IN PARTNERSHIP WITH:
Institut polytechnique de Grenoble

Université Joseph Fourier (Grenoble)

Activity Report 2018

Project-Team CONVECS

Construction of verified concurrent systems

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

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Proofs and Verification

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:

A1.3. - Distributed Systems
A1.3.5. - Cloud
A1.3.6. - Fog, Edge
A2.1.1. - Semantics of programming languages
A2.1.7. - Distributed programming
A2.4.1. - Analysis
A2.4.2. - Model-checking
A2.5. - Software engineering
A2.5.1. - Software Architecture & Design
A2.5.4. - Software Maintenance & Evolution
A2.5.5. - Software testing
A7.1.1. - Distributed algorithms
A7.1.3. - Graph algorithms
A7.2. - Logic in Computer Science
A8.9. - Performance evaluation

Other Research Topics and Application Domains:

B6.1.1. - Software engineering
B6.3.2. - Network protocols
B6.4. - Internet of things
B6.6. - Embedded systems
B8.1. - Smart building/home

1. Team, Visitors, External Collaborators

Research Scientists

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

Faculty Member

Gwen Salaün [UGA, Professor, HDR]

PhD Students

Gianluca Barbon [UGA]
Lina Marsso [Inria]
Ajay Muroor Nadumane [Inria]
Umar Ozeer [Orange Labs]

Technical staff

Lian Apostol [Inria, until Nov. 2018]

Interns

Pierre Bouvier [Inria, from Feb. 2018 until Jun. 2018]
Lucie Colpart [Inria, from Jun. 2018 until Aug. 2018]
Philippe Ledent [Inria, from Feb. 2018 until Jun. 2018]
Ernesto Yah Lopez [Inria, from Feb. 2018 until Jun. 2018]

Administrative Assistant

Myriam Etienne [Inria]

External Collaborator

Viet Anh Nguyen [IRT Saint-Exupéry]

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 [5] 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*) [55], 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 [41], 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,
- *Cloud computing:* self-deployment protocols, dynamic reconfiguration protocols,
- *Fog and IoT:* stateful IoT applications in the fog,
- *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. CADP Pro

Construction and Analysis of Distributed Processes
KEYWORDS: Formal methods - Verification
FUNCTIONAL DESCRIPTION: CADP (*Construction and Analysis of Distributed Processes* – formerly known as *CAESAR/ALDEBARAN Development Package*) [4] is a toolbox for protocols and distributed systems engineering.

In this toolbox, we develop and maintain the following tools:

- CAESAR.ADT [40] 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 [46], [45] 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 [41] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently of any particular high level language. In this respect, OPEN/CAESAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CAESAR consists of a set of 16 code libraries with their programming interfaces, such as:
  - CAESAR_GRAPH, which provides the programming interface for graph exploration,
  - CAESAR_HASH, which contains several hash functions,
  - CAESAR_SOLVE, which resolves Boolean equation systems on the fly,
  - CAESAR_STACK, which implements stacks for depth-first search exploration, and
  - CAESAR_TABLE, which handles tables of states, transitions, labels, etc.

A number of on-the-fly analysis tools have been developed within the OPEN/CAESAR environment, among which:

- BISIMULATOR, which checks bisimulation equivalences and preorders,
- CUNCTATOR, which performs steady-state simulation of continuous-time Markov chains,
- DETERMINATOR, which eliminates stochastic nondeterminism in normal, probabilistic, or stochastic systems,
- DISTRIBUTOR, which generates the graph of reachable states using several machines,
- EVALUATOR, which evaluates MCL formulas,
- EXECUTOR, which performs random execution,
- 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 [49], CTL [37], ACTL [38], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [44] 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 [53] 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 Inria Rennes – Bretagne-Atlantique.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [42] 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.

- Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu and Wendelin Serwe
- Contact: Hubert Garavel
- URL: http://cadp.inria.fr/

5.2. TRAIAN

KEYWORDS: Compilation - LOTOS NT
FUNCTIONAL DESCRIPTION: TRAIAN is a compiler for translating LOTOS NT descriptions into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

The current version of TRAIAN, which handles LOTOS NT types and functions only, has useful applications in compiler construction [43], being used in all recent compilers developed by CONVECS.

- Participants: Hubert Garavel, Frédéric Lang and Wendelin Serwe
- Contact: Hubert Garavel
- URL: http://convecs.inria.fr/software/traian/

6. New Results

6.1. New Formal Languages and their Implementations

6.1.1. LOTOS and LNT Specification Languages

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

LNT [5] [36] is a next-generation formal description language for asynchronous concurrent systems. The design of LNT at CONVECS is the continuation of the efforts undertaken in the 80s to define sound languages for concurrency theory and, indeed, LNT is derived from the ISO standards LOTOS (1989) and E-LOTOS (2001). In a nutshell, LNT attempts to combine the best features of imperative programming languages, functional languages, and value-passing process calculi.

LNT is not a frozen language: its definition started in 2005, as part of an industrial project. Since 2010, LNT has been systematically used by CONVECS for numerous case studies (many of which being industrial applications — see § 6.5). LNT is also used as a back-end by other research teams who implement various languages by translation to LNT. It is taught in university courses, e.g., at University Grenoble Alpes and ENSIMAG, where it is positively accepted by students and industry engineers. Based on the feedback acquired by CONVECS, LNT is continuously improved.

In 2018, the CADP tools that translate LNT to LOTOS have been enhanced in various ways. In the warning and error messages emitted by LNT2LOTOS, line numbers have been made more precise. In addition to a bug fix, the LNT_DEPEND tool, which computes dependencies between LNT modules has been entirely rewritten and made much faster. Also, the LNT language has been simplified by removing "!external" pragmas for constructors, as "!external" pragmas for types are sufficient.

We also continued improving the TRAIAN compiler for the LOTOS NT language (a predecessor of LNT), which is used for the construction of most CADP compilers and translators.

In February 2018, we released version 2.9 of TRAIAN. We scrutinized the source code of TRAIAN, deleting all parts of code corresponding to those features of the LOTOS NT language that were either not fully implemented or seldom used in practice. This reduced the source code of TRAIAN by 40% and the binaries by 50%. External LOTOS NT functions are now allowed to return a non-void result. Support for 64-bit macOS executables was added. A few bugs have been fixed and the reference manual of TRAIAN was entirely revised.

The main limitation of TRAIAN 2.x is that it is a 20-year-old compiler that is increasingly difficult to maintain. It consists in a large collection of attribute grammars and is built using the FNC-2 compiler generation system, which is no longer supported. For this reason, TRAIAN only exists in 32-bit version, and sometimes hits the 3–4 GB RAM limit when dealing with large compiler specifications, such as those of LNT2LOTOS or EVALUATOR 5.

For this reason, we undertook a complete rewrite of TRAIAN to get rid of FNC-2. Two main design decisions behind TRAIAN 3.0 are the following: (i) it supports (most of) the LOTOS NT language currently accepted by TRAIAN 2.9, but also extensions belonging to LNT, so as to allow a future migration from LOTOS NT to LNT; and (ii) TRAIAN 3.0 is currently written in LOTOS NT and compiled using TRAIAN 2.9, but should be ultimately capable of bootstrapping itself.

So far, a lexer and parser for LOTOS NT have been developed using the SYNTAX compiler-generation system [1] developed at Inria Paris. This work triggered an in-depth reexamination of the programming interfaces offered by SYNTAX and led to enhancements of these interfaces (see § 6.1.6).

The abstract syntax tree of LOTOS NT, and the library of predefined LOTOS NT types and functions have been redesigned; previously specified as FNC-2 attribute grammars, they are now themselves written in LOTOS NT, so as to allow bootstrap, using the current version of TRAIAN to build the next one. The construction of the abstract syntax tree has also been completed. Finally, we set several non-regression test bases gathered all available programs written in LOTOS NT.

6.1.2. NUPN

Participant: Hubert Garavel.

[1] http://syntax.gforge.inria.fr

Nested-Unit Petri Nets (NUPNs) is an upward-compatible extension of P/T nets, which are enriched with structural information on their concurrent structure. Such additional information can easily be produced when NUPNs are generated from higher-level specifications (e.g., process calculi); quite often, such information allows logarithmic reductions in the number of bits required to represent states, thus enabling verification tools to perform better. The principles of NUPNs are exposed in [39] and its PNML representation is described here [2].

The NUPN model has been adopted by the Model Checking Contest and the Rigorous Examination of Reactive Systems challenge. It has been so far implemented in thirteen different tools developed in four countries.

