IN PARTNERSHIP WITH:
Institut polytechnique de Grenoble

Université Joseph Fourier (Grenoble)

Activity Report 2016

Project-Team CONVECS

Construction of verified concurrent systems

IN COLLABORATION WITH: Laboratoire d'Informatique de Grenoble (LIG)

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Embedded and Real-time Systems

Table of contents

Project-Team CONVECS

Creation of the Team: 2012 January 01, updated into Project-Team: 2014 January 01

Keywords:

Computer Science and Digital Science:

1.1.6. - Cloud
1.3. - Distributed Systems
2.1.1. - Semantics of programming languages
2.1.7. - Distributed programming
2.4.1. - Analysis
2.4.2. - Model-checking
2.5. - Software engineering
5.11.1. - Human activity analysis and recognition
7.1. - Parallel and distributed algorithms
7.4. - Logic in Computer Science
7.11. - Performance evaluation

Other Research Topics and Application Domains:

6.3.2. - Network protocols
6.4. - Internet of things
6.6. - Embedded systems
8.1. - Smart building/home
9.4.1. - Computer science

1. Members

Research Scientists

Radu Mateescu [Team leader, Inria, Senior Researcher, HDR]
Hubert Garavel [Inria, Senior Researcher]
Frédéric Lang [Inria, Researcher]
Wendelin Serwe [Inria, Researcher]

Faculty Member

Gwen Salaün [UGA, Professor, HDR]

Engineers

Jingyan Jourdan-Lu [Inria, until Jan 2016, granted by Conseil Région Auvergne Rhône-Alpes]
Eric Léo [Inria, until Feb 2016, granted by Conseil Région Auvergne Rhône-Alpes]
Hugues Evrard [Inria, until Jul 2016]

PhD Students

Gianluca Barbon [UGA]
Lina Marsso [Inria, from Oct 2016]
Umar Ozeer [Orange Labs, from Nov 2016]
Fatma Jebali [Inria, until Jul 2016, granted by Conseil Région Auvergne Rhône-Alpes]

Administrative Assistant

Myriam Etienne [Inria]

Others

Sai Srikar Kasi [Inria, Intern, from May 2016 until Jul 2016]
Ajay Muroor-Nadumane [Inria, Intern, from Feb 2016 until Jul 2016]
Mohammad-Ali Tabikh [Inria, Intern, from Feb 2016 until Jul 2016]

2. Overall Objectives

2.1. Overview

The CONVECS project-team addresses the rigorous design of concurrent asynchronous systems using formal methods and automated analysis. These systems comprise several activities that execute simultaneously and autonomously (i.e., without the assumption about the existence of a global clock), synchronize, and communicate to accomplish a common task. In computer science, asynchronous concurrency arises typically in hardware, software, and telecommunication systems, but also in parallel and distributed programs.

Asynchronous concurrency is becoming ubiquitous, from the micro-scale of embedded systems (asynchronous logic, networks-on-chip, GALS – *Globally Asynchronous, Locally Synchronous* systems, multi-core processors, etc.) to the macro-scale of grids and cloud computing. In the race for improved performance and lower power consumption, computer manufacturers are moving towards asynchrony. This increases the complexity of the design by introducing nondeterminism, thus requiring a rigorous methodology, based on formal methods assisted by analysis and verification tools.

There exist several approaches to formal verification, such as theorem proving, static analysis, and model checking, with various degrees of automation. When dealing with asynchronous systems involving complex data types, verification methods based on state space exploration (reachability analysis, model checking, equivalence checking, etc.) are today the most successful way to detect design errors that could not be found otherwise. However, these verification methods have several limitations: they are not easily accepted by industry engineers, they do not scale well while the complexity of designs is ever increasing, and they require considerable computing power (both storage capacity and execution speed). These are the challenges that CONVECS seeks to address.

To achieve significant impact in the design and analysis of concurrent asynchronous systems, several research topics must be addressed simultaneously. There is a need for user-friendly, intuitive, yet formal specification languages that will be attractive to designers and engineers. These languages should provide for both functional aspects (as needed by formal verification) and quantitative ones (to enable performance evaluation and architecture exploration). These languages and their associated tools should be smoothly integrated into large-scale design flows. Finally, verification tools should be able to exploit the parallel and distributed computing facilities that are now ubiquitous, from desktop to high-performance computers.

3. Research Program

3.1. New Formal Languages and their Concurrent Implementations

We aim at proposing and implementing new formal languages for the specification, implementation, and verification of concurrent systems. In order to provide a complete, coherent methodological framework, two research directions must be addressed:

- *Model-based specifications*: these are operational (i.e., constructive) descriptions of systems, usually expressed in terms of processes that execute concurrently, synchronize together and communicate. Process calculi are typical examples of model-based specification languages. The approach we promote is based on LOTOS NT (LNT for short), a formal specification language that incorporates most constructs stemming from classical programming languages, which eases its acceptance by students and industry engineers. LNT [29] is derived from the ISO standard E-LOTOS (2001), of which it represents the first successful implementation, based on a source-level translation from

LNT to the former ISO standard LOTOS (1989). We are working both on the semantic foundations of LNT (enhancing the language with module interfaces and timed/probabilistic/stochastic features, compiling the m among n synchronization, etc.) and on the generation of efficient parallel and distributed code. Once equipped with these features, LNT will enable formally verified asynchronous concurrent designs to be implemented automatically.

- *Property-based specifications*: these are declarative (i.e., non-constructive) descriptions of systems, which express *what* a system should do rather than *how* the system should do it. Temporal logics and μ-calculi are typical examples of property-based specification languages. The natural models underlying value-passing specification languages, such as LNT, are Labeled Transition Systems (LTSs or simply *graphs*) in which the transitions between states are labeled by actions containing data values exchanged during handshake communications. In order to reason accurately about these LTSs, temporal logics involving data values are necessary. The approach we promote is based on MCL (*Model Checking Language*) [51], which extends the modal μ-calculus with data-handling primitives, fairness operators encoding generalized Büchi automata, and a functional-like language for describing complex transition sequences. We are working both on the semantic foundations of MCL (extending the language with new temporal and hybrid operators, translating these operators into lower-level formalisms, enhancing the type system, etc.) and also on improving the MCL on-the-fly model checking technology (devising new algorithms, enhancing ergonomy by detecting and reporting vacuity, etc.).

We address these two directions simultaneously, yet in a coherent manner, with a particular focus on applicable concurrent code generation and computer-aided verification.

3.2. Parallel and Distributed Verification

Exploiting large-scale high-performance computers is a promising way to augment the capabilities of formal verification. The underlying problems are far from trivial, making the correct design, implementation, fine-tuning, and benchmarking of parallel and distributed verification algorithms long-term and difficult activities. Sequential verification algorithms cannot be reused as such for this task: they are inherently complex, and their existing implementations reflect several years of optimizations and enhancements. To obtain good speedup and scalability, it is necessary to invent new parallel and distributed algorithms rather than to attempt a parallelization of existing sequential ones. We seek to achieve this objective by working along two directions:

- *Rigorous design:* Because of their high complexity, concurrent verification algorithms should themselves be subject to formal modeling and verification, as confirmed by recent trends in the certification of safety-critical applications. To facilitate the development of new parallel and distributed verification algorithms, we promote a rigorous approach based on formal methods and verification. Such algorithms will be first specified formally in LNT, then validated using existing model checking algorithms of the CADP toolbox. Second, parallel or distributed implementations of these algorithms will be generated automatically from the LNT specifications, enabling them to be experimented on large computing infrastructures, such as clusters and grids. As a side-effect, this "bootstrapping" approach would produce new verification tools that can later be used to self-verify their own design.
- *Performance optimization:* In devising parallel and distributed verification algorithms, particular care must be taken to optimize performance. These algorithms will face concurrency issues at several levels: grids of heterogeneous clusters (architecture-independence of data, dynamic load balancing), clusters of homogeneous machines connected by a network (message-passing communication, detection of stable states), and multi-core machines (shared-memory communication, thread synchronization). We will seek to exploit the results achieved in the parallel and distributed computing field to improve performance when using thousands of machines by reducing the number of connections and the messages exchanged between the cooperating processes carrying out the verification task. Another important issue is the generalization of existing LTS representations (explicit, implicit, distributed) in order to make them fully interoperable, such that compilers and verification tools can handle these models transparently.

3.3. Timed, Probabilistic, and Stochastic Extensions

Concurrent systems can be analyzed from a *qualitative* point of view, to check whether certain properties of interest (e.g., safety, liveness, fairness, etc.) are satisfied. This is the role of functional verification, which produces Boolean (yes/no) verdicts. However, it is often useful to analyze such systems from a *quantitative* point of view, to answer non-functional questions regarding performance over the long run, response time, throughput, latency, failure probability, etc. Such questions, which call for numerical (rather than binary) answers, are essential when studying the performance and dependability (e.g., availability, reliability, etc.) of complex systems.

Traditionally, qualitative and quantitative analyzes are performed separately, using different modeling languages and different software tools, often by distinct persons. Unifying these separate processes to form a seamless design flow with common modeling languages and analysis tools is therefore desirable, for both scientific and economic reasons. Technically, the existing modeling languages for concurrent systems need to be enriched with new features for describing quantitative aspects, such as probabilities, weights, and time. Such extensions have been well-studied and, for each of these directions, there exist various kinds of automata, e.g., discrete-time Markov chains for probabilities, weighted automata for weights, timed automata for hard real-time, continuous-time Markov chains for soft real-time with exponential distributions, etc. Nowadays, the next scientific challenge is to combine these individual extensions altogether to provide even more expressive models suitable for advanced applications.

Many such combinations have been proposed in the literature, and there is a large amount of models adding probabilities, weights, and/or time. However, an unfortunate consequence of this diversity is the confuse landscape of software tools supporting such models. Dozens of tools have been developed to implement theoretical ideas about probabilities, weights, and time in concurrent systems. Unfortunately, these tools do not interoperate smoothly, due both to incompatibilities in the underlying semantic models and to the lack of common exchange formats.

