IN PARTNERSHIP WITH:
Institut polytechnique de Grenoble

Université Joseph Fourier (Grenoble)

Activity Report 2015

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. - Verification
- 2.5. - Software engineering
- 5.1.1. - Engineering of interactive systems
- 7.1. - Parallel and distributed algorithms
- 7.11. - Performance evaluation
- 7.4. - Logic in Computer Science

Other Research Topics and Application Domains:
- 5.2.4. - Aerospace
- 6.3.2. - Network protocols
- 6.6. - Embedded systems
- 9.4.1. - Computer science

1. Members

Research Scientists

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

Faculty Member

Gwen Salaün [Grenoble INP, Associate Professor, HdR]

Engineers

Soraya Arias [Inria, Engineer]
Jingyan Jourdan-Lu [Inria, granted by Conseil Régional Rhône-Alpes]
Eric Léo [Inria, granted by Conseil Régional Rhône-Alpes]

PhD Students

Raquel Oliveira [Univ. Grenoble I, until Sep 2015]
Gianluca Barbon [ENSIMAG, from Oct 2015]
Hugues Evrard [Inria, until Jul 2015, granted by Caisse des Dépôts et Consignations]
Fatma Jebali [Inria, granted by Conseil Régional Rhône-Alpes]
Abderahman Kriouile [Inria, until Jul 2015]
Rim Sakka Abid [Inria]

Post-Doctoral Fellows

Lakhdar Akroun [Inria]

José Ignacio Requeno [Inria]

Administrative Assistants

Isabelle Allegre [Inria, until Jun 2015]

Myriam Etienne [Inria]

Others

Matthias Güdemann [Systerel/freelance]

Imad-Seddick Arrada [Inria, Intern, from Jun 2015 until Jul 2015]

Remy Delanaux [ENSIMAG, Intern, from Feb 2015 until May 2015]

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 [36] 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*) [57], 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 [43], 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 [42] 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 [47], [46] 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 [43] 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.

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

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,
  - 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 [51], CTL [39], ACTL [41], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [45] 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 [56] 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 [44] 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 PMC Partial Model Checker

Participants: Radu Mateescu, Frédéric Lang.

We develop a tool named PMC (*Partial Model Checker*, see § 6.4), which performs the compositional model checking of dataless MCL formulas on networks of communicating automata described in the EXP language.

PMC can be freely downloaded from the CONVECS Web site [2].

6. New Results

6.1. New Formal Languages and their Implementations

6.1.1. *Definition of LNT*

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.

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

In 2015, the theoretical foundations of LNT have been explored in a journal article [14] that, after examining the various ways sequential composition is handled in mainstream value-passing process calculi, shows that these various approaches are subsumed by the LNT approach, which is easier to learn and leads to more readable and more concise specifications.

The LNT language has also been enhanced in several aspects:

- The "`case`" construct now supports multiple (tuple-like) expressions and patterns.
- Two new parameter-passing modes "`in var`" and "`out var`" have been introduced to allow finer data-flow analyses.
- Exceptions are better handled and a new "`assert`" statement was added to LNT.
- The "`none`" channel is now implicitly predefined.
- Finally, the LNT reference manual has been extended and updated at many places.

6.1.2. Translation from LNT to LOTOS

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

In 2015, the translator from LNT to LOTOS was further improved. In addition to 22 bug fixes and improved error messages, the following enhancements have been brought:

- The "`-root`" option of LNT2LOTOS now accepts value parameters for LNT processes and supports gate parameters in named style. It also accepts the name of a process not present in the current module.
- Negative number constants of the form "-2^{k-1}", where integer numbers are represented using k bits, are now supported.
- Better warning messages are emitted for "`in`" and "`in out`" (formerly "`inout`") parameters.

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 2015, in addition to a few bug fixes, the "`-root`" option of the CAESAR compiler has been generalized to support processes having value parameters; this makes compositional verification easier by removing the need for introducing extra wrapper processes having no value parameters. The EXEC/CAESAR interface has been extended with two new primitives "`CAESAR_KERNEL_DELAY`" and "`CAESAR_KERNEL_EXIT()`". Also, optimizations have been introduced to generate shorter and simpler C code, and to make sure that this C code compiles without spurious warnings.

A systematic comparison between CAESAR.ADT and available interpreters/compilers for other languages that support rewrite rules or pattern matching has been undertaken. This comparison reuses the benchmarks developed for the three Rewrite Engine Competitions (2006, 2009, and 2010). As a preliminary step, we developed a tenth translators from the REC formalism in which these benchmarks are written to languages such as Haskell, LOTOS, Maude, mCRL, OCAML, Opal, Rascal, Scala, and Tom.

6.1.4. NUPN

Participants: Hubert Garavel, Frédéric Lang.

The CAESAR.BDD tool that analyzes NUPN (*Nested-Unit Petri Nets*) models and serves to prepare the yearly Model Checking Contest [3] has been enhanced in several ways. In addition to 7 bug fixes, 14 new command-line options have been added to CAESAR.BDD ("`-arcs`", "`-bits`", "`-creator`", "`-density`", "`-encodings`", "`-height`", "`-hwb`", "`-multiple-arcs`", "`-multiple-initial-tokens`", "`-places`", "`-redundant-units`", "`-transitions`", "`-units`", and "`-width`"). The output format produced by the "`-exclusive-places`" option has been revised. The "`-mcc`" option now computes the extended free choice property. A new option "`-network nupn`" was also added to EXP.OPEN to produce NUPN models from automata networks.

Particular efforts have been put to increase the scalability of CAESAR.BDD for large models. Reading large NUPN files was made much faster. The "`-exclusive-places`" option of CAESAR.BDD was made faster too. The size of the largest data structure allocated by CAESAR.BDD, has been divided by four. CAESAR.BDD has also been optimized to save memory when handling NUPN models having a simple hierarchical structure. Finally, user-specified timeouts are better supported.

A conference article was published [24], which formally defines the NUPN model and investigates its mathematical properties. Additionally, the assembly of a collection of large NUPN models was undertaken, and various prototype tools to handle NUPN models ("nupn_pack", "nupn_reduce", and "nupn_merge") have been developed.

6.1.5. Translation from GRL to LNT

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

In the context of the Bluesky project (see § 8.1.2.1), 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) makes possible the on-the-fly exploration of the LTS underlying a GRL specification using CADP.

In 2015, we have revised the GRL syntax to make GRL easier to learn and to understand. Our data base of examples has been updated to take those changes into account. We have also added some language features, such as named constants, and a dedicated construct called *activation signal* to define constraints on the asynchronous activation of blocks. This new construct is easier to use than the previous solution based on ad-hoc data signals, and semantically more elegant as it avoids unexpected deadlocks. Activation signals permit realistic situations such as halting, priorities, scenarios, and pace relations between synchronous components to be modeled at a suitable level of abstraction. They can be smoothly translated into LNT without affecting the rest of the translation.

As regards the specification of properties, to reduce the complexity of using full-fledged temporal logics, we have also proposed a property specification language dedicated to GALS systems, based upon a set of temporal logic patterns, which capture frequently encountered behaviours, encompassing both time-critical and untimed aspects of GALS systems. Those patterns include deadlock, livelock, safety, liveness, and fairness properties. The semantics of the proposed patterns have been defined by translation into the MCL language.