In 2018, a journal article (to appear in 2019) has been written to formalize the complete theory of NUPNs. The CAESAR.BDD tool for NUPNs has been extended with twelve new options. A new tool named NUPN_INFO has been added to CADP to perform three normalizing transformations of NUPNs.

6.1.3. MCL and XTL Property Specification Languages

Participants: Hubert Garavel, Radu Mateescu.

CADP provides two different languages, named MCL and XTL, for expressing data-handling temporal properties of concurrent systems. MCL is an extension of alternation-free modal μ-calculus with data values, programming language constructs, generalized regular formulas on transition sequences, and fairness operators. XTL is a functional-like programming language interpreted on Labeled Transition Systems, enabling the definition of temporal operators by computing their interpretation using fixed point iterations over sets of states and transitions.

In 2018, we enhanced these languages and their associated tools as follows:

- The MCL v4 language was enhanced with a new operator "**loop**" on regular formulas over transition sequences. This general iteration operator parameterized by data variables is able to characterize complex (recursively definable) sequences in an LTS. Two auxiliary regular operators "**continue**" and "**exit**" carrying data values were also introduced to express the repetition and the termination of a loop regular formula, respectively. These operators are particularly useful for specifying transition sequences having a particular cumulated cost (e.g., number of transitions, sum of weights associated to actions, etc.) in the context of probabilistic verification (see § 6.3.2).
- The MCL v3 language was modified and aligned on MCL v4 by removing syntactic differences that existed between both languages concerning the infinite repetition operator ("@") and the respective precedences of the concatenation (".") and choice ("|") operators in regular formulas. MCL v3 has also been enriched with the option operator ("?") on regular formulas already present in MCL v4.
- Consequently, the two versions of MCL_EXPAND for MCL v3 and MCL v4 have been unified in one single tool, which is now invoked by both EVALUATOR 3 and EVALUATOR 4. The corresponding manual pages have been simplified accordingly, with the introduction of two overarching manual pages ("mcl" and "evaluator"). In addition to five bug fixes, the memory footprint of MCL_EXPAND has been reduced. The error messages displayed by MCL_EXPAND, EVALUATOR 3, and EVALUATOR 4 have been improved in terms of accuracy and explanatory contents.
- In addition to four bug fixes, the XTL model checker now performs consistency checks on the C identifiers specified by the pragmas "!implementedby", "!comparedby", "!enumeratedby", and "!printedby".
- Two new options were added to the EVALUATOR and XTL model checkers: "-depend", which displays the libraries transitively included in an MCL or XTL file, and "-source", which is used by SVL to display correct file names and line numbers for MCL or XTL formulas embedded in SVL scenarios.

6.1.4. Translation of Term Rewrite Systems

Participant: Hubert Garavel.

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

We pursued the development undertaken in 2015 of a software platform for systematically comparing the performance of rewrite engines and pattern-matching implementations in algebraic specification and functional programming languages. Our platform reuses the benchmarks of the three Rewrite Engine Competitions (2006, 2009, and 2010). Such benchmarks are term-rewrite systems expressed in a simple formalism named REC, for which we developed automated translators that convert REC benchmarks into many languages, among which AProVE, Clean, Haskell, LNT, LOTOS, Maude, mCRL, MLTON, OCAML, Opal, Rascal, Scala, SML-NJ, Stratego/XT, and Tom.

In 2018, we corrected and/or enhanced several of the existing REC translators and finalized experiments. The results of this study have been presented during an invited talk at WRLA'2018 (*12th International Workshop on Rewriting Logic and its Applications*) and an article [15] was published in the WRLA post-proceedings.

6.1.5. Formal Modeling and Analysis of BPMN

Participant: Gwen Salaün.

A business process is a set of structured activities that provide a certain service or product. Business processes can be modeled using the BPMN standard, and several industrial platforms have been developed for supporting their design, modeling, and simulation.

In collaboration with Francisco Durán and Camilo Rocha (University of Málaga, Spain), we proposed a rewriting logic executable specification of BPMN with time and extended with probabilities. Duration times and delays for tasks and flows can be specified as stochastic expressions, while probabilities are associated to various forms of branching behavior in gateways. These quantities enable discrete-event simulation and automatic stochastic verification of properties such as expected processing time, expected synchronization time at merge gateways, and domain-specific quantitative assertions. The mechanization of the stochastic analysis tasks is done with Maude's statistical model checker PVeStA. These results led to a publication in an international journal [10].

We also worked on an extension of BPMN with data, which is convenient for describing real-world processes involving complex behavior and data descriptions. By considering this level of expressiveness due to the new features, challenging questions arise regarding the choice of the semantic framework for specifying such an extension of BPMN, as well as how to carry out the symbolic simulation, validation, and assess the correctness of the process models. These issues were addressed first by providing a symbolic executable rewriting logic semantics of BPMN using the rewriting modulo SMT framework, where the execution is driven by rewriting modulo axioms and by querying SMT decision procedures for data conditions. Second, reachability properties, such as deadlock freedom and detection of unreachable states with data exhibiting certain values, can be specified and automatically checked with the help of Maude, thanks to its support for rewriting modulo SMT. These results led to a publication in an international conference [21].

6.1.6. Other Language Developments

Participants: Hubert Garavel, Frédéric Lang, 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.

In 2018, we enhanced the SYNTAX compiler generator [3] in various ways: (i) The "string manager" has been generalized to allow several symbol tables to be handled simultaneously; (ii) The "source manager" has been extended with new relocation primitives that enable the caller to specify alternative file names and line numbers for the source file being parsed; for instance, this is typically useful for implementing the "#line" pragma of the C preprocessor; this mechanism has been extended to transparently handle multiple relocations (triggered by the lexer) while recognizing the right-hand side of a syntax rule in the grammar; (iii) The "include manager" has been modified to store file names in a distinct symbol table than the table of identifiers, and to provide the list of all files transitively included from the principal module; (iv) Finally, the main programming interface of SYNTAX has been extended with new primitives, so that at present only 5 calls (rather than 9–13 calls, formerly) are required to launch a compiler written using SYNTAX.

[3] http://syntax.gforge.inria.fr

All the CADP compilers have been modified to take advantage of the improvements of the SYNTAX library.

Also, a master student started to study an automated translation from Event-B to LNT. He reviewed the syntax and semantics of Event-B and proposed a pencil-paper translation of most Event-B operators. He applied it to a small example consisting of a bank system, where accounts can be created and closed, and money can be deposited or withdrawn. This was a preliminary work that did not lead to a full implementation, due to lack of time. However, this work is a solid basis for a later implementation.

6.2. Parallel and Distributed Verification

6.2.1. Distributed State Space Manipulation

Participant: Wendelin Serwe.

For distributed verification, CADP provides the PBG format, which implements the theoretical concept of *Partitioned LTS* [44] and provides a unified access to an LTS distributed over a set of remote machines.

In 2018, we improved the usability of distributed state space manipulation tools. In particular:

- A memory shortage error that occurs on a computing node now triggers a distributed termination of the computation, producing proper error messages in the log file of that node.
- A similar naming scheme for log files produced by computing nodes was enforced for all distributed verification tools, which prevents interferences between different invocations of the tools.

6.2.2. Debugging of Concurrent Systems using Counterexample Analysis

Participants: Gianluca Barbon, Gwen Salaün.

Model checking is an established technique for automatically verifying that a model satisfies a given temporal property. When the model violates the property, the model checker returns a counterexample, which is a sequence of actions leading to a state where the property is not satisfied. Understanding this counterexample for debugging the specification is a complicated task for several reasons: (i) the counterexample can contain hundreds of actions, (ii) the debugging task is mostly achieved manually, (iii) the counterexample does not explicitly highlight the source of the bug that is hidden in the model, (iv) the most relevant actions are not highlighted in the counterexample, and (v) the counterexample does not give a global view of the problem.

We proposed an approach that improves the usability of model checking by simplifying the comprehension of counterexamples. Our solution aims at keeping only actions in counterexamples that are relevant for debugging purposes. This is achieved by detecting in the models some specific choices between transitions leading to a correct behaviour or falling into an erroneous part of the model. These choices, which we call "neighbourhoods", provide key information for understanding the bug behind the counterexample. To extract such choices, we proposed a first method for debugging the counterexamples of safety property violations. To do so, it builds a new model from the original one containing all the counterexamples, and then compares the two models to identify neighbourhoods.

In 2018, we proposed a different method for debugging the counterexamples of liveness property violations. Given a liveness property, it extends the model with prefix and suffix information w.r.t. that property. This enriched model is then analysed to identify neighbourhoods. A set of abstraction techniques we developed exploit the enriched model annotated with neighbourhoods to extract relevant actions from counterexamples, which makes their comprehension easier. This work led to a publication in an international conference [16].