To address these issues, CONVECS follows two research directions:

- *Unifying the semantic models*. Firstly, we will perform a systematic survey of the existing semantic models in order to distinguish between their essential and non-essential characteristics, the goal being to propose a unified semantic model that is compatible with process calculi techniques for specifying and verifying concurrent systems. There are already proposals for unification either theoretical (e.g., Markov automata) or practical (e.g., PRISM and MODEST modeling languages), but these languages focus on quantitative aspects and do not provide high-level control structures and data handling features (as LNT does, for instance). Work is therefore needed to unify process calculi and quantitative models, still retaining the benefits of both worlds.
- *Increasing the interoperability of analysis tools*. Secondly, we will seek to enhance the interoperability of existing tools for timed, probabilistic, and stochastic systems. Based on scientific exchanges with developers of advanced tools for quantitative analysis, we plan to evolve the CADP toolbox as follows: extending its perimeter of functional verification with quantitative aspects; enabling deeper connections with external analysis components for probabilistic, stochastic, and timed models; and introducing architectural principles for the design and integration of future tools, our long-term goal being the construction of a European collaborative platform encompassing both functional and non-functional analyzes.

3.4. Component-Based Architectures for On-the-Fly Verification

On-the-fly verification fights against state explosion by enabling an incremental, demand-driven exploration of LTSs, thus avoiding their entire construction prior to verification. In this approach, LTS models are handled implicitly by means of their *post* function, which computes the transitions going out of given states and thus serves as a basis for any forward exploration algorithm. On-the-fly verification tools are complex software artifacts, which must be designed as modularly as possible to enhance their robustness, reduce their development effort, and facilitate their evolution. To achieve such a modular framework, we undertake research in several directions:

- *New interfaces for on-the-fly LTS manipulation*. The current application programming interface (API) for on-the-fly graph manipulation, named OPEN/CAESAR [35], provides an "opaque" representation of states and actions (transitions labels): states are represented as memory areas of fixed size and actions are character strings. Although appropriate to the pure process algebraic setting, this representation must be generalized to provide additional information supporting an efficient construction of advanced verification features, such as: handling of the types, functions, data values, and parallel structure of the source program under verification, independence of transitions in the LTS, quantitative (timed/probabilistic/stochastic) information, etc.
- *Compositional framework for on-the-fly LTS analysis*. On-the-fly model checkers and equivalence checkers usually perform several operations on graph models (LTSs, Boolean graphs, etc.), such as exploration, parallel composition, partial order reduction, encoding of model checking and equivalence checking in terms of Boolean equation systems, resolution and diagnostic generation for Boolean equation systems, etc. To facilitate the design, implementation, and usage of these functionalities, it is necessary to encapsulate them in software components that could be freely combined and replaced. Such components would act as graph transformers, that would execute (on a sequential machine) in a way similar to coroutines and to the composition of lazy functions in functional programming languages. Besides its obvious benefits in modularity, such a component-based architecture will also make it possible to take advantage of multi-core processors.
- *New generic components for on-the-fly verification*. The quest for new on-the-fly components for LTS analysis must be pursued, with the goal of obtaining a rich catalog of interoperable components serving as building blocks for new analysis features. A long-term goal of this approach is to provide an increasingly large catalog of interoperable components covering all verification and analysis functionalities that appear to be useful in practice. It is worth noticing that some components can be very complex pieces of software (e.g., the encapsulation of an on-the-fly model checker for a rich temporal logic). Ideally, it should be possible to build a novel verification or analysis tool by assembling on-the-fly graph manipulation components taken from the catalog. This would provide a flexible means of building new verification and analysis tools by reusing generic, interoperable model manipulation components.

3.5. Real-Life Applications and Case Studies

We believe that theoretical studies and tool developments must be confronted with significant case studies to assess their applicability and to identify new research directions. Therefore, we seek to apply our languages, models, and tools for specifying and verifying formally real-life applications, often in the context of industrial collaborations.

4. Application Domains

4.1. Application Domains

The theoretical framework we use (automata, process algebras, bisimulations, temporal logics, etc.) and the software tools we develop are general enough to fit the needs of many application domains. They are applicable to virtually any system or protocol that consists of distributed agents communicating by asynchronous messages. The list of recent case studies performed with the CADP toolbox (see in particular § 6.5) illustrates the diversity of applications:

- *Bioinformatics:* genetic regulatory networks, nutritional stress response, metabolic pathways,
- *Component-based systems:* Web services, peer-to-peer networks,
- *Databases:* transaction protocols, distributed knowledge bases, stock management,
- *Distributed systems:* virtual shared memory, dynamic reconfiguration algorithms, fault tolerance algorithms, cloud computing,

- *Embedded systems:* air traffic control, avionic systems, medical devices,
- *Hardware architectures:* multiprocessor architectures, systems on chip, cache coherency protocols, hardware/software codesign,
- *Human-machine interaction:* graphical interfaces, biomedical data visualization, plasticity,
- *Security protocols:* authentication, electronic transactions, cryptographic key distribution,
- *Telecommunications:* high-speed networks, network management, mobile telephony, feature interaction detection.

5. New Software and Platforms

5.1. The CADP Toolbox

Participants: Hubert Garavel [correspondent], Frédéric Lang, Radu Mateescu, Wendelin Serwe.

We maintain and enhance CADP (*Construction and Analysis of Distributed Processes* – formerly known as *CAESAR/ALDEBARAN Development Package*) [1], a toolbox for protocols and distributed systems engineering [1]. In this toolbox, we develop and maintain the following tools:

- CAESAR.ADT [34] is a compiler that translates LOTOS abstract data types into C types and C functions. The translation involves pattern-matching compiling techniques and automatic recognition of usual types (integers, enumerations, tuples, etc.), which are implemented optimally.
- CAESAR [40], [39] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purposes). The translation is done using several intermediate steps, among which the construction of a Petri net extended with typed variables, data handling features, and atomic transitions.
- OPEN/CAESAR [35] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently of any particular high level language. In this respect, OPEN/CAESAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CAESAR consists of a set of 16 code libraries with their programming interfaces, such as:
  - CAESAR_GRAPH, which provides the programming interface for graph exploration,
  - CAESAR_HASH, which contains several hash functions,
  - CAESAR_SOLVE, which resolves Boolean equation systems on the fly,
  - CAESAR_STACK, which implements stacks for depth-first search exploration, and
  - CAESAR_TABLE, which handles tables of states, transitions, labels, etc.

  A number of on-the-fly analysis tools have been developed within the OPEN/CAESAR environment, among which:
  - BISIMULATOR, which checks bisimulation equivalences and preorders,
  - CUNCTATOR, which performs steady-state simulation of continuous-time Markov chains,
  - DETERMINATOR, which eliminates stochastic nondeterminism in normal, probabilistic, or stochastic systems,
  - DISTRIBUTOR, which generates the graph of reachable states using several machines,
  - EVALUATOR, which evaluates MCL formulas,
  - EXECUTOR, which performs random execution,

[1] http://cadp.inria.fr

  - EXHIBITOR, which searches for execution sequences matching a given regular expression,
  - GENERATOR, which constructs the graph of reachable states,
  - PROJECTOR, which computes abstractions of communicating systems,
  - REDUCTOR, which constructs and minimizes the graph of reachable states modulo various equivalence relations,
  - SIMULATOR, XSIMULATOR, and OCIS, which enable interactive simulation, and
  - TERMINATOR, which searches for deadlock states.
- BCG (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. BCG also plays a key role in CADP as many tools rely on this format for their inputs/outputs. The BCG environment consists of various libraries with their programming interfaces, and of several tools, such as:
  - BCG_CMP, which compares two graphs,
  - BCG_DRAW, which builds a two-dimensional view of a graph,
  - BCG_EDIT, which allows the graph layout produced by BCG_DRAW to be modified interactively,
  - BCG_GRAPH, which generates various forms of practically useful graphs,
  - BCG_INFO, which displays various statistical information about a graph,
  - BCG_IO, which performs conversions between BCG and many other graph formats,
  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph,
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc.

    For instance, one can define recursive functions on sets of states, which allow evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [42], CTL [30], ACTL [32], etc.) to be defined in XTL.
- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [38] and providing a unified access to a graph partitioned in fragments distributed over a set of remote machines, possibly located in different countries. The PBG format is supported by several tools, such as:
  - PBG_CP, PBG_MV, and PBG_RM, which facilitate standard operations (copying, moving, and removing) on PBG files, maintaining consistency during these operations,
  - PBG_MERGE (formerly known as BCG_MERGE), which transforms a distributed graph into a monolithic one represented in BCG format,
  - PBG_INFO, which displays various statistical information about a distributed graph.
- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CAESAR-compliant compilers, e.g.:
  - BCG_OPEN, for models represented as BCG graphs,
  - CAESAR.OPEN, for models expressed as LOTOS descriptions,
  - EXP.OPEN, for models expressed as communicating automata,
  - FSP.OPEN, for models expressed as FSP [50] descriptions,
  - LNT.OPEN, for models expressed as LNT descriptions, and
  - SEQ.OPEN, for models represented as sets of execution traces.

The CADP toolbox also includes TGV (*Test Generation based on Verification*), which has been developed by the VERIMAG laboratory (Grenoble) and the VERTECS project-team at Inria Rennes – Bretagne-Atlantique.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [36] scripting language. Both EUCALYPTUS and SVL provide users with an easy and uniform access to the CADP tools by performing file format conversions automatically whenever needed and by supplying appropriate command-line options as the tools are invoked.

5.2. The TRAIAN Compiler

Participants: Hubert Garavel [correspondent], Frédéric Lang, Wendelin Serwe.

We develop a compiler named TRAIAN, see § 5.2), for translating LOTOS NT descriptions into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

The current version of TRAIAN, which handles LOTOS NT types and functions only, has useful applications in compiler construction [37], being used in all recent compilers developed by CONVECS.

The TRAIAN compiler can be freely downloaded from the CONVECS Web site [2].

6. New Results

6.1. New Formal Languages and their Implementations

The ability to compile and verify formal specifications with complex, user-defined operations and data structures is a key feature of the CADP toolbox since its very origins. A long-run effort has been recently undertaken to ensure a uniform treatment of types, values, and functions across all the various CADP tools.

