IN PARTNERSHIP WITH:
Université de Bologne (Italie)

Activity Report 2018

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

Post-Doctoral Fellow

Akira Yoshimizu [Inria]

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]
Stefano Pio Zingaro [University Bologna]
Gabriele Vanoni [University Bologna]

Administrative Assistant

Christine Claux [Inria]

Visiting Scientist

Emma Kerinec [Inria, from Oct 2018]

External Collaborators

Claudio Guidi [Italiana Software]
Daniel Hirschkoff [Ecole Normale Supérieure Lyon]
Fabrizio Montesi [University of Southern Denmark]
Saverio Giallorenzo [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 [42], [41], [43], 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. New Software and Platforms

5.1. 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/

5.2. 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 2018 Jolie was complemented by 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. Jolie currently supports some of the main technologies used in SOAs (e.g., HTTP). However, only a limited amount of IoT devices uses the media and protocols already supported by Jolie. Indeed, protocols such as CoAP and MQTT, which are widely used in IoT scenarios, are not implemented in Jolie. Integrating these protocols, as we have done, is essential in order to allow Jolie programs to directly interact with the majority of IoT devices. We note that emerging frameworks for interoperability, such as the Web of Things, rely on the same protocols we mentioned for IoT, thus JIoT is also compliant with them. Concretely, work in 2018 comprised the inclusion of the CoAP/UDP and MQTT/TCP protocols among the communication technologies supported by the language. The Jolie implementation of MQTT and CoAP, as well as the UDP transport protocol used by CoAP, are based on the JAVA framework Netty.

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

5.3. 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/

5.4. 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 distributed system as an AIOC, 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 2018 we did minor changes to AIOCJ, including the possibility of generating code only for a few roles, thus avoiding the need for deployment information for other roles.

- 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

5.5. 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. The debugger also provides commands to automatically find and undo consequences of a given action. 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.

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

5.6. 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. New Results

6.1. Service-Oriented Computing

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

6.1.1. 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. Usage of contracts and session types in orchestration languages guarantees correct communication [22]. Asynchronous subtyping for binary session types has been thought to be decidable for 8 years, before we proved in previous work it to be undecidable. We have now highlighted some practically-relevant fragments of session types where asynchronous subtyping is indeed decidable [14].

We also studied practical aspects of choreographies, that is multiparty contracts. On the one hand we extended the classical proof of correctness of the projection of a choreography on one participant, which is the cornerstone of the theory of choreographies. Indeed, the classical proof considers projection to a model based on channels in the CCS style, while practice mainly relies on correlation sets. We bridged this gap by giving a correctness proof towards a real-world execution model based on correlation sets [31].

We then studied how to apply choreographies for cross-organizational system integration. More precisely, we proposed a software development process to build integrations composed by distributed, independent connectors whose global behaviour is correct by construction [30]. Choreographies and choreography projection are at the heart of the proposed development process.

6.1.2. Microservices

We continued the study of microservice-oriented computing started in past years, in particular by using our microservice-oriented language Jolie. We focused, in particular, on the use of Jolie in an Internet of Things (IoT) setting [28]. Technically, a key feature of Jolie is that it supports in a uniform way multiple service-oriented communication protocols such as HTTP and SOAP. We extended Jolie in order to support, uniformly as well, also lightweight protocols such as MQTT and CoAP, which are largely used in IoT. These are very different from service-oriented protocols which are point-to-point and based on TCP since MQTT is publish-subscribe while CoAP is based on UDP.

6.2. Models for Reliability

Participant: Ivan Lanese.

6.2.1. Reversibility

We have continued the study of reversibility started in the past years, concentrating on contracts and debugging, and applying the results related to debugging to the Erlang programming language. Concerning contracts, in [12] we further studied the retractable contracts that we defined in previous work. The main novelty consists in showing how retractable contracts can be obtained by taking standard contracts, making them reversible using the general approach presented by Phillips and Ulidowski [44], and then applying suitable control policies.

Concerning debugging, we highlighted in [18] the general approach that can be used to build a causal-consistent reversible debugger for a given language. We then instantiated this general approach to a relevant subset of the language Erlang, first defining uncontrolled and controlled reversible semantics for it [16], and then building an actual causal-consistent reversible debugger called CauDEr [32].