Both approaches are fully automated by a tool we implemented and that has been validated on real-world case studies from various application areas. We extended the methodology and tool with 3D visualization techniques to visualize the erroneous part of the model with a specific focus on neighbourhoods, in order to have a global view of the bug behaviour. This work led to a publication to appear in an international conference.

A detailed description of the proposed methodology is available in G. Barbon's PhD thesis [8].

6.3. Timed, Probabilistic, and Stochastic Extensions

6.3.1. Tools for Probabilistic and Stochastic Systems

Participants: Hubert Garavel, Frédéric Lang.

Formal models and tools dealing with quantitative aspects (such as time, probabilities, and other continuous physical quantities) have become unavoidable for a proper study and computer-aided verification of functional and non-functional properties of cyber-physical systems. The wealth of such formal models is sometimes referred to as a quantitative "zoo" [48].

The CADP toolbox already implements some of these probabilistic/stochastic models, namely DTMCs and CTMCs (*Discrete-Time* and *Continuous-Time Markov Chains*), and IMCs (*Interactive Markov Chains*) [50]. Our long-term goal is to increase the capability and flexibility of the CADP tools, so as to support other quantitative models more easily.

In 2018, BCG_STEADY and BCG_TRANSIENT were enhanced along the following lines:

- They were extended to handle single-state Markov chains and to properly compute state solution vectors and transition throughputs on such models.
- Their command-line options were simplified and warnings are emitted when the input Markov chain contains no stochastic transition.
- A problem which caused correct Markov chains to be rejected was corrected. This problem was due to floating point conversion and rounding errors.
- A confusion between state numbers and matrix indices was fixed in the output and error messages.
- Models containing probabilistic self-loops are now rejected, as was already the case of longer circuits of probabilistic transitions, as both represent similar "timelock" situations.

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

Participant: Radu Mateescu.

Specifying and verifying quantitative properties of concurrent systems requires expressive and user-friendly property languages combining temporal, data-handling, and quantitative aspects. In collaboration with José Ignacio Requeno (Univ. Zaragoza, Spain), we undertook the quantitative analysis of concurrent systems modeled as PTSs (*Probabilistic Transition Systems*), whose actions contain data values and probabilities. We proposed a new regular probabilistic operator that extends naturally the Until operators of PCTL (*Probabilistic Computation Tree Logic*) [47], by specifying the probability measure of a path characterized by a generalized regular formula involving arbitrary computations on data values. We integrated the regular probabilistic operator into MCL, we devised an associated on-the-fly model checking method based on a combined local resolution of linear and Boolean equation systems, and we implemented the method in a prototype extension of the EVALUATOR model checker.

In 2018, we continued improving and using the extended model checker as follows:

- The model checker now determinizes the dataless regular formulas contained in regular probabilistic operators, ensuring automatically that the linear equation systems produced by the verification of these operators have a unique solution.
- For nondeterministic data-handling regular formulas contained in regular probabilistic operators, the model checker now produces a warning message informing the user that the determinization has to be done manually.
- We carried out further experiments to analyze the quantitative behaviour of the Bounded Retransmission Protocol, namely the variation of the probability of transmission failure w.r.t. the total number of retransmissions attempts.

A paper describing the probabilistic extension of MCL and of the on-the-fly model checker was published in an international journal [13].

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 2018, we improved these tools along the following lines:

- SVL now invokes EVALUATOR 3, EVALUATOR 4, and XTL with their new "-source" option, so that error and warning messages regarding temporal logic formulas now display line numbers in the SVL file itself, rather than in the temporary files generated to contain the temporal logic formulas, making it easier for users to modify incorrect MCL and XTL formulas contained in SVL files.
- SVL has been modified so that both EVALUATOR 3 and EVALUATOR 4 can now be used to compute "deadlock" and "livelock" statements.
- SVL does not require anymore that every "property" statement contains at least one verification statement, namely "comparison", "verify", "deadlock", "livelock", or a shell-line command with an "expected" clause.
- In addition to a bug fix, the EXP.OPEN tool was enhanced with a new option "-depend", displaying both the list of EXP files included (directly or transitively) in the input EXP file, and the list of automata, hide, rename, and cut files used (directly or transitively) in the input EXP file.

A paper containing both a tutorial and a survey on compositional verification was published in an international conference [14].

6.4.2. On-the-Fly Test Generation

Participants: Lina Marsso, Radu Mateescu, Wendelin Serwe.

The CADP toolbox provides support for conformance test case generation by means of the TGV tool. Given a formal specification of a system and a test purpose described as an input-output LTS (IOLTS), TGV automatically generates test cases, which assess using black box testing techniques the conformance of a system under test w.r.t. the formal specification. A test purpose describes the goal states to be reached by the test and enables one to indicate parts of the specification that should be ignored during the testing process. TGV does not generate test cases completely on the fly (i.e., *online*), because it first generates the complete test graph (CTG) and then traverses it backwards to produce controllable test cases.

To address these limitations, we developed the prototype tool TESTOR [4] to extract test cases completely on the fly. TESTOR presents several advantages w.r.t. TGV: (i) it has a more modular architecture, based on generic graph transformation components taken from the OPEN/CAESAR libraries (τ-compression, τ-confluence, τ-closure, determinization, resolution of Boolean equation systems); (ii) it is capable of extracting a test case completely on the fly, by exploiting the diagnostic generation features of the Boolean equation system resolution algorithms; (iii) it enables a more flexible expression of test purposes, taking advantage of the multiway rendezvous, a primitive to express communication and synchronization among a set of distributed processes.

[4] http://convecs.inria.fr/software/testor

In 2018, we improved TESTOR and TGV as follows:

- TESTOR has been ported to the Windows operating system.
- TESTOR can now be directly connected (by means of Unix pipes) to a system under test (SUT), executing the test case, rather than generating an abstract test-case that has to be connected to the SUT.
- We revised the architecture of TESTOR, so that the interface for the user is more similar to the one of TGV. This enables a user to easily switch between both tools.
- Taking advantage of the similar interfaces, we merged the non-regression test bases of TESTOR and TGV.
- We also fixed a bug and added a new option "-self" to TGV, reducing the number of warning messages.

These activities led to a new version 3.0 of TESTOR and two publications in international conferences [24], [18].

6.4.3. Other Component Developments

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

In 2018, several components of CADP have been improved as follows:

- The CADP toolbox now contains a new tool named SCRUTATOR for pruning Labeled Transition Systems on the fly.
- The OPEN/CAESAR environment was enriched with a new SOLVE_2 library for solving linear equation systems on the fly.
- Two manual pages ("bes" and "seq") have been added, which provide standalone definitions of CADP's BES format for Boolean Equation Systems and SEQ format for execution traces. The OPEN/CAESAR manual pages have been enhanced to give full prototypes for function parameters.
- The CADP toolbox has been ported to Solaris 11 and to SunOS 5.11 OpenIndiana "Hipster". CADP has also been ported to macOS 10.14 "Mojave" and a 64-bit version of CADP is now available for macOS.
- We also designed new C functions for handling path names in order to replace the traditional POSIX primitives basename(), dirname(), and realpath(), which suffer from limitations and ambiguities.

6.5. Real-Life Applications and Case Studies

6.5.1. Autonomous Resilience of Distributed IoT Applications in a Fog Environment

Participants: Umar Ozeer, Gwen Salaün.

Fog computing provides computing, storage and communication resources (and devices) at the edge of the network, near the physical world (PW). These end-devices nearing the physical world can have interesting properties such as short delays, responsiveness, optimized communications and privacy, which are especially appealing to IoT (Internet of Things) applications. However, IoT devices in the fog have low stability and are prone to failures.

In the framework of the collaboration with Orange Labs (see § 7.1.1), we are working on the key challenge of providing reliable services. This may be critical in this context since the non-containment of failures may impact the physical world. For instance, the failure of a smoke detector or a lamp in a smart home for elderly/medicated people may be hazardous. The design of such resilience solutions is complex due to the specificities of the environment, i.e., (i) dynamic infrastructure, where entities join and leave without synchronization; (ii) high heterogeneity in terms of functions, communication models, network, processing and storage capabilities; and (iii) cyber-physical interactions, which introduce non-deterministic and physical world's space and time dependent events.

In 2018, our work focused on proposing an end-to-end resilience approach for stateful IoT applications in the fog taking into account the three specificities mentioned above. The resilience protocol is functionally divided into four phases: (i) state-saving; (ii) monitoring and failure detection; (iii) failure notification and reconfiguration; and (iv) decision and recovery. The protocol implements a combination of different state-saving techniques based on rules and policies to cope with the heterogeneous nature of the environment and recover from failures in a consistent way, including PW-consistency. This work led to a publication in an international conference [25].