6.1.1. Translation from LNT to LOTOS

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

LNT is a next generation formal description language for asynchronous concurrent systems, which attempts to combine the best features of imperative programming languages and value-passing process algebras. LNT is increasingly used by CONVECS for industrial case studies and applications (see § 6.5) and serves also in university courses on concurrency, in particular at ENSIMAG (Grenoble) and at Saarland University.

In 2016, the long-term effort to enhance the LNT language and its tools has been pursued. LNT has been enriched with a new statement “use X” that suppresses compiler warnings when a variable X is assigned but not used. The syntax of LNT expressions has been modified so that field selections (“E.X”), field updates (“E1.X = E2”), and array accesses (“E1 [E2]”) can now be freely combined without extra parentheses. LNT programs can now import predefined libraries, and two such libraries (BIT.lnt and OCTET.lnt) have been introduced.

A move towards “safer” LNT exceptions has started. The syntax for exceptions in function declarations has been modified and the semantics of LNT has shifted from “unchecked” to “checked” exceptions: exception parameters, if any, must be explicitly mentioned when a function is called. Such exception parameters can now be passed using either the named style or the positional style.

A few static-semantics constraints have been relaxed; for instance, it is no longer required that actual gate parameters be different when calling a process. Various bugs have been fixed. Several error/warning messages have been made more precise, and the format of LNT error/warning messages has been aligned on that of GCC. Finally, the LNT2LOTOS Reference Manual has been updated and enhanced.

6.1.2. Translation from LOTOS NT to C

Participants: Hubert Garavel, Sai Srikar Kasi, Wendelin Serwe.

[2] http://convecs.inria.fr/software/traian

The TRAIAN compiler is used to build many compilers and translators of the CADP toolbox. This compiler itself is built using the FNC-2 compiler generator that, unfortunately, is no longer available. For this reason, TRAIAN only exists in 32-bit version, and sometimes hits the 3-4 GByte RAM limit when dealing with complex compilers such as LNT2LOTOS.

In 2016 we addressed this issue, in several steps. As a first step, we released a stable version 2.8 of TRAIAN. Then, we gathered all programs written in LOTOS NT, the input language of TRAIAN, and organized them in non-regression test bases. We entirely scrutinized the source code of TRAIAN, which consists in a large collection of attribute grammars, deleting all parts of code corresponding to those features of the LOTOS NT language that were either not fully implemented or seldom used in practice. This reduced the source code of TRAIAN by 40% and divided by two the size of TRAIAN executables. A few other bugs have been fixed and the reference manual of TRAIAN was entirely revised and updated.

In parallel, we undertook a complete rewrite of TRAIAN to get rid of the FNC-2 deprecated attribute grammar tool. We developed lexical and syntactic descriptions of the input language using the SYNTAX compiler-generation system developed at Inria Paris. The syntax tree of LOTOS NT and the library of predefined LOTOS NT types and functions are now themselves defined in LOTOS NT, as we plan to follow a bootstraping approach, using the current version of TRAIAN to build the next one. To this aim, a large fraction of the TRAIAN attribute grammars has been rewritten in LOTOS NT.

6.1.3. Translation from LOTOS to Petri nets and C

Participants: Hubert Garavel, Wendelin Serwe.

The LOTOS compilers CAESAR and CAESAR.ADT, which were once the flagship of CADP, now play a more discrete role since LNT (rather than LOTOS) has become the recommended specification language of CADP. Thus, CAESAR and CAESAR.ADT are mostly used as back-end translators for LOTOS programs automatically generated from LNT or other formalisms such as Fiacre, and are only modified when this appears to be strictly necessary.

In 2016, following the writing of the new CADP manual page for LOTOS, the common front-end of CAESAR and CAESAR.ADT was carefully inspected, which led to various bug fixes regarding type signatures, error messages for overloaded functions, renaming/actualization of sorts and operations, equations for renamed operations, C-language reserved keywords, implementation of numeral sorts, and iterators over these sorts. Another bug was fixed for the "`-external`" option of CAESAR and a new "`-numeral`" option was introduced. Also, the C identifiers automatically generated by CAESAR.ADT for sorts, operations, tester and selector macros have been simplified; as the new conventions are not backward compatible, migration tools were developed to ease transitioning the existing LOTOS and C files.

6.1.4. NUPN

Participant: Hubert Garavel.

Nested-Unit Petri Nets (NUPNs) is an upward-compatible extension of P/T nets, which are enriched with structural information on their concurrent structure. Such additional information can easily be produced when NUPNs are generated from higher-level specifications (e.g., process calculi); quite often, such information allows logarithmic reductions in the number of bits required to represent states, thus enabling verification tools to perform better. The principles of NUPNs are exposed in [33] and its PNML representation is described here [3].

In 2016, the NUPN principles have been presented in an invited talk at D-CON, the German national conference on concurrency theory. The collection of NUPN models used for experimentation has been enlarged and reorganized; it now contains more than $10,000$ models. A new beta-version of the VLPN (*Very Large Petri Nets*) benchmark suite, which contains 350 large models has been produced. Also, new prototype tools have been developed that try to convert P/T nets into NUPNs, which requires to automatically infer the concurrent structure of flat, unstructured nets.

[3] http://mcc.lip6.fr/nupn.php

The CAESAR.BDD tool that analyzes NUPN models and serves to prepare the yearly Model Checking Contest [4] has been enhanced with two new options "`-initial-places`" and "`-initial-tokens`". It now properly handles the case where the initial marking contains more than 2^{31} tokens. The output of the "`-mcc`" option has been made more precise when the NUPN under study is conservative or sub-conservative.

6.1.5. Translation from BPMN to LNT

Participants: Gwen Salaün, Ajay Muroor-Nadumane.

Evolution has become a central concern in software development and in particular in business processes, which support the modeling and the implementation of software as workflows of local and inter-process activities. We advocate that business process evolution can be formally analyzed in order to compare different versions of processes, identify precisely the differences between them, and ensure the desired consistency.

In collaboration with Pascal Poizat (LIP6, Paris), we worked on checking the evolution of BPMN processes. To promote its adoption by business process designers, we have implemented it in a tool, VBPMN, that can be used through a Web application. We have defined different kinds of atomic evolutions that can be combined and formally verified. We have defined a BPMN to LNT model transformation, which, using the LTS operational semantics of LNT, enables us to automate our approach using existing LTS model checking and equivalence checking tools, such as those provided by CADP. We have applied our approach to many examples for evaluation purposes. These results have been published in an international conference [23].

6.1.6. Translation from GRL to LNT

Participants: Hubert Garavel, Fatma Jebali, Jingyan Jourdan-Lu, Frédéric Lang, Eric Léo, Radu Mateescu, Wendelin Serwe.

In the context of the Bluesky project (see § 8.2.2.1), we study the formal modeling of GALS (*Globally Asynchronous, Locally Synchronous*) systems, which are composed of several synchronous subsystems evolving cyclically, each at its own pace, and communicating with each other asynchronously. Designing GALS systems is challenging due to both the high level of (synchronous and asynchronous) concurrency and the heterogeneity of computations (deterministic and nondeterministic). To bring our formal verification techniques and tools closer to the GALS paradigm, we designed a new formal language named GRL (*GALS Representation Language*), as an intermediate format between GALS models and purely asynchronous concurrent models. GRL combines the main features of synchronous dataflow programming and asynchronous process calculi into one unified language, while keeping the syntax homogeneous for better acceptance by industrial GALS designers. GRL allows a modular composition of synchronous systems (blocks), environmental constraints (environments), and asynchronous communication mechanisms (mediums), to be described at a level of abstraction that is appropriate to verification. GRL also supports external C and LNT code. A translator named GRL2LNT has been developed, allowing an LNT program to be generated from a GRL specification automatically. Additionally, an OPEN/CAESAR-compliant compiler named GRL.OPEN (based on GRL2LNT and LNT.OPEN) enables the on-the-fly exploration of the LTS underlying a GRL specification using CADP.

In 2016, a new version 3.3 of the GRL2LNT translator has been released, with an improved LNT code generation exploiting the "`use`" construct newly added to LNT. Also, a non-regression test base containing hundreds of GRL specifications has been set up. This also contributes to the non-regression testing of the compilation chain for LNT by providing new LNT descriptions generated automatically by GRL2LNT.

An overview paper about GRL and its translation to LNT was published in an international journal [14]. The complete definition of GRL and its applications to GALS systems are available in F. Jebali's PhD thesis [44].

6.1.7. Translation of Term Rewrite Systems

Participants: Hubert Garavel, Lina Marsso, Mohammad-Ali Tabikh.

[4] http://mcc.lip6.fr/

In 2016, we pursued the development undertaken in 2015 of a software platform for systematically comparing the performance of rewrite engines and pattern-matching implementations in algebraic specification and functional programming languages. Our platform reuses the benchmarks of the three Rewrite Engine Competitions (2006, 2009, and 2010). Such benchmarks are term-rewrite systems expressed in a simple formalism named REC, for which we developed automated translators that convert REC benchmarks into various languages.

In 2016, we corrected a number of benchmarks and added many new ones, to reach a total of 85 benchmarks in December 2016. Among these new benchmarks, one can mention a formalization of arithmetic operations on signed integers, a collection of (8-bit, 16-bit, and 32-bit) binary adders and multipliers, and a complete model of the MAA ("Message Authenticator Algorithm"), a Message Authentication Code used for financial transactions (ISO 8731-2) between 1987 and 2002.

The existing translators (for Haskell, LOTOS, Maude, mCRL, OCAML, Opal, Rascal, Scala, SML-NJ, and Tom) have been enhanced and new translators (for AProVE, Clean, LNT, MLTON, Stratego/XT) have been developed. Tools for automatically extracting and synthesizing performance statistics have also been developed.

6.2. Parallel and Distributed Verification

6.2.1. Distributed State Space Manipulation

Participants: Hubert Garavel, Hugues Evrard, Wendelin Serwe.

For distributed verification, CADP provides the PBG format, which implements the theoretical concept of *Partitioned LTS* [38] and provides a unified access to an LTS distributed over a set of remote machines.

In 2016, the code of the CAESAR_NETWORK_1 library, which is a building block for the distributed verification tools of CADP, has been carefully scrutinized and split into logically-independent modules. Nine problems have been detected and solved, among which a flaw in the distributed termination algorithm: the entire network could freeze if a worker process crashed too early, before the opening of TCP sockets. From now on, a better distributed termination algorithm is used, which supports the coexistence of several networks, ensures that all connections are closed before terminating, and produces more informative traces indicating which worker has triggered termination. Also, the improved CAESAR_NETWORK_1 library checks that all workers operate in directories that are pairwise distinct, mutually disjoint, and different from the working directory of the coordinator process.

6.2.2. Distributed Code Generation for LNT

Participants: Hugues Evrard, Frédéric Lang, Wendelin Serwe.

Rigorous development and prototyping of a distributed algorithm using LNT involves the automatic generation of a distributed implementation. For the latter, a protocol realizing process synchronization is required. As far as possible, this protocol must itself be distributed, so as to avoid the bottleneck that would inevitably arise if a unique process would have to manage all synchronizations in the system. Using a synchronization protocol that we verified formally in 2013, we developed a prototype distributed code generator, named DLC (*Distributed LNT Compiler*), which takes as input the model of a distributed system described as a parallel composition of LNT processes.

In 2016, we improved the user documentation of the DLC distribution, and added support for structured data types, enabling experiments of DLC on the LNT model of the CAESAR_SOLVE_2 library (see § 6.2.3). An overview paper about DLC has been accepted in an international journal [12].

6.2.3. Distributed Resolution of Boolean Equation Systems

Participant: Wendelin Serwe.

The BES_SOLVE tool of CADP enables to solve BESs (*Boolean Equation Systems*) using the various resolution algorithms provided by the CAESAR_SOLVE library (see 5.1), including a distributed on-the-fly resolution algorithm described in pseudo-code in [45].

In 2016, we modeled the pseudo-code of the distributed resolution algorithm in LNT (about $1,000$ lines). For a set of BES examples (encoded as LNT data types and functions), we experimented the generation of the LTS corresponding to the distributed resolution algorithm applied to each BES. We also experimented with the DLC tool [21] to generate a prototype distributed implementation of the resolution algorithm from its LNT specification. These experiments uncovered some errors in the original pseudo-code.

We also simplified the C implementation included in the BES_SOLVE tool to closer match the corrected LNT model, mainly by removing additional synchronization messages. We started to evaluate the simplified implementation using our non-regression test base (more than $15,000$ BESs), with promising results.

6.2.4. *Stability of Communicating Systems*

Participant: Gwen Salaün.

Analyzing systems communicating asynchronously via reliable FIFO buffers is an undecidable problem. A typical approach is to check whether the system is bounded, and if not, the corresponding state space can be made finite by limiting the presence of communication cycles in behavioral models or by imposing an upper bound for the size of communication buffers.

In 2016, our focus was on systems that are likely to be unbounded and therefore result in infinite systems. We did not want to restrict the system by imposing any arbitrary bound. We introduced a notion of stability and proved that once the system is stable for a specific buffer bound, it remains stable whatever larger bounds are chosen for buffers. This enables one to check certain properties on the system for that bound and to ensure that the system will preserve them for arbitrarily larger buffer bounds. We also proved that computing this bound is undecidable but we showed how we succeed in computing these bounds for many examples using heuristics and equivalence checking. These results have been published in an international conference [18].

In collaboration with Carlos Canal (University of Málaga, Spain), we have also shown how the stability approach can be used for composition and adaptation of component-based software. This led to a publication in an international conference [20].

6.2.5. *Debugging of Concurrent Systems*

Participants: Gianluca Barbon, Gwen Salaün.

Model checking is an established technique for automatically verifying that a model satisfies a given temporal property. When the model violates the property, the model checker returns a counterexample, which is a sequence of actions leading to a state where the property is not satisfied. Understanding this counterexample for debugging the specification is a complicated task for several reasons: (i) the counterexample can contain hundreds of actions, (ii) the debugging task is mostly achieved manually, and (iii) the counterexample does not give any clue on the state of the system (e.g., parallelism or data expressions) when the error occurs.

In 2016, we proposed a new approach that improves the usability of model checking by simplifying the comprehension of counterexamples. Our solution aims at keeping only actions in counterexamples that are relevant for debugging purposes. To do so, we first extract in the model all the counterexamples. Second, we define an analysis algorithm that identifies actions that makes the behaviour skip from incorrect to correct behaviours, making these actions relevant from a debugging perspective. Our approach is fully automated by a tool that we implemented and applied on real-world case studies from various application areas for evaluation purposes. A paper presenting these results has been accepted at an international conference.

6.3. Timed, Probabilistic, and Stochastic Extensions

6.3.1. *On-the-fly Model Checking for Extended Regular Probabilistic Operators*

Participant: Radu Mateescu.

In the context of the SENSATION project (see § 8.3.1.1), we study the specification and verification of quantitative properties of concurrent systems, which requires expressive and user-friendly property languages combining temporal, data-handling, and quantitative aspects.

In 2016, in collaboration with José Ignacio Requeno (Univ. Zaragoza, Spain), we aimed at facilitating the quantitative analysis of systems modeled as PTSs (Probabilistic Transition Systems) labeled by actions containing data values and probabilities. We proposed a new regular probabilistic operator that computes the probability measure of a path specified by a generalized regular formula involving arbitrary computations on data values. This operator, which subsumes the Until operators of PCTL (Probabilistic Computation Tree Logic) [41] and their action-based counterparts, can provide useful quantitative information about paths having certain (e.g., peak) cost values. We integrated the regular probabilistic operator into MCL and we devised an associated on-the-fly model checking method, based on a combined local resolution of linear and Boolean equation systems. We implemented the method in a prototype extension of the EVALUATOR model checker and experimented it on realistic PTSs modeling concurrent systems. This work led to a publication [22].

6.4. Component-Based Architectures for On-the-Fly Verification

6.4.1. Compositional Verification

Participant: Frédéric Lang.

The CADP toolbox contains various tools dedicated to compositional verification, among which EXP.OPEN, BCG_MIN, BCG_CMP, and SVL play a central role. EXP.OPEN explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). BCG_MIN and BCG_CMP respectively minimize and compare behavior graphs modulo strong or branching bisimulation and their stochastic extensions. SVL (*Script Verification Language*) is both a high-level language for expressing complex verification scenarios and a compiler dedicated to this language.

In 2016, the *n among m* parallel composition operator "`par`" of the EXP language has been extended. Before the extension, the set of m processes among which any subset of size n could be synchronized on a given action was necessarily the set of all parallel processes composed by the "`par`" operator. From now on, by a slight extension of the syntax, this set of m processes can be defined as a subset of the parallel processes. Also, while n had to be strictly greater than 1, it can now also have value 0 (meaning that none of the m processes can perform the corresponding action) or 1 (meaning that each process can perform the corresponding action on its own, without synchronization). A bug in EXP.OPEN has been fixed and better messages are now emitted to warn the user about dubious usage of the "`par`" operator.

The SVL language has been extended to include the extended "`par`" operator. Two bugs have also been corrected.

6.4.2. Other Component Developments

Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu, Wendelin Serwe.

Sustained effort was made to improve the documentation of the CADP toolbox. Various corrections have been brought to the CADP manual pages. A 27-page manual page explaining how the LOTOS language is implemented has been written, and the manual pages of the CAESAR and CAESAR.ADT compilers have been also updated. To make documentation more readable, the EVALUATOR3, and EVALUATOR4 manual pages have been splitted each in two parts, so as to better distinguish between the languages (namely, MCL3 and MCL) and their model checkers. The CADP distribution has been made leaner by keeping only the essential papers, and the "publications" and "tutorial" pages of the CADP Web site have been enriched and reorganized.

The conventions for string notations to represent "raw" values (i.e., values whose type is not a predefined one, but a custom type defined by the user) have been improved, together with the associated conversion algorithms for reading/writing raw values from/to strings. The BCG_WRITE manual page has been updated to more accurately describe how label fields of type "raw" are parsed. The behaviour of the functions `bcg_character_scan()`, `bcg_string_scan()`, `bcg_real_scan()`, and `bcg_raw_scan()` has been carefully revised, and all the BCG libraries and tools (especially BCG_IO) have been modified to follow the new conventions and emit better diagnostics when label fields contain incorrect notations of raw values. Also, BCG_IO has been enhanced so that very long execution sequences can be converted into SEQ or LOTOS files without causing stack overflow.

Finally, enhancements and bug fixes have been brought to other CADP components, including CADP_MEMORY, EUCALYPTUS, INSTALLATOR, OCIS, RFL, TST, and XTL. The style files for the various editors supported by CADP have been updated to take into account the recent features added to LNT. The predefined MCL libraries of the EVALUATOR model checker have been modified to generate more explanatory diagnostics for the inevitability operators.

Although CADP is mostly used on Linux, specific effort has been made to target other execution platforms. Concerning macOS: CADP now supports the recent versions 10.10 ("Yosemite"), 10.11 ("El Capitan"), and 10.12 ("Sierra"). Concerning Windows: changes have been brought to support Windows 10 and the 64-bit version of Cygwin (previously, only the 32-bit version was supported). Other adaptations were required to handle the recent versions of Cygwin packages, MinGW C compiler, and Mintty shell, as well as the case where Cygwin is not installed in "`C:\`", but in either "`C:\Cygwin`" or "`C:\Cygwin64`".

6.5. Real-Life Applications and Case Studies

6.5.1. Reconfiguration and Resilience of Distributed Cloud Applications

Participants: Umar Ozeer, Gwen Salaün.

In the context of a collaboration with Orange Labs, an Ensimag student (Bakr Derrazi) supervised by Xavier Etchevers and Gwen Salaün, has made his internship from February 2016 to July 2016 at Orange Labs. As a result, we have proposed a first solution and prototype for detecting and repairing failures of data-centric applications distributed in the cloud. A PhD thesis (Umar Ozeer) has started on this subject in November 2016.

