IN PARTNERSHIP WITH:
Institut polytechnique de Grenoble

Université Joseph Fourier (Grenoble)

Activity Report 2019

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

1. Team, Visitors, External Collaborators **1**
2. Overall Objectives **2**
3. Research Program **3**
3.1. New Formal Languages and their Concurrent Implementations 3
3.2. Parallel and Distributed Verification 3
3.3. Timed, Probabilistic, and Stochastic Extensions 4
3.4. Component-Based Architectures for On-the-Fly Verification 5
3.5. Real-Life Applications and Case Studies 5
4. Application Domains **6**
5. Highlights of the Year **6**
6. New Software and Platforms **6**
6.1. CADP 6
6.2. TRAIAN 9
7. New Results **9**
7.1. New Formal Languages and their Implementations 9
7.1.1. LOTOS and LNT Specification Languages 9
7.1.2. Nested-Unit Petri Nets 10
7.1.3. Formal Modeling and Analysis of BPMN 10
7.2. Parallel and Distributed Verification 11
7.2.1. Debugging of Concurrent Systems using Counterexample Analysis 11
7.2.2. Eliminating Data Races in Parallel Programs using Model Checking 11
7.3. Timed, Probabilistic, and Stochastic Extensions 12
7.4. Component-Based Architectures for On-the-Fly Verification 12
7.4.1. Compositional Verification 12
7.4.2. Other Component Developments 13
7.5. Real-Life Applications and Case Studies 14
7.5.1. Autonomous Resilience of Distributed IoT Applications in a Fog Environment 14
7.5.2. Verified Composition and Deployment of IoT Applications 14
7.5.3. Autonomous Car 15
8. Bilateral Contracts and Grants with Industry **16**
8.1.1. Orange Labs 16
8.1.2. Nokia Bell Labs 16
9. Partnerships and Cooperations **16**
9.1. Regional Initiatives 16
9.2. National Initiatives 16
9.2.1. PIA (Programme d'Investissements d'Avenir) 16
9.2.2. Competitivity Clusters 17
9.2.3. Other National Collaborations 17
9.3. European Initiatives 17
9.4. International Initiatives 17
9.4.1. Inria International Partners 17
9.4.2. Other International Collaborations 18
9.5. International Research Visitors 18
10. Dissemination **18**
10.1. Promoting Scientific Activities 18
10.1.1. Scientific Events: Organisation 18
10.1.1.1. General Chair, Scientific Chair 18
10.1.1.2. Member of the Organizing Committees 19
10.1.2. Scientific Events: Selection 19

10.1.2.1. Chair of Conference Program Committees 19
10.1.2.2. Member of the Conference Program Committees 19
10.1.2.3. Reviewer 20
10.1.3. Journal 20
10.1.3.1. Member of the Editorial Boards 20
10.1.3.2. Reviewer - Reviewing Activities 20
10.1.4. Software Dissemination and Internet Visibility 20
10.1.5. Invited Talks 21
10.1.6. Research Administration 21
10.2. Teaching - Supervision - Juries 21
10.2.1. Teaching 21
10.2.2. Supervision 22
10.2.3. Juries 22
10.3. Popularization 22
11. Bibliography **23**

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
A2.1.6. - Concurrent programming
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
B7.2.1. - Smart vehicles
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

Pierre Bouvier [UGA, from Oct 2019]
Lina Marsso [Inria]
Ajay Muroor Nadumane [Inria]
Umar Ozeer [Orange Labs, until Nov 2019]

Technical staff
Armen Inants [Inria, Engineer]

Interns and Apprentices
Pierre Bouvier [Inria, from Feb 2019 until Jun 2019]
Arthur Feyt [Inria, until Jun 2019]
Philippe Ledent [Inria, from Feb 2019 until Jun 2019]
Alejandro Martinez Rivero [UGA, until Jun 2019]

Administrative Assistant
Myriam Etienne [Inria, Administrative Assistant]

Visiting Scientist
Luca Di Stefano [Gran Sasso Science Institute, from Mar 2019 until Jul 2019]

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 [6] 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*) [47], which extends the modal μ-calculus with data-handling primitives, fairness operators encoding generalized Büchi automata, and a functional-like language for describing complex transition sequences. We are working both on the semantic foundations of MCL (extending the language with new temporal and hybrid operators, translating these operators into lower-level formalisms, enhancing the type system, etc.) and also on improving the MCL on-the-fly model checking technology (devising new algorithms, enhancing ergonomy by detecting and reporting vacuity, etc.).