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

As regards the GRL2LNT tool, nine successive versions have been released, to take into account the syntactic changes in the GRL language, to correct about 20 bugs, to eliminate compilation warnings, and to implement the following new features:

- The generated LNT code has been corrected so as to eliminate compilation warnings and to take into account recent changes in the syntax of LNT (see § 6.1.1).
- GRL system specifications can now be parameterized with data values and instantiated using the new "`-root`" option of GRL2LNT.
- The interface between GRL and external C code has been revised in two ways: (1) external blocks with several outputs are now mapped to a single external function instead of one function per output previously, and (2) conversion functions between GRL and C numeric types have been defined, handling runtime verification of overflows. Those conversion functions have been packaged in a new code library, which is automatically included by GRL2LNT.
- Several verifications on the usage of signals and communication channels have been implemented, leading either to error messages, or to warnings corresponding to potential errors. About 20 new error messages and 10 new warnings have been added.

In addition, three manual pages have been written to document respectively the GRL language, the GRL2LNT translator tool, and the GRL.OPEN shell script. The GRL non-regression test base has been extended and now contains 230 correct examples and 400 incorrect examples.

An article describing the GRL language and its associated tools has been submitted to an international journal.

6.1.6. *Translation from BPMN to LNT*

Participant: Gwen Salaün.

Business processes support the modeling and the implementation of software as workflows of local and inter-process activities. Taking over structuring and composition, evolution has become a central concern in software development. We believe this should be taken into account as soon as the modeling of business processes, which can thereafter be made executable using process engines or model-to-code transformations. 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.

To reach this objective, we developed, in collaboration with Pascal Poizat (LIP6, Paris), a model transformation from the BPMN standard notation to the LNT process algebra. We then proposed a set of relations for comparing business processes at the formal model level. With reference to related work, we proposed a richer set of comparison primitives supporting renaming, refinement, property- and context-awareness. Thanks to the implementation of a tool that integrates with the Eclipse IDE and behind-the-scene interaction with the CADP verification toolbox, we put the checking of evolution within the reach of business process designers. Our approach is fully automated and has been applied for evaluation to a large set of BPMN processes.

6.1.7. *Other Language Developments*

Participants: Hugues Evrard, Hubert Garavel, Frédéric Lang, Eric Léo, Wendelin Serwe.

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.

In 2015, the connection to external software development tools has progressed. The support of the LOTOS and LNT languages in the Emacs/XEmacs, jEdit, and Vim editors has improved. More text editors are now supported, including Nano, Notepad++, and all the text editors compliant with GtkSourceView 3.0 (including the Gedit editor of Gnome). Pretty-printers such as a2ps and the LaTeX "listings" package are also supported. Configuration files for three CADP languages (MCL, SVL, and XTL) and three CADP formats (BES, NUPN, and RBC) have been added.

Also, with the help of its principal author Pierre Boullier (Inria, Alpage), we corrected a memory allocation bug in the SYNTAX parser generator, which is used in most of the compilers developed by the CONVECS team.

6.2. Parallel and Distributed Verification

6.2.1. Distributed Code Generation for LNT

Participants: Hugues Evrard, Frédéric Lang.

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. A particularity of such a protocol is its ability to support branching synchronizations, corresponding to situations where a process may offer a choice of synchronizing actions (which themselves may nondeterministically involve several sets of synchronizing processes) instead of a single one. Therefore, a classical barrier protocol is not sufficient and a more elaborate synchronization protocol is needed.

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 2015, we finalized the development of DLC: the code was cleaned and the different compiler components were better integrated. A new option was added for the generated executables to dump at runtime an execution trace in the SEQUENCE format of CADP, for further analysis. A complete description of DLC, its synchronization protocol, performance data and usage examples were presented in Hugues Evrard's PhD thesis [9], defended in July 2015. An overview of DLC was presented in an international conference paper [23], and an extended version has been prepared for a journal special issue currently under construction. A tool paper was accepted in an international conference that will take place in 2016 [22].

6.2.2. Verification of Asynchronously Communicating Systems

Participants: Lakhdar Akroun, 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, whether the corresponding state space can be made finite by limiting the presence of communication cycles in behavioral models or by fixing the buffer size. In this work, our focus is on systems that are likely to be unbounded and therefore result in infinite systems. We do 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 whatever larger bounds are used for buffers. We also proved that computing this bound is undecidable but we showed how we can succeed in computing these bounds for many typical examples using heuristics and equivalence checking.

6.2.3. Analysis of Verification Counterexamples

Participants: Gianluca Barbon, Gwen Salaün.

Model checking is an established technique for automatically verifying that a model, e.g., a Labelled Transition System (LTS), obtained from higher-level specification languages (such as process algebras) satisfies a given temporal property, e.g., the absence of deadlocks. 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 (even thousands) 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 collaboration with the SLIDE team of the LIG laboratory, we work on new solutions for simplifying the comprehension of counterexamples and thus favouring usability of model checking techniques. To do so, we apply pattern mining techniques to a set of correct traces (extracted from the LTS) and incorrect traces (corresponding to counterexamples), to identify specific patterns indicating more precisely the source of the problem.

6.3. Timed, Probabilistic, and Stochastic Extensions

6.3.1. Model Checking for Extended PCTL

Participants: Radu Mateescu, José Ignacio Requeno.

In the context of the SENSATION project (see § 8.2.1.1), we study the specification and verification of quantitative properties of concurrent systems.

In 2015, we developed a probabilistic version of ACTL (Action-based CTL) [41], named PACTL. This logic represents an action-based counterpart for PCTL (*Probabilistic Computation Tree Logic*) [50] and is interpreted naturally over DTMCs with labeled transitions, such as those produced from IPCs (*Interactive Probabilistic Chains*) [40]. The PACTL operators generalize those of ACTL: they characterize sequences of transitions in the DTMC by specifying both the states and the actions labeling the transitions. We implemented PACTL as an XTL library, which allows the designer to combine properties on actions, data, probabilities, and discrete time. We have experimented the PACTL library on several DTMCs imported from the probabilistic model checker PRISM [55] to ensure that both implementations provide the same numerical results.

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

6.4.1. Compositional Verification

Participants: Hubert Garavel, 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 2015, we corrected one bug in BCG_CMP and eight bugs in SVL. We extended the SVL language and compiler as follows:

- A new statement was added to translate a LOTOS file or a process in a LOTOS file to an LNT file automatically.
- LNT processes with data parameters can now be instantiated directly in the SVL script, without requiring a parameterless intermediate process to be defined.
- LNT processes with gate parameters can now be instantiated in the SVL script using the named parameter-passing style of LNT.
- Specification of a diagnostic file is now optional in the "`comparison`", "`deadlock`", and "`live-lock`" statements of SVL.
- The "`property`" statement has been extended so that it can now contain any kind of statement, provided it contains at least one verification statement.
- Within SVL properties, it is now possible to define shell lines followed by an "`expected`" clause to specify the expected result of the shell line.
- It is now possible to add a "`result`" clause after a verification statement, so as to store the result of the verification in a shell variable that can be subsequently used in the SVL script.

We improved several demo examples of CADP by using these new SVL constructs, and we added a new demo example on the verification of an airplane-ground communication protocol.

We also improved the PMC tool, by correcting five bugs and adding a new "`-order`" option, which permits the user to define a particular order for quotienting. We improved the presentation of the demo examples released in the PMC distribution. Those examples are now given in LNT and translated automatically into networks of automata in the EXP language, instead of being given directly as networks of automata.

6.4.2. On-the-Fly Test Generation

Participants: Hubert Garavel, Radu Mateescu, Wendelin Serwe.