6.3. Probabilistic Systems and Resource Control

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

6.3.1. Probabilistic Rewriting and Computation

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 sized-type system. In 2018, Focus has produced another work along these lines, based on intersection types [23]. In the usual, pure, lambda-calculus, various notions of terminating terms can be characterised by way of intersection types, in such a way that the class of terminating terms *coincides* with the one of typable terms. The presence of probabilistic choices together with the aforementioned recursion theoretical limitations prevents the same scenario to happen in probabilistic lambda-calculi, i.e., lambda-calculi endowed with some form of probabilistic choice. Nevertheless, Breuvart and Dal Lago proved that capturing the probability of termination in an approximate way by means of intersection types is indeed possible [23].

In 2018, we have also been active in laying out a novel foundation for *probabilistic abstract reduction systems* (*probabilistic ARSs*). ARSs constitute a general framework to study fundamental properties of computations, such as termination or confluence. These properties are intricately related to the well-definedness of functions, and consequently, play key roles in the formal study of programming languages. Specifically, in collaboration with Yamada, Avanzini and Dal Lago [20] introduced a new notion of probabilistic computation by means of a reduction relation over multidistributions. This relation enables the seamless combination of non-deterministic and probabilistic choice, thereby, considerably simplifying earlier notions of reduction semantics by means of schedulers and Markov chains. On top of this, a partially flawed characterisation of positive almost sure termination by means of Lyapunov ranking functions, initially due to Bournez and Garnier, could be clarified. Moreover, the *interpretation method*, which is maybe the most fundamental technique to investigate termination and runtime complexity of term rewrite systems, could be lifted to probabilistic systems.

Finally, we have been able to propose a novel and natural way of giving the reduction semantics for Markovian process algebras [13], a model of concurrent interaction which is particularly appropriate to the performance analysis of concurrent systems.

6.3.2. Complexity Analysis of Functional Programs

A research topic which lies at the core of Focus since its inception is the complexity analysis of functional programs, through tools like implicit complexity and linear logic. During 2018, we have published an extended version of a paper in which we proved that the most general form of ramified recursion, a key tool in term rewriting, remains sound for polynomial time computation, although requiring some nontrivial machinery based on sharing and memoisation [11]. We have also started to investigate along a new and promising research direction concerning the efficient implementation of functional programming languages through randomised strategies. We have shown that even the simplest strategy is nontrivial for the pure, untyped lambda-calculus [25], being in certain cases more efficient than both innermost and outermost strategies.

6.3.3. Reasoning About Effectful and Concurrent Programs

Pure functional programs are relatively easy to reason about due to referential transparency, a property inherent to functional programs that renders their semantics close to the one of mathematical expressions.

Quite recently, functional programming has found its way into main stream programming languages. Thus functional programming is *combined* with various forms of computational effects, such as exceptions, state, or even nondeterministic choice. Since a couple of years, we are interested in studying the impact of effects on the metatheory of functional programming languages, with an eye to coinductive methodologies akin to those employed in concurrency, coinduction *in primis*. In 2018, Francesco Gavazzo has extended some of our previous work about a generic approach to behavioural equivalences for higher-order effectual languages to *metrics*, themselves a much more natural way to compare programs in many cases (e.g., in the presence of probabilistic choice). His contribution has been published in the top conference in logic in computer science in 2018 [29].

6.4. Verification Techniques

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

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

6.4.1. Deadlock detection

We have continued the work on deadlock detection of previous years, on languages of concurrent objects. Thus in [33] we have applied and refined previous techniques so to handle multi-threaded programs with reentrant locks. For this we have defined a simple calculus featuring recursion, threads and synchronizations that guarantee exclusive access to objects. We detect deadlocks by associating an abstract model to programs and we define an algorithm for verifying that a problematic object dependency (e.g. a circularity) between threads will not be manifested.

In [15] we give two different notions of deadlock for systems based on active objects and futures. One is based on blocked objects and conforms with the classical definition of deadlock. The other one is an extended notion of deadlock based on blocked processes which is more general than the classical one. We introduce a technique to prove deadlock freedom in which an abstract version of the program is translated into Petri nets. Extended deadlocks, and then also classical deadlock, can be detected via checking reachability of a certain forms of marking.

6.4.2. Proof techniques based on unique solutions

We study bisimilarity, a behavioural equivalence whose success is much due to the associated bisimulation proof method. In particular, we discuss different proof methods, based on unique solution of equations or of special forms of inequations called contractions, and inspired by Milner's theorem on unique solution of equations. The techniques are at least as powerful as the bisimulation proof method and its up-to context enhancements. The techniques can be transferred onto other behavioural equivalences, possibly contextual and non-coinductive. This enables a coinductive reasoning style on such equivalences. An overview paper on these techniques is [19].

The paper [36] discusses a rather comprehensive formalisation of the core of the theory of CCS in the HOL theorem prover (HOL4), with a focus towards the theory of unique solutions of contractions. (The formalisation consists of about 20,000 lines of proof scripts in Standard ML.) Some refinements of the theory itself are obtained. In particular we remove the constraints on summation, which must be weakly-guarded, by moving to rooted contraction, that is, the coarsest precongruence contained in the contraction preorder.

In [26] we apply the above techniques to study Milner's encoding of the call-by-value λ-calculus into the π-calculus. We show that, by tuning the encoding to two subcalculi of the π-calculus (Internal π and Asynchronous Local π), the equivalence on λ-terms induced by the encoding coincides with Lassen's eager normal-form bisimilarity, extended to handle η-equality. As behavioural equivalence in the π-calculus we consider contextual equivalence and barbed congruence. We also extend the results to preorders.

On a different, but related, strand of work [17], we study the tree structures that result when writing call-by-name functions as processes, and give general conditions under which this representation produces Lévy-Longo Trees and Böhm Trees, the best known tree structures on the lambda-calculus.

6.5. 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).

6.5.1. Computational thinking and constructionism

In the last ten years, the expression “computational thinking” has been used to talk about the introduction of CS in K-12 education. The expression was originally used in the 1980s by Seymour Papert, a pioneer in Math education using programming (he is the principal inventor of the LOGO programming language). We analysed [37] the original context in which the expression originated: the constructionist learning theory, that promotes an active way of learning by constructing meaningful computational artifacts. Papert aimed to teach Math and Physics, but we think CS too is a breeding ground for applying constructionist practices like creative learning, iterative and incremental development, learning by doing, learning by trial and error, project-based learning [35].

6.5.2. CS in the school curriculum

As there is no established practice in teaching CS, academics should facilitate the introduction of CS principles in the school curriculum, to avoid misconceptions and to focus mainly on scientific principles, rather than on technical aspects. Within a CINI (Italian National Interuniversity Consortium for Informatics) group, we designed a proposal [27] for CS teaching in Italian K-10 schools, that focuses on CS principles, and gives space to the use of digital technologies only as tools for self-expression through computation. When introducing a new discipline, often misconceptions arise. In a large sample of primary teachers, we investigate [38], [24] the ideas about the “buzzword” *coding*, that is more and more used to talk about CS at school. Only 60% of teachers correctly linked “coding” to “programming” (some of them implicitly), and many misconceptions (e.g. “coding is only for children”, or “coding is the transversal use of computational thinking at school”, “programming is only for professionals”) were found. After defining a curriculum, one should also provide some materials to concretely teach the discipline and ensure learning objectives will be achieved. We presented [21] the structure of a nationwide initiative by the Italian Ministry of Education: Problem Solving Olympics (OPS). Preliminary analysis of students’ results in the last five editions suggests the competition fosters learning of computational thinking knowledge and skills.

6.5.3. Growth mindset and teacher training

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. Teachers’ mindset directly affects students’ one. By contrast, applying constructionists approaches seems to foster a growth mindset. In facts, we found a statistically significative, albeit little, increase of pre-service primary teacher’s growth mindset after a “creative computing and computational thinking” course [34].

6.6. 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 this area a *portfolio solver* combines a variety of different constraint solvers for solving a given problem. This fairly recent approach enables to significantly boost the performance of single solvers, especially when multicore architectures are exploited. In [10] we give a brief overview of the portfolio solver sunny-cp, and we discuss its performance in the MiniZinc Challenge —the annual international competition for CP solvers —where it won two gold medals in 2015 and 2016.

7. Partnerships and Cooperations

7.1. National Initiatives

- ELICA (Expanding Logical Ideas for Complexity Analysis) is an ANR project that started on October 2014 and that finished on September 2018. ELICA focused on methodologies for the static analysis of programs and their resource consumption. The project's aim was 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, Martini.
- 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.