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

3.2. Parallel and Distributed Verification

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

- *Rigorous design:* Because of their high complexity, concurrent verification algorithms should themselves be subject to formal modeling and verification, as confirmed by recent trends in the certification of safety-critical applications. To facilitate the development of new parallel and distributed verification algorithms, we promote a rigorous approach based on formal methods and verification. Such algorithms will be first specified formally in LNT, then validated using existing model checking algorithms of the CADP toolbox. Second, parallel or distributed implementations of these algorithms will be generated automatically from the LNT specifications, enabling them to be experimented on large computing infrastructures, such as clusters and grids. As a side-effect, this “bootstrapping” approach would produce new verification tools that can later be used to self-verify their own design.

- *Performance optimization:* In devising parallel and distributed verification algorithms, particular care must be taken to optimize performance. These algorithms will face concurrency issues at several levels: grids of heterogeneous clusters (architecture-independence of data, dynamic load balancing), clusters of homogeneous machines connected by a network (message-passing communication, detection of stable states), and multi-core machines (shared-memory communication, thread synchronization). We will seek to exploit the results achieved in the parallel and distributed computing field to improve performance when using thousands of machines by reducing the number of connections and the messages exchanged between the cooperating processes carrying out the verification task. Another important issue is the generalization of existing LTS representations (explicit, implicit, distributed) in order to make them fully interoperable, such that compilers and verification tools can handle these models transparently.

3.3. Timed, Probabilistic, and Stochastic Extensions

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

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

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

To address these issues, CONVECS follows two research directions:

- *Unifying the semantic models*. Firstly, we will perform a systematic survey of the existing semantic models in order to distinguish between their essential and non-essential characteristics, the goal being to propose a unified semantic model that is compatible with process calculi techniques for specifying and verifying concurrent systems. There are already proposals for unification either theoretical (e.g., Markov automata) or practical (e.g., PRISM and MODEST modeling languages), but these languages focus on quantitative aspects and do not provide high-level control structures and data handling features (as LNT does, for instance). Work is therefore needed to unify process calculi and quantitative models, still retaining the benefits of both worlds.

- *Increasing the interoperability of analysis tools*. Secondly, we will seek to enhance the interoperability of existing tools for timed, probabilistic, and stochastic systems. Based on scientific exchanges with developers of advanced tools for quantitative analysis, we plan to evolve the CADP toolbox as follows: extending its perimeter of functional verification with quantitative aspects; enabling deeper connections with external analysis components for probabilistic, stochastic, and timed models; and introducing architectural principles for the design and integration of future tools, our long-term goal being the construction of a European collaborative platform encompassing both functional and non-functional analyzes.

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

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

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

3.5. Real-Life Applications and Case Studies

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

4. Application Domains

4.1. Application Domains

The theoretical framework we use (automata, process algebras, bisimulations, temporal logics, etc.) and the software tools we develop are general enough to fit the needs of many application domains. They are applicable to virtually any system or protocol that consists of distributed agents communicating by asynchronous messages. The list of recent case studies performed with the CADP toolbox (see in particular § 7.5) 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, train supervision 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. Highlights of the Year

5.1. Highlights of the Year

Frédéric Lang, together with Franco Mazzanti from CNR-ISTI/FMT (Pisa, Italy), won all the gold medals for the “Parallel CTL” and “Parallel LTL” tracks of the RERS’2019 (*Rigorous Evaluation of Reactive Systems*) challenge [1]. The goal of these two tracks was to verify 180 properties expressed in the branching-time temporal logic CTL and 180 properties expressed in the linear-time temporal logic LTL. These properties had to be evaluated on various complex systems, having up to 70 concurrent processes and 234 synchronization actions. To attack such difficult problems, Lang and Mazzanti decided to join forces, and managed to evaluate all the 360 properties correctly, by designing new verification algorithms and exploiting the compositional verification techniques of CADP.

6. New Software and Platforms

6.1. CADP

Construction and Analysis of Distributed Processes

KEYWORDS: Formal methods - Verification

[1] http://rers-challenge.org/2019

FUNCTIONAL DESCRIPTION: CADP (*Construction and Analysis of Distributed Processes* – formerly known as *CAESAR/ALDEBARAN Development Package*) [5] is a toolbox for protocols and distributed systems engineering.