In the context of the collaboration with STMicroelectronics, we study techniques for testing if a (hardware) implementation is conform to a formal model described in LNT. Our approach is inspired by the theory of conformance testing [63], as implemented for instance in TGV [53] and JTorX [33]. We have developed three prototype tools to support this approach. The first tool implements a dedicated OPEN/CAESAR-compliant compiler for the particular asymmetric synchronous product between the model and the test purpose. The second tool, based on slightly extended generic components for graph manipulation (τ-compression, τ-confluence reduction, determinization) and resolution of Boolean equation systems, generates the complete test graph (CTG), which can be used to extract concrete test cases or to drive the test of the implementation. A third prototype tool takes as input a CTG and extracts either a single test case (randomly chosen or the first encountered one), or the set of *all* test cases. The principal advantage of our approach compared to existing tools is the use of LNT for describing test purposes, which facilitates the manipulation of data values.

In 2015, we corrected the prototype tools to properly handle timers and failure transitions, improved the documentation, and simplified internal data structures.

These prototype tools were used in the case study with STMicroelectronics (see § 6.5.1) and the EnergyBus (see § 6.5.4).

6.4.3. Other Component Developments

Participants: Soraya Arias, Hubert Garavel, Frédéric Lang, Radu Mateescu.

We separated the MCL library defining the operators of ACTL (Action-based CTL) [41] in two parts: the first one defines the operators of ACTL$\setminus$X (the fragment of ACTL without the next-time operators), including optimized definitions of derived temporal operators, and the second one defines the next-time operators, including the definitions of silent next-time operators, which complement the visible next-time operators already present in the library.

We also added an MCL library defining the operators of the Lμ-dsbr fragment of modal μ-calculus [6], which includes the ACTL$\setminus$X library. The Lμ-dsbr library also defines the absence of deadlock property as an MCL formula adequate w.r.t. divergence-sensitive branching bisimulation (divbranching for short) and allowing one to hide all visible actions in the LTS and to reduce it modulo divbranching prior to verification, which may bring significant performance gains.

A new major version 1.2 of the BCG format for storing Labelled Transition Systems was released as part of CADP 2015-a. Following this change, various minor residual bugs have been identified and fixed in 2015, and the type system of XTL has been modified to require fewer explicit type coercions.

In addition to bug fixes in various tools (e.g., CUNCTATOR, EUCALYPTUS, TST, XTL, etc.), the installation procedures of CADP have been revisited and updated; in particular, work is going on and many preliminary changes have been silently brought to ease installation of CADP on Windows.

6.5. Real-Life Applications and Case Studies

6.5.1. ACE Cache Coherency Protocol

Participants: Abderahman Kriouile, Radu Mateescu, Wendelin Serwe.

In the context of a CIFRE convention with STMicroelectronics, we studied system-level cache coherency, a major challenge faced in the current System-on-Chip architectures. Because of their increasing complexity (mainly due to the significant number of computing units), the validation effort using current simulation-based techniques grows exponentially. As an alternative, we study formal verification.

We focused on the ACE (AXI Coherency Extensions) cache coherency protocol, a system-level coherency protocol proposed by ARM [31]. In previous years, we developed a parametric formal model (about $3,400$ lines of LNT) of a system consisting of an ACE-based cache coherent interconnect, processors, and a main memory. We also specified temporal properties expressing cache coherence, data integrity, and successful completion of each transaction. Note that the former property required to transform state-based properties into action-based ones, by adding information about the cache state to the actions executed by the cache.

In 2015, we continued to exploit the formal model to improve the validation of the architecture under design at STMicroelectronics, in particular by integrating tests derived from the formal model into the official test plans. This work led to a publication in an international conference [25], and the defense of the PhD corresponding to the CIFRE convention [10].

6.5.2. Deployment and Reconfiguration Protocols for Cloud Applications

Participants: Rim Sakka Abid, Gwen Salaün.

Cloud applications are complex applications composed of a set of interconnected software components running on different virtual machines, hosted on remote physical servers. Deploying and reconfiguring this kind of applications are very complicated tasks especially when one or multiple virtual machines fail when achieving these tasks. Hence, there is a need for protocols that can dynamically reconfigure and manage running distributed applications.

In 2015, we proposed a novel protocol, which aims at reconfiguring cloud applications. This protocol is able to ensure communication between virtual machines and resolve dependencies by exchanging messages, (dis)connecting, and starting/stopping components in a specific order. The interaction between machines is assured via a publish-subscribe messaging system. Each machine reconfigures itself in a decentralized way. The protocol supports virtual machine failures, and the reconfiguration always terminates successfully even in the presence of a finite number of failures. Due to the high degree of parallelism inherent to these applications, the protocol was specified in LNT and verified using CADP. The use of formal specification languages and tools helped to detect several bugs and to improve the protocol. These results were published in [12].

Another line of work concerns autonomic computing in cloud environments. Managing distributed cloud applications is a challenging problem because manual administration is no longer realistic for these complex distributed systems. Thus, autonomic computing is a promising solution for monitoring and updating these applications automatically. This is achieved through the automation of administration functions and the use of control loops called autonomic managers. An autonomic manager observes the environment, detects changes, and reconfigures dynamically the application. Multiple autonomic managers can be deployed in the same system and must make consistent decisions. Using them without coordination may lead to inconsistencies and error-prone situations.

In 2015, we propose an approach for coordinating stateful autonomic managers, which relies on a simple coordination language, new techniques for asynchronous controller synthesis and Java code generation. We used our approach for coordinating real-world cloud applications. These results were published in [19].

6.5.3. Networks of Programmable Logic Controllers

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

In the context of the Bluesky project (see § 8.1.2.1), we study the software applications embedded on the PLCs (Programmable Logic Controllers) manufactured by Crouzet Automatismes. One of the objectives of Bluesky is to enable the rigorous design of complex control applications running on several PLCs connected by a network. Such applications are instances of GALS (*Globally Asynchronous, Locally Synchronous*) systems composed of several synchronous automata embedded on individual PLCs, which interact asynchronously by exchanging messages. A formal analysis of these systems can be naturally achieved by using the formal languages and verification techniques developed in the field of asynchronous concurrency.

For describing the applications embedded on individual PLCs, Crouzet provides a dataflow language with graphical syntax and synchronous semantics, equipped with an ergonomic user-interface that facilitates the learning and use of the language by non-experts. To equip the PLC language of Crouzet with functionalities for automated verification, the solution adopted in Bluesky was to translate it into GRL (see § 6.1.5), which enables the connection to testing and verification tools covering the synchronous and asynchronous aspects.

In 2015, we have provided support to Crouzet, who started to integrate GRL in the PLC design process by developing both a library of GRL blocks corresponding to function blocks present in their PLC programming tool, and an automated translation from a PLC program into a GRL block. The GRL2LNT and GRL.OPEN tools (see § 6.1.5) provide a direct connection to all verification functionalities of CADP, in particular model checking and equivalence checking.

We also investigated the equivalence checking for networks of PLCs, with the objective of proposing a general methodology usable in industrial context. We identified several rules (formalized as templates) for describing the asynchronous and synchronous parts of PLC networks, as well as their external behaviour (service), in order to facilitate the equivalence checking modulo branching bisimulation.

6.5.4. EnergyBus Standard for Connecting Electric Components

Participants: Hubert Garavel, Wendelin Serwe.

The EnergyBus [4] is an upcoming industrial standard for electric power transmission and management, based on the CANopen field bus. It is developed by a consortium assembling all major industrial players (such as Bosch, Panasonic, and Emtas) in the area of light electric vehicles (LEV); their intention is to ensure interoperability between all electric LEV components. At the core of this initiative is a universal plug integrating a CAN-Bus [5] with switchable power lines. The central and innovative role of the EnergyBus is to manage the safe electricity access and distribution inside an EnergyBus network.

In the framework of the European FP7 project SENSATION (see § 8.2.1.1) a formal specification in LNT of the main EnergyBus protocols is being developed by Alexander Graf-Brill and Holger Hermanns at Saarland University [48], with the active collaboration of CONVECS.