To illustrate our protocol at work, we mounted a smart home testbed with objects that can be found in real-life smart homes to test our solution. Our resilience approach was also implemented as a framework and deployed onto the testbed. The empirical results showed that multiple failures are recovered in an acceptable time in regard to end users. This work led to a publication to appear in an international conference.

6.5.2. *Verified Composition and Deployment of IoT Applications*

Participants: Radu Mateescu, Ajay Muroor Nadumane, Gwen Salaün.

The Internet of Things (IoT) is an interconnection of physical devices and software entities that can communicate and perform meaningful tasks largely without human intervention. The design and development of IoT applications is an interesting problem as these applications are typically dynamic, distributed and, more importantly, heterogeneous in nature.

In the framework of the collaboration with Nokia Bell Labs (see § 7.1.2), we proposed to build and deploy reliable IoT applications using a series of steps: (i) IoT objects and compositions are described using an interface-based behavioural model; (ii) the correctness of the composition is ensured by checking a behavioural compatibility notion that we proposed for IoT systems; and (iii) finally, a deployment plan respecting the dependencies between the objects is generated to facilitate automated deployment and execution of the application.

Regarding implementation, behavioural models and composition are specified in LNT and we take advantage of the CADP toolbox to perform compatibility checks. The deployment is automated using the Majord'Home platform developed by Nokia Bell Labs. The entire implementation is packaged as a Web tool available for end-users. This work led to a publication to appear in an international conference.

6.5.3. *Memory Protection Unit*

Participants: Hubert Garavel, Radu Mateescu, Wendelin Serwe.

Asynchronous circuits have key advantages in terms of low energy consumption, robustness, and security. However, the absence of a global clock makes the design prone to deadlock, livelock, synchronization, and resource-sharing errors. Formal verification is thus essential for designing such circuits, but it is not widespread enough, as many hardware designers are not familiar with it and few verification tools can cope with asynchrony on complex designs. In the framework of the SECURIOT-2 project (see § 8.2.2.1), we are interested in the rigorous design of asynchronous circuits used in the secure elements for IoT devices developed in the project.

In collaboration with Aymane Bouzafour and Marc Renaudin (Tiempo Secure), we suggested an extension of Tiempo's industrial design flow for asynchronous circuits, based upon the standard Hardware Description Language SystemVerilog (SV), with the formal verification capabilities provided by CADP. This was achieved by translating SV descriptions into LNT, expressing correctness properties in MCL, and verifying them using the EVALUATOR model checker of CADP. It turned out that the constructs of SV and LNT are in close correspondence, and that the synthesizable SV subset can be entirely translated into LNT. The MCL language was also shown adequate for expressing all property patterns relevant for asynchronous circuits.

The practicality of the approach was demonstrated on an asynchronous circuit (4000 lines of SV) implementing a memory protection unit (MPU). The MPU block exhibits a high degree of internal concurrency, comprising 660 parallel execution flows and 250 internal communication channels. The corresponding state space was generated compositionally, by identifying a suitable minimization and composition strategy described in

SVL (the largest intermediate state space had more than 116 million states and 862 million transitions). A set of 184 MCL properties were successfully verified on the state space, expressing the correct initialization of the MPU configuration registers, the mutual exclusion of read and write operations on registers, the correct responses to stimuli, and the security requirements related to the many access-control policies enforced by the MPU. This work led to a publication in an international conference [17].

6.5.4. TLS 1.3 Handshake Protocol

Participants: Lina Marsso, Radu Mateescu.

Security services are extensively used in fields like online banking, e-government, online shopping, etc. To ensure a secure communication between peers in terms of authenticity, privacy, and data integrity, cryptographic protocols are applied to regulate the data transfer. These protocols provide a standardized set of rules and methods for the interaction between peers. The Transport Layer Security (TLS) is a widely used security protocol, encompassing a set of rules for the communication between clients and servers, and relying on public-key cryptography to ensure integrity of exchanged data. However, despite multiple prevention measurements, several vulnerabilities (such as Heartbleed and DROWN), have been discovered recently. Therefore, testing the implementations of security protocols is still a crucial issue.

In the framework of the RIDINGS PHC project (see § 8.3.1), we are interested in testing protocols and distributed systems. In collaboration with Josip Bozic and Franz Wotawa (TU Graz, Austria), we undertook the formal modelling of the draft TLS 1.3 handshake protocol [5]. Taking as input the informal description of TLS 1.3 in the draft standard, we developed a formal model (1293 lines of LNT) specifying the handshake messages and client-server interactions. As far as we are aware, this is the first formal model of the draft TLS 1.3 handshake.

We used our LNT model for conformance testing with the OpenSSL version 1.0.1e implementation of the TLS protocol [6]. We defined three test purposes specifying requirements from the draft TLS 1.3 handshake, and applied the newly developed TESTOR tool (see § 6.4.2) to generate the test cases from the LNT model and each test purpose. The execution of these test cases on the OpenSSL implementation spotted a discrepancy of the server's response to a client certificate request w.r.t. the draft TLS 1.3 standard. This work led to a publication in an international workshop [18].

6.5.5. Message Authenticator Algorithm

Participants: Hubert Garavel, Lina Marsso.

The Message Authenticator Algorithm (MAA) is one of the first cryptographic functions for computing a Message Authentication Code. Between 1987 and 2001, the MAA was adopted in international standards (ISO 8730 and ISO 8731-2) to ensure the authenticity and integrity of banking transactions. The MAA also played a role in the history of formal methods, as the National Physical Laboratory (NPL, United Kingdom) developed, in the early 90s, three formal, yet non-executable, specifications of the MAA in VDM, Z, and LOTOS abstract data types.

In 2018, we examined how the new generation of formal methods can cope with the MAA case study. We specified the MAA in both LOTOS and LNT and checked these specifications using the CADP tools. The C code generated by the CADP compilers was executed w.r.t. a set of reference MAA test vectors, as well as supplementary test vectors devised to improve the coverage of byte permutations and message segmentation. This enabled us to detect and correct several errors in the reference test vectors given in the ISO 8730 and ISO 8731-2 standards. This work led to a publication in an international workshop [22].

6.5.6. Other Case Studies

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

[5] https://tools.ietf.org/html/draft-ietf-tls-tls13-24
[6] https://www.openssl.org/

Based on the work described above, the demo examples of the CADP toolbox have been enriched. Two new demo examples have been added: demo_06 (Transport Layer Security v1.3 handshake protocol specified in LNT), and demo_11 (a hardware block implementing a Dynamic Task Dispatcher). The demo_12 (Message Authenticator Algorithm) is now documented in a publication [22]. The demo_17 (distributed leader election protocol) has been converted from LOTOS to LNT. Finally, most existing demo examples have been updated to reflect the evolution of the MCL v3 and SVL languages.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Grants with Industry

7.1.1. Orange Labs

Participants: Umar Ozeer, Gwen Salaün.

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

7.1.2. Nokia Bell Labs

Participants: Radu Mateescu, Ajay Muroor Nadumane, Gwen Salaün.

Ajay Muroor Nadumane is supported by a PhD grant (from October 2017 to October 2020) from Nokia Bell Labs (Nozay) on IoT service composition supported by formal methods, under the supervision of Gwen Salaün (CONVECS), Radu Mateescu (CONVECS), Ludovic Noirie, and Michel Le Pallec (Nokia Bell Labs).

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. ARC6 Programme

Participants: Lina Marsso, Radu Mateescu [correspondent], Wendelin Serwe.

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

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

8.2. National Initiatives

8.2.1. PIA (Programme d'Investissements d'Avenir)

8.2.1.1. CAPHCA

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

CAPHCA (*Critical Applications on Predictable High-Performance Computing Architectures*) is a project funded by the PIA. The project, led by IRT Saint-Exupéry (Toulouse), involves a dozen of industrial partners (among which Airbus, CS Systèmes d'Information, Synopsis, and Thalès Avionics), the University Paul Sabatier (Toulouse), and Inria Grenoble – Rhône-Alpes (CONVECS and SPADES project-teams). CAPHCA addresses the dual problem of achieving performance and determinism when using new, high performance, multicore System-on-Chip (SoC) platforms for the deployment of real-time, safety-critical applications. The methodology adopted by CAPHCA consists in building a pragmatic combination of methods, tools, design constraints and patterns deployable at a short-term horizon in the industrial domains targeted in the project.

CAPHCA started in December 2017 for four years. The main contributions of CONVECS to CAPHCA are the detection of concurrency errors in parallel applications by means of formal methods and verification techniques.