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

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

  A number of on-the-fly analysis tools have been developed within the OPEN/CAESAR environment, among which:
  - BISIMULATOR, which checks bisimulation equivalences and preorders,
  - CUNCTATOR, which performs steady-state simulation of continuous-time Markov chains,
  - DETERMINATOR, which eliminates stochastic nondeterminism in normal, probabilistic, or stochastic systems,
  - DISTRIBUTOR, which generates the graph of reachable states using several machines,
  - EVALUATOR, which evaluates MCL formulas,
  - EXECUTOR, which performs random execution,
  - 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 [43], CTL [32], ACTL [33], etc.) to be defined in XTL.
- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [38] and providing a unified access to a graph partitioned in fragments distributed over a set of remote machines, possibly located in different countries. The PBG format is supported by several tools, such as:
  - PBG_CP, PBG_MV, and PBG_RM, which facilitate standard operations (copying, moving, and removing) on PBG files, maintaining consistency during these operations,
  - PBG_MERGE (formerly known as BCG_MERGE), which transforms a distributed graph into a monolithic one represented in BCG format,
  - PBG_INFO, which displays various statistical information about a distributed graph.
- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CAESAR-compliant compilers, e.g.:
  - BCG_OPEN, for models represented as BCG graphs,
  - CAESAR.OPEN, for models expressed as LOTOS descriptions,
  - EXP.OPEN, for models expressed as communicating automata,
  - FSP.OPEN, for models expressed as FSP [45] 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 [36] scripting language. Both EUCALYPTUS and SVL provide users with an easy and uniform access to the CADP tools by performing file format conversions automatically whenever needed and by supplying appropriate command-line options as the tools are invoked.

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

6.2. 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 [37], 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/

7. New Results

7.1. New Formal Languages and their Implementations

7.1.1. LOTOS and LNT Specification Languages

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

LNT [6] [31] 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 § 7.5). 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 2019, a new option `-depend` has been added to the LNT_DEPEND, LNT2LOTOS, and LNT.OPEN tools. LNT_DEPEND now supports the case where the user replaces the predefined LNT modules (e.g., BOOLEAN, NATURAL, etc.) with custom versions. LNT_DEPEND has been made faster and displays better error messages. The LOTOS code generated by LNT2LOTOS for parallel compositions could be semantically incorrect and has been fixed.

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

The version 2.x of TRAIAN that we have been developing for almost 20 years is increasingly difficult to maintain. It consists of 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 2.x only exists in a 32-bit version, and sometimes hits the 4 GB RAM limit when dealing with large compiler specifications, such as those of LNT2LOTOS or EVALUATOR 5.

For this reason, we undertook in 2018 a complete rewrite of TRAIAN (version 3.0) 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.x, 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.x, but should be ultimately capable of bootstrapping itself.

In 2019, we continued the development of TRAIAN 3.0, whose grammar and syntax analysis phase was already almost complete. We fully implemented several static program analysis phases, among which the following:

- binding analysis, which associates a declaration to every identifier occurring in the program (e.g., type, channel, variable, event, etc.)
- typing analysis (including resolution of function name overloading), which associates a type to every expression in the program
- type-productivity and type-finiteness analysis, which check respectively whether a type has at least one value and whether a type has a finite number of values

We also fully implemented the C function generation phase and started to implement the C type generation phase. To avoid problems when switching from TRAIAN 2.x to TRAIAN 3.0, TRAIAN 3.0 generates almost exactly the same code as TRAIAN 2.x. The principal differences concern the numbers used to uniquely identify symbols (variables and functions) in the generated C code, because these are often derived from the syntax tree.

TRAIAN 3.0 is checked regularly against a non-regression test suite consisting of 845 correct and 1545 incorrect programs.

In total, the functionalities that remain to be implemented in TRAIAN 3.0 represent less than 32% of the code of TRAIAN 2.x.

7.1.2. Nested-Unit Petri Nets

Participants: Pierre Bouvier, Hubert Garavel.

Nested-Unit Petri Nets (NUPNs) is a model of computation that can be seen as an upward-compatible extension of P/T nets, which are enriched with structural information on their concurrent and hierarchical structure. Such structural information can easily be produced when NUPNs are generated from higher-level specifications (e.g., process calculi) and allows logarithmic reductions in the number of bits required to represent reachable states, thus enabling verification tools to perform better. For this reason, NUPNs have been so far implemented in thirteen verification tools developed in four countries, and adopted by two international competitions (the Model Checking Contest and the Rigorous Examination of Reactive Systems challenge).

In 2019, a journal article [13] has been published, which formalizes the complete theory of NUPNs.

The development of software tools for NUPNs has steadily progressed. The file format for NUPNs has been enhanced and made more precise; the NUPN_INFO tool has been extended with two new options; the CAESAR.BDD tool as been extended with six new options and its capabilities and efficiency improved in many respects.

We also revisited the problem of decomposing a Petri net into a network of automata, a problem that has been around since the early 70s. We reformulated this problem as the transformation of an ordinary, one-safe Petri net into a unit-safe NUPN. We developed various transformation methods, all of which we implemented in a tool chain that combines NUPN tools with third-party software, such as SAT solvers, SMT solvers, and tools for graph colouring and finding maximal cliques. We performed an extensive evaluation of these methods on a collection of more than 12,000 nets from diverse sources, including nets whose marking graph is too large for being explored exhaustively.