7.2. European Initiatives

7.2.1. 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 proposed project aims to bring the existing prototype tools based on these technologies to mainstream programming languages and development frameworks used in industry.

7.2.2. 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.

7.2.3. 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.
- 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.

7.3. International Initiatives

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

7.3.1.1. 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.

7.3.2. 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.

7.3.2.1. AYAME

CRECOGI

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

International Partner (Institution - Laboratory - Researcher):

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

Duration: 2015 - 2020 The description of the project can be found in Section 7.3.1.1.

7.4. International Research Visitors

7.4.1. 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.

- Filippo Bonchi (ENS Lyon and University of Pisa) "Sound up-to techniques and complete abstract domain".

- Luis Fernando Llana Díaz (Universidad Complutense de Madrid) “Probabilistic software product lines”.
- Claudia Faggian (Université Paris-Diderot – Paris 7) : “Probabilistic Lambda Calculus – beyond deterministic evaluation”
- Nao Hirokawa (Japan Advanced Institute of Science and Technology): “Transformations for Lazy Evaluation and Theorem Proving”.
- Guilhem Jaber (University of Nantes): “Game semantics for higher-order functions with state”.
- Thomas Leventis (Institut de Mathematiques de Marseille): “Taylor Expansion of lambda terms and differential linear logic.”
- Gabriel Scherer (Inria Parsifal). “Keep (re)playing until your get all the successes”.
- Emilio Tuosto (University of Leicester): “On pomsets as models of asynchronous message-passing languages”.
- Akihisa Yamada (NII Tokyo): “Mathematics for Complexity in Isabelle/HOL”.

7.4.2. Visits to International Teams

- Francesco Gavazzo visited the Faculty of Mathematics and Physics (University of Ljubljana) hosted by Alex Simpson, from 02/10/2017 to 31/01/2018.
- U. Dal Lago has spent overall a few weeks in Japan (University of Kyoto and University of Tokyo), collaborations with Naohiko Hoshino and Naoki Kobayashi.

7.4.2.1. Sabbatical programme

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

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Organisation

8.1.1.1. General Chair, Scientific Chair

S. Martini: Workshop “Formalisms at the interface with machines, languages and systems”, Bertinoro October 16-17 2018. https://programme.hypotheses.org/autumn-workshop-formalisms-at-the-interface-with-machines-languages-and-systems

U. Dal Lago: First Workshop on Probabilistic Interactive and Higher-Order Computation, http://pihoc2018.cs.unibo.it/

8.1.1.2. Member of the Organizing Committees

Steering Committee membersip:

I. Lanese: Conference on Reversible Computation (RC); Microservices, DevOps, and Service-Oriented Architecture (MiDOS) track of the Annual ACM Symposium on Applied Computing (SAC); IFIP Int. Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE)

8.1.2. Scientific Events Selection

8.1.2.1. Chair of Conference Program Committees

U. Dal Lago: 21st Int. Conference on Foundations of Software Science and Computation Structures (FoSSaCS)

8.1.2.2. Member of the Conference Program Committees

M. Bravetti: IEEE Int. Conference on Big Data (BigData 2018); 12th IEEE Int. Conference On Big Data Science and Engineering (BigDataSE 2018); 18th IEEE Int. Conference on Software Quality, Reliability, and Security (QRS 2018); 30th IFIP Int. Conference on Testing Software and Systems (ICTSS 2018); 16th Int. Conference on Software Engineering and Formal Methods (SEFM 2018); 15th Int. Conference on Applied Computing (AC 2018)

U. Dal Lago: Thirty-Third Annual ACM/IEEE Symposium on Logic in Computer Science (LICS); Third Int. Conference on Formal Structures for Computation and Deduction (FSCD); 13th Workshop on Logical and Semantic Frameworks with Applications (LSFA);

I. Lanese: 11th IEEE Int. Conference on Service-Oriented Computing and Applications (SOCA 2018); 11th Interaction and Concurrency Experience (ICE 2018); 10th Conference on Reversible Computation (RC 2018); 15th Int. Conference on Formal Aspects of Component Software (FACS 2018); Second Int. Workshop on Microservices: Agile and DevOps Experience (MADE18)

D. Sangiorgi: 27th European Symposium on Programming (ESOP); 13th int. conf. on Software Technologies (ICSOFT'18); Symposium on Dependable Software Engineering (SETTA'18); IFIP Int. Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE); int. conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-22).