In 2015, we pursued the analysis of the LNT model, involving both verification (by means of state-space exploration and model checking techniques) and validation (using test cases automatically derived from the formal LNT model).

6.5.5. AutoFlight Control System

Participant: Fatma Jebali.

In collaboration with Eric Jenn (IRT Saint Exupery, Toulouse), we studied an AutoFlight Control System (AFCS), provided by Thales Avionics. The goal of an AFCS is to improve the quality of flight and enhance the operational capability of the aircraft. The architecture of the AFCS comprises two parts. The first part (FCP, Flight Control Panel) consists of a control panel, which enables the pilot to interact with the system. The second part (AFS, Automatic Flight System) is in charge of performing functions such as guidance and automatic pilot. For safety purposes, each part is organized into a command and monitoring channels. The command channel ensures the function allocated to the component. The monitoring channel ensures that the command channel operates correctly. To ensure a sufficient availability level, a high level of redundancy is built inside the system. Components communicate using various communication means with different latencies (AFDX, A429, discrete).

[4] http://www.energybus.org
[5] http://www.can-cia.org

Since AFCSs have stringent safety and time-critical requirements, formal verification is required to ensure their correctness. To this aim, we have applied the GRL approach for the formal modelling and verification of GALS systems (see § 6.1.5). In a first step, we have addressed the AFCS without redundancy. We have written a GRL description (750 lines), which can be parameterized by the activation paces of different synchronous components. We have written a set of correctness properties in MCL, which we have verified on the GRL model.

6.5.6. Graphical User-Interfaces and Plasticity

Participants: Hubert Garavel, Frédéric Lang, Raquel Oliveira.

In the context of the Connexion project (see § 8.1.1.2) and in close collaboration with Gaëlle Calvary and Sophie Dupuy-Chessa (IIHM team of the LIG laboratory), we study the formal description and validation of graphical user-interfaces using the most recent features of the CADP toolbox. The case study assigned to LIG in this project is a prototype graphical user-interface [38] designed to provide human operators with an overview of a running nuclear plant. The main goal of the system is to inform the operators about alarms resulting from faults, disturbances, or unexpected events in the plant. Contrary to conventional control rooms, which employ large desks and dedicated hardware panels for supervision, this new-generation interface uses standard computer hardware (i.e., smaller screen(s), keyboard, and mouse), thus raising challenging questions on how to best provide synthetic views of the plant status. Another challenge is to introduce plasticity in such interface, so as to enable several supervision operators, including mobile ones outside of the control room, to get accurate information in real time.

We formally specified this new-generation interface in LNT, encompassing not only the usual components traditionally found in graphical user-interfaces, but also a model of the physical world (namely, a nuclear reactor with various fault scenarios) and a cognitive model of a human operator in charge of supervising the plant. Also, several desirable properties of the interface have been expressed in MCL and verified on the LNT model using CADP. At last, we used our formal model to check conformance of execution traces generated by an industrial control room prototype provided by a partner in the project.

In 2015, we finalized our approach to formally verifying safety critical interactive systems provided with plastic user interfaces, either using equivalence checking (to check whether different versions of user interfaces present the same interaction capabilities and appearance) or model checking (to check a set of properties over a model of the system). The results have been published in international conferences [26], [27] and journals [17], and in Raquel Oliveira's PhD thesis [11].

6.5.7. Fault-Tolerant Routing for Network-on-Chip Architectures

Participant: Wendelin Serwe.

Fault-tolerant architectures provide adaptivity for on-chip communications, but also increase the complexity of the design, so that formal verification techniques are needed to check their correctness. In collaboration with Chris Myers and Zhen Zhang (University of Utah, USA), we studied an extension of the link-fault tolerant Network-on-Chip (NoC) architecture introduced by Wu *et al* [64] that supports multiflit wormhole routing. A major difference with similar architectures existing in the literature is that the considered routing algorithm is not statically proven free of deadlocks, but rather implements deadlock avoidance (by dynamically detecting possible deadlock situations and avoiding them by dropping packets).

In 2015, we detected a potential livelock in the previously developed formal LNT model [65]. The correction of this problem led to an improved routing algorithm, for which the state space for 2x2 NoCs could be generated compositionally. We also experimented with the analysis of larger configurations on Grid'5000, but even a 2x3 NoC is still too large, so that compositional state space generation fails with an intermediate state space of several billions of states. This work led to a publication accepted in an international journal [18] and a PhD thesis [66].

6.5.7.1. *Other Case Studies*

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 2015, the effort to maintain and enhance these demos has been pursued. The progressive migration to LNT has continued, by translating five demos (16, 21, 22, 36, and 38) from LOTOS to LNT. A new demo 05 (airplane-ground communication protocol) has been added. The code of many demos was updated to use the latest features of LNT, such as "`in var`" parameters and "`assert`" statements. Demos 14 and 16 have been greatly simplified by inlining MCL and XTL temporal logic formulas in SVL scripts, using the "`property`", "`check`", and "`|=`" statements recently added to SVL. Nine demos (02, 08, 17, 20, 27, 28, 31, 33, and 36) have been simplified by using the new possibility to pass value parameters to LOTOS and LNT processes directly in SVL scripts. XTL formulas have been shortened in demos 23 and 27. The illustration of the EXEC/CAESAR framework in demo 38 has been integrated as a property into the SVL script. Finally, demo 38 led to a publication in an international workshop [29].

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Grants with Industry

Participants: Hubert Garavel, Abderahman Kriouile, Radu Mateescu, Wendelin Serwe.

Abderahman Kriouile is supported by a CIFRE PhD grant (from March 2012 to March 2015) from STMicroelectronics (Grenoble) on the verification of cache coherency in systems on chip (see § 6.5.1), under the supervision of Guilhem Barthes (STMicroelectronics), Christophe Chevallaz (STMicroelectronics), Grégory Faux (STMicroelectronics), Radu Mateescu (CONVECS), Wendelin Serwe (CONVECS), and Massimo Zendri (STMicroelectronics).

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. *FSN (Fonds national pour la Société Numérique)*

8.1.1.1. *OpenCloudware*

Participants: Rim Sakka Abid, Hugues Evrard, Frédéric Lang, Gwen Salaün [correspondent].

OpenCloudware [6] is a project funded by the FSN. The project is led by France Telecom / Orange Labs (Meylan, France) and involves 18 partners (among which Bull, OW2, Thalès, Inria, etc.). OpenCloudware aims at providing an open software platform enabling the development, deployment and administration of cloud applications. The objective is to provide a set of integrated software components for: (i) modeling distributed applications to be executed on cloud computing infrastructures; (ii) developing and constructing multi-tier virtualized applications; and (iii) deploying and administrating these applications (PaaS platform) possibly on multi-IaaS infrastructures.

OpenCloudware started in January 2012 for three years and nine months. The main contributions of CONVECS to OpenCloudware (see § 6.5.2) are the formal specification of the models, architectures, and protocols (self-deployment, dynamic reconfiguration, self-repair, etc.) underlying the OpenCloudware platform, the automated generation of code from these specifications for rapid prototyping purposes, and the formal verification of the aforementioned protocols.

8.1.1.2. *Connexion*

Participants: Hubert Garavel [correspondent], Frédéric Lang, Raquel Oliveira.

[6] http://www.opencloudware.org

Connexion [7] (*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. 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 (see § 6.5.6).

8.1.2. Competitivity Clusters

8.1.2.1. Bluesky for I-Automation

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

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 month until June 2016. The main contributions of CONVECS to Bluesky (see § 6.1.5 and § 6.5.3) 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.