7.1.3. 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 (*Business Process Model and Notation*) standard, and several industrial platforms have been developed for supporting their design, modeling, and simulation.

In collaboration with Francisco Durán (University of Málaga, Spain) and Camilo Rocha (University of Cali, Colombia), we proposed an approach for the modeling and analysis of resource allocation for business processes. Our approach enables the automatic computation of measures for precisely identifying and optimizing the allocation of resources in business processes, including resource usage over time. The proposed analysis, especially suited to support decision-making strategies, is illustrated with a case study of a parcel ordering and delivery by a fleet of drones. This work comprises an encoding of a significant and expressive subset of BPMN in rewriting logic, an executable logic of concurrent change that can naturally deal with states and concurrent computations. The encoding is by itself a formal semantics and interpreter of the BPMN subset that captures all concurrent behavior and thus is used to simulate the concurrent evolution of any business process with a given number of resources and replicas. This work led to two publications, in an international conference [19] and an international journal [12].

7.2. Parallel and Distributed Verification

7.2.1. Debugging of Concurrent Systems using Counterexample Analysis

Participant: 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 a novel approach to improve the usability of model checking by simplifying the comprehension of counterexamples. Our approach takes as input an LTS model and an (unsatisfied) temporal logic property, and operates in four steps. First, all counterexamples for the property are extracted from the model. Second, the model is analyzed to identify the actions that skip from correct to incorrect behaviours (intuitively, these are the most relevant actions from a debugging perspective). Third, using a panel of abstraction techniques, these actions are extracted from the counterexamples. Fourth, 3D visualization techniques are used for highlighting specific regions in the model, where a choice is possible between executing a correct behaviour or falling into an erroneous part of the model, according to the property under analysis. We developed a tool named CLEAR to fully automate our approach, and we applied it on real-world case studies from various application areas for evaluation purposes. This allowed us to identify several patterns corresponding to typical cases of errors (e.g., interleaving, iteration, causality, etc.).

These results led to two publications in international conferences [15] [16] and a publication to appear in an international journal [11].

7.2.2. Eliminating Data Races in Parallel Programs using Model Checking

Participants: Radu Mateescu, Wendelin Serwe.

Parallelization of existing sequential programs to increase their performance and exploit recent multi- and many-core architectures is a challenging but inevitable effort. One increasingly popular parallelization approach is based on OpenMP, which enables the designer to annotate a sequential program with constructs specifying the parallel execution of code blocks. These constructs are then interpreted by the OpenMP compiler and runtime, which assigns blocks to threads running on a parallel architecture. Although this scheme is very flexible and not (very) intrusive, it does not prevent the occurrence of synchronization errors (e.g., deadlocks) or data races on shared variables.

In the framework of the CAPHCA project (see § 9.2.1.1), in collaboration with Eric Jenn and Viet Anh Nguyen (IRT Saint-Exupéry, Toulouse), we proposed an iterative method to assist the OpenMP parallelization by using formal methods and verification. In each iteration, potential data races are identified by applying to the OpenMP program a lockset analysis, which computes the set of shared variables that potentially need to be protected by locks. To avoid the insertion of superfluous locks, an abstract, action-based formal model of the OpenMP program in LNT is extracted and analyzed using the EVALUATOR model checker of CADP. Spurious locks are detected by checking ACTL formulas expressing the absence of concurrent execution of shared variables accesses. This work led to an international publication [28].

7.3. Timed, Probabilistic, and Stochastic Extensions

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

Participants: Armen Inants, Radu Mateescu.

Specifying and verifying quantitative properties of concurrent systems requires expressive and user-friendly property languages combining temporal, data-handling, and quantitative aspects. To this aim, we undertook the quantitative analysis of concurrent systems modeled as PTSs (*Probabilistic Transition Systems*), whose actions contain channel names, data values, and probabilities. We proposed a new regular probabilistic operator that extends naturally the Until operators of PCTL (*Probabilistic Computation Tree Logic*) [41], by specifying the probability measure of a path characterized by a generalized regular formula involving arbitrary computations on data values. We devised an on-the-fly model checking method for this new operator, based on a combined local resolution of linear and Boolean equation systems.

In 2019, we continued this work as follows:

- The MCL v4 language was conservatively extended with the new probabilistic operator, leading to a new version MCL v5.
- A new version 5 of the EVALUATOR model checker that handles the MCL v5 language, was added to CADP. EVALUATOR 5 is backward compatible with EVALUATOR 4, to which it adds a new option "`-epsilon`" specifying the precision of floating-point computations. A new version 5 of the MCL_EXPAND tool, the front-end common to the EVALUATOR 3, 4, and 5 model checkers, was added to CADP. This version is upