Adrián Palacios: “A Reversible Semantics for Erlang." (2 visits, during the year)
- Matteo Acclavio: "Proof Diagrams for Multiplicative Linear Logic: Syntax and Semantics."
- Ken Sakayori: "A Truly Concurrent Game Model of the Asynchronous pi-Calculus."
- Marco Carbone: "Multiparty Session types and Linear Logic."
- Beniamino Accattoli: "The Complexity of Abstract Machines."
- Ulrich Schoepp, on Complexity analysis of probabilistic programs.

8.4.2. Visits to International Teams

U. Dal Lago has spent two weeks in Japan (University of Kyoto and University of Tokyo). Topics: geometry of interaction for continuous probabilistic programming languages, and categorical models for multitoken machines.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

I. Lanese: Scientific coordinator of the International Training School on Reversible Computation (28-31/8/2017, Torun, Poland), organized by the COST Action IC1405 on Reversible Computation - Extending Horizons of Computing

9.1.1.2. Member of the Organizing Committees (conferences and schools)

S. Giallorenzo: Publicity Chair for 1st Conference on Microservices 2017 (Microservices 2017)

I. Lanese: Publicity Chair for 12th International Federated Conference on Distributed Computing Techniques (DisCoTec 2017)

9.1.2. Scientific Events Selection

9.1.2.1. Member of the Conference Program Committees

M. Bravetti: 15th International Conference on Software Engineering and Formal Methods (SEFM 2017); 2017 IEEE International Conference on Big Data (IEEE BigData 2017); 29th IFIP International Conference on Testing Software and Systems (ICTSS 2017).

U. Dal Lago: 26th Annual Conference of the European Association for Computer Science Logic (CSL 2017); 2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017).

S. Giallorenzo: 32nd Annual ACM/SIGAPP Symposium On Applied Computing (SAC 2017)

I. Lanese: 14th International Conference on Formal Aspects of Component Software (FACS 2017); 9th Conference on Reversible Computation (RC 2017); 10th Interaction and Concurrency Experience (ICE 2017).

S. Martini: 14th Annual Conference on Theory and Applications of Models of Computation (TAMC 2017); Fourth International Conference on History and Philosophy of Computing (HAPOC 2017); DIDAMATICA - Informatica per la Didattica (DIDAMATICA 2017).

D. Sangiorgi: 11th edition A.P. Ershov Informatics Conference (PSI Conference Series); 11th Int. Conference on Language and Automata Theory and Applications (LATA); 28th International Conference on Concurrency Theory (CONCUR 2017); 18th Italian Conference on Theoretical Computer Science (ICTCS 2017); 45th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL).

G. Zavattaro: 5th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems (FOCLASA 2017), Trento, Italy, 6–10 September, 2017; 6th European Conference on Service-oriented and Cloud Computing (ESOCC'17), Oslo, Norway, 27–29 September, 2017; 28th International Conference on Concurrency Theory (CONCUR'17), Berlin, Germany, 5–8 September 2017.

9.1.2.2. Member of the Editorial Boards

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

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.

9.1.3. Invited Talks

U. Dal Lago: lecturer at the International School "A Brief Introduction to Probabilistic and Quantum Programming", Universidade do Minho.

I. Lanese: lecturer at the International Training School on Reversible Computation (28-31/8/2017, Torun, Poland), organized by the COST Action IC1405 on Reversible Computation - Extending Horizons of Computing

D. Sangiorgi: invited talk at the event "Celebrating 20 years of IRAFS (The International Research Area on Foundations of the Sciences)", Pontifical Lateran University, Rome.

G. Zavattaro: Invited talk at '15th International Workshop on Foundations of Coordination Languages and Self-Adaptative Systems (FOCLASA'17)".

9.1.4. 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 Executive Board of EQANIE (European Quality Assurance Network for Informatics Education); he has been nominated vice-president, starting 1 January 2018.

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.

S. Martini is a member of the Board of CINI (Italian National Interuniversity Consortium for Informatics), designated by the Ministry for Semplificazione e Pubblica Amministrazione, from 2015.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Mario Bravetti

  Master: "Linguaggi, Compilatori e Modelli Computazionali", 120 hours, 1st year, University of Bologna, Italy.

- Ugo Dal Lago

  Undergraduate: "Introduction to Programming in Python", 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", 36 Hours, 2nd 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.

- Saverio Giallorenzo

  Undergraduate: "Laboratorio di Operating Systems", 40 hours, 2nd year, University of Bologna, Italy.

- Ivan Lanese

  Undergraduate: "Architettura degli Elaboratori", 56 hours, 1st year, University of Bologna, Italy.

Master: “Ingegneria del Software Orientata ai Servizi”, 22 hours, 2nd 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

  Undergraduate: “Introduction to programming in Python”, 78 hours, 1st year, University of Bologna, Italy.

  Undergraduate: “Programming in Python”, 72 hours, 1st year, University of Bologna, Italy.

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

- Davide Sangiorgi

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

- Gianluigi Zavattaro

  Undergraduate: “Computer Architectures”, 60 hours, 1st year, University of Bologna, Italy

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

9.2.2. Supervision

PhD thesis completed in 2016:

- Abel Garcia Celestrin, “Analysis of Cloud Computing Systems”. Supervisor C. Laneve.
- Vincenzo Mastandrea, “Deadlock analysis in ASP”. Supervisors: Cosimo Laneve and Ludovic Henrio (CNRS Sophia Antipolis).
- Valeria Vignudelli, “Behavioral Equivalences for Higher-Order Languages with Probabilities”. Supervisor D. Sangiorgi.

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

- Raphaelle Crubillé, October 2015, “Bisimulation Metrics and Probabilistic Lambda Calculi”, Université Denis Diderot and University of Bologna. Supervisors Thomas Ehrhard and 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.
- Francesco Gavazzo, October 2015, “Coinductive Techniques for Effectful Lambda Calculi”. Supervisors U. Dal Lago and D. Sangiorgi.
- Michael Lodi, January 2017, “Growth Mindset and Computational Thinking”. Supervisor: S. Martini.
- Tong Liu, November 2015, “Constraint based languages for Software Defined Networks”. Supervisor: Maurizio Gabbrielli.
- Stefano Pio Zingaro, November 2016, “High level languages for Internet of Things applications”. Supervisors: Maurizio Gabbrielli and Ivan Lanese.

9.2.3. Juries

M. Bravetti has been member of the PhD jury of Federica Panarotto, University of Verona, Italy, May 2017.
U. Dal Lago has been member of the PhD jury of Pierre Vial, Université Denis-Diderot, December 2017.
D. Hirschkoff has been member of the PhD evaluation committee for Yannick Zakowski, ENS Rennes, December 2017.

9.3. 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;
- S. Martini has given various talks at institutes and workshops on the teaching methods for Computer Science, including a talk on "Educare all'informatica nella scuola?", working day on "La cultura informatica come fattore di sviluppo" Roma, Camera dei Deputati, December 2017.
- S. Martini is coordinator of some initiatives for the 'Hour of Code' (see e.g., http://www.programmailfuturo.it). M. Lodi is also involved.
- M. Lodi is in charge of preparing material "unplugged" to teach the principles of computational thinking in elementary schools.
- M. Lodi has been involved in meetings aimed at forming teachers of elementary and high schools in the Bologna province. These include a 15-hour course "Coding e robotica: tecnologie e metodologie per una didattica integrata nella scuola secondaria di I grado", and a 12-hour course "Scratch avanzato: funzionalità e applicazione alla didattica", both organised by Fondazione Golinelli (Bologna).
- M. Lodi has given talks on computational thinking and programming languages for teaching computer science to children, including: "Coding - Che cosa è e perché?" and "Coding con Scratch", at Mathesis Piacenza; "CoderDojo: Pensiero computazionale e informatica creativa... anche a scuola?", Convention Diesse; introduction to computer science for high school students, as part of an event of introduction to management ("Giardino delle imprese"), Fondazione Golinelli, Bologna; "L'illusione del coding" and "TeacherDojo", at the contest "CoderDojo Coolest Projects Milano", Milan; "Ragazze, siate coraggiose: la vostra intelligenza puo' crescere (ma il "coding" non basta!)", at "Festival della Cultura Tecnica 2017", Bologna; a talk and youtube video "Quattro domande sul pensiero computazionale" (https://www.youtube.com/watch?v=ANlwOckhcVU).
- M. Lodi has has been member of the evaluation commission for the awards "Programma le Regole", set by Italian Ministry of Education and Research (MIUR) and "Programma il Futuro".

D. Hirschkoff takes part in several popularization activities in schools, in Lyon (association "Maths en Jeans").

9.3.1. Other duties

S. Martini is Head of the Department of Computer Science and Engineering, University of Bologna, for the term 2015-2018.

G. Zavattaro is coordinator of undergraduate studies at the Department of Computer Science and Engineering, University of Bologna (Informatica per il Management).

10. 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° 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] A. G. CELESTRÍN. *Static analysis of concurrent programs based on behavioral type systems*, University of Bologna, May 2017, https://hal.inria.fr/tel-01660749

[11] V. MASTANDREA. *Analysis of synchronisation patterns in active object based on behavioural types*, UCA, I3S ; UCA, Inria, December 2017, https://hal.archives-ouvertes.fr/tel-01651649

[12] V. VIGNUDELLI. *Behavioral Equivalences for Higher-Order Languages with Probabilities*, University of Bologna, May 2017, https://hal.inria.fr/tel-01644462

Articles in International Peer-Reviewed Journals

[13] M. AVANZINI, U. DAL LAGO. *Automating sized-type inference for complexity analysis*, in "Proceedings of the ACM on Programming Languages", August 2017, vol. 1, n° ICFP, pp. 1 - 29 [*DOI :* 10.1145/3110287], https://hal.inria.fr/hal-01639200

[14] J. BOUBETA-PUIG, M. BRAVETTI, L. LLANA, M. MERAYO. *Analysis of temporal complex events in sensor networks*, in "Journal of Information and Telecommunication", July 2017, vol. 1, n° 3, pp. 273-289 [*DOI :* 10.1080/24751839.2017.1347763], https://hal.inria.fr/hal-01637941

[15] M. BRAVETTI, M. CARBONE, G. ZAVATTARO. *Undecidability of asynchronous session subtyping*, in "Information and Computation", October 2017, vol. 256, pp. 300 - 320 [*DOI :* 10.1016/J.IC.2017.07.010], https://hal.inria.fr/hal-01637935

[16] M. DALLA PREDA, M. GABBRIELLI, S. GIALLORENZO, I. LANESE, J. MAURO. *Dynamic Choreographies: Theory And Implementation*, in "Logical Methods in Computer Science", May 2017, vol. 13, pp. 1 - 57 [*DOI :* 10.23638/LMCS-13(2:1)2017], https://hal.inria.fr/hal-01631394

[17] O. DARDHA, E. GIACHINO, D. SANGIORGI. *Session Types Revisited*, in "Information and Computation", October 2017, vol. 256, pp. 253 - 286 [*DOI :* 10.1016/J.IC.2017.06.002], https://hal.inria.fr/hal-01647086

[18] A. GARCIA, C. LANEVE, M. LIENHARDT. *Static analysis of cloud elasticity*, in "Science of Computer Programming", November 2017, vol. 147, pp. 27 - 53 [*DOI :* 10.1016/J.SCICO.2017.03.008], https://hal.inria.fr/hal-01643175

[19] E. GIACHINO, I. LANESE, C. A. MEZZINA, F. TIEZZI. *Causal-consistent rollback in a tuple-based language*, in "Journal of Logical and Algebraic Methods in Programming", April 2017, vol. 88, pp. 99 - 120 [*DOI :* 10.1016/J.JLAMP.2016.09.003], https://hal.inria.fr/hal-01633260

[20] N. KOBAYASHI, C. LANEVE. *Deadlock analysis of unbounded process networks*, in "Information and Computation", February 2017, vol. 252, pp. 48 - 70 [*DOI :* 10.1016/J.IC.2016.03.004], https://hal.inria.fr/hal-01643152

[21] S. MARTINI. *Computational thinking: a fourth competence after writing, reading, and accounting*, in "IL NODO, SCUOLA IN RETE", December 2017, n^{o} 47, pp. 18-28, https://hal.inria.fr/hal-01643699

[22] D. SANGIORGI. *Equations, Contractions, and Unique Solutions*, in "ACM Transactions on Computational Logic", April 2017, vol. 18, n^{o} 1, pp. 1-36 [*DOI :* 10.1145/2971339], https://hal.inria.fr/hal-01647063

Articles in National Peer-Reviewed Journals

[23] M. LODI, S. MARTINI, E. NARDELLI. *Do we really need computational thinking?* , in "Mondo Digitale", November 2017, n^{o} 72, pp. 1-15, https://hal.inria.fr/hal-01656340

International Conferences with Proceedings

[24] F. BARBANERA, I. LANESE, U. DE 'LIGUORO. *Retractable and Speculative Contracts*, in "19th International Conference on Coordination Languages and Models (COORDINATION)", Neuchâtel, Switzerland, J.-M. JACQUET, M. MASSINK (editors), Coordination Models and Languages, Springer, June 2017, vol. 10319, pp. 119-137, Part 3: Types [*DOI :* 10.1007/978-3-319-59746-1_7], https://hal.inria.fr/hal-01633262