8.1.3. Other National Collaborations

Additionally, we collaborated in 2015 with the following Inria project-teams:

- PAREO (Inria Nancy — Grand Est): Pierre-Etienne Moreau

Beyond Inria, we had sustained scientific relations with the following researchers:

- Gaëlle Calvary and Sophie Dupuy-Chessa (LIG, Grenoble),
- Fabrice Kordon and Lom Messan Hillah (LIP6, Paris),
- Noël De Palma and Fabienne Boyer (LIG, Grenoble),
- Xavier Etchevers (Orange Labs, Meylan),
- Matthias Güdemann (Systerel, Aix-en-Provence),
- Christophe Deleuze, Ioannis Parissis, and Mouna Tka Mnad (LCIS, Valence),
- Pascal Poizat (LIP6, Paris).

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. SENSATION

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

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

SENSATION [8] (*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 contributes 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 (see § 6.5.4).

8.2.2. Collaborations with Major European Organizations

The CONVECS project-team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM [9]. H. Garavel and R. Mateescu are members of the FMICS board, H. Garavel being in charge of dissemination actions.

8.3. 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.

8.3.1. Other International Collaborations

In 2015, we had scientific relations with several universities abroad, including:

- CWI, The Netherlands (Jurgen Vinju and Paul Klint),
- University of Málaga, Spain (F. Duran and C. Canal),
- University of Colorado, USA (Fabio Somenzi), and
- University of Utah, USA (Chris Myers and Zhen Zhang).

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- The annual CONVECS seminar was held in Charavines (France) on May 27–29, 2015. The following invited scientists attended the seminar:
  - Eric Jenn (IRT Saint-Exupéry / Thales Avionics) gave on May 27, 2015 a talk entitled "*The INGEQUIP Project and the TwIRTee demonstrator*".
  - Alexandre Hamez (IRT Saint-Exupéry) gave on May 29, 2015 a talk entitled "*CAESAR.SDD*".
- Chris Myers (University of Utah, USA) visited us from June 8–12, 2015. He gave a talk entitled "*An Integrated Verification Architecture*" on June 9, 2015.
- Hernan Ponce de Leon (Aalto University, Finland) visited us from June 29 to July 1, 2015. He gave a talk entitled "*Unfolding Based Testing for Multithreaded Programs*" on June 29, 2015.

[8] http://sensation-project.eu/
[9] http://fmics.inria.fr

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 of MCC'2015 (*Model Checking Contest*). In 2015, he verified the forms describing the benchmark models and enhanced the format of these forms by adding new model properties. He contributed to the formalization of the contest's rules and to the writing of the two contest calls (call for models and call for tools). He also participated in migrating the PetriWeb model base (developed at the Technical University of Eindhoven) to the format and conventions of the contest. A journal article describing the achievements of the 5th Model Checking Contest has been written and submitted for publication.
- G. Salaün was a member of the organizing committee of SEFM'2015 (*13th International Conference on Software Engineering and Formal Methods*), York, United Kingdom, September 7–11, 2015.

9.1.2. Scientific events selection

9.1.2.1. Chair of conference program committees

- G. Salaün was programme committee co-chair of SVT-SAC'2015 (the *Software Verification Track* of the *Symposium on Applied Computing*), Salamanca, Spain, April 13–17, 2015.

9.1.2.2. Member of the conference program committees

- H. Garavel was program committee member of TACAS'2015 (*21th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*), London, United Kingdom, April 11–19, 2015.
- G. Salaün and W. Serwe were program committee members of FSEN'2015 (*6th International Conference on Fundamentals of Software Engineering*), Tehran, Iran, April 22–24, 2015.
- H. Garavel was program committee member of DCDS'2015 (*5th International Workshop on Dependable Control of Discrete Systems*), Cancun, Mexico, May 27–29, 2015.
- G. Salaün was program committee member of ICE'2015 (*8th Interaction and Concurrency Experience*), Grenoble, France, June 4-5, 2015.
- F. Lang, R. Mateescu, and W. Serwe were program committee members of FMICS'2015 (*20th International Workshop on Formal Methods for Industrial Critical Systems*), Oslo, Norway, June 22–23, 2015.
- G. Salaün was program committee member of WWV'2015 (*11th International Workshop on Automated Specification and Verification of Web Systems*), Oslo, Norway, June 23, 2015.
- F. Lang was program committee member of ETR'2015 (*École d'été Temps Réel*), Rennes, France, August 24–28, 2015.
- H. Garavel and G. Salaün were program committee members of SEFM'2015 (*13th International Conference on Software Engineering and Formal Methods*), York, United Kingdom, September 7–11, 2015.
- G. Salaün was program committee member of SCART'2015 (*1st International Workshop on the ART of Software Composition*), York, United Kingdom, September 8, 2015.
- G. Salaün was program committee member of FACS'2015 (*12th International conference on Formal Aspects of Component Software*), Rio de Janeiro, Brazil, October 14–16, 2015.
- H. Garavel was program committee member of MARS'2015 (*Workshop on Models for Formal Analysis of Real Systems*), Suva, Fiji, November 23, 2015.

9.1.2.3. Reviewer

- H. Garavel was a reviewer for CONCUR'2015 (*26th Conference on Concurrency Theory*), Madrid, Spain, September 1–4, 2015.
- R. Mateescu was a reviewer for FORTE'2015 (*35th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems*), Grenoble, France, June 2–5, 2015.
- G. Salaün was a reviewer for SOCA'2015 (*8th IEEE International Conference on Service Oriented Computing and Applications*), Rome, Italy, October 19–21, 2015.
- W. Serwe was a reviewer for ATVA'2015 (*13th International Symposium on Automated Technology for Verification and Analysis*, October 12–15, 2015, Shanghai, China), MARS'2015, and TACAS'2016.

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*).
- G. Salaün is an editorial board member of SOCA (*Springer International Journal on Service Oriented Computing and Applications*).

9.1.3.2. Reviewer - Reviewing activities

- H. Garavel was a reviewer for the Mathematical Reviews (MathSciNet) of the American Mathematical Society.
- F. Lang was a reviewer for SOSYM (*International Journal on Software and Systems Modeling*).
- R. Mateescu was a reviewer for JISA (*Journal of Internet Services and Applications*), IEEE TSE (*Transactions on Software Engineering*), JLAMP (*Journal of Logical and Algebraic Methods in Programming*), FAoC (*Formal Aspects of Computing*), and STTT.
- G. Salaün was a reviewer for FAoC, JLAMP, JSS (*Journal of Systems and Software*), and MPE (*Mathematical Problems in Engineering*).

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, the PIC2LNT translator, and the PMC model checker (see § 5.2).

In 2015, the main facts are the following:

- We prepared and distributed twelve successive versions (2015-a to 2015-l) of CADP.
- We were requested to grant CADP licenses for 434 different computers in the world.

The CONVECS Web site [10] was updated with scientific contents, announcements, publications, etc. Following a request from the computer staff of Inria Grenoble, we worked on the migration of the former Web site "http://www.inrialpes.fr/vasy" to a more recent web infrastructure. This former site was split into three new, distinct Web sites "http://vasy.inria.fr", "http://cadp.inria.fr", and "http://fmics.inria.fr". Dedicated effort was made to properly redirect all former URLs so as not to create dead links.

By the end of December 2015, the CADP forum [11], opened in 2007 for discussions regarding the CADP toolbox, had over 350 registered users and over 1600 messages had been exchanged.

Also, for the 2015 edition of the Model Checking Contest, we provided two families of models (totalling 15 Nested-Unit Petri Nets) derived from our LNT models.