6.5.2. Activity Detection in a Smart Home

Participants: Frédéric Lang, Radu Mateescu.

In collaboration with Paula-Andrea Lago-Martinez and Claudia Roncancio (SIGMA team, LIG) and with Nicolas Bonnefond (PERVASIVE INTERACTION team, Inria and LIG), we study how formal methods can help to analyze logs of events obtained from the many sensors and actuators installed in the Amiqual4Home smart home.

In 2016, we considered using the MCL temporal logic to detect the start and end of activities in a log, such as cooking or taking a shower. We applied our tools on a log containing about $140,000$ events that had been generated over 10 days of living in the smart home. This preliminary study has shown that the MCL temporal logic is sufficiently rich to enable an easy specification of the searched activities, notably thanks to its multiple extensions such as macro definitions, parameterized fixed point operators, and data handling mechanisms. The particularly long length of the analyzed logs also enabled us to improve some of the CADP tools, so that they better scale up. This work led to an article submitted to an international conference.

6.5.3. Other Case Studies

Participant: Hubert Garavel.

The demo examples of CADP, which have been progressively accumulated since the origins of the toolbox, are a showcase for the multiple capabilities of CADP, as well as a test bed to assess the new features of the toolbox. In 2016, the effort to maintain and enhance these demos has been pursued. The demo 12 (Message Authentication Algorithm) and demo 31 (SCSI-2 bus arbitration protocol) have been manually translated from LOTOS to LNT. Additionally, demo 12 has been deeply revised by simplifying its LOTOS, LNT, and C code, by taking advantage of the imperative-programming features of LNT, and by enriching the LNT specification with the test cases contained in the original MAA description. This allowed to detect and correct a mistake in the C code implementing function `HIGH_MUL()`. Other CADP demos (namely demos 05, 16, and 36) have also been simplified and/or enhanced in various ways.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Grants with Industry

Participants: Umar Ozeer, Gwen Salaün.

Umar Ozeer is supported by a PhD grant (from November 2016 to November 2019) from Orange Labs (Grenoble) on detecting and repairing failures of data-centric applications distributed in the cloud and the Internet of Things (see § 6.5.1), under the supervision of Xavier Etchevers (Orange Labs), Gwen Salaün (CONVECS), François Gaël Ottogalli (Orange Labs), and Jean-Marc Vincent (POLARIS project-team).

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. ARC6 Programme

Participants: Lina Marsso, Radu Mateescu, Wendelin Serwe.

ARC6 is an academic research community funded by the Auvergne Rhône-Alpes region, whose objective is to foster the scientific collaborations between different academic institutions of the region working in the domain of information and communication technologies. ARC6 organizes various scientific animations (conferences, working groups, summer schools, etc.) and issues a yearly call for PhD and post-doctorate research project proposals.

Lina Marsso is supported by an ARC6 grant (from October 2016 to October 2019) on formal methods for testing networks of programmable logic controllers, under the supervision of Radu Mateescu and Wendelin Serwe (CONVECS), Ioannis Parissis and Christophe Deleuze (LCIS, Valence).

8.2. National Initiatives

8.2.1. FSN (Fonds national pour la Société Numérique)

8.2.1.1. Connexion

Participants: Hubert Garavel [correspondent], Frédéric Lang.

Connexion [5] (*COntrôle commande Nucléaire Numérique pour l'EXport et la rénovatION*) is a project funded by the FSN, within the second call for projects "*Investissements d'Avenir — Briques génériques du logiciel embarqué*". The project, led by EDF and supported by the *Pôles de compétitivité* Minalogic, Systematic, and *Pôle Nucléaire Bourgogne*, involves many industrial and academic partners, namely All4Tech, Alstom Power, ArevA, Atos Worldgrid, CEA-LIST, CNRS/CRAN, Corys Tess, ENS Cachan, Esterel Technologies, Inria, LIG, Predict, and Rolls-Royce. Connexion aims at proposing and validating an innovative architecture dedicated to the design and implementation of control systems for new nuclear power plants in France and abroad.

Connexion started in April 2012 for four years, and was extended for 6 months until September 2016. In this project, CONVECS assisted another LIG team, IIHM, in specifying human-machine interfaces formally using the LNT language and in verifying them using CADP.

8.2.2. Competitivity Clusters

8.2.2.1. Bluesky for I-Automation

Participants: Hugues Evrard, Hubert Garavel, Fatma Jebali, Jingyan Jourdan-Lu, Frédéric Lang, Eric Léo, Radu Mateescu [correspondent].

[5] http://www.cluster-connexion.fr

Bluesky for I-Automation is a project funded by the FUI (*Fonds Unique Interministériel*) within the *Pôle de Compétitivité* Minalogic. The project, led by Crouzet Automatismes (Valence), involves the SMEs (*Small and Medium Enterprises*) Motwin and VerticalM2M, the LCIS laboratory of Grenoble INP, and CONVECS. Bluesky aims at bringing closer the design of automation applications and the Internet of things by providing an integrated solution consisting of hardware, software, and services enabling a distributed, Internet-based design and development of automation systems. The automation systems targeted by the project are networks of programmable logic controllers, which belong to the class of GALS (*Globally Asynchronous, Locally Synchronous*) systems.

Bluesky started in September 2012 for three years and was extended for nine months until June 2016. The main contributions of CONVECS to Bluesky (see § 6.1.6) are the definition of GRL, the formal pivot language for describing the asynchronous behavior of logic controller networks, and the automated verification of the behavior using compositional model checking and equivalence checking techniques.

8.2.3. Other National Collaborations

We had sustained scientific relations with the following researchers:

- Pierre Boullier (Inria, team ALPAGE),
- Pierre-Etienne Moreau (LORIA, team PAREO),
- Fabrice Kordon and Lom Messan Hillah (LIP6, Paris),
- Noël De Palma and Fabienne Boyer (LIG, Grenoble),
- Xavier Etchevers (Orange Labs, Meylan),
- Christophe Deleuze and Ioannis Parissis (LCIS, Valence),
- Pascal Poizat (LIP6, Paris),
- Lina Ye (LRI, Paris).

8.3. European Initiatives

8.3.1. FP7 & H2020 Projects

8.3.1.1. SENSATION

Participants: Hubert Garavel [correspondent], Radu Mateescu, Wendelin Serwe.

SENSATION [6] (*Self ENergy-Supporting Autonomous computaTION*) is a European project no. 318490 funded by the FP7-ICT-11-8 programme. It gathers 9 participants: Inria (ESTASYS and CONVECS project-teams), Aalborg University (Denmark), RWTH Aachen and Saarland University (Germany), University of Twente (The Netherlands), GomSpace (Denmark), and Recore Systems (The Netherlands). The main goal of SENSATION is to increase the scale of systems that are self-supporting by balancing energy harvesting and consumption up to the level of complete products. In order to build such Energy Centric Systems, embedded system designers face the quest for optimal performance within acceptable reliability and tight energy bounds. Programming systems that reconfigure themselves in view of changing tasks, resources, errors, and available energy is a demanding challenge.

SENSATION started on October 1st, 2012 for three years, and has been extended for five months until February 29, 2016. CONVECS contributed to the project regarding the extension of formal languages with quantitative aspects (see § 6.3.1), studying common semantic models for quantitative analysis, and applying formal modeling and analysis to the case studies provided by the industrial partners.

8.3.2. Collaborations with Major European Organizations

The CONVECS project-team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM [7]. H. Garavel and R. Mateescu are members of the FMICS board, H. Garavel being in charge of dissemination actions.

[6] http://sensation-project.eu/
[7] http://fmics.inria.fr

8.4. International Initiatives

H. Garavel is a member of IFIP (*International Federation for Information Processing*) Technical Committee 1 (*Foundations of Computer Science*) Working Group 1.8 on Concurrency Theory chaired successively by Luca Aceto and Jos Baeten.

At Saarland University (Germany), H. Garavel is a guest scientist of the DEPEND research group headed by Holger Hermanns, who received an ERC Advanced Grant ("POWVER") in 2016.

In 2016, we had scientific relations with several universities and companies abroad, including:

- SRI International, California, USA (Steven Eker),
- Technical University of Eindhoven, The Netherlands (Jan Friso Groote),
- University of Málaga, Spain (Francisco Durán and Carlos Canal),
- Aalto University, Finland (Hernan Ponce de Leon),
- Technical University of Graz, Austria (Franz Wotawa),
- University of Zaragoza, Spain (José Ignacio Requeno),
- University of Utah, USA (Chris Myers and Zhen Zhang),
- DiffBlue, Oxford, UK (Matthias Güdemann).

8.5. International Research Visitors

8.5.1. Visits of International Scientists

- Hernan Ponce de Leon (Aalto University, Finland) visited us from February 15 to February 19, 2016.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. Member of the Organizing Committees

- H. Garavel is a member of the model board [8] of MCC (*Model Checking Contest*). In 2016, he helped preparing new models (especially those in the NUPN format) and verified, using the CÆSAR.BDD tool of CADP, the forms describing all benchmark models submitted by the contest participants; this revealed a number of inconsistencies. The mission and activities of the model board are described in a journal paper [15].
- Together with Peter Höfner (Data61, CSIRO, Sydney, Australia), H. Garavel set up a model repository (hosted on the Gforge of Inria) to collect and archive formal models of real systems; this infrastructure is used by the series of MARS workshops [9]. The first model deposited in this repository was W. Serwe's description in LOTOS and LNT of an asynchronous circuit implementing the Data Encryption Standard.
- G. Salaün is member of the steering committee of the SEFM (*International Conference on Software Engineering and Formal Methods*) conference series since 2014.

[8] http://mcc.lip6.fr/models.php
[9] http://www.mars-workshop.org/

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

- R. Mateescu was the tool chair of TACAS'2016 (*22th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*), Eindhoven, The Netherlands, April 2–8, 2016.
- G. Salaün was co-chair of SVT-SAC'2016 (the *Software Verification and Testing* track of the *31st ACM Symposium on Applied Computing*), Pisa, Italy, April 4–8, 2016.

9.1.2.2. Member of the Conference Program Committees

- H. Garavel was program committee member of the 6th FMF (*Forum Methodes Formelles*), Toulouse-Grenoble-Saclay, France, January 26, 2016.
- F. Lang was program committee member of GaM'2016 (*Graphs as Models*), Eindhoven, The Netherlands, April 2–3, 2016.
- G. Salaün was program committee member of SOAP-SAC'2016 (the *Service-Oriented Architecture and Programming* track) of SAC'2016, Pisa, Italy, April 4–8, 2016.
- R. Mateescu was program committee member of SPIN'2016 (*23rd International SPIN Symposium on Model Checking of Software*), Eindhoven, The Netherlands, April 7–8, 2016.
- G. Salaün was program committee member of CIEL'2016 (*5ème Conférence en IngénieriE du Logiciel*), Besançon, France, June 7, 2016.
- G. Salaün was program committee member of COMPSAC'2016 (*40th IEEE International Conference on Computers, Software and Applications*), Atlanta, Georgia, USA, June 10–14, 2016.
- G. Salaün was program committee member of WWV'2016 (*12th International Workshop on Automated Specification and Verification of Web Systems*), Porto, Portugal, June 26, 2016.
- H. Garavel and G. Salaün were program committee members of SEFM'2016 (*14th International Conference on Software Engineering and Formal Methods*), Vienna, Austria, July 4–8, 2016.
- G. Salaün was program committee member of RV'2016 (*16th International Conference on Runtime Verification*), Madrid, Spain, September 23–30, 2016.
- R. Mateescu was program committee member of FMICS-AVoCS'2016 (*International Workshop on Formal Methods for Industrial Critical Systems and Automated Verification of Critical Systems*), Pisa, Italy, September 26–29, 2016.
- H. Garavel was program committee member of HILT'2016 (*Workshop on High Integrity Language Technology*), Pittsburgh, PA, October 6–7, 2016.
- R. Mateescu was program committee member of ICTSS'2016 (*28th International Conference on Testing Software and Systems*), Graz, Austria, October 17–19, 2016.
- G. Salaün was program committee member of FACS'2016 (*13th International Conference on Formal Aspects of Component Software*), Besançon, France, October 19–21, 2016.

9.1.2.3. Reviewer

- G. Barbon was a reviewer for COMPSAC'2016, SVT-SAC'2017, and FSEN'2017 (*7th IPM International Conference on Fundamentals of Software Engineering*), Tehran, Iran, April 26–28, 2017.
- H. Garavel was a reviewer for SPIN'2016.
- F. Lang was a reviewer for FORTE'2016 (*36th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems*), Heraklion, Crete, Greece, June 6–9, 2016.
- R. Mateescu was a reviewer for FACS'2016 and SEFM'2016.
- G. Salaün was a reviewer for DAIS'2016 (*16th IFIP International Conference on Distributed Applications and Interoperable Systems*), Heraklion, Crete, Greece, June 6–9, 2016.

- W. Serwe was a reviewer for SEFM'2016, FMICS-AVoCS'2016, RV'2016, and DATE'2017 (*20th International Conference on Design, Automation and Test in Europe*), Lausanne, Switzerland, March 27–31, 2017.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

- H. Garavel is an editorial board member of STTT (*Springer International Journal on Software Tools for Technology Transfer*).

9.1.3.2. Reviewer - Reviewing Activities

- G. Barbon was a reviewer for JSS (*Journal of Systems and Software*).
- H. Garavel was a reviewer for the Mathematical Reviews (MathSciNet) of the American Mathematical Society.
- F. Lang was a reviewer for STTT.
- R. Mateescu was a reviewer for Acta Informatica and STTT.
- G. Salaün was a reviewer for IJCIS (*International Journal of Cooperative Information Systems*), JLAMP (*Journal of Logic and Algebraic Methods in Programming*), IEEE TSE (*Transactions on Software Engineering*), and TSI (*Technique et Science Informatiques*).
- W. Serwe was a reviewer for SPE (*Journal on Software: Practice and Experience*) and STTT.

9.1.4. Software Dissemination and Internet Visibility

The CONVECS project-team distributes several software tools: the CADP toolbox (see § 5.1), the TRAIAN compiler (see § 5.2), the PIC2LNT translator, the PMC model checker, and the DLC compiler.

In 2016, the main facts are the following:

- We prepared and distributed twelve successive versions (2016-a to 2016-l) of CADP.
- A new version 2.8 of TRAIAN was released on February 29, 2016.
- We were requested to grant CADP licenses for 507 different computers in the world.

The CONVECS Web site [10] was updated with scientific contents, announcements, publications, etc.

By the end of December 2016, the CADP forum [11], opened in 2007 for discussions regarding the CADP toolbox, had over 389 registered users and over 1769 messages had been exchanged.

Also, for the 2016 edition of the Model Checking Contest, we provided four families of models (totalling 62 Nested-Unit Petri Nets) derived from our LNT models. A journal article presenting the achievements of the Model Checking Contest since its origins has been published [15].

Other research teams took advantage of the software components provided by CADP (e.g., the BCG and OPEN/CAESAR environments) to build their own research software. We can mention the following developments:

- An approach for specifying formally the composition of Web services [27]
- The Vercors integrated environment for verifying and running distributed components [43], [46]
- The PN2MC tool for modeling and verifying RTCP-nets [53]
- The Alvis tool for designing hierarchical communication diagrams [54]
- The IDCM tool for designing and integrating complex systems [48], [47]
- The Availability Analyzer tool for software architecture decomposition alternatives [55]

[10] http://convecs.inria.fr
[11] http://cadp.inria.fr/forum.html

Other teams also used the CADP toolbox for various case studies:

- Formal specification and verification of TCP extended with the Window Scale Option [49]
- Performance evaluation of concurrent data structures [56]
- Heuristic search for equivalence checking [31]
- Using formal models to cross check an implementation [52]
- Proving linearizability via branching bisimulation [57]
- Computing maximal weak and other bisimulations [28]
- Verification methodologies for fault-tolerant Network-on-Chip systems [58]

9.1.5. Awards and Distinctions

H. Garavel is an invited professor at Saarland University (Germany) as a holder of the Gay-Lussac Humboldt Prize.

9.1.6. Invited Talks

- H. Garavel gave a talk entitled "*Process Calculi: Towards the Great Unification*" on January 12, 2016 at the Formal Methods seminar of Inria Grenoble.
- R. Mateescu gave a talk entitled "*On-the-Fly Verification for Extended Action-Based Temporal Logics*" on March 3rd, 2016 at IST Graz, Austria.
- H. Garavel was a guest speaker at the 9th German national D-CON meeting that took place in Saarbrücken, Germany, on March 7–8, 2016, where he gave a talk entitled "*Nested-Unit Petri Nets*".
- G. Salaün gave a talk entitled "*Automated Verification of Asynchronously Communicating Systems*" on March 15, 2016 at the LORIA laboratory, Nancy, France.
- H. Garavel gave a talk entitled "*On the Simplest Way to Build Integers from Naturals*" on April 28, 2016 at Saarland University, Germany.
- G. Barbon gave a talk entitled "*Debugging Concurrent Programs using Model Checking and Mining Techniques*" at MOVEP'2016 (*12th Summer School on Modelling and Verification of Parallel Processes*) that took place in Genova, Italy, on June 17–July 1, 2016.
- H. Garavel gave a talk entitled "*On the Most Suitable Axiomatization of Signed Integers Using Free Constructors*" during WADT'2016 (*23rd International Workshop on Algebraic Development Techniques*) that took place in Gregynog, Wales, UK, on September 21–24, 2016.
- H. Garavel gave two talks entitled "*Term Rewrite Systems and the Definition of Signed Integers*" and "*Benchmarking Implementations of Conditional Term Rewrite Systems*" on September 30, 2016 at the LSV laboratory, Cachan, France.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

CONVECS is a host team for MOSIG [12] (*Master of Science in Informatics at Grenoble*), the international master programme in computer science, common to Grenoble INP and Université Grenoble Alpes.

[12] http://mosig.imag.fr

In 2016, we carried out the following teaching activities:

- H. Garavel, together with Laurence Pierre (TIMA, Grenoble), created a new curriculum HECS [13] ("*High-confidence Embedded and Cyberphysical Systems*") for 2nd-year MOSIG students. This curriculum opened for the first time in September 2016.
- F. Lang, R. Mateescu, G. Salaün, and W. Serwe gave lectures on models for concurrency, temporal logics, equivalences, formal languages and verification (36 hours) as part of the MOSIG/HECS-2 course ("*Modeling and Analysis of Concurrent Systems*") led by G. Salaün.
- H. Garavel gave lectures on probabilistic models, stochastic models, and static/dynamic fault trees (6 hours) as part of the MOSIG/HECS-3 course ("*Performance and quantitative properties*") led by Goran Frehse (Verimag).
- H. Garavel gave lectures on the synchronous languages Lustre and SCADE, and on model-driven engineering (7.5 hours) as part of the MOSIG/HECS-4 course ("*Industrial processes for high-confidence design*") led by Laurence Pierre (TIMA).
- G. Salaün was co-responsible of the ISI (*Ingéniérie des Systèmes d'Information*) department of ENSIMAG ("*École Nationale Supérieure d'Informatique et de Mathématiques Appliquées*", Grenoble INP) from September 1, 2011 until August 31st, 2016.
- G. Salaün is Professor at Université Grenoble Alpes since September 1st, 2016, and teaches algorithmics, programming, and Web development at the MMI department of IUT1. He is also headmaster of the SMIN professional licence (L3, 3rd year of university), 192 hours.
- W. Serwe supervised a group of six teams in the context of the "*projet Génie Logiciel*" (55 hours "*équivalent TD*", consisting in 16.5 hours of lectures, plus supervision and evaluation).
- F. Lang gave a lecture on formal methods (9 hours "*équivalent TD*") in the framework of the software engineering course given to the first year students of MOSIG.
- F. Lang gave a lecture on "*Modélisation et vérification de systèmes concurrents et temps-réel*" (27 hours "*équivalent TD*") to the third year computer science engineering students of ENSIMAG.
- G. Barbon gave a lecture on "*Théorie des Langages 1*" at ENSIMAG (18 hours "*équivalent TD*").