[25] D. BRESOLIN, I. LANESE. *Most General Property-Preserving Updates*, in "LATA 2017 - Language and Automata Theory and Applications", Umea, Sweden, March 2017, https://hal.inria.fr/hal-01635801

[26] F. BREUVART, U. DAL LAGO, A. HERROU. *On Higher-Order Probabilistic Subrecursion*, in "FoSSaCS 2017 - 20th International Conference on Foundations of Software Science and Computation Structures", Uppsala, Sweden, LNCS, April 2017, vol. 10203, pp. 370-386 [*DOI :* 10.1007/978-3-662-54458-7_22], https://hal.inria.fr/hal-01639379

[27] F. CALLEGATI, M. GABBRIELLI, S. GIALLORENZO, A. MELIS, M. PRANDINI. *Smart Mobility for All: A Global Federated Market for Mobility-as-a-Service Operators*, in "ITSC2017- 20th International Conference on Intelligent Transportation", Yokohama, Japan, October 2017, https://hal.inria.fr/hal-01631427

[28] F. CALLEGATI, S. GIALLORENZO, A. MELIS, M. PRANDINI. *Insider Threats in Emerging Mobility-as-a-Service Scenarios*, in "HICSS 2017 - 50th annual Hawaii International Conference on System Science", Hilton Waikoloa Village, United States, January 2017, https://hal.inria.fr/hal-01631388

[29] I. CORRADINI, M. LODI, E. NARDELLI. *Computational Thinking in Italian Schools: Quantitative Data and Teachers' Sentiment Analysis after Two Years of "Programma il Futuro" Project*, in "ITiCSE '17 - Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education", Bologna, Italy, July 2017 [*DOI :* 10.1145/3059009.3059040], https://hal.inria.fr/hal-01636232

[30] I. CORRADINI, M. LODI, E. NARDELLI. *Conceptions and Misconceptions about Computational Thinking among Italian Primary School Teachers*, in "ICER '17 - Proceedings of the 2017 ACM Conference on International Computing Education Research", Tacoma (WA), United States, August 2017 [*DOI :* 10.1145/3105726.3106194], https://hal.inria.fr/hal-01636235

[31] R. CRUBILLÉ, U. DAL LAGO. *Metric Reasoning About λ-Terms: The General Case*, in "ESOP 2017 - 26th European Symposium on Programming", Uppsala, Sweden, H. YANG (editor), LNCS - Lecture Notes in Computer Science, Springer, April 2017, vol. 10201, pp. 341-367 [*DOI :* 10.1007/978-3-662-54434-1_13], https://hal.inria.fr/hal-01639369

[32] U. DAL LAGO, C. FAGGIAN, B. VALIRON, A. YOSHIMIZU. *The Geometry of Parallelism: Classical, Probabilistic, and Quantum Effects*, in "POPL 2017 - Principles of Programming Languages", Paris, France, Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, January 2017, pp. 833-845, https://arxiv.org/abs/1610.09629 [*DOI :* 10.1145/3009837.3009859], https://hal.archives-ouvertes.fr/hal-01474620

[33] U. DAL LAGO, F. GAVAZZO, P. BLAIN LEVY. *Effectful Applicative Bisimilarity: Monads, Relators, and Howe's Method*, in "LICS 2017 - ACM/IEEE Symposium on Logic in Computer Science", Reykjavik, Iceland, IEEE, June 2017, pp. 1-12 [*DOI :* 10.1109/LICS.2017.8005117], https://hal.inria.fr/hal-01636365

[34] U. DAL LAGO, F. GAVAZZO, R. TANAKA. *Effectful applicative similarity for call-by-name lambda calculi*, in "ICTCS 2017 - 18th Italian Conference on Theoretical Computer Science", Naples, Italy, September 2017, https://hal.inria.fr/hal-01636368

[35] U. DAL LAGO, C. GRELLOIS. *Probabilistic Termination by Monadic Affine Sized Typing*, in "ESOP 2017 - 26th European Symposium on Programming", Uppsala, Sweden, ESOP 2017, April 2017, https://hal.archives-ouvertes.fr/hal-01635077

[36] U. DAL LAGO, R. TANAKA, A. YOSHIMIZU. *The Geometry of Concurrent Interaction: Handling Multiple Ports by Way of Multiple Tokens*, in "LICS 2017 - Thirty-Second Annual ACM/IEEE Symposium on Logic in Computer Science", Reykjavik, Iceland, June 2017, https://hal.inria.fr/hal-01639411

[37] N. DRAGONI, I. LANESE, S. T. LARSEN, M. MAZZARA, R. MUSTAFIN, L. SAFINA. *Microservices: How To Make Your Application Scale*, in "A.P. Ershov Informatics Conference (the PSI Conference Series, 11th edition)", Moscow, Russia, June 2017, https://hal.inria.fr/hal-01636132

[38] A. DURIER, D. HIRSCHKOFF, D. SANGIORGI. *Divergence and unique solution of equations*, in "CONCUR 2017 - 28th International Conference on Concurrency Theory", Berlin, Germany, September 2017, vol. 7, pp. 1 - 7 [*DOI :* 10.4230/LIPIcs.CONCUR.2017.7], https://hal.archives-ouvertes.fr/hal-01643502

[39] L. HENRIO, C. LANEVE, V. MASTANDREA. *Analysis of Synchronisations in Stateful Active Objects*, in "IFM 2017 - 13th International Conference on Integrated Formal Methods", Torino, France, September 2017 [*DOI :* 10.1007/978-3-540-74792-5_5], https://hal.archives-ouvertes.fr/hal-01627866

[40] T. LIU, R. AMADINI, J. MAURO. *SUNNY with Algorithm Configuration*, in "Open Algorithm Selection Challenge (OASC 2017)", Brussels, Belgium, September 2017, pp. 12 - 14, https://hal.inria.fr/hal-01674691

[41] T. LIU, R. DI COSMO, M. GABBRIELLI, J. MAURO. *NightSplitter: a scheduling tool to optimize (sub)group activities*, in "CP 2017 - 23rd International Conference on Principles and Practice of Constraint Programming", Melbourne, Australia, Lecture Notes in Computer Science, Springer, August 2017, vol. 10416, pp. 370-386 [*DOI :* 10.1007/978-3-319-66158-2_24], https://hal.inria.fr/hal-01648192

[42] M. LODI. *Growth Mindset in Computational Thinking Teaching and Teacher Training*, in "ICER '17 International Computing Education Research Conference", Tacoma (WA), United States, August 2017 [*DOI :* 10.1145/3105726.3105736], https://hal.inria.fr/hal-01636236

Scientific Books (or Scientific Book chapters)

[43] N. DRAGONI, S. GIALLORENZO, A. LAFUENTE, M. MAZZARA, F. MONTESI, R. MUSTAFIN, L. SAFINA. *Microservices: yesterday, today, and tomorrow*, in "Present and Ulterior Software Engineering", M. MAZZARA, B. MEYER (editors), Springer, September 2017, https://hal.inria.fr/hal-01631455

[44] A. GARCIA, C. LANEVE. *JaDA – the Java Deadlock Analyzer*, in "Behavioural Types: from Theory to Tools", S. GAY, A. RAVARA (editors), River Publishers, 2017, pp. 169-192, https://hal.inria.fr/hal-01643216

[45] S. GIALLORENZO, I. LANESE, J. MAURO, M. GABBRIELLI. *Programming Adaptive Microservice Applications: an AIOCJ Tutorial*, in "Behavioural Types: from Theory to Tools", S. GAY, A. RAVARA (editors), River Publishers, June 2017, https://hal.inria.fr/hal-01631422

[46] C. GUIDI, I. LANESE, M. MAZZARA, F. MONTESI. *Microservices: a Language-based Approach*, in "Present and Ulterior Software Engineering", M. MAZZARA, B. MEYER (editors), Springer, November 2017, https://hal.inria.fr/hal-01635817

Research Reports

[47] L. HENRIO, C. LANEVE, V. MASTANDREA. *Analysis of synchronisation patterns in stateful active objects*, I3S ; Inria - Sophia antipolis, June 2017, https://hal.archives-ouvertes.fr/hal-01542595

References in notes

[48] F. BARBANERA, U. DE'LIGUORO. *Sub-behaviour relations for session-based client/server systems*, in "Mathematical Structures in Computer Science", 2015, vol. 25, n^{o} 6, pp. 1339–1381

[49] D. GELERNTER. *Generative Communication in Linda*, in "ACM Trans. Program. Lang. Syst.", 1985, vol. 7, n^{o} 1, pp. 80-112

[50] D. GORLA, R. PUGLIESE. *Resource Access and Mobility Control with Dynamic Privileges Acquisition*, in "ICALP", J. C. M. BAETEN, J. K. LENSTRA, J. PARROW, G. J. WOEGINGER (editors), Lect. Notes in Comput. Sci., Springer, 2003, vol. 2719, pp. 119-132