8.2.2. Competitivity Clusters

8.2.2.1. SECURIOT-2

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

SECURIOT-2 is a project funded by the FUI (*Fonds Unique Interministériel*) within the *Pôle de Compétitivité* Minalogic. The project, led by Tiempo Secure (Grenoble), involves the SMEs (*Small and Medium Enterprises*) Alpwise, Archos, Sensing Labs, and Trusted Objects, the Institut Fourier and the VERIMAG laboratories of Université Grenoble Alpes, and CONVECS. SECURIOT-2 aims at developing a secure micro-controller unit (SMCU) that will bring to the IoT a high level of security, based on the techniques used for smart cards or electronic passports. The SMCU will also include an original power management scheme adequate with the low power consumption constraints of the IoT.

SECURIOT-2 started in September 2017 for three years. The main contributions of CONVECS to SECURIOT-2 are the formal modeling and verification of the asynchronous hardware implementing the secure elements developed by the project partners.

8.2.3. Other National Collaborations

We had sustained scientific relations with the following researchers:

- Xavier Etchevers (Orange Labs, Meylan),
- Fabrice Kordon and Lom Messan Hillah (LIP6, Paris),
- Eric Jenn and Viet Anh Nguyen (IRT Saint-Exupéry, Toulouse),
- Ioannis Parissis and Oum-El-Kheir Aktouf (LCIS, Valence),
- Pascal Poizat (LIP6, Paris).

8.3. European Initiatives

8.3.1. Collaborations in European Programs, Except FP7 & H2020

Program: PHC Amadeus

Project acronym: RIDINGS

Project title: Rigourous Development of GALS Systems

Duration: January 2017 – December 2018

Coordinator: Inria Grenoble – Rhône-Alpes / CONVECS

Other partners: TU Graz, Institute of Software Technology (Austria)

Abstract: GALS systems, composed of synchronous components (driven by local clocks) that communicate through a network, are increasingly spreading with the development of the IoT. GALS systems are intrinsically complex due to the interplay of synchronous and asynchronous aspects, which make their development and debugging difficult. Therefore, it is necessary to adopt rigorous design methodologies, based on formal methods assisted by efficient validation tools. The RIDINGS project aims at enhancing the design flow of a GALS system by integrating the automatic generation of conformance tests from the formal model and the temporal properties used for verifying the system. This yields a double benefit for the designer: (i) it makes possible to check that a physical implementation conforms to the verified model; (ii) the development cost of the model and properties is distributed on the verification and testing phases of the design process, therefore increasing the return on investment.

8.3.2. Collaborations with Major European Organizations

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

8.4. International Initiatives

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

8.4.1. Inria International Partners

8.4.1.1. Informal International Partners

Saarland University (Germany): we collaborate on a regular basis with the DEPEND (*Dependable Systems and Software*) research group headed by Holger Hermanns, who received an ERC Advanced Grant ("POWVER") in 2016.

8.4.2. Other International Collaborations

In 2018, we had scientific relations with several universities and institutes abroad, including:

- University of Málaga, Spain (Francisco Durán),
- University of Cali, Colombia (Camilo Rocha),
- University of Zaragoza, Spain (José Ignacio Requeno),
- ISTI/CNR, Pisa, Italy (Franco Mazzanti),
- RWTH Aachen, Germany (Joost-Pieter Katoen),
- Saarland University, Germany (Holger Hermanns),
- Eindhoven University of Technology, The Netherlands (Anton Wijs and Sander de Putter).

8.5. International Research Visitors

8.5.1. Visits of International Scientists

- H. Garavel is an invited professor at Saarland University (Germany) as a holder of the Gay-Lussac Humboldt Prize.
- Josip Bozic, Birgit Hofer, Hermann Felbinger, and Franz Wotawa (TU Graz, Austria) visited us from March 5 to March 9, 2018 in the framework of the RIDINGS PHC project (see § 8.3.1).
- G. Salaün visited the University of Málaga (Spain) from May 30 to June 13 and from December 16 to December 22, 2018.

[7] http://fmics.inria.fr

- L. Marsso and R. Mateescu visited TU Graz (Austria) from August 20 to August 24, 2018 in the framework of the PHC RIDINGS project.

The annual CONVECS seminar was held in Dullin (France) on July 10–12, 2018. The following invited scientists attended the seminar:

- Eric Jenn (IRT Saint-Exupéry / Thales Avionics) gave on July 10, 2018 a talk entitled "*The CAPHCA Project, or How to Be Fast and Reasonable*".
- Viet Anh Nguyen (IRT Saint-Exupéry) gave on July 12, 2018 a talk entitled "*Cache-conscious Offline Real-time Scheduling for Multi-core Platforms: Algorithms and Implementation*".
- Yliès Falcone (CORSE project-team) gave on July 11, 2018 a talk entitled "*Some Recent Work on the Runtime Monitoring of Systems*".

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Member of the Organizing Committees

- H. Garavel is a member of the model board [8] of MCC (*Model Checking Contest*). In 2018, he helped preparing new models (especially those in the NUPN format) and verified, using the CÆSAR.BDD tool of CADP, the forms describing all benchmark models submitted by the contest participants; this revealed a number of inconsistencies. The results of MCC'2018 have been published online [51].
- Together with Peter Höfner (Data61, CSIRO, Sydney, Australia), H. Garavel set up a model repository (hosted on the Gforge of Inria) to collect and archive formal models of real systems; this infrastructure is used by the series of MARS workshops [9]. This repository currently contains 21 models, one of which (a Transport Layer Security protocol) was deposited in 2018 by CONVECS.
- G. Salaün is member of the steering committee of the ACM SAC-SVT (*Symposium of Applied Computing – Software Verification and Testing Track*) conference series since 2018.
- G. Salaün is member of the steering committee of the SEFM (*International Conference on Software Engineering and Formal Methods*) conference series since 2014.
- G. Salaün is member of the steering committee of the FOCLASA (*International Workshop on Foundations of Coordination Languages and Self-Adaptative Systems*) workshop series since 2011.

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

- G. Salaün was co-chair of ACM SAC-SOAP'2018 (*33rd ACM Symposium of Applied Computing – Service-Oriented Architectures and Programming Track*), Pau, France, April 9–13, 2018.
- G. Salaün was workshops co-chair at STAF'2018 (*Software Technologies: Applications and Foundations*), Toulouse, France, June 25-29, 2018.
- W. Serwe was co-chair of MARS'2018 (*3rd Workshop on Models for Formal Analysis of Real Systems*) affiliated with ETAPS'2018 (*European Joint Conferences on Theory and Practice of Software*), Thessaloniki, Greece, April 20, 2018.

9.1.2.2. Member of the Conference Program Committees

- H. Garavel was program committee member of FVPS'2018 (*International Workshop on Formal Verification of Physical Systems*), Hagenberg, Austria, August 17, 2018.

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

- F. Lang was program committee member of SPIN'2018 (*25th International SPIN Symposium on Model Checking of Software*), Málaga, Spain, June 20–22, 2018.
- R. Mateescu was program committee member of ICTSS'2018 (*30th IFIP International Conference on Testing Software and Systems*), Cádiz, Spain, October 1–3, 2018.
- R. Mateescu was program committee member of FMICS'2018 (*23rd International Conference on Formal Methods for Industrial Critical Systems*), Maynooth, Ireland, September 3–4, 2018.
- G. Salaün was program committee member of CAL'2018 (*11ème Conférence francophone sur les Architectures Logicielles*), Grenoble, France, June 14–15, 2018.
- G. Salaün was program committee member of COMPSAC'2018 (*IEEE International Conference on Computers, Software, and Applications*), Tokyo, Japan, July 23–27, 2018.
- G. Salaün was program committee member of DATAMOD'2018 (*7th International Symposium "From Data to Models and Back"*), Toulouse, France, June 25–26, 2018.
- G. Salaün was program committee member of FACS'2018 (*15th International Conference on Formal Aspects of Component Software*), Pohang, Korea, October 10–12, 2018.
- G. Salaün was program committee member of FOCLASA'2018 (*16th International Workshop on Foundations of Coordination Languages and Self-Adaptative Systems*), Toulouse, France, June 26, 2018.
- G. Salaün was program committee member of HPCS-4PAD'2018 (*5th International Symposium on Formal Approaches to Parallel and Distributed Systems*), Orléans, France, July 16–20, 2018.
- G. Salaün was program committee member of SEFM'2018 (*16th International Conference on Software Engineering and Formal Methods*), Toulouse, France, June 27–29, 2018.
- G. Salaün was program committee member of SAC-SVT'2018 (*33rd ACM Symposium on Applied Computing - Software Verification and Testing Track*), Pau, France, April 9–13, 2018.

9.1.2.3. Reviewer

- G. Barbon was a reviewer for COMPSAC'2018, DATAMOD'2018, and SEFM'2018.
- F. Lang was a reviewer for FoSSaCS'2018 (*21st International Conference on Foundations of Software Science and Computation Structures*).
- A. Muroor Nadumane was a reviewer for COMPSAC'2018, FACS'2018, FOCLASA'2018, SAC-SVT'2018, and SEFM'2018.
- U. Ozeer was a reviewer for COMPSAC'2018.
- R. Mateescu and W. Serwe were reviewers for the Festschrift in honor of Bernhard Steffen.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

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

9.1.3.2. Reviewer - Reviewing Activities

- F. Lang was a reviewer for FAoC (*Formal Aspects of Computing*) and ToCL (*ACM Transactions on Computational Logic*).
- R. Mateescu was a reviewer for STTT, ToCL, and TOR (*IEEE Transactions on Reliability*).
- A. Muroor Nadumane was a reviewer for STTT.
- G. Salaün was a reviewer for FAoC, JCC (*Journal of Computer and Communications*), JLAMP (*Journal of Logical and Algebraic Methods in Programming*), SCP (*Science of Computer Programming*), TSE (*IEEE Transactions on Software Engineering*), TSI (*Technique et Science Informatiques*).

9.1.4. Software Dissemination and Internet Visibility

The CONVECS project-team distributes several software tools, among which the CADP toolbox.

In 2018, the main facts are the following:

- We prepared and distributed twelve successive versions (2018-a to 2018-l) of CADP.
- We were requested to grant CADP licenses for 381 different computers in the world.

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

By the end of December 2018, the CADP forum [11], opened in 2007 for discussions regarding the CADP toolbox, had over 426 registered users and over 1847 messages had been exchanged.

Also, for the 2018 edition of the Model Checking Contest, 4 families of models generated using CADP (totalling 101 Nested-Unit Petri Nets) were provided.

We contributed to Wikipedia as follows:

- We created a new page about the Message Authenticator Algorithm (https://en.wikipedia.org/wiki/Message_Authenticator_Algorithm)
- We added the three last paragraphs of the section about constructing integer numbers (https://en.wikipedia.org/wiki/Integer#Construction)

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 RichTest Tool for Message-Passing Concurrent Programs [33]
- The REFINER Tool for Verifying Behavioural Model-to-Model Transformations [63]
- The ALVIS Tool for Modelling and Verification of Real-Time Systems [60]
- The COSTO Tool for Component-Based Software [30]
- The IDCM Tool for Analyzing UML Architectures [52]
- The OCARINA Tool and its Extension AADL2LNT for Analysing AADL Descriptions [58]
- The aZiZa Tool for Heterogeneous Behavioural Models [31]
- The Papyrus-RT Tool for Model-driven Engineering with UML-RT [62]
- Formal Analysis of Distributed Reactive Applications [35], [34]

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

- Formal Modelling and Verification of an Automatic Train Supervision System [56], [57]
- Verification of Highly-Optimized Concurrent Data Structures [61]
- Detection of Data Breaches in Banking Transaction Processes [54]
- Verification of Visibility-Based Properties on Multiple Moving Robots [59]
- Experimental Analysis of Compositional State Space Generation Strategies [64]
- Product-Line for Families of Program Translators [32]

9.1.5. Invited Talks

- H. Garavel participated in the workshop "*Safety of Future Systems: Science meets Industry*" organized by the Lorentz Center (Leiden, The Netherlands) on April 9–13, 2018. He gave a lecture entitled "*Concurrency Theory Meets IoT*".
- H. Garavel gave an invited talk entitled "*Benchmarking Implementations of Term Rewriting and Pattern Matching in Algebraic, Functional, and Object-Oriented Languages - The 4th Rewrite Engines Competition*" at WRLA'2018 (*12th International Workshop on Rewriting Logic and its Applications*), Thessaloniki, Greece, April 14–15, 2018.

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

- L. Marsso gave a talk and presented a poster entitled "*Automated Test Generation for GALS Systems*" on March 8, 2018 at the 2nd year PhD LIG Day.
- L. Marsso gave a talk entitled "*Generation with CADP of Relevant Scenarios for Testing Autonomous Cars*" at the seminar of the group TransForm (*Méthodes formelles pour les systèmes de transport*) held in Villeneuve d'Ascq on November 22, 2018.
- L. Marsso gave a talk entitled "*Automated Test Generation for GALS Systems*" at the Scientific day of ARC 6 held in Lyon on November 29, 2018.
- R. Mateescu participated to the Kobe-Grenoble workshop organized by UGA in Grenoble on February 26–27, 2018. He gave a talk entitled "*Rigorous Design of PLC Networks using Formal Methods*" on February 26.
- A. Muroor Nadumane gave a talk entitled "*Building Reliable IoT Application and Beyond*" at the Inria-Nokia Bell Labs meeting held in Paris on November 27, 2018.
- U. Ozeer gave a talk and presented a poster entitled "*Autonomous Resilience of Distributed IoT Applications in a Fog Environment*" on March 8, 2018 at the 2nd year PhD LIG Day.
- U. Ozeer gave a talk entitled "*Autonomous Resilience of Distributed IoT Applications in a Fog Environment*" at the seminar on IoT research projects held at Orange Labs, Meylan, on March 29–30, 2018.
- U. Ozeer presented a poster entitled "*Autonomous Resilience of Distributed IoT Applications in a Fog Environment*" at the LIG seminar "*Regards sur le futur de l'informatique*" held in Grenoble on April 6, 2018.
- U. Ozeer gave a talk entitled "*Resilience of Distributed IoT Applications in a Dynamic Fog Environment*" at the IO Labs seminar held in Paris on October 30–31, 2018.
- G. Salaün gave a keynote talk entitled "*Safe Composition of Software Services*" at DATAMOD'2018, Toulouse, France, on June 26, 2018.

9.1.6. Research Administration

- H. Garavel was appointed to the Executive Commission in charge of International Relations at COMUE Université Grenoble Alpes.
- F. Lang is chair of the "*Commission du développement technologique*", which is in charge of selecting R&D projects for Inria Grenoble – Rhône-Alpes.
- R. Mateescu is the scientific correspondent of the European and International Partnerships 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 Committee of the PCS (*Pervasive Computing Systems*) action of the PERSYVAL Labex.
- W. Serwe is (together with Laurent Lefèvre from the AVALON Inria project-team) correspondent in charge of the 2017 Inria activity reports at Inria Grenoble – Rhône-Alpes.
- W. Serwe is a member of the "*Comité de Centre*" at Inria Grenoble – Rhone-Alpes.
- W. Serwe is "*chargé de mission*" for the scientific axis *Formal Methods, Models, and Languages* of the LIG laboratory.

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 Université Grenoble Alpes (UGA).

In 2018, we carried out the following teaching activities:

G. Barbon and W. Serwe supervised each a group of six teams in the context of the "*projet Génie Logiciel*" (55 hours "*équivalent TD*", consisting in 16 hours of lectures, plus supervision and evaluation), ENSIMAG, January 2018.

F. Lang and R. Mateescu gave a lecture on "*Modeling and Analysis of Concurrent Systems: Models and Languages for Model Checking*" (27 hours "*équivalent TD*") to third year students of ENSIMAG and second year students of the MOSIG (*Master of Science in Informatics at Grenoble*).

F. Lang gave a course on "*Formal Software Development Methods*" (7.5 hours "*équivalent TD*") in the framework of the "*Software Engineering*" lecture given to first year students of the MOSIG.

F. Lang and W. Serwe provided a 6-hour training about the CADP toolbox to Eric Jenn, Nicolas Hili, and Sun Wei Tsun (IRT Saint-Exupéry, Toulouse, France) on June 28, 2018.

L. Marsso gave a course on "*Algorithms and Web Programming*" (64 hours "*équivalent TD*") at the department MMI of IUT1 (UGA).

A. Muroor Nadumane gave a course on "*Object Oriented Programming*" (42 hours "*équivalent TD*") at the department MMI of IUT1 (UGA).

G. Salaün taught about 230 hours of classes (algorithmics, Web development, object-oriented programming, iOS programming) at the department MMI of IUT1 (UGA). He is also headmaster of the "*Services Mobiles et Interface Nomade*" (SMIN) professional licence (3rd year of university) at IUT1/UGA.

9.2.2. Supervision

PhD in progress: L. Marsso, "*Formal Methods for Testing Networks of Controllers*, Université Grenoble Alpes, since October 2016, R. Mateescu, W. Serwe, I. Parissis, and Ch. Deleuze

PhD in progress: A. Muroor Nadumane, "*Softwarization of Everything: IoT Service Composition*, Université Grenoble Alpes, since October 2017, G. Salaün, R. Mateescu, L. Noirie, and M. Le Pallec

PhD in progress: U. Ozeer, "*Autonomous Resilience of Applications in a Largely Distributed Cloud Environment*, Université Grenoble Alpes, since November 2016, L. Letondeur, G. Salaün, F.-G. Ottogalli, and J.-M. Vincent

9.2.3. Juries

- G. Salaün was PhD committee member for Gustavo García Pascual's PhD thesis, entitled "*Optimizing Mobile Applications by Exploiting Variability Models at Runtime*", defended at University of Málaga (Spain) on December 18, 2018.

10. Bibliography

Major publications by the team in recent years

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

[2] H. EVRARD, F. LANG. *Automatic Distributed Code Generation from Formal Models of Asynchronous Processes Interacting by Multiway Rendezvous*, in "Journal of Logical and Algebraic Methods in Programming", March 2017, vol. 88, 33 p. [*DOI :* 10.1016/J.JLAMP.2016.09.002], https://hal.inria.fr/hal-01412911

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

[4] 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

[5] H. GARAVEL, F. LANG, W. SERWE. *From LOTOS to LNT*, in "ModelEd, TestEd, TrustEd - Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday", J.-P. KATOEN, R. LANGERAK, A. RENSINK (editors), Lecture Notes in Computer Science, Springer, October 2017, vol. 10500, pp. 3 - 26 [*DOI :* 10.1007/978-3-319-68270-9_1], https://hal.inria.fr/hal-01621670

[6] F. JEBALI, F. LANG, R. MATEESCU. *Formal Modelling and Verification of GALS Systems Using GRL and CADP*, in "Formal Aspects of Computing", April 2016, vol. 28, n° 5, pp. 767–804 [*DOI :* 10.1007/s00165-016-0373-3], https://hal.inria.fr/hal-01290449

[7] 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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[8] G. BARBON. *Debugging of Behavioural Models using Counterexample Analysis*, Université Grenoble Alpes, December 2018

Articles in International Peer-Reviewed Journals

[9] L. AKROUN, G. SALAÜN. *Automated verification of automata communicating via FIFO and bag buffers*, in "Formal Methods in System Design", June 2018, vol. 52, n° 3, pp. 260 - 276 [*DOI :* 10.1007/s10703-017-0285-8], https://hal.inria.fr/hal-01898159

[10] F. DURÁN, C. ROCHA, G. SALAÜN. *Stochastic Analysis of BPMN with Time in Rewriting Logic*, in "Science of Computer Programming", December 2018, vol. 168, pp. 1-17 [*DOI :* 10.1016/j.scico.2018.08.007], https://hal.inria.fr/hal-01866289

[11] F. KORDON, H. GARAVEL, L. HILLAH, E. PAVIOT-ADET, L. JEZEQUEL, F. HULIN-HUBARD, E. AMPARORE, M. BECCUTI, B. BERTHOMIEU, H. EVRARD, P. G. JENSEN, D. LE BOTLAN, T. LIEBKE, J. MEIJER, J. SRBA, Y. THIERRY-MIEG, J. VAN DE POL, K. WOLF. *MCC'2017-The Seventh Model Checking Contest*, in "LNCS Transactions on Petri Nets and Other Models of Concurrency", 2018, vol. 11090, pp. 181-209 [*DOI :* 10.1007/978-3-662-58381-4_9], https://hal.inria.fr/hal-01917492

[12] A. KRISHNA, P. POIZAT, G. SALAÜN. *Checking Business Process Evolution*, in "Science of Computer Programming", January 2019, vol. 170, pp. 1-26 [*DOI :* 10.1016/j.scico.2018.09.007], https://hal.inria.fr/hal-01920273

[13] R. MATEESCU, J. I. REQUENO. *On-the-Fly Model Checking for Extended Action-Based Probabilistic Operators*, in "International Journal on Software Tools for Technology Transfer", October 2018, vol. 20, n° 5, pp. 563–587 [*DOI :* 10.1007/s10009-018-0499-0], https://hal.inria.fr/hal-01862754

Invited Conferences

[14] H. GARAVEL, F. LANG, L. MOUNIER. *Compositional Verification in Action*, in "FMICS 2018 - 23rd International Conference on Formal Methods for Industrial Critical Systems", Maynooth, Ireland, LNCS, Springer, September 2018, vol. 11119, pp. 189-210 [*DOI :* 10.1007/978-3-030-00244-2_13], https://hal.inria.fr/hal-01890246

[15] H. GARAVEL, M.-A. TABIKH, I.-S. ARRADA. *Benchmarking Implementations of Term Rewriting and Pattern Matching in Algebraic, Functional, and Object-Oriented Languages - The 4th Rewrite Engines Competition*, in "Proceedings of the 12th International Workshop on Rewriting Logic and its Applications (WRLA'18)", Thessaloniki, Greece, 2018, https://hal.inria.fr/hal-01883212

International Conferences with Proceedings

[16] G. BARBON, V. LEROY, G. SALAÜN. *Counterexample Simplification for Liveness Property Violation*, in "SEFM 2018 - 16th International Conference on Software Engineering and Formal Methods", Toulouse, France, LNCS, June 2018, vol. 10886, pp. 173-188 [*DOI :* 10.1007/978-3-319-92970-5_11], https://hal.inria.fr/hal-01818790

[17] A. BOUZAFOUR, M. RENAUDIN, H. GARAVEL, R. MATEESCU, W. SERWE. *Model-checking Synthesizable SystemVerilog Descriptions of Asynchronous Circuits*, in "ASYNC'18 - 24th IEEE International Symposium on Asynchronous Circuits and Systems", Vienne, Austria, May 2018, https://hal.inria.fr/hal-01777093

[18] J. BOZIC, L. MARSSO, R. MATEESCU, F. WOTAWA. *A Formal TLS Handshake Model in LNT*, in "MARS/VPT 2018 - 3nd Workshop on Models for Formal Analysis of Real Systems and 6th International Workshop on Verification and Program Transformation", Thessaloniki, Greece, April 2018, vol. 268, pp. 1 - 40 [*DOI :* 10.4204/EPTCS.268.1], https://hal.inria.fr/hal-01779151

[19] M. CORTES-CORNAX, A. KRISHNA, A. MOS, G. SALAÜN. *Automated Analysis of Industrial Workflow-based Models*, in "SAC 2018 - 33rd Annual ACM Symposium on Applied Computing", Pau, France, Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018, ACM, April 2018, pp. 120-127 [*DOI :* 10.1145/3167132.3167142], https://hal.inria.fr/hal-01781315

[20] F. DURÁN, C. ROCHA, G. SALAÜN. *Computing the Parallelism Degree of Timed BPMN Processes*, in "FOCLASA 2018 - 16th International Workshop on Foundations of Coordination Languages and Self-Adaptative Systems", Toulouse, France, June 2018, pp. 1-16 [*DOI :* 10.1007/978-3-030-04771-9_24], https://hal.inria.fr/hal-01961952

[21] F. DURÁN, C. ROCHA, G. SALAÜN. *Symbolic Specification and Verification of Data-aware BPMN Processes using Rewriting Modulo SMT*, in "WRLA 2018 : 12th International Workshop on Rewriting Logic and its Applications", Thessaloniki, Greece, April 2018, pp. 1-20, https://hal.inria.fr/hal-01866268

[22] H. GARAVEL, L. MARSSO. *Comparative Study of Eight Formal Specifications of the Message Authenticator Algorithm*, in "MARS/VPT 2018 - 3nd Workshop on Models for Formal Analysis of Real Systems and the 6th International Workshop on Verification and Program Transformation", Thessaloniki, Greece, April 2018, vol. 268, pp. 41 - 87 [*DOI :* 10.4204/EPTCS.268.2], https://hal.inria.fr/hal-01775332

[23] B. HOFER, R. MATEESCU, W. SERWE, F. WOTAWA. *Using LNT Formal Descriptions for Model-Based Diagnosis*, in "DX 2018 - 29th International Workshop on Principles of Diagnosis", Warsaw, Poland, August 2018, pp. 1-8, https://hal.inria.fr/hal-01877693

[24] L. MARSSO, R. MATEESCU, W. SERWE. *TESTOR: A Modular Tool for On-the-Fly Conformance Test Case Generation*, in "TACAS 2018 - 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems", Thessaloniki, Greece, Lecture Notes in Computer Science, Springer, April 2018, vol. 10806, pp. 211-228 [*DOI :* 10.1007/978-3-319-89963-3_13], https://hal.inria.fr/hal-01777861

[25] U. OZEER, X. ETCHEVERS, L. LETONDEUR, F.-G. OTTOGALLI, G. SALAÜN, J.-M. VINCENT. *Resilience of Stateful IoT Applications in a Dynamic Fog Environment*, in "EAI International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '18)", New York, United States, November 2018, pp. 1-10 [*DOI :* 10.1145/3286978.3287007], https://hal.archives-ouvertes.fr/hal-01927286

Conferences without Proceedings

[26] U. OZEER, L. LETONDEUR, F.-G. OTTOGALLI, G. SALAÜN, J.-M. VINCENT. *Designing and Implementing Resilient IoT Applications in the Fog: A Smart Home Use Case*, in "ICIN 2019 2019 - 22nd Conference on Innovation in Clouds, Internet and Networks", Paris, France, February 2019, pp. 1-3, https://hal.archives-ouvertes.fr/hal-01979686

Books or Proceedings Editing

[27] C. CANAL, G. SALAÜN (editors). *Preface: Special issue on Foundations of Coordination Languages and Self-adaptive Systems*, Elsevier, December 2018, vol. 168, pp. 169 - 170 [*DOI :* 10.1016/J.SCICO.2018.09.003], https://hal.inria.fr/hal-01898179

[28] R. MATEESCU (editor). *Recent advances in interactive and automated analysis*, Springer Verlag, April 2018, vol. 20, n^{o} 2, pp. 119 - 123 [*DOI :* 10.1007/S10009-017-0477-Y], https://hal.inria.fr/hal-01766570

[29] M. MAZZARA, I. OBER, G. SALAÜN (editors). *Software Technologies: Applications and Foundations (STAF 2018)*, 2018 [*DOI :* 10.1007/978-3-030-04771-9], https://hal.inria.fr/hal-01961961

References in notes

[30] P. ANDRÉ, O. CARDIN. *Trusted Services for Cyber Manufacturing Systems*, in "Proceedings of the 7th Workshop on Service Orientation in Holonic and Multi-agent Manufacturing (SOHOMA'2017), Nantes, France", T. BORANGIU, D. TRENTESAUX, A. THOMAS, O. CARDIN (editors), Springer International Publishing, October 2018, pp. 359–370

[31] C. ATTIOGBÉ. *Mastering Heterogeneous Behavioural Models*, in "Proceedings of the 7th International Conference on Model and Data Engineering (MEDI'2017), Barcelona, Spain", Y. OUHAMMOU, M. IVANOVIC, A. ABELLÓ, L. BELLATRECHE (editors), Lecture Notes in Computer Science, Springer Verlag, October 2017, vol. 10563, pp. 291–299

[32] D. A. O. CAMACHO. *A Product-Line for Families of Program Translators: A Grammar-Based Approach*, Université Catholique de Louvain, Belgium, August 2010

[33] R. CARVER, Y. LEI. *Stateless Techniques for Generating Global and Local Test Oracles for Message-Passing Concurrent Programs*, in "Journal of Systems and Software", February 2018, vol. 136, pp. 237–265

[34] S. CHABANE, R. AMEUR-BOULIFA, M. MEZGHICHE. *Formal Framework for Automated Analysis and Verification of Distributed Reactive Applications*, in "Proceedings of the 1st International Conference on

Embedded and Distributed Systems (EDiS'2017), Oran, Algeria", IEEE Computer Society Press, December 2017

[35] S. CHABANE, R. AMEUR-BOULIFA, M. MEZGHICHE. *Rethinking of I/O-Automata Composition*, in "Proceedings of the 2017 Forum on Specification and Design Languages (FDL'2017), Verona, Italy", F. FUMMI, H. D. PATEL, S. CHAKRABORTY (editors), IEEE Computer Society Press, September 2017, pp. 1–7

[36] D. CHAMPELOVIER, X. CLERC, H. GARAVEL, Y. GUERTE, C. MCKINTY, V. POWAZNY, F. LANG, W. SERWE, G. SMEDING. *Reference Manual of the LNT to LOTOS Translator (Version 6.8)*, January 2019, Inria, Grenoble, France

[37] 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

[38] 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

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

[40] 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

[41] 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

[42] 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

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

[44] 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)

[45] 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

[46] 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

[47] 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

[48] A. HARTMANNS, H. HERMANNS. *In the Quantitative Automata Zoo*, in "Science of Computer Programming", 2015, vol. 112, pp. 3–23

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

[50] H. HERMANNS. *Interactive Markov Chains and the Quest for Quantified Quality*, Lecture Notes in Computer Science, Springer Verlag, 2002, vol. 2428

[51] F. KORDON, H. GARAVEL, L. M. HILLAH, F. HULIN-HUBARD, E. AMPARORE, M. BECCUTI, B. BERTHOMIEU, G. CIARDO, S. DAL ZILIO, T. LIEBKE, A. LINARD, J. MEIJER, A. MINER, J. SRBA, Y. THIERRY-MIEG, J. VAN DE POL, K. WOLF. *Complete Results for the 2018 Edition of the Model Checking Contest*, June 2018, http://mcc.lip6.fr/2018/results.php

[52] T. LAMBOLAIS, A.-L. COURBIS. *Development and Verification of UML Architectures by Refinement and Extension Techniques*, in "Proceedings of the 9th European Congress on Embedded Real Time Software and Systems (ERTS2'2018), Toulouse, France", January 2018

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

[54] F. MARTINELLI, F. MERCALDO, V. NARDONE, A. ORLANDO, A. SANTONE, G. VAGLINI. *Safety Critical Systems Formal Verification Using Execution Traces*, in "Proceedings of the 27th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'2018), Paris, France", IEEE Computer Society Press, June 2018, pp. 247–250

[55] 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

[56] F. MAZZANTI, A. FERRARI. *Ten Diverse Formal Models for a CBTC Automatic Train Supervision System*, in "Proceedings of the 3rd Workshop on Models for Formal Analysis of Real Systems and 6th International Workshop on Verification and Program Transformation (MARS/VPT@ETAPS'2018), Thessaloniki, Greece", J. P. GALLAGHER, R. VAN GLABBEEK, W. SERWE (editors), EPTCS, April 2018, vol. 268, pp. 104–149

[57] F. MAZZANTI, A. FERRARI, G. O. SPAGNOLO. *Towards Formal Methods Diversity in Railways: an Experience Report with Seven Frameworks*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", June 2018, vol. 20, n^{o} 3, pp. 263–288

[58] H. MKAOUAR, B. ZALILA, J. HUGUES, M. JMAIEL. *An Ocarina Extension for AADL Formal Semantics Generation*, in "Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC'2018), Pau, France", ACM Computer Society Press, April 2018, pp. 1402–1409

[59] A. N. SHESHKALANI, R. KHOSRAVI. *Verification of Visibility-Based Properties on Multiple Moving Robots in an Environment with Obstacles*, in "International Journal of Advanced Robotic Systems", July 2018, vol. 15, n^{o} 4, pp. 1–13

[60] M. SZPYRKA, Ł. PODOLSKI, M. WYPYCH. *Modelling and Verification of Real-Time Systems with Alvis*, in "Towards a Synergistic Combination of Research and Practice in Software Engineering", P. KOSIUCZENKO, L. MADEYSKI (editors), Springer International Publishing, 2018, pp. 165–178

[61] X. YANG, J. KATOEN, H. LIN, G. LIU, H. WU. *Branching Bisimulation and Concurrent Object Verification*, in "Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'2018), Luxembourg City, Luxembourg", IEEE Computer Society Press, June 2018, pp. 267–278

[62] R. A. DE OLIVEIRA, J. DINGEL. *Supporting Model Refinement with Equivalence Checking in the Context of Model-driven Engineering with UML-RT*, in "Proceedings of the 15th Workshop on Model-Driven Engineering, Verification and Validation (MoDeVVa'2017) co-located with ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS'2017), Austin, TX, USA", CEUR Workshop Proceedings, September 2017, vol. 2019, pp. 307–314

[63] S. DE PUTTER, A. WIJS. *A Formal Verification Technique for Behavioural Model-to-Model Transformations*, in "Formal Aspects of Computing", January 2018, vol. 30, n^{o} 1, pp. 3–43

[64] S. DE PUTTER, A. WIJS. *To Compose, or Not to Compose, That Is the Question: An Analysis of Compositional State Space Generation*, in "Proceedings of the 22nd International Symposium on Formal Methods (FM'2018), Oxford, UK", K. HAVELUND, J. PELESKA, B. ROSCOE, E. DE VINK (editors), Lecture Notes in Computer Science, Springer Verlag, July 2018, vol. 10951, pp. 485–504