9.2.2. Supervision

- Fatma Jebali, "*A Formal Framework for Modelling and Verifying Globally Asynchronous Locally Synchronous Systems*", Université Grenoble Alpes, September 12, 2016, F. Lang and R. Mateescu
- PhD in progress: G. Barbon, "*Debugging Concurrent Programs using Model Checking and Mining Techniques*", Université Grenoble Alpes, since October 2015, G. Salaün and V. Leroy
- PhD in progress: L. Marsso, "*Formal Methods for Testing Networks of Controllers*", Université Grenoble Alpes, since October 2016, R. Mateescu, W. Serwe, I. Parissis, and Ch. Deleuze
- PhD in progress: U. Ozeer, "*Autonomous Resilience of Applications in a Largely Distributed Cloud Environment*", Université Grenoble Alpes, since November 2016, X. Etchevers, G. Salaün, F.-G. Ottogalli, and J.-M. Vincent

9.2.3. Juries

- R. Mateescu was reviewer of Oleksandra Kulankhina's PhD thesis, entitled "*A Framework for Rigorous Development of Distributed Components: Formalisation and Tools*", defended at University of Nice Sophia-Antipolis on October 14, 2016.
- G. Salaün was reviewer of Guillaume Verdier's PhD Thesis, entitled "*Variants of Acceptance Specifications for Modular System Design*", defended at University of Toulouse on March 29, 2016.

[13] http://hecs.imag.fr

9.3. Popularization

H. Garavel participates to the program committee and organization committee of FMF (Formal Methods Forum [14]), a series of industrial conferences on formal methods set up by the competitivity clusters Aerospace Valley and Minalogic, with the support of Inria and many other partners. The 6th FMF conference, devoted to safety engineering, was held on January 26, 2016. The 7th FMF conference, devoted to formal methods and cybersecurity, is scheduled on January 31, 2017.

H. Garavel and R. Mateescu co-operated with Gérard Berry to prepare his lecture on explicit-state enumerative verification given at Collège de France on April 22, 2016.

9.4. Miscellaneous Activities

H. Garavel was a member of the 2016 Inria selection committee for hiring research officers (*chargés de recherche*).

In 2016, H. Garavel was appointed to the Executive Commission in charge of International Relations at COMUE University Grenoble Alpes.

F. Lang is chair of the "*Commission du développement technologique*", which is in charge of selecting R&D projects for Inria Grenoble – Rhône-Alpes.

R. Mateescu is the correspondent of the "*Département des Partenariats Européens*" for Inria Grenoble – Rhône-Alpes.

R. Mateescu is a member of the "*Comité d'orientation scientifique*" for Inria Grenoble – Rhône-Alpes.

R. Mateescu is a member of the "*Bureau*" of the LIG laboratory.

G. Salaün was an elected member of the council of the LIG laboratory until August 31, 2016.

G. Salaün is a member of the Scientific Committee of the PCS action of the PERSYVAL Labex.

W. Serwe is "*chargé de mission*" for the scientific axis *Formal Methods, Models, and Languages* of the LIG laboratory.

W. Serwe is (together with Laurent Lefèvre from the AVALON Inria project-team) correspondent in charge of the 2016 Inria activity reports at Inria Grenoble – Rhône-Alpes.

10. Bibliography

Major publications by the team in recent years

[1] H. GARAVEL, F. LANG, R. MATEESCU, W. SERWE. *CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes*, in "International Journal on Software Tools for Technology Transfer", 2013, vol. 15, n° 2, pp. 89-107 [*DOI :* 10.1007/s10009-012-0244-z], http://hal.inria.fr/hal-00715056

[2] F. LANG, R. MATEESCU. *Partial Model Checking using Networks of Labelled Transition Systems and Boolean Equation Systems*, in "Logical Methods in Computer Science", October 2013, vol. 9, n° 4, pp. 1–32, http://hal.inria.fr/hal-00872181/en

[3] R. MATEESCU, P. POIZAT, G. SALAÜN. *Adaptation of Service Protocols using Process Algebra and On-the-Fly Reduction Techniques*, in "IEEE Transactions on Software Engineering", 2012 [*DOI :* 10.1109/TSE.2011.62], http://hal.inria.fr/hal-00717252

[14] http://projects.laas.fr/IFSE/FMF

[4] R. MATEESCU, W. SERWE. *Model Checking and Performance Evaluation with CADP Illustrated on Shared-Memory Mutual Exclusion Protocols*, in "Science of Computer Programming", February 2012 [*DOI :* 10.1016/J.SCICO.2012.01.003], http://hal.inria.fr/hal-00671321

[5] R. MATEESCU, A. WIJS. *Sequential and Distributed On-the-Fly Computation of Weak Tau-Confluence*, in "Science of Computer Programming", 2012, vol. 77, n° 10–11, pp. 1075–1094, http://hal.inria.fr/hal-00676451/en

[6] R. MATEESCU, A. WIJS. *Property-Dependent Reductions Adequate with Divergence-Sensitive Branching Bisimilarity*, in "Science of Computer Programming", December 2014, vol. 96, n° 3, pp. 354–376

[7] G. SALAÜN, T. BULTAN, N. ROOHI. *Realizability of Choreographies using Process Algebra Encodings*, in "IEEE Transactions on Services Computing", August 2012, vol. 5, n° 3, pp. 290-304, http://hal.inria.fr/hal-00726448

Publications of the year

Articles in International Peer-Reviewed Journals

[8] R. ABID, G. SALAÜN, N. DE PALMA. *Formal Design of Dynamic Reconfiguration Protocol for Cloud Applications*, in "Science of Computer Programming", 2016, vol. 117, pp. 1-16 [*DOI :* 10.1016/J.SCICO.2015.12.001], https://hal.inria.fr/hal-01246152

[9] G. BARBON, M. MARGOLIS, F. PALUMBO, F. RAIMONDI, N. WELDIN. *Taking Arduino to the Internet of Things: the ASIP programming model*, in "Computer Communications", 2016, vol. 00, pp. 1 - 15 [*DOI :* 10.1016/J.COMCOM.2016.03.016], https://hal.inria.fr/hal-01416490

[10] F. DURÁN, G. SALAÜN. *Robust and reliable reconfiguration of cloud applications*, in "Journal of Systems and Software", 2016, vol. 122, pp. 524-537 [*DOI :* 10.1016/J.JSS.2015.09.020], https://hal.inria.fr/hal-01245555

[11] X. ETCHEVERS, G. SALAÜN, F. BOYER, T. COUPAYE, N. DE PALMA. *Reliable Self-deployment of Distributed Cloud Applications*, in "Software: Practice and Experience", 2017, vol. 47, n° 1, pp. 3-20 [*DOI :* 10.1002/SPE.2400], https://hal.inria.fr/hal-01290465

[12] H. EVRARD, F. LANG. *Automatic Distributed Code Generation from Formal Models of Asynchronous Processes Interacting by Multiway Rendezvous*, in "Journal of Logical and Algebraic Methods in Programming", September 2016, https://hal.inria.fr/hal-01412911

[13] M. GÜDEMANN, P. POIZAT, G. SALAÜN, L. YE. *VerChor: A Framework for the Design and Verification of Choreographies*, in "IEEE Transactions on Services Computing", July 2016, vol. 9, n° 4, pp. 647-660 [*DOI :* 10.1109/TSC.2015.2413401], https://hal.archives-ouvertes.fr/hal-01198918

[14] F. JEBALI, F. LANG, R. MATEESCU. *Formal Modelling and Verification of GALS Systems Using GRL and CADP*, in "Formal Aspects of Computing", April 2016, vol. 28, n° 5, pp. 767–804 [*DOI :* 10.1007/S00165-016-0373-3], https://hal.inria.fr/hal-01290449

[15] F. KORDON, H. GARAVEL, L. M. HILLAH, E. PAVIOT-ADET, L. JEZEQUEL, C. RODRÍGUEZ, F. HULIN-HUBARD. *MCC'2015 – The Fifth Model Checking Contest*, in "Transactions on Petri Nets and Other Models

of Concurrency", 2016, vol. 9930, pp. 262-273 [*DOI :* 10.1007/978-3-662-53401-4_12], https://hal.inria.fr/hal-01361274

[16] D. VEKRIS, F. LANG, C. DIMA, R. MATEESCU. *Verification of EB3 specifications using CADP* , in "Formal Aspects of Computing", March 2016, vol. 28, n° 1, pp. 145-178 [*DOI :* 10.1007/s00165-016-0362-6], https://hal.inria.fr/hal-01290460

[17] Z. ZHANG, W. SERWE, J. WU, T. YONEDA, H. ZHENG, C. MYERS. *An improved fault-tolerant routing algorithm for a Network-on-Chip derived with formal analysis*, in "Science of Computer Programming", 2016 [*DOI :* 10.1016/J.SCICO.2016.01.002], https://hal.inria.fr/hal-01261234

International Conferences with Proceedings

[18] L. AKROUN, G. SALAÜN, L. YE. *Automated Analysis of Asynchronously Communicating Systems*, in "23rd International SPIN symposium on Model Checking of Software", Eindhoven, Netherlands, SPIN'2016, Springer Verlag, April 2016 [*DOI :* 10.1007/978-3-319-32582-8_1], https://hal.inria.fr/hal-01280164

[19] G. BARBON, A. CORTESI, P. FERRARA, E. STEFFINLONGO. *DAPA: Degradation-Aware Privacy Analysis of Android Apps*, in "STM 2016 - 12th International Workshop on Security and Trust Management", Heraklion, Greece, September 2016, pp. 32 - 46 [*DOI :* 10.1007/978-3-319-46598-2_3], https://hal.inria.fr/hal-01416504

[20] C. CANAL, G. SALAÜN. *Stability-Based Adaptation of Asynchronously Communicating Software*, in "14th International Conference on Software Engineering and Formal Methods", Vienne, Austria, July 2016 [*DOI :* 10.1007/978-3-319-41591-8_22], https://hal.inria.fr/hal-01359044