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

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:

- The Ocarina tool for analysing AADL descriptions [58]
- The MIstRAL tool for middleware reconfiguration based on formal methods [61]
- The DFTCalc tool for analysing dynamic fault trees [49]
- The Vercors integrated environment for verifying and running distributed components [52]
- The IMCReo tool for Interactive Markov Chains in Stochastic Reo [60]
- The GROOVE tool for verification based on graph rewriting [54]

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

- Improving design patterns finder precision using model checking [34]
- Applying automata learning to embedded control software [62]
- Testing autonomous systems using communicating extended finite-state machines [32]
- Assisting refinement in system-on-chip design [59]
- Formal verification of plastic user interfaces exploiting domain ontologies [37]
- Action-based verification of a fire alarm system [35]

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

- G. Salaün gave a talk entitled "*Reliable Deployment, Reconfiguration, and Control of Cloud Applications*" on February 12, 2015 at the University of Málaga, Spain.
- H. Garavel gave a talk entitled "*Nested-Unit Petri Nets: A Useful Concept for PseuCo?*" on March 9, 2015 at Saarland University, Saarbrücken, Germany.
- W. Serwe gave a talk entitled "*Using a Formal Model to Improve Verification of a Cache-Coherent System-on-Chip*" on March 19, 2015 at the Cadence Club Formal, Grenoble, France.
- R. Mateescu gave a talk entitled "*Overview of MCL, a Data-Based Model Checking Language*" on March 31, 2015 at Inria Grenoble — Rhône-Alpes, Montbonnot, France.
- G. Salaün gave a talk entitled "*Reliable Deployment, Reconfiguration, and Control of Cloud Applications*" on March 26, 2015 at LIRMM, Montpellier, France.
- G. Salaün gave a talk entitled "*Reliable Deployment, Reconfiguration, and Control of Cloud Applications*" on April 2, 2015 at LABRI, Bordeaux, France.
- G. Salaün gave a talk entitled "*Verification of Asynchronously Communicating Systems*" on April 9, 2015 at IRISA, Rennes, France.
- H. Garavel gave a talk entitled "*Nested-Unit Petri Nets: Combining Hierarchy with Concurrency*" on May 6, 2015 at LIP6, Université Pierre et Marie Curie, Paris, France.
- G. Salaün gave a talk entitled "*Formal Modelling and Analysis of BPMN Processes*" on July 9, 2015 at XEROX, Meylan, France.
- F. Lang gave two talks entitled "*Vérification de systèmes concurrents asynchrones par des méthodes compositionnelles*" and "*CADP : une boîte à outils pour la construction et l'analyse de processus distribués*" on August 25, 2015 at ETR (*Ecole d'Été Temps Réel*), Rennes, France. He also held a lab session on CADP.
- G. Salaün gave a talk entitled "*Formal Modelling and Analysis of BPMN Processes*" on October 27, 2015 at the University of Málaga, Spain.

- H. Garavel gave a talk entitled "*Les aspects probabilistes dans CADP*" on December 8, 2015 at the "*Journées Inria*" held at LIP6, Université Pierre et Marie Curie, Paris, France.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

CONVECS is a host team for the computer science master entitled "*Mathématiques, Informatique, spécialité : Systèmes et Logiciels*", common to Grenoble INP and University Joseph Fourier.

In 2015, we carried out the following teaching activities:

- G. Salaün is co-responsible of the ISI (*Ingéniérie des Systèmes d'Information*) department of ENSIMAG since September 1, 2011.
- F. Lang gave a lecture on formal methods (6 hours) in the framework of the software engineering course given to the first year students of the MOSIG (Master of Science in Informatics at Grenoble).
- G. Salaün gave lectures (192 hours "*équivalent TD*") to computer science engineering students of ENSIMAG ("*École Nationale Supérieure d'Informatique et de Mathématiques Appliquées*", Grenoble INP) on "*Théorie des langages*" (first year), "*Programmation orientée objet*" (second year), and "*Modélisation et vérification de systèmes concurrents*" (thrid year). He also organized a series of "*Conférences technologiques*" (third year).
- H. Garavel, together with Laurence Pierre (TIMA, Grenoble), created a new curriculum "*High-confidence Embedded and Cyberphysical Systems*" of the international masters programme.

9.2.2. Supervision

- PhD: Hugues Evrard, "*Génération automatique d'implémentation distribuée à partir de modèles formels de processus concurrents asynchrones*", Université Grenoble-Alpes, July 10, 2015, G. Salaün, F. Lang
- PhD: Abderahman Kriouile, "*Formal Methods for Functional Verification of Cache-Coherent System-on-Chip*", Université Grenoble-Alpes, September 17, 2015, R. Mateescu, W. Serwe, M. Zendri
- PhD: Raquel Oliveira, "*Formal Specification and Verification of Interactive Systems with Plasticity: Applications to Nuclear-Plant Supervision*", Université Grenoble-Alpes, December 3, 2015, H. Garavel, S. Dupuy-Chessa, G. Calvary, F. Lang
- PhD: Rim Sakka Abid, "*Coordination et Reconfiguration des Applications Distribuées Cloud*", Université Grenoble-Alpes, December 16, 2015, G. Salaün, N. de Palma
- PhD in progress: Fatma Jebali, "*A Framework for the Formal Specification and Verification of Globally Asynchronous Locally Synchronous Systems*", since December 2012, F. Lang, R. Mateescu
- PhD in progress: Gianluca Barbon, "*Debugging Concurrent Programs using Model Checking and Mining Techniques*", since October 2015, G. Salaün

9.2.3. Juries

- H. Garavel was member of the jury for David Lugato's habilitation thesis, entitled "*Formaliser par la modelisation : Applications au calcul haute performance et à la génération de tests par exécution symbolique*", defended at Université de Bordeaux, France, on March 5, 2015.
- W. Serwe was PhD committee member for Zhen Zhang's PhD thesis, entitled "*Verification Methodologies for Fault-Tolerant Network-on-Chip Systems*", defended at University of Utah, Salt Lake City, USA, on October 26, 2015.

9.3. Popularization

H. Garavel participates to the committee in charge of organizing the Aerospace Valley series of industrial conferences on formal methods.

The fifth conference [12], devoted to testing, held on June 16 in Toulouse and retransmitted by video-conference in Grenoble and Saclay, attracted participants from industry and academia. W. Serwe gave a talk (with Thierry Jéron, Inria Rennes), entitled "*TGV: Génération de tests de conformité à partir de modèles formels*", and a talk (with Massimo Zendri, STMicroelectronics), entitled "*Génération de tests basés sur les modèles pour des systèmes sur puce avec cohérence de caches*".

The sixth conference [13], devoted to safety, is to be held on January 26, 2016.

9.4. Miscellaneous Activities

H. Evrard is a member of the council of the MSTII doctoral school.

H. Evrard and G. Salaün are members of the council of the LIG laboratory.

H. Garavel is a member of the LIG commission in charge of preparing candidates selected for recruitment interviews at CNRS.

H. Garavel is an expert evaluator for Minalogic "*pôle de compétitivité mondial*".

H. Garavel was a reviewer for various ongoing ANR (*Agence Nationale de la Recherche*) pre-propositions and projects evaluated in 2015.

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

F. Lang is (together with Laurent Lefèvre from the AVALON Inria project-team) correspondent in charge of the 2014 and 2015 Inria activity reports at Inria Grenoble Rhône-Alpes.

E. Léo and W. Serwe are members of the "*Comité de centre*" of 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 is a member of the scientific council of Grenoble INP (*Conseil scientifique de l'institut*).

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

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

[12] http://projects.laas.fr/IFSE/FMF/J5/index.html
[13] http://projects.laas.fr/IFSE/FMF/J5/index.html

[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^{o} 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^{o} 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^{o} 3, pp. 290-304, http://hal.inria.fr/hal-00726448

Publications of the year

Doctoral Dissertations and Habilitation Theses

[8] R. ABID. *Coordination and Reconfiguration of Distributed Cloud Applications*, Université de Grenoble, December 2015, https://hal.inria.fr/tel-01258795

[9] H. EVRARD. *Automatic Distributed Code Generation from Formal Models of Asynchronous Concurrent Processes*, Université Grenoble Alpes, July 2015, https://hal.inria.fr/tel-01215634

[10] A. KRIOUILE. *Formal Methods for Functional Verification of Cache-Coherent System-on-Chip*, Université Grenoble Alpes, September 2015, https://hal.inria.fr/tel-01258784

[11] R. OLIVEIRA. *Formal Specification and Verification of Interactive Systems with Plasticity : Applications to Nuclear-Plant Supervision*, Université Grenoble Alpes, December 2015, https://tel.archives-ouvertes.fr/tel-01253619

Articles in International Peer-Reviewed Journals

[12] R. ABID, G. SALAÜN, N. DE PALMA. *Formal Design of Dynamic Reconfiguration Protocol for Cloud Applications*, in "Science of Computer Programming", 2015 [*DOI :* 10.1016/J.SCICO.2015.12.001], https://hal.inria.fr/hal-01246152

[13] F. DURÁN, G. SALAÜN. *Robust and reliable reconfiguration of cloud applications*, in "Journal of Systems and Software", 2015 [*DOI :* 10.1016/J.JSS.2015.09.020], https://hal.inria.fr/hal-01245555

[14] H. GARAVEL. *Revisiting sequential composition in process calculi*, in "Journal of Logical and Algebraic Methods in Programming", 2015 [*DOI :* 10.1016/J.JLAMP.2015.08.001], https://hal.inria.fr/hal-01247770

[15] H. GARAVEL, F. LANG, R. MATEESCU. *Compositional Verification of Asynchronous Concurrent Systems using CADP*, in "Acta Informatica", June 2015, vol. 52, n^{o} 4, 56 p. [*DOI :* 10.1007/S00236-015-0226-1], https://hal.inria.fr/hal-01247507

[16] M. GÜDEMANN, P. POIZAT, G. SALAÜN, L. YE. *VerChor: A Framework for the Design and Verification of Choreographies*, in "IEEE Transactions on Services Computing", 2015, forthcoming [*DOI :* 10.1109/TSC.2015.2413401], https://hal.archives-ouvertes.fr/hal-01198918

[17] R. OLIVEIRA, S. DUPUY-CHESSA, G. CALVARY. *Verification of Plastic Interactive Systems*, in "Journal of Interactive Media (i-com)", December 2015, vol. 14, n^{o} 3, pp. 192–204 [*DOI :* 10.1515/ICOM-2015-0036], https://hal.inria.fr/hal-01254590

[18] 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

[19] R. ABID, G. SALAÜN, N. DE PALMA, S. MAK-KARE GUEYE. *Asynchronous Coordination of Stateful Autonomic Managers in the Cloud*, in "12th International Symposium on Formal Aspects of Components and Systems FACS'2015", Niterói, Rio de Janeiro, Brazil, October 2015, https://hal.inria.fr/hal-01245754

[20] L. AKROUN, F. TOUMANI, L. NOURINE. *Reasoning in description logics with variables: preliminary results regarding the EL logic*, in "28th International Workshop on Description Logics", Athenes, Greece, June 2015, 12 p. , https://hal.inria.fr/hal-01163342

[21] C. CANAL, G. SALAÜN. *Model-Based Adaptation of Software Communicating via FIFO Buffers*, in "18th International Conference on Fundamental Approaches to Software Engineering (FASE 2015)", Londres, United Kingdom, April 2015 [*DOI :* 10.1007/978-3-662-46675-9_17], https://hal.inria.fr/hal-01150353

[22] 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, Springer-Verlag, April 2016, https://hal.inria.fr/hal-01250925

[23] H. EVRARD, F. LANG. *Automatic Distributed Code Generation from Formal Models of Asynchronous Concurrent Processes*, in "23rd Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP 2015)", Turku, Finland, March 2015, https://hal.inria.fr/hal-01086522

[24] H. GARAVEL. *Nested-Unit Petri Nets: A Structural Means to Increase Efficiency and Scalability of Verification on Elementary Nets*, in "36th International Conference on Application and Theory of Petri Nets and Concurrency PETRI NETS'2015", Brussels, Belgium, June 2015 [*DOI :* 10.1007/978-3-319-19488-2_9], https://hal.inria.fr/hal-01142198

[25] A. KRIOUILE, W. SERWE. *Using a Formal Model to Improve Verification of a Cache-Coherent System-on-Chip*, in "21th International Conference on Tools and Algorithms for the Construction and Analysis of Systems", London, UK, France, April 2015 [*DOI :* 10.1007/978-3-662-46681-0_62], https://hal.inria.fr/hal-01104747

[26] R. OLIVEIRA, S. DUPUY-CHESSA, G. CALVARY. *Equivalence Checking for Comparing User Interfaces*, in "7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems EICS'2015", Duisburg, Germany, ACM, June 2015 [*DOI :* 10.1145/2774225.2774844], https://hal.inria.fr/hal-01247496

[27] R. OLIVEIRA, S. DUPUY-CHESSA, G. CALVARY. *Plasticity of User Interfaces: Formal Verification of Consistency*, in "7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems EICS'2015", Duisburg, Germany, ACM, June 2015 [*DOI :* 10.1145/2774225.2775078], https://hal.inria.fr/hal-01247489

[28] G. SALAÜN, L. YE. *Debugging Process Algebra Specifications*, in "VMCAI 2015", Mumbai, India, Springer, January 2015, vol. 8931, 18 p. [*DOI :* 10.1007/978-3-662-46081-8_14], https://hal.inria.fr/hal-01087505

[29] W. SERWE. *Formal Specification and Verification of Fully Asynchronous Implementations of the Data Encryption Standard*, in "Proceedings of the first Workshop on Models for Formal Analysis of Real Systems (MARS 2015)", Suva, Fiji, R. VAN GLABBEEK, J. F. GROOTE, P. HÖFNER (editors), Electronic Proceedings in Theoretical Computer Science, November 2015, vol. 196 [*DOI :* 10.4204/EPTCS.196.6], https://hal.inria.fr/hal-01227999

Research Reports

[30] H. GARAVEL, F. LANG, R. MATEESCU. *Compositional Verification of Asynchronous Concurrent Systems using CADP (extended version)*, Inria Grenoble - Rhône-Alpes, April 2015, n^{o} RR-8708, https://hal.inria.fr/hal-01138749

References in notes

[31] *AMBA AXI and ACE Protocol Specification*, ARM IHI 0022D (ID102711), ARM, October 22 2011

[32] A. ANDREWS, M. ABDELGAWAD, A. GARIO. *Active World Model for Testing Autonomous Systems Using CEFSM*, in "Proceedings of the 12th Workshop on Model-Driven Engineering, Verification and Validation MoDeVVa'2015 (Ottawa, Canada)", M. FAMELIS, D. RATIU, M. SEIDL, G. M. K. SELIM (editors), CEUR Workshop Proceedings, September 2015, vol. 1514, pp. 1–10

[33] A. BELINFANTE. *JTorX: A Tool for On-Line Model-Driven Test Derivation and Execution*, in "Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2010 (Paphos, Cyprus)", Lecture Notes in Computer Science, Springer Verlag, March 2010, vol. 6015, pp. 266–270