A. Yoshimizu: Games for Logic and Programming Languages XIII (GaLoP 2018).

8.1.3. Journal

8.1.3.1. 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.

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.

8.1.4. Invited Talks

I. Lanese: 10th Conference on Reversible Computation (RC 2018)

D. Sangiorgi: 19th Italian Conference on Theoretical Computer Science (ICTCS'18)

Schools:

U. Dal Lago: "Lambda Calcolo e Teoria dei Tipi", Scuola AILA, Gargnano 2018

8.1.5. 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 the vice-presiden of EQANIE (European Quality Assurance Network for Informatics Education), since 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.

8.2. Teaching - Supervision - Juries

8.2.1. 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.

- 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”, 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: “Programming in Python”, 72 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

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

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

8.2.2. Supervision

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”. Supervisor U. Dal Lago.
- 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”. Supervisor: Maurizio Gabbrielli.
- Gabriele Vanoni, November 2018. “Optimal Reduction, Geometry of Interaction, and the Space-Time Tradeoff”. Supervisor Ugo Dal Lago.

8.2.3. Juries

Daniel Hirschkoff was reviewer (rapporteur) for the PhD thesis Guvan Cabon, in december 2018 (Univ. Rennes).

Simone Martini has been member of the PhD jury of Francesco di Giacomo, Università Ca’ Foscari Venezia (June 2018) and Tilburg University (November 2018)

8.3. Popularization

8.3.1. Education

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.
- S. Martini has given various talks at institutes and workshops on the teaching methods for Computer Science, including a talk on ”Abbiamo davvero bisogno del pensiero computazionale?”, at the conference ”I bit rotolano dovunque”, Università di Roma Tor Vergata, May 2018.

8.3.2. Interventions

Daniel Hirschkoff was part of the organising committee for the "École Jeunes Chercheurs en Programmation", 25-29 june 2018, in Lyon.

8.3.3. Other duties

S. Martini has been Head of the Department of Computer Science and Engineering, University of Bologna, until May 2018.

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

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

9. 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° 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° 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° 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

Articles in International Peer-Reviewed Journals

[10] R. AMADINI, M. GABBRIELLI, J. MAURO. *SUNNY-CP and the MiniZinc challenge*, in "Theory and Practice of Logic Programming", January 2018, vol. 18, n° 01, pp. 81 - 96 [*DOI :* 10.1017/S1471068417000205], https://hal.inria.fr/hal-01931324

[11] M. AVANZINI, U. DAL LAGO. *On sharing, memoization, and polynomial time*, in "Information and Computation", August 2018, vol. 261, pp. 3 - 22 [*DOI :* 10.1016/J.IC.2018.05.003], https://hal.archives-ouvertes.fr/hal-01926431

[12] F. BARBANERA, I. LANESE, U. DE'LIGUORO. *A theory of retractable and speculative contracts*, in "Science of Computer Programming", December 2018, vol. 167, pp. 25 - 50 [*DOI :* 10.1016/J.SCICO.2018.06.005], https://hal.inria.fr/hal-01912858

[13] M. BRAVETTI. *Reduction Semantics in Markovian Process Algebra*, in "Journal of Logical and Algebraic Methods in Programming", 2018, vol. 96, pp. 41-64 [*DOI :* 10.1016/J.JLAMP.2018.01.002], https://hal.inria.fr/hal-01921194

[14] M. BRAVETTI, M. CARBONE, G. ZAVATTARO. *On the Boundary between Decidability and Undecidability of Asynchronous Session Subtyping*, in "Theoretical Computer Science", 2018, vol. 722, pp. 19-51 [*DOI :* 10.1016/J.TCS.2018.02.010], https://hal.inria.fr/hal-01921168

[15] F. S. DE BOER, M. BRAVETTI, M. D. LEE, G. ZAVATTARO. *A Petri Net Based Modeling of Active Objects and Futures*, in "Fundamenta Informaticae", 2018, vol. 159, n^{o} 3, pp. 197-256 [*DOI :* 10.3233/FI-2018-1663], https://hal.inria.fr/hal-01919136