[21] H. EVRARD. *DLC: Compiling a Concurrent System Formal Specification to a Distributed Implementation*, in "TACAS'2016", Eindhoven, Netherlands, 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2016, Springer-Verlag, April 2016 [*DOI :* 10.1007/978-3-662-49674-9_34], https://hal.inria.fr/hal-01250925

[22] R. MATEESCU, J. I. REQUENO. *On-the-Fly Model Checking for Extended Action-Based Probabilistic Operators*, in "23rd International SPIN symposium on Model Checking of Software", Eindhoven, Netherlands, S. VERLAG (editor), SPIN'2016, April 2016 [*DOI :* 10.1007/978-3-319-32582-8_13], https://hal.inria.fr/hal-01280129

Conferences without Proceedings

[23] P. POIZAT, G. SALAÜN, A. KRISHNA. *Checking Business Process Evolution*, in "13th International Conference on Formal Aspects of Component Software (FACS)", Besançon, France, October 2016, https://hal.inria.fr/hal-01366641

Books or Proceedings Editing

[24] D. GIANNAKOPOULOU, G. SALAÜN, M. BUTLER (editors). *Special Issue of the Formal Aspects of Computing Journal on Software Engineering and Formal Methods (SEFM'14)*, Springer, April 2016, https://hal.inria.fr/hal-01419890

[25] F. LANG, F. FLAMMINI (editors). *Preface to the Special issue on Formal Methods for Industrial Critical Systems (FMICS'2014)*, Special Issue on Formal Methods for Industrial Critical Systems (FMICS'2014), Elsevier, March 2016, vol. 118 [*DOI :* 10.1016/J.SCICO.2016.01.004], https://hal.inria.fr/hal-01271895

[26] G. SALAÜN, M. STOELINGA (editors). *Preface: Special Issue on Software Verification and Testing (Selected Papers from SAC-SVT'15)*, ACM, December 2016, https://hal.inria.fr/hal-01419302

References in notes

[27] N. ADADI, M. BERRADA, D. CHENOUNI. *Formal Specification of Web Services Composition Using LOTOS*, in "International Journal of Computer Technology & Applications", September 2016, vol. 7, n^{o} 5, pp. 636–642

[28] A. BOULGAKOV, T. GIBSON-ROBINSON, A. W. ROSCOE. *Computing Maximal Weak and Other Bisimulations*, in "Formal Aspects of Computing", May 2016, vol. 28, n^{o} 3, pp. 381–407

[29] D. CHAMPELOVIER, X. CLERC, H. GARAVEL, Y. GUERTE, C. MCKINTY, V. POWAZNY, F. LANG, W. SERWE, G. SMEDING. *Reference Manual of the LOTOS NT to LOTOS Translator (Version 5.7)*, November 2012, Inria/VASY, 153 pages

[30] E. M. CLARKE, E. A. EMERSON, A. P. SISTLA. *Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications*, in "ACM Transactions on Programming Languages and Systems", April 1986, vol. 8, n^{o} 2, pp. 244–263

[31] N. DE FRANCESCO, G. LETTIERI, A. SANTONE, G. VAGLINI. *Heuristic Search for Equivalence Checking*, in "Software & Systems Modeling", May 2016, vol. 15, n^{o} 2, pp. 513–530

[32] R. DE NICOLA, F. W. VAANDRAGER. *Action versus State Based Logics for Transition Systems*, Lecture Notes in Computer Science, Springer Verlag, 1990, vol. 469, pp. 407–419

[33] H. GARAVEL. *Nested-Unit Petri Nets: A Structural Means to Increase Efficiency and Scalability of Verification on Elementary Nets*, in "Proceedings of the 36th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS'15), Brussels, Belgium", R. R. DEVILLERS, A. VALMARI (editors), Lecture Notes in Computer Science, Springer Verlag, June 2015, vol. 9115, pp. 179–199

[34] H. GARAVEL. *Compilation of LOTOS Abstract Data Types*, in "Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)", S. T. VUONG (editor), North Holland, December 1989, pp. 147–162

[35] H. GARAVEL. *OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing*, in "Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)", Berlin, B. STEFFEN (editor), Lecture Notes in Computer Science, Springer Verlag, March 1998, vol. 1384, pp. 68–84, Full version available as Inria Research Report RR-3352

[36] H. GARAVEL, F. LANG. *SVL: a Scripting Language for Compositional Verification*, in "Proceedings of the 21st IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2001 (Cheju Island, Korea)", M. KIM, B. CHIN, S. KANG, D. LEE (editors), Kluwer Academic Publishers, August 2001, pp. 377–392, Full version available as Inria Research Report RR-4223

[37] H. GARAVEL, F. LANG, R. MATEESCU. *Compiler Construction using LOTOS NT*, in "Proceedings of the 11th International Conference on Compiler Construction CC 2002 (Grenoble, France)", N. HORSPOOL (editor), Lecture Notes in Computer Science, Springer Verlag, April 2002, vol. 2304, pp. 9–13

[38] H. GARAVEL, R. MATEESCU, I. SMARANDACHE-STURM. *Parallel State Space Construction for Model-Checking*, in "Proceedings of the 8th International SPIN Workshop on Model Checking of Software SPIN'2001 (Toronto, Canada)", Berlin, M. B. DWYER (editor), Lecture Notes in Computer Science, Springer Verlag, May 2001, vol. 2057, pp. 217–234, Revised version available as Inria Research Report RR-4341 (December 2001)

[39] H. GARAVEL, W. SERWE. *State Space Reduction for Process Algebra Specifications*, in "Theoretical Computer Science", February 2006, vol. 351, n^{o} 2, pp. 131–145

[40] H. GARAVEL, J. SIFAKIS. *Compilation and Verification of LOTOS Specifications*, in "Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)", L. LOGRIPPO, R. L. PROBERT, H. URAL (editors), North Holland, June 1990, pp. 379–394

[41] H. HANSSON, B. JONSSON. *A Logic for Reasoning about Time and Reliability*, in "Formal Aspects of Computing", 1994, vol. 6, n^{o} 5, pp. 512–535

[42] M. HENNESSY, R. MILNER. *Algebraic Laws for Nondeterminism and Concurrency*, in "Journal of the ACM", 1985, vol. 32, pp. 137–161

[43] L. HENRIO, O. KULANKHINA, S. LI, E. MADELAINE. *Integrated Environment for Verifying and Running Distributed Components*, in "Proceedings of the 19th International Conference on Fundamental Approaches to Software Engineering FASE'2016 (Eindhoven, The Netherlands)", P. STEVENS, A. WĄSOWSKI (editors), Lecture Notes in Computer Science, Springer, April 2016, vol. 9633, pp. 66–83

[44] F. JEBALI. *Formal Framework for Modelling and Verifying Globally Asynchronous Locally Synchronous Systems*, Université Grenoble Alpes, September 2016

[45] C. JOUBERT. *Vérification distribuée à la volée de grands espaces d'états*, Institut National Polytechnique de Grenoble, December 2005

[46] O. KULANKHINA. *A Framework for Rigorous Development of Distributed Components: Formalisation and Tools*, Université Nice Sophia-Antipolis, October 2016

[47] T. LAMBOLAIS, A.-L. COURBIS, H.-V. LUONG, C. PERCEBOIS. *IDF: A Framework for the Incremental Development and Conformance Verification of UML Active Primitive Components*, in "Journal of Systems and Software", March 2016, vol. 113, pp. 275 –295

[48] T. LAMBOLAIS, A.-L. COURBIS, H.-V. LUONG, T.-L. PHAN. *Designing and Integrating Complex Systems: Be Agile Through Liveness Verification and Abstraction*, G. AUVRAY, J.-C. BOCQUET, E. BONJOUR, D. KROB (editors), Springer International Publishing, 2016, pp. 69–81

[49] L. LOCKEFEER, D. M. WILLIAMS, W. FOKKINK. *Formal Specification and Verification of TCP Extended with the Window Scale Option*, in "Science of Computer Programming", March 2016, vol. 118, pp. 3–23

[50] J. MAGEE, J. KRAMER. *Concurrency: State Models and Java Programs*, 2006, Wiley, April 2006

[51] R. MATEESCU, D. THIVOLLE. *A Model Checking Language for Concurrent Value-Passing Systems*, in "Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland)", J. CUELLAR, T. MAIBAUM, K. SERE (editors), Lecture Notes in Computer Science, Springer Verlag, May 2008, vol. 5014, pp. 148–164

[52] R. OLIVEIRA, S. DUPUY-CHESSA, G. CALVARY, D. DADOLLE. *Using Formal Models to Cross Check an Implementation*, in "Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems EICS'2016 (Brussels, Belgium)", ACM, June 2016, pp. 126–137

[53] M. SZPYRKA, J. BIERNACKI, A. BIERNACKA. *Tools and Methods for RTCP-Nets Modeling and Verification*, in "Archives of Control Sciences", September 2016, vol. 26, n^{o} 3, pp. 339–365

[54] M. SZPYRKA, P. MATYASIK, J. BIERNACKI, A. BIERNACKA, M. WYPYCH, L. KOTULSKI. *Hierarchical Communication Diagrams*, in "Computing and Informatics", 2016, vol. 35, n^{o} 1, pp. 55–83

[55] H. SÖZER, M. STOELINGA, H. BOUDALI, M. AKŞIT. *Availability Analysis of Software Architecture Decomposition Alternatives for Local Recovery*, in "Software Quality Journal", May 2016, pp. 1–27

[56] H. WU, X. YANG, J.-P. KATOEN. *Performance Evaluation of Concurrent Data Structures*, M. FRÄNZLE, D. KAPUR, N. ZHAN (editors), Springer, 2016, pp. 38–49

[57] X. YANG, J. KATOEN, H. LIN, H. WU. *Proving Linearizability via Branching Bisimulation*, in "CoRR", 2016, vol. abs/1609.07546

[58] Z. ZHANG. *Verification Methodologies for Fault-Tolerant Network-on-Chip Systems*, University of Utah, USA, May 2016