[34] M. L. BERNARDI, M. CIMITILE, G. D. RUVO, G. A. D. LUCCA, A. SANTONE. *Improving Design Patterns Finder Precision Using a Model Checking Approach*, in "Proceedings of the CAiSE 2015 Forum at the 27th International Conference on Advanced Information Systems Engineering CAiSE'2015 (Stockholm, Sweden)", J. GRABIS, K. SANDKUHL (editors), CEUR Workshop Proceedings, June 2015, vol. 1367, pp. 113–120

[35] J. BIERNACKI, A. BIERNACKA, M. SZPYRKA. *Action-based verification of RTCP-nets with CADP*, in "Proceedings of the International Conference of Computational Methods in Sciences and Engineering ICCMSE'2015 (Athens, Greece)", T. E. SIMOS, Z. KALOGIRATOU, T. MONOVASILIS (editors), AIP Conference Proceedings, March 2015, vol. 1702

[36] 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

[37] A. CHEBIEB, Y. A. AMEUR. *Formal Verification of Plastic User Interfaces Exploiting Domain Ontologies*, in "Proceedings of the International Symposium on Theoretical Aspects of Software Engineering TASE'2015 (Nanjing, China)", IEEE Computer Society, September 2015, pp. 79–86

[38] F. CHÉRIAUX, D. GALARA, M. VIEL. *Interfaces for Nuclear Power Plant Overview*, in "Proceedings of the 8th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies NPIC & HMIT 2012 (San Diego, California, USA)", American Nuclear Society, July 2012

[39] 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

[40] N. COSTE, H. HERMANNS, E. LANTREIBECQ, W. SERWE. *Towards Performance Prediction of Compositional Models in Industrial GALS Designs*, in "Proceedings of the 21th International Conference on Computer Aided Verification CAV'2009 (Grenoble, France)", A. BOUAJJANI, O. MALER (editors), Lecture Notes in Computer Science, Springer Verlag, July 2009, vol. 5643, pp. 204–218, http://hal.inria.fr/inria-00381657

[41] 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

[42] 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

[43] 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

[44] 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

[45] 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)

[46] 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

[47] 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

[48] A. GRAF-BRILL. *Model-based Testing Approaches for the EnergyBus*, Department of Computer Science, Faculty of Natural Sciences and Technology I, Saarland University, October 2013

[49] D. GUCK, J. SPEL, M. STOELINGA. *DFTCalc: Reliability Centered Maintenance via Fault Tree Analysis*, in "Proceedings of the 17th International Conference on Formal Engineering Methods ICFEM'2015 (Paris,

France)", M. BUTLER, S. CONCHON, F. ZAÏDI (editors), Lecture Notes in Computer Science, Springer Verlag, November 2015, vol. 9407, pp. 304–311

[50] 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

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

[52] L. HENRIO, O. KULANKHINA, S. LI, E. MADELAINE. *Integrated environment for verifying and running distributed components*, Inria, December 2015, n^o 8841

[53] C. JARD, T. JÉRON. *TGV: Theory, Principles and Algorithms — A Tool for the Automatic Synthesis of Conformance Test Cases for Non-Deterministic Reactive Systems*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", August 2005, vol. 7, n^o 4, pp. 297–315

[54] S. JUNGES, D. GUCK, J. KATOEN, A. RENSINK, M. STOELINGA. *Fault Trees on a Diet - Automated Reduction by Graph Rewriting*, in "Proceedings of the 1st International Symposium on Dependable Software Engineering: Theories, Tools, and Applications SETTA'2015 (Nanjing, China)", X. LI, Z. LIU, W. YI (editors), Lecture Notes in Computer Science, Springer Verlag, November 2015, vol. 9409, pp. 3–18

[55] M. KWIATKOWSKA, G. NORMAN, D. PARKER. *PRISM: Probabilistic Symbolic Model Checker*, in "Computer Performance Evaluation: Modelling Techniques and Tools", T. FIELD, P. HARRISON, J. BRADLEY, U. HARDER (editors), Lecture Notes in Computer Science, Springer Verlag, April 2002, vol. 2324, pp. 200–204

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

[57] 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

[58] H. MKAOUAR, B. ZALILA, J. HUGUES, M. JMAIEL. *From AADL Model to LNT Specification*, in "Proceedings of the 20th Ada-Europe International Conference on Reliable Software Technologies (Madrid, Spain)", J. A. DE LA PUENTE, T. VARDANEGA (editors), Lecture Notes in Computer Science, Springer Verlag, June 2015, vol. 9111, pp. 146–161

[59] H. MOKRANI, R. AMEUR-BOULIFA, E. ENCRENAZ-TIPHENE. *Assisting Refinement in System-on-Chip Design*, in "Languages, Design Methods, and Tools for Electronic System Design", M.-M. LOUERAT, T. MAEHNE (editors), Lecture Notes in Electrical Engineering, Springer Verlag, 2015, vol. 311, pp. 21–42

[60] N. OLIVEIRA, A. SILVA, L. S. BARBOSA. *IMCReo: Interactive Markov Chains for Stochastic Reo*, in "Journal of Internet Services and Information Security", 2015, vol. 5, n^o 1, pp. 3–28

[61] N. ROSA. *Middleware Reconfiguration Relying on Formal Methods*, in "Proceedings of the 15th IEEE International Conference on Computer and Information Technology; 14th IEEE International Conference on Ubiquitous Computing and Communications; 13th IEEE International Conference on Dependable, Autonomic and Secure Computing; 13th IEEE International Conference on Pervasive Intelligence and Computing CIT/IUCC/DASC/PICom'2015 (Liverpool, United Kingdom)", Y. WU, G. MIN, N. GEORGALAS, J. HU, L.

ATZORI, X. JIN, S. A. JARVIS, L. (. LIU, R. A. CALVO (editors), IEEE Computer Society Press, October 2015, pp. 648–655

[62] W. SMEENK, J. MOERMAN, F. W. VAANDRAGER, D. N. JANSEN. *Applying Automata Learning to Embedded Control Software*, in "Proceedings of the 17th International Conference on Formal Engineering Methods ICFEM'2015 (Paris, France)", M. BUTLER, S. CONCHON, F. ZAÏDI (editors), Lecture Notes in Computer Science, Springer Verlag, November 2015, vol. 9407, pp. 67–83

[63] J. TRETMANS. *Model Based Testing with Labelled Transition Systems*, in "Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers", R. M. HIERONS, J. P. BOWEN, M. HARMAN (editors), Lecture Notes in Computer Science, Springer Verlag, 2008, vol. 4949, pp. 1–38

[64] J. WU, Z. ZHANG, C. MYERS. *A Fault-Tolerant Routing Algorithm for a Network-on-Chip Using a Link Fault Model*, in "Virtual Worldwide Forum for PhD Researchers in Electronic Design Automation", 2011, http://www.async.ece.utah.edu/oldsite/publications/VW-FEDA2.pdf

[65] Z. ZHANG, W. SERWE, J. WU, T. YONEDA, H. ZHENG, C. MYERS. *Formal Analysis of a Fault-Tolerant Routing Algorithm for a Network-on-Chip*, in "Proceedings of the 19th International Workshop on Formal Methods for Industrial Critical Systems FMICS'2014 (Florence, Italy)", F. LANG, F. FLAMMINI (editors), Lecture Notes in Computer Science, Springer Verlag, September 2014, vol. 8718, pp. 48–62

[66] Z. ZHANG. *Verification Methodologies for Fault-Tolerant Network-on-Chip Systems*, University of Utah, Salt Lake City, October 2015