[16] I. LANESE, N. NISHIDA, A. PALACIOS, G. VIDAL. *A theory of reversibility for Erlang*, in "Journal of Logical and Algebraic Methods in Programming", November 2018, vol. 100, pp. 71 - 97 [*DOI :* 10.1016/J.JLAMP.2018.06.004], https://hal.inria.fr/hal-01912856

[17] D. SANGIORGI, X. XU. *Trees from functions as processes*, in "Logical Methods in Computer Science", August 2018, https://arxiv.org/abs/1804.05797 [*DOI :* 10.2168/LMCS-14(3:11)2018], https://hal.inria.fr/hal-01931186

Invited Conferences

[18] I. LANESE. *From Reversible Semantics to Reversible Debugging*, in "Reversible Computation", Leicester, United Kingdom, Lecture Notes in Computer Science, September 2018, vol. 11106, https://hal.inria.fr/hal-01912920

[19] D. SANGIORGI. *Bisimilarity via unique-solution techniques*, in "19th Italian Conference on Theoretical Computer Science", Urbino, Italy, September 2018, https://hal.inria.fr/hal-01931203

International Conferences with Proceedings

[20] M. AVANZINI, U. DAL LAGO, A. YAMADA. *On Probabilistic Term Rewriting*, in "Functional and Logic Programming - 14th International Symposium, Proceedings", Nagoya, Japan, May 2018, https://hal.archives-ouvertes.fr/hal-01926502

[21] R. BORCHIA, A. CARBONARO, G. CASADEI, L. FORLIZZI, M. LODI, S. MARTINI. *Problem Solving Olympics: an inclusive education model for learning Informatics*, in "Informatics in Schools. Fundamentals of Computer Science and Software Engineering - 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018", St. Petersburg, Russia, Lecture Notes in Computer Science, October 2018, vol. 11169, pp. 319–335 [*DOI :* 10.1007/978-3-030-02750-6_25], https://hal.inria.fr/hal-01913064

[22] M. BRAVETTI, G. ZAVATTARO. *Foundations of Coordination and Contracts and Their Contribution to Session Type Theory*, in "20th International Conference on Coordination Languages and Models (COORDINATION)", Madrid, Spain, G. D. M. SERUGENDO, M. LORETI (editors), Coordination Models and Languages, Springer International Publishing, June 2018, vol. LNCS-10852, pp. 21-50 [*DOI :* 10.1007/978-3-319-92408-3_2], https://hal.inria.fr/hal-01821498

[23] F. BREUVART, U. DAL LAGO. *On Intersection Types and Probabilistic Lambda Calculi*, in "20th International Symposium on Principles and Practice of Declarative Programming", Frankfurt am Main, Germany, ACM Press, September 2018 [*DOI :* 10.1145/3236950.3236968], https://hal.archives-ouvertes.fr/hal-01926420

[24] I. CORRADINI, M. LODI, E. NARDELLI. *An Investigation of Italian Primary School Teachers' View on Coding and Programming*, in "Informatics in Schools. Fundamentals of Computer Science and Software Engineering - 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018", St. Petersburg, Russia, Lecture Notes in Computer Science, October 2018, vol. 11169, pp. 228–243 [*DOI :* 10.1007/978-3-030-02750-6_18], https://hal.inria.fr/hal-01913059

[25] U. DAL LAGO, G. VANONI. *On Randomised Strategies in the λ-Calculus*, in "19th Italian Conference on Theoretical Computer Science", Urbino, Italy, September 2018, https://hal.archives-ouvertes.fr/hal-01926512

[26] A. DURIER, D. HIRSCHKOFF, D. SANGIORGI. *Eager Functions as Processes*, in "the 33rd Annual ACM/IEEE Symposium", Oxford, United Kingdom, Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, ACM Press, July 2018 [*DOI :* 10.1145/3209108.3209152], https://hal.archives-ouvertes.fr/hal-01917255

[27] L. FORLIZZI, M. LODI, V. LONATI, C. MIROLO, M. MONGA, A. MONTRESOR, A. MORPURGO, E. NARDELLI. *A Core Informatics Curriculum for Italian Compulsory Education*, in "Informatics in Schools. Fundamentals of Computer Science and Software Engineering - 11th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2018", St. Petersburg, Russia, Lecture Notes in Computer Science, October 2018, vol. 11169, pp. 141–153 [*DOI :* 10.1007/978-3-030-02750-6_11], https://hal.inria.fr/hal-01913057

[28] M. GABBRIELLI, S. GIALLORENZO, I. LANESE, S. P. ZINGARO. *A Language-based Approach for Interoperability of IoT Platforms*, in "Hawaii International Conference on System Science", Waikoloa Village, United States, November 2018, https://hal.inria.fr/hal-01912958

[29] F. GAVAZZO. *Quantitative Behavioural Reasoning for Higher-order Effectful Programs: Applicative Distances*, in "LICS '18- Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science", Oxford, United Kingdom, July 2018 [*DOI :* 10.1145/3209108.3209149], https://hal.inria.fr/hal-01926069

[30] S. GIALLORENZO, I. LANESE, D. RUSSO. *ChIP: a Choreographic Integration Process*, in "On the Move to Meaningful Internet Systems", La Valletta, Malta, October 2018, https://hal.inria.fr/hal-01912917

[31] S. GIALLORENZO, F. MONTESI, M. GABBRIELLI. *Applied Choreographies*, in "38th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE)", Madrid, Spain, C. BAIER, L. CAIRES (editors), Formal Techniques for Distributed Objects, Components, and Systems, Springer International Publishing, June 2018, vol. LNCS-10854, pp. 21-40 [*DOI :* 10.1007/978-3-319-92612-4_2], https://hal.inria.fr/hal-01824812

[32] I. LANESE, N. NISHIDA, A. PALACIOS, G. VIDAL. *CauDEr: A Causal-Consistent Reversible Debugger for Erlang*, in "Functional and Logic Programming", Nagoya, Japan, May 2018, https://hal.inria.fr/hal-01912894

[33] C. LANEVE. *A lightweight deadlock analysis for programs with threads and reentrant locks*, in "22nd International Symposium on Formal Methods", Oxford, United Kingdom, July 2018, https://hal.inria.fr/hal-01926509

[34] M. LODI. *Can Creative Computing foster Growth Mindset?*, in "Joint Proceedings of the 1st Co-Creation in the Design, Development and Implementation of Technology-Enhanced Learning workshop (CC-TEL 2018) and Systems of Assessments for Computational Thinking Learning workshop (TACKLE 2018) co-located with 13th European Conference on Technology Enhanced Learning (ECTEL 2018)", Leeds, United Kingdom, CEUR Workshop Proceedings, September 2018, vol. 2190, https://hal.inria.fr/hal-01913053

[35] M. MONGA, M. LODI, D. MALCHIODI, A. MORPURGO, B. SPIELER. *Learning to program in a constructionist way*, in "Proceedings of Constructionism 2018", Vilnius, Lithuania, August 2018, https://hal.inria.fr/hal-01913065

[36] C. TIAN, D. SANGIORGI. *Unique solutions of contractions, CCS, and their HOL formalisation*, in "Combined 25th International Workshop on Expressiveness in Concurrency and 15th Workshop on Structural Operational Semantics", Beijing, China, September 2018, vol. 276, pp. 122 - 139 [*DOI :* 10.4204/EPTCS.276.10], https://hal.inria.fr/hal-01931199

National Conferences with Proceedings

[37] M. LODI. *Computational Thinking: from "samba schools of computation" to CoderDojos*, in "Atti del convegno DIDAMATICA 2018", Cesena, Italy, April 2018, https://hal.inria.fr/hal-01913063

Other Publications

[38] I. CORRADINI, M. LODI, E. NARDELLI. *Coding and Programming: What Do Italian Primary School Teachers Think? (Abstract Only)*, ACM Press, February 2018, SIGCSE '18 - Proceedings of the 49th ACM Technical Symposium on Computer Science Education, Poster [*DOI :* 10.1145/3159450.3162268], https://hal.inria.fr/hal-01913062

[39] A. DURIER, D. HIRSCHKOFF, D. SANGIORGI. *Eager Functions as Processes*, March 2018, working paper or preprint [*DOI :* 10.1145/3209108.3209152], https://hal.archives-ouvertes.fr/hal-01736696

[40] A. DURIER, D. HIRSCHKOFF, D. SANGIORGI. *Towards 'up to context' reasoning about higher-order processes*, August 2018, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01857391

References in notes

[41] 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

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

[43] 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

[44] 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