IN PARTNERSHIP WITH:
Université de Bologne (Italie)

Activity Report 2016

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:

1. - Architectures, systems and networks
1.3. - Distributed Systems
1.4. - Ubiquitous Systems
2.1.1. - Semantics of programming languages
2.1.6. - Concurrent programming
2.1.7. - Distributed programming
2.4.3. - Proofs

Other Research Topics and Application Domains:

6.1. - Software industry
6.3. - Network functions
6.4. - Internet of things
9.4.1. - Computer science

1. Members

Research Scientist

Elena Giachino [University Bologna, Researcher]

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, Associate Professor]

PhD Students

Francesco Gavazzo [University Bologna]
Alberto Cappai [University Bologna, until Jun 2016]
Raphaelle Crubille [Univ. Paris VII]
Vincenzo Mastandrea [Univ. Nice]
Alessandro Rioli [University Bologna, until Jun 2016]
Marco Solieri [University Paris 13]
Valeria Vignudelli [University Bologna]

Post-Doctoral Fellows

Saverio Giallorenzo [University Bologna]
Charles Grellois [Inria]
Martin Avanzini [University Bologna]
Flavien Breuvart [Inria, until Aug 2016]

Visiting Scientists

Ryo Tanaka [University Tokyo, Aug 2016]
Akira Yoshimizu [University Tokyo, Aug 2016]

Administrative Assistant

Christine Riehl

Others

Daniel Hirschkoff [Collaborateur Extérieur, ENS Lyon]
Claudio Guidi [Collaborateur Extérieur, italianaSoftware s.r.l.]
Fabrizio Montesi [Collaborateur Extérieur, 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 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: 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

- Valeria Vignudelli has won the “Outstanding Master Thesis Award”, for best master thesis in logic in computer science. Awarded by the Vienna Center for Logic and Algorithms, as part of the VCLA International Student Awards (http://logic-cs.at/award/)

6. New Software and Platforms

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

6.2. DF4ABS

Deadlock Framework for ABS

SCIENTIFIC DESCRIPTION

We have prototyped a framework for statically detecting deadlocks in a concurrent object-oriented language with asynchronous method calls and cooperative scheduling of method activations (the language is ABS, which has been developed in the EU project HATS and is currently extended with primitives for cloud-computing in the EU project ENVISAGE. ABS is very similar to ASP, developed by the former OASIS team.). Since this language features recursion and dynamic resource creation, deadlock detection is extremely complex and state-of-the-art solutions either give imprecise answers or do not scale. In order to augment precision and scalability we propose a modular framework that allows several techniques to be combined. The basic component of the framework is a front-end inference algorithm that extracts abstract behavioural descriptions of methods that retain resource dependency information. Then these behavioural descriptions are analyzed by a back-end that uses a fix-point technique to derive in a deterministic way the deadlock information.

- Contact: Cosimo Laneve
- URL: http://df4abs.nws.cs.unibo.it/

6.3. HoCA

Higher-Order 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 analyzed 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. All these techniques have been specifically suited to the methods integrated in

modern first-order complexity analyzers. 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), and call-by-value reductions of the PCF program are simulated by the ATRS step-by-step. On the ATRS, various complexity reflecting transformations are performed: inlining, dead-code-elimination, instantiation of higher-order variables through a call-flow-analysis and finally uncurrying. This results in a first-order rewrite system, whose runtime-complexity asymptotically reflects the complexity of the initial program.

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

6.4. 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.
DEVELOPMENTS IN 2016

Jolie has transitioned from version 1.4.1 to version 1.6. The last version of Jolie that supports Java 1.6 is Jolie 1.5. Jolie 1.6 transitions from Java 1.6 to Java 1.8 and makes use of the new features and libraries found in the new version of Java. Version 1.6 of Jolie features:

- general performance improvements and bug fixes, in particular regarding concurrent data structures using Java lambdas,
- improvements of the standard library of the language,
- better error messages and improved compatibility with the main operating systems,
- support for type choices (AKA type sums),
- support of for-loop construct to iterate over arrays without explicit indexes,
- improved support for the HTTP protocol (and, by extension, web applications).
- Participants: Claudio Guidi, Fabrizio Montesi, Saverio Giallorenzo and Maurizio Gabbrielli
- Contact: Fabrizio Montesi
- URL: http://www.jolie-lang.org/

6.5. SRA

Static Resource Analyzer for ABS
SCIENTIFIC DESCRIPTION

We prototype a static analysis technique that computes upper bounds of virtual machine usages in a concurrent language with explicit acquire and release operations of 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, a feature that is used by Amazon Elastic Cloud Computing or by the Docker FiWare). Our technique is modular and consists of (i) a type system associating programs with behavioural descriptions 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.

- Contact: Cosimo Laneve
- URL: http://sra.cs.unibo.it/

6.6. SUNNY-CP

SCIENTIFIC DESCRIPTION

Within the Constraint Programming paradigm, a portfolio solver combines different constraint solvers in order to create a globally better solver. Sunny-cp is a parallel portfolio solver that allows one to solve a Constraint (Satisfaction/Optimization) Problem defined in the MiniZinc language. It essentially implements the SUNNY algorithm introduced in the team. Sunny-cp is built on top of state-of-the-art constraint solvers, including: Choco, Chuffed, CPX, G12/LazyFD, G12/FD, G12/Gurobi, G12/CBC, Gecode, HaifaCSP, iZplus, MinisatID, Opturion, OR-Tools.

FUNCTIONAL DESCRIPTION

SUNNY-CP is a portfolio solver for solving both Constraint Satisfaction Problems and Constraint Optimization Problems. The goal of SUNNY-CP is to provide a flexible, configurable, and usable CP portfolio solver that can be set up and executed just like a regular individual CP solver.

- Contact: Maurizio Gabbrielli
- URL: https://github.com/CP-Unibo/sunny-cp

6.7. Blender

Aeolus Blender
KEYWORDS: Automatic deployment - Cloud applications management
SCIENTIFIC DESCRIPTION

The various tools developed in the Aeolus project (Zephyrus, Metis, Armonic) have been combined in this software which represents an integrated solution for the declarative specification of cloud applications, and its subsequent automatic deployment on an OpenStack cloud system. In particular, a web-based interface is used to specify the basic software artifacts to include in the application, indicate their level of replication, and specify co-installability conflicts (i.e. when two components cannot be installed on the same virtual machines). The tool Zephyrus is then used to synthesize the final architecture of the application, the tool Metis indicates the plan of configuration actions, and the Armonic platform provides the library of components and the low-level scripts to actually install and configure the entire application.

- Partners: IRILL - Mandriva
- Contact: Gianluigi Zavattaro
- URL: https://github.com/aeolus-project/blender

7. New Results

7.1. Service-oriented computing

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

7.1.1. Microservices

Microservices is an emerging paradigm for the development of distributed systems that, originating from Service-Oriented Architecture, fosters the creation of an ecosystem of reusable components by focusing on the small dimension, the loose coupling, and the dynamic topology of services. Their dynamic nature calls for suitable techniques that support automatic deployment. In [40] we address this problem and we propose JRO (Jolie Redeployment Optimiser), a tool for the automatic and optimised deployment of microservices written in the Jolie language. The flexibility of microservices is their key advantage, yet it poses many security issues. In [39] we classify the most relevant vulnerabilities related to data reliability, integrity, and authenticity, and we investigate directions for their mitigation.

7.1.2. Orchestrations and choreographies

The practice of programming distributed systems is one of the most 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, choreographies can be compiled to obtain projected systems that enjoy freedom from deadlocks and races by construction. In [10] we studied how to make choreographies, and extensions of them that allow one to perform dynamic updates, a suitable tool for real-world programming.

7.2. Models for reliability

Participants: Elena Giachino, Ivan Lanese.

7.2.1. Reversibility

We have continued the study of causal-consistent reversibility started in the past years. In particular, we concentrated on uncontrolled reversibility, where one specifies how a concurrent computation can go back to past states, without giving policies about when to do that. In [25] we thoroughly studied the problem for higher-order pi-calculus. In particular, we studied the causality structures needed to enable reversibility, and we related them with the causal semantics of Boreale and Sangiorgi. In [30] we proposed a modular approach that, given a formal model equipped with both an LTS semantics and an independence relation capturing causality, defines a causal-consistent reversible semantics for it. The approach is very general, capturing models as different as CCS and concurrent X-machines, but it is not fully automatic.

7.3. Cloud Computing and Deployment

Participants: Elena Giachino, Saverio Giallorenzo, Claudio Guidi, Cosimo Laneve, Gianluigi Zavattaro.

7.3.1. Static deployment

We have continued our foundational investigation of the Aeolus component model for the automatic deployment of a component-based application in a cloud environment. In [42] we have refined a previous Turing completeness result for the Aeolus model. In fact, a previous proof of undecidability of the deployment problem assumes the possibility of performing in a synchronized way atomic configuration actions on a set of interdependent components: this feature is usually not supported by actual deployment frameworks. To make the theoretical model used for our undecidability result closer to the real deployment infrastructures, in [42] we have proved that even without synchronized configuration actions the Aeolus component model is still Turing complete.

7.3.2. Dynamic deployment

We have analyzed linguistic mechanisms for expressing and managing dynamic aspects of deployment, in particular the possibility to dynamically modify the architecture of an application.

In [17] we propose a new mechanism for Dynamic Rebinding, a particular kind of Dynamic Software Updating that focuses on modifying the workflow of a program. This mechanism is built upon the model of Concurrent Object Groups, which is adopted in programming languages like Coboxes, Creol or ABS. Using this model, which extends and solves some of the limitations of Active Objects, it becomes possible for an update to wait for the program to reach a local quiescent state and then perform the update without creating any inconsistency in the program's state.

In [34] we show how deployment can be added as a first-class citizen in the object-oriented modeling language ABS. We follow a declarative approach: programmers specify deployment constraints and a solver synthesizes ABS classes exposing methods like deploy (resp. undeploy) that executes (resp. cancels) configuration actions changing the current deployment towards a new one satisfying the programmer's desiderata. Differently from previous works, this novel approach supports the specification of incremental modifications, thus supporting the declarative modeling of elastic applications.

7.4. Probabilistic Systems and Resource Control

Participants: Martin Avanzini, Flavien Breuvart, Alberto Cappai, Raphaelle Crubillé, Ugo Dal Lago, Francesco Gavazzo, Charles Grellois, Simone Martini, Alessandro Rioli, Davide Sangiorgi, Marco Solieri, Valeria Vignudelli.

7.4.1. Probabilistic Systems

7.4.1.1. Behavioural Equivalences and Metrics

Finding effective methodologies to check program equivalence is one of the oldest problems in the theory of programming languages, and has been studied also in the realm of probabilistic programming idioms. One particularly fruitful research direction consists in *characterising* context equivalence, the most natural way to *define* equivalence in higher-order languages, by way of *coinductive* notions of equivalence akin to bisimulation. In 2016, Focus has been involved in defining notions of *environmental* bisimulation for probabilistic lambda-calculi [37], proving them not only adequate, but also fully-abstract. Environmental bisimulation, contrarily to *applicative* bisimulation, is robust enough to be applicable to languages with local store. Moreover, the proof of adequacy of environmental bisimulation turns out to be simpler than that of applicative bisimulation, the latter requiring sophisticated arguments from linear programming. In a probabilistic setting, programs are more naturally compared through metrics rather than through equivalences, due to their intrinsic quantitative nature. Nicely, coinductive methodologies for program equivalence can be generalised to metrics by way of so-called *behavioural metrics*. This year, we have studied behavioural metrics in the context of concurrent processes, and defined enhancements of the proof method based on bisimulation metrics, by extending the theory of up-to techniques to premetrics on discrete probabilistic concurrent processes [32].

7.4.1.2. Programming Languages for Machine Learning

In recent years, higher-order functional programming languages like Church, Anglican, and Venture, have proved to be extremely effective as ways to specify not algorithms but rather bayesian models in the context of machine learning. The operational semantics of these languages, and learning algorithms when applied to programs in these languages, have been so far defined only informally. In 2016, we developed the operational semantics of an untyped probabilistic lambda-calculus with continuous distributions, as a foundation for universal probabilistic programming languages like those cited above [31]. Our first contribution was to adapt the classic operational semantics of lambda-calculus to a continuous setting. Our second contribution was to formalise the implementation technique of trace Markov chain Monte Carlo (MCMC) for our calculus and to show its correctness.

7.4.2. Resource Control

7.4.2.1. Complexity Analysis of Higher-Order Functional Programs

Complexity analysis of higher-order programs have been one of the main research themes inside Focus since its inception. It remains so today, although the emphasis is progressively shifting towards problems related to the *implementation* of complexity analysis methodologies rather than on their foundations. One issue with most existing complexity analysis methodologies is that they are insensitive to the sharing of computations among subprograms. We have studied how the interpretation method and dependency tuples, two prominent complexity analysis techniques can be adapted to graph-rewriting, thus accounting for the possible performance gains due to sharing [38]. We have also collaborated to the development of TCT, the Tyrolean Complexity Tool [29], a state-of-the-art complexity analyzer for term rewrite systems, making it capable to efficiently apply not one but *many* methodologies to the input program. Finally, we studied how the geometry of interaction can provide effective ways to compile higher-order functional programs into circuits, thus guaranteeing space efficiency [21].

7.4.2.2. On the Foundations of Complexity Analysis

One of the main foundational issues in complexity analysis is whether simple time cost models can be proved invariant, i.e., polynomially related to low-level models like those traditionally defined on Turing machines. We have solved a long-standing open problem, and proved that the unitary cost model, namely that attributing unitary cost to each beta-reduction step, is invariant for the pure lambda-calculus when evaluated leftmost-outermost [12]. We have also studied to which extent traditional methodologies like the interpretation method and light logics can be adapted to higher-order languages [16] and processes [20], respectively.

7.5. Verification techniques

Participants: Daniel Hirschkoff, Elena Giachino, Cosimo Laneve, Davide Sangiorgi.

7.5.1. Deadlock detection

In [22] we present a framework for statically detecting deadlocks in a concurrent object-oriented language with asynchronous method calls and cooperative scheduling of method activations. Since this language features recursion and dynamic resource creation, deadlock detection is extremely complex and state-of-the-art solutions either give imprecise answers or do not scale. In order to augment precision and scalability we propose a modular framework that allows several techniques to be combined. The basic component of the framework is a front-end inference algorithm that extracts abstract behavioural descriptions of methods, called contracts, which retain resource dependency information. This component is integrated with a number of possible different back-ends that analyze contracts and derive deadlock information. As a proof-of-concept, we discuss two such back-ends: (i) an evaluator that computes a fixpoint semantics and (ii) an evaluator using abstract model checking.

In [36] we study deadlock detection in an actor model with wait-by-necessity synchronizations, a lightweight technique that synchronizes invocations when the corresponding values are strictly needed. This approach relies on the use of futures that are not given an explicit Future type. The approach we adopt allows for the implicit synchronization on the availability of some value (where the producer of the value might be decided at runtime), whereas previous work allowed only explicit synchronization on the termination of a well-identified request. This way we are able to analyze the data-flow synchronization inherent to languages that feature wait-by-necessity. We provide a type-system and a solver inferring the type of a program so that deadlocks can be identified statically. As a consequence we can automatically verify the absence of deadlocks in actor programs with wait-by-necessity synchronizations.

7.5.2. Service Level Agreement

There is a gap between run-time service behaviours and the contracted quality expectations with the customers that is due to the informal nature of service level agreements. In [41] we explain how to bridge the gap by formalizing service level agreements with metric functions. We therefore discuss an end-to-end analysis flow that can either statically verify if a service code complies with a metric function or use run-time monitoring systems to report possible misbehaviours. In both cases, our approach provides a feedback loop to fix and improve the metrics and eventually the resource configurations of the service itself.

7.6. Type Systems

Participants: Daniel Hirschkoff, Simone Martini, Davide Sangiorgi.

7.6.1. Surveys

In [27], Martini elaborates the history of type systems, focusing on that fundamental period covering the seventies and the early eighties. It was then that types became the cornerstone of the programming language design, passing first from the abstract data type (ADT) movement and blossoming then into the object-oriented paradigm. The paper also discusses how it has been possible that a concept like ADTs, with its clear mathematical semantics, neat syntax, and straightforward implementation, can have given way to objects, a lot dirtier from any perspective the language theorist may take.

In another paper [45], the same author compares the notion of "type" as found in programming languages with that found in mathematical logic, pointing out also some important historical remarks such as the role of the Curry-Howard isomorphism. It is argued that there are three different characters at play in programming languages, all of them now called types: the technical concept used in language design to guide implementation; the general abstraction mechanism used as a modelling tool; the classifying tool inherited from mathematical logic.

Two further surveys concerns behavioural types. The successful application of behavioural types requires a solid understanding of several practical aspects, from their representation in a concrete programming language, to their integration with other programming constructs such as methods and functions, to design and monitoring methodologies that take behaviours into account. The survey [15] provides an overview of the state of the art of these aspects.

The behavioural type of a software component specifies its expected patterns of interaction using expressive type languages, so that types can be used to determine automatically whether the component interacts correctly with other components. Two related important notions of behavioural types are those of session types and behavioural contracts. The paper [24] surveys the main accomplishments of the last twenty years within these two approaches.

7.6.2. Subtyping and dualities in name-passing concurrency

The fusion calculi are simplifications of the π-calculus in which input and output are symmetric and restriction is the only binder. In [23], Hirschkoff et al. highlight a major difference between these calculi and the π-calculus from the point of view of types, proving some impossibility results for subtyping in fusion calculi. A modification of fusion calculi is then proposed that allows one to import subtype systems, and related results, from the π-calculus, and examine the consequences of such modifications on theory and expressiveness of the languages.

8. Partnerships and Cooperations

8.1. National Initiatives

- REVER (Programming Reversible Recoverable Systems) is an ANR project with a 4-year duration. REVER aims to study the possibility of defining semantically well-founded and composable abstractions for dependable computing on the basis of a reversible programming language substrate, where reversibility means the ability to undo any distributed program execution, possibly step by step. The critical assumption behind REVER is that by adopting a reversible model of computation, and by combining it with appropriate notions of compensation and modularity, one can develop systematic and composable abstractions for recoverable and dependable systems. Main persons involved: Giachino, Lanese, Laneve, Zavattaro.

- PACE (Processus non-standard: Analyse, Coinduction, et Expressivité) is an ANR project with a 4-year duration. The project targets three fundamental ingredients in theories of concurrent processes, namely coinduction, expressiveness, and analysis techniques. The project targets processes that are beyond the realm of "traditional" processes. Specifically, the models studied exhibit one or more of the following features: probabilities, higher-order, quantum, constraints, knowledge, and confidentiality. These models are becoming increasingly important for today's applications. Coinduction is intended to play a pivotal role. Indeed, the approaches to expressiveness and the analysis techniques considered in the project are based on coinductive equalities. Main persons involved: Hirschkoff (project coordinator), Dal Lago, Lanese, Sangiorgi, Zavattaro.
- 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, Cappai, 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, Martini, 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. FP7 & H2020 Projects

- ENVISAGE (Engineering Virtualized Services) is a EU FP7 project, with starting date October 1st, 2013, and with a 3-year duration. The project is about model-based development of virtualized services, including tool support for resource analysis. Most Focus members are involved.

8.2.2. Collaborations in European Programs, Except FP7 & H2020

- The ICT COST Action BETTY (Behavioural Types for Reliable Large-Scale Software Systems). initiated in October 2012 and with a 4-year duration, uses behavioural type theory as the basis for new foundations, programming languages, and software development methods for communication-intensive distributed systems. Behavioural type theory encompasses concepts such as interfaces, communication protocols, contracts, and choreographies. Main persons involved: Bravetti, Giachino, Hirschkoff, Lanese, Laneve, Mauro, Sangiorgi, Zavattaro.
- 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: Giachino, 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.
- SMAll (Smart Mobility for All) SMAll is an EIT project that runs during 2016.

  The aim of the project is to develop a service-oriented platform, called the SMAll platform [1], to support the creation of a liquid market for transportation, facilitating the publication, retrieval, and orchestration of functionalities for transportation, owned by different operators.

  Jolie plays a prominent part in the development of the SMAll platform: it is the main language for the development of both the platform — the architecture of services that support publishing, organisation, and interaction among the functionalities for transportation — and of the services for mobility.

8.2.3. Collaborations with Major European Organizations

Simone Martini is a member of the Executive Board of EQANIE (European Quality Assurance Network for Informatics Education), from October 2014.

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 new joint PhD started in september 2016 (Adrien Durier).
- Inria EPI Spades (on models and languages for components, reversibility). 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. An Italian PhD student (Marco Solieri) is working on his PhD thesis with joint supervision (Martini, Guerrini).
- 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: Mauro, Zavattaro. Some short visits in both directions during the year.
- EPI Carte, Inria-Nancy Grand Est and LORIA (on implicit computational complexity). Contact person(s) in Focus: Dal Lago.
- 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.

[1] https://github.com/small-dev/SMAll.Wiki/wiki

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.

8.4. 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.

- Marco Bernardo: "Disjunctive Probabilistic Modal Logic is Enough for Bisimilarity on Reactive Probabilistic Systems."
- Guillermo Roman Diez: "Resource Analysis."
- Lukasz Mikulski: "On Concurrency Paradigms."
- Jean-Pierre Jouannaud: "Coq modulo theory."
- Paul Blain Levy: "Trace semantics of well-founded processes via commutativity."
- Akihisa Yamada: "Certifying Safety and Termination Proofs for Integer Transition Systems" (long visit).
- Ryo Tanaka: semantics of higher-order functional languages (long visit).
- Antonio Ravara: “Behavioural Type Inference for Object-Oriented Languages”.
- Lukasz Mikulski: "On concurrency paradigms".
- Ludovic Henrio, “Deadlock analysis in distributed object systems”.

8.4.2. Visits to International Teams

- Ugo Dal Lago: has spent two weeks in April at ENS Lyon, and 1 month at IRIF, Université Paris 7, in May/June.
- Charles Grellois has taken part in the programme "Automata, Logic and Games", 5 weeks (August-September), Singapore.
- Valeria Vignudelli has spent six months in the Inria Comète team, Inria Saclay/Ecole Polytechnique, Paris.
- Abel Garcia has spent 6 months at the Department of Computer Science, TUD Darmstadt.

8.4.2.1. Sabbatical programme

Maurizio Gabbrielli is, since 15 September 2014, Head of the EIT ICT Labs Doctoral School with Paris as his principal location.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

I. Lanese: Co-chair and local organizer for the 8th Conference on Reversible Computation (RC 2016), Bologna, July 7th-8th, 2016.

9.1.1.2. Chair of Conference Program Committees

I. Lanese: 36th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE 2016)

D. Sangiorgi: 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), Rome, July 2016

9.1.1.3. Member of the Conference Program Committees

M. Bravetti: 5th European Conference on Service-Oriented and Cloud Computing (ESOCC 2016); 14th International Conference on Software Engineering and Formal Methods (SEFM 2016); 10th IEEE International Conference on Big Data Science and Engineering (IEEE BigDataSE 2016); 2016 IEEE International Conference on Big Data (IEEE BigData 2016) 32nd Symposium on Applied Computing, track on Software Verification and Testing (SAC-SVT 2016); 7th International Conference on Fundamentals of Software Engineering (FSEN 2017).

U. Dal Lago: 13th IFIP WG 1.3 International Workshop on Coalgebraic Methods in Computer Science (CMCS 2016); 1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016); 15th International Workshop on Termination (WST 2016); 17th International Workshop on Logic and Computational Complexity (LCC 2016, Chair).

I. Lanese: 1st International Workshop on Pre- and Post-Deployment Verification Techniques (PrePost 2016); 9th Interaction and Concurrency Experience (ICE 2016); 25th European Symposium on Programming (ESOP 2016); Special Track on Microservices: Science and Engineering of the 9th IEEE International Conference on Service-Oriented Computing and Applications (MSE @ SOCA 2016).

S. Martini: DIDAMATICA - Informatica per la Didattica (DIDAMATICA 2016).

D. Sangiorgi: 10th International Conference on Language and Automata Theory and Applications (LATA'16).

V. Vignudelli: 9th Interaction and Concurrency Experience (ICE 2016).

G. Zavattaro: 21st International Symposium on Formal Methods (FM2016); 5th European Conference on Service-oriented and Cloud Computing (ESOCC'16); 27th International Conference on Concurrency Theory (CONCUR'16); 31st ACM/SIGAPP Symposium On Applied Computing - track on Software Verification and Testing (ACM-SAC 2016).

9.1.1.4. Member of the Editorial Boards

U. Dal Lago: Logical Methods 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).

The Scientific World Journal (Hindawi Publishing Corporation): editor till August 3, 2016

F. Montesi: Guest editor for "Special issue on Service-Oriented Architecture and Programming (SOAP 2013)". Science of Computer Programming.

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

9.1.2. Invited Talks

U. Dal Lago: gave an invited course "Implicit Complexity and the Curry-Howard Correspondence" at 'Days in Logic 2016', Lisboa, Portugal.

D. Hirschkoff: gave an invited course "Mechanised Coinductive Proofs" at East China Normal University (Shanghai, China), in november 2016 (a course mainly targeted towards master students).

S. Martini: gave the invited talk "Types in programming languages, between modelling, abstraction and correctness" at 'Computability in Europe' (Cie 2016), special session on History and Philosophy of Computing, Paris, June 2016.

G. Zavattaro: gave an invited course on cloud computing management at the 'Bertinoro International Spring School 2016' (BISS'16), Bertinoro, Italy, 6-11 March, 2016.

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: “Cryptography”, 36 Hours, 2nd Year, University of Bologna, Italy.
- Daniel Hirschkoff

  Master: “Coinductive Methods in Computer Science”, 2nd year, Ecole Normale Supérieure de Lyon.
- Maurizio Gabbrielli

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

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

  Undergraduate: “Programmazione”, 40 hours, 1st year, University of Bologna, Italy.
- Saverio Giallorenzo

  Undergraduate: “Laboratorio di Operating Systems”, 40 hours, 2nd year, University of Bologna, Italy.
- Ivan Lanese

  Undergraduate: “Programmazione”, 32 hours, 1st year, University of Bologna, Italy.

  Undergraduate: “Algoritmi e Strutture Dati”, 45 hours, 2nd 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”, 58 hours, 1st year, University of Bologna, Italy.

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

  Master: “Logical Foundations of Computer Science”, 48 hours, 2nd year, University of Bologna, Italy.
- Davide Sangiorgi

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

  Master: “Models for concurrency”, 15 hours, 2nd year, University of Bologna, Italy.
- Marco Solieri

  Undergraduate: “UNIX system programming”, 48 hours, 3rd year, Université Paris Diderot

Undergraduate: “Network programming”, 48 hours, 3rd year, Université Paris Diderot

- Gianluigi Zavattaro

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

  Undergraduate: “Programming Languages”, 60 hours, 2nd year, University of Bologna, Italy

9.2.2. Supervision

PhD thesis completed in 2016:

- Alberto Cappai, " On Equivalences, Metrics, and Computational Indistinguishability", University of Bologna, April 2016.
- Alessandro Rioli, " Coinductive Techniques on a Linear Quantum λ-calculus", University of Bologna, April 2016.
- Marco Solieri, “Sharing, Superposition and Expansion — Geometrical Studies on the Semantics and Implementation of λ-calculi and Proof-Nets”, S. Guerrini (Paris Nord) and S. Martini. [11].
- Saverio Giallorenzo, “Real-World Choreographies”, University of Bologna, April 2016 [10].

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.

- 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.
- Abel Garcia, January 2014, “Analysis of Cloud Computing Systems”. Supervisor C. Laneve.
- Francesco Gavazzo, October 2015, “Coinductive Techniques for Effectful Lambda Calculi”. Supervisors U. Dal Lago and D. Sangiorgi.
- Raphaelle Crubillé, October 2015, “Bisimulation Metrics and Probabilistic Lambda Calculi”, Université Denis Diderot and University of Bologna. Supervisors Thomas Ehrhard and Ugo Dal Lago.
- Vincenzo Mastandrea, October 2015, “Deadlock analysis in ASP”. Supervisors Cosimo Laneve and Ludovic Henrio (CNRS Sophia Antipolis).
- Valeria Vignudelli, January 2014, “Probabilities in Higher-Order languages”. Supervisor D. Sangiorgi.

9.2.3. Juries

U. Dal Lago has been member of the PhD evaluation committee for Charles Grellois (April 2016, Université Paris 7), and Thomas Leventis (December 2016, Université Aix-Marseille).

D. Hirschkoff has been referee (rapporteur) for the Habilitation à Diriger les Recherches of Nicolas Tabareau, Université de Nantes), November 2016, and member of the PhD committee of Martin Bodin (Université Rennes 1), Nov. 25, 2016.

S. Martini has been supervisor and member of the PhD evaluation committee for Marco Solieri (Université Paris 13 and Università di Bologna).

G. Zavattaro: has been member of the PhD jury of Laura Nenzi and Andrea Morichetta, IMT Lucca, Italy, July 2016.

9.3. Popularization

Simone Martini has carried out extended work of scientific divulgation, including

- member of the technical committee of Olimpiadi del Problem Solving (at Italian Ministry of Education), http://www.olimpiadiproblemsolving.com;
- invited talks ("Il pensiero computazionale spiegato ai manager", given at 'Il codice del futuro', Teatro anatomico dell'Archiginnasio, Bologna, May 2016, and "Pensare computazionale: una quarta competenza dopo scrivere, leggere e far di conto", given at 'Trasformazioni sociali e trasmissione delle conoscenze nell'Università italiana', Bologna, November 2016);
- various talks at institutes and workshops on the teaching methods for Computer Science;
- coordinator of some initiatives for the 'Hour of Code', see https://italia.code.org/ and http://www.programmailfuturo.it.

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

9.3.1. Other duties

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.

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

D. Sangiorgi has been coordinator of undergraduate studies at the Department of Computer Science and Engineering, University of Bologna (Informatica per il Management), till October 2016, when G. Zavattaro has taken over from him.

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^{o} 2

[6] U. D. Lago, M. Gaboardi. *Linear Dependent Types and Relative Completeness*, in "Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011", IEEE Computer Society, 2011, pp. 133-142, https://hal.inria.fr/hal-00906347/en

[7] I. Lanese, C. A. Mezzina, A. Schmitt, J.-B. Stefani. *Controlling Reversibility in Higher-Order Pi*, in "Proc. of CONCUR 2011", J.-P. Katoen, B. König (editors), Lecture Notes in Computer Science, Springer, 2011, vol. 6901, pp. 297-311, http://www.cs.unibo.it/~lanese/publications/fulltext/concur2011.pdf.gz

[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] S. GIALLORENZO. *Real-World Choreographies*, Università degli studi di Bologna, May 2016, https://hal.inria.fr/tel-01336757

[11] M. SOLIERI. *Sharing, Superposition and Expansion: Geometrical Studies on the Semantics and Implementation of λ-calculi and Proof-nets*, Université Paris XIII, Sorbonne Paris Cité ; Università di Bologna, November 2016, https://hal.archives-ouvertes.fr/tel-01400369

Articles in International Peer-Reviewed Journals

[12] B. ACCATTOLI, U. DAL LAGO. *(Leftmost-outermost) beta reduction is invariant, indeed*, in "Logical Methods in Computer Science", 2016 [*DOI :* 10.2168/LMCS-12(1:4)2016], https://hal.inria.fr/hal-01337712

[13] R. AMADINI, M. GABBRIELLI, J. MAURO. *An Extensive Evaluation of Portfolio Approaches for Constraint Satisfaction Problems*, in "International Journal of Interactive Multimedia and Artificial Intelligence", 2016, https://hal.inria.fr/hal-01336684

[14] R. AMADINI, M. GABBRIELLI, J. MAURO. *Portfolio Approaches for Constraint Optimization Problems*, in "Annals of Mathematics and Artificial Intelligence", 2016, https://hal.inria.fr/hal-01336686

[15] D. ANCONA, V. BONO, M. BRAVETTI, J. CAMPOS, G. CASTAGNA, P.-M. DENIÉLOU, S. J. GAY, N. GESBERT, E. GIACHINO, R. HU, E. B. JOHNSEN, F. MARTINS, V. MASCARDI, F. MONTESI, R. NEYKOVA, N. NG, L. PADOVANI, V. T. VASCONCELOS, N. YOSHIDA. *Behavioral Types in Programming Languages*, in "Foundations and Trends in Programming Languages", July 2016, vol. 3, n^{o} 2-3, pp. 95-230 [*DOI :* 10.1561/2500000031], https://hal.inria.fr/hal-01348054

[16] P. BAILLOT, U. DAL LAGO. *Higher-order interpretations and program complexity*, in "Information and Computation", 2016 [*DOI :* 10.1016/J.IC.2015.12.008], https://hal.inria.fr/hal-01337728

[17] M. BRAVETTI, E. GIACHINO, M. LIENHARDT, P. WONG. *Dynamic Rebinding for Concurrent Object Groups: Theory and Practice*, in "Journal of Logical and Algebraic Methods in Programming", April 2016 [*DOI :* 10.1016/J.JLAMP.2016.03.002], https://hal.inria.fr/hal-01337333

[18] F. BREUVART, G. MANZONETTO, A. POLONSKY, D. RUOPPOLO. *New Results on Morris's Observational Theory: the benefits of separating the inseparable*, in "Leibniz International Proceedings in Informatics (LIPIcs)", June 2016, 560 p. [*DOI :* 10.4230/LIPICS.FSCD.2016.70], https://hal.inria.fr/hal-01337192

[19] S. CAPECCHI, E. GIACHINO, N. YOSHIDA. *Global Escape in Multiparty Sessions*, in "Mathematical Structures in Computer Science", 2016 [*DOI :* 10.1017/S0960129514000164], https://hal.inria.fr/hal-01336832

[20] U. DAL LAGO, S. MARTINI, D. SANGIORGI. *Light Logics and Higher-Order Processes*, in "Mathematical Structures in Computer Science", 2016, vol. 26, n^{o} 06, pp. 969 - 992 [*DOI :* 10.1017/S0960129514000310], https://hal.inria.fr/hal-01400903

[21] U. DAL LAGO, U. SCHÖPP. *Computation by interaction for space-bounded functional programming*, in "Information and Computation", 2016, vol. 248 [*DOI :* 10.1016/J.IC.2015.04.006], https://hal.inria.fr/hal-01337724

[22] E. GIACHINO, C. LANEVE, M. LIENHARDT. *A framework for deadlock detection in core ABS*, in "Software and Systems Modeling", 2016 [*DOI :* 10.1007/S10270-014-0444-Y], https://hal.inria.fr/hal-01229046

[23] D. HIRSCHKOFF, J.-M. MADIOT, D. SANGIORGI. *Name-passing calculi: From fusions to preorders and types*, in "Journal of Information and Computation", 2016, vol. 251, 26 p. [*DOI :* 10.1016/J.IC.2016.10.003], https://hal.inria.fr/hal-01419632

[24] H. HÜTTEL, E. TUOSTO, H. T. VIEIRA, G. ZAVATTARO, I. LANESE, V. T. VASCONCELOS, L. CAIRES, M. CARBONE, P.-M. DENIÉLOU, D. MOSTROUS, L. PADOVANI, A. RAVARA. *Foundations of Session Types and Behavioural Contracts*, in "ACM Computing Surveys", June 2016, vol. 49, n^{o} 1 [*DOI :* 10.1145/2873052], https://hal.archives-ouvertes.fr/hal-01336707

[25] I. LANESE, C. A. MEZZINA, J.-B. STEFANI. *Reversibility in the higher-order π-calculus*, in "Theoretical Computer Science", 2016, vol. 625, pp. 25-84 [*DOI :* 10.1016/J.TCS.2016.02.019], https://hal.inria.fr/hal-01303090

[26] M. SOLIERI. *Geometry of resource interaction and Taylor–Ehrhard–Regnier expansion: a minimalist approach*, in "Mathematical Structures in Computer Science", November 2016, pp. 1 - 43 [*DOI :* 10.1017/S0960129516000311], https://hal.archives-ouvertes.fr/hal-01400359

Invited Conferences

[27] S. MARTINI. *Types in Programming Languages, between Modelling, Abstraction, and Correctness*, in "Computability in Europe, CiE 2016: Pursuit of the Universal", Paris, France, LNCS, Springer, June 2016, vol. 9709, https://hal.inria.fr/hal-01335657

International Conferences with Proceedings

[28] E. ALBERT, F. DE BOER, R. HÄHNLE, E. B. JOHNSEN, C. LANEVE. *Envisage: Developing SLA-aware Deployed Services with Formal Methods*, in "ESOCC 2016:Fifth European Conference on Service-Oriented and Cloud Computing", Wien, Austria, September 2016, https://hal.inria.fr/hal-01345020

[29] M. AVANZINI, G. MOSER, M. SCHAPER. *Complexity of Acyclic Term Graph Rewriting*, in "Proceedings of FSCD'16", Porto, Portugal, D. KESNER, B. PIENTKA (editors), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, June 2016, vol. 52, pp. 10:1 - 10:18 [*DOI :* 10.4230/LIPICS.FSCD.2016], https://hal.inria.fr/hal-01392188

[30] A. BERNADET, I. LANESE. *A Modular Formalization of Reversibility for Concurrent Models and Languages*, in "ICE 2016", Heraklion, Greece, EPTCS, June 2016, https://hal.inria.fr/hal-01337423

[31] J. BORGSTRÖM, U. DAL LAGO, A. D. GORDON, M. SZYMCZAK. *A lambda-calculus foundation for universal probabilistic programming*, in "International Conference on Functional Programming", Nara, Japan, 2016, pp. 33 - 46 [*DOI :* 10.1145/2951913.2951942], https://hal.inria.fr/hal-01400890

[32] K. CHATZIKOKOLAKIS, C. PALAMIDESSI, V. VIGNUDELLI. *Up-To Techniques for Generalized Bisimulation Metrics*, in "27th International Conference on Concurrency Theory (CONCUR 2016)", Québec City, Canada, Leibniz International Proceedings in Informatics (LIPIcs), August 2016, vol. 59, pp. 35:1–35:14 [*DOI :* 10.4230/LIPIcs.CONCUR.2016.35], https://hal.inria.fr/hal-01335234

[33] U. DAL LAGO. *Infinitary Lambda Calculi from a Linear Perspective*, in "Logic in Computer Science", New York, United States, July 2016 [*DOI :* 10.1145/2933575.2934505], https://hal.inria.fr/hal-01400883

[34] S. DE GOUW, J. MAURO, B. NOBAKHT, G. ZAVATTARO. *Declarative Elasticity in ABS*, in "Proceedings of the European Conference on Service-Oriented and Cloud Computing 2016", Vienna, Austria, September 2016, pp. 118 - 134 [*DOI :* 10.1007/978-3-319-44482-6_8], https://hal.inria.fr/hal-01399887

[35] M. FALASCHI, M. GABBRIELLI, C. OLARTE, C. PALAMIDESSI. *Slicing Concurrent Constraint Programs*, in "Pre-proceedings of the 26th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2016)", Edinburgh, United Kingdom, M. V. HERMENEGILDO, P. LOPEZ-GARCIA (editors), 2016, https://hal.inria.fr/hal-01421407

[36] E. GIACHINO, L. HENRIO, C. LANEVE, V. MASTANDREA. *Actors may synchronize, safely!* *, in "PPDP 2016 18th International Symposium on Principles and Practice of Declarative Programming", Edinburgh, United Kingdom, September 2016, https://hal.inria.fr/hal-01345315

[37] D. SANGIORGI, V. VIGNUDELLI. *Environmental Bisimulations for Probabilistic Higher-Order Languages*, in "POPL '16", St. Petersburg, United States, Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, January 2016 [*DOI :* 10.1145/2837614.2837651], https://hal.inria.fr/hal-01337665

Conferences without Proceedings

[38] M. AVANZINI, G. MOSER. *Complexity of Acyclic Term Graph Rewriting*, in "Proceedings of FSCD'16", Porto, Portugal, June 2016 [*DOI :* 10.4230/LIPIcs.FSCD.2016], https://hal.inria.fr/hal-01336582

[39] F. CALLEGATI, S. GIALLORENZO, A. MELIS, M. PRANDINI. *Data Security Issues in MaaS-enabling Platforms*, in "International Forum on Research and Technologies for Society and Industry", Bologna, Italy, September 2016, https://hal.inria.fr/hal-01336700

Scientific Books (or Scientific Book chapters)

[40] M. GABBRIELLI, S. GIALLORENZO, C. GUIDI, J. MAURO, F. MONTESI. *Self-Reconfiguring Microservices*, in "Theory and Practice of Formal Methods", E. ÁBRAHÁM, M. BONSANGUE, E. B. JOHNSEN (editors), Lecture Notes in Computer Science, Springer, 2016, n^o 9660, pp. 194-210 [*DOI :* 10.1007/978-3-319-30734-3_14], https://hal.inria.fr/hal-01336688

[41] E. GIACHINO, S. DE GOUW, C. LANEVE, B. NOBAKHT. *Statically and Dynamically Verifiable SLA Metrics*, in "Theory and Practice of Formal Methods - Essays Dedicated to Frank de Boer on the Occasion of His 60th Birthday", Lecture Notes in Computer Science, Springer, 2016, vol. 9660, pp. 211-225 [*DOI :* 10.1007/978-3-319-30734-3_15], https://hal.archives-ouvertes.fr/hal-01336836

[42] J. MAURO, G. ZAVATTARO. *On the Expressiveness of Synchronization in Component Deployment*, in "Theory and Practice of Formal Methods", Lecture Notes in Computer Science, 2016, vol. 9660, pp. 344-359 [*DOI :* 10.1007/978-3-319-30734-3], https://hal.inria.fr/hal-01334772

Research Reports

[43] R. AMADINI, M. GABBRIELLI, J. MAURO. *Parallelizing Constraint Solvers for Hard RCPSP Instances*, Inria Sophia Antipolis, March 2016, https://hal.inria.fr/hal-01295061

[44] R. AMADINI, M. GABBRIELLI, J. MAURO. *SUNNY-CP: a Portfolio Solver for Constraint Programming*, Inria Sophia Antipolis, 2016, https://hal.inria.fr/hal-01336407

References in notes

[45] S. MARTINI. *Several Types of Types in Programming Languages*, in "HaPoC 2015 - Third International Conference History and Philosophy of Computing", Pisa, Italy, Springer, October 2015, vol. 487, pp. 216 - 227 [*DOI :* 10.1007/978-3-319-47286-7_15], https://hal.inria.fr/hal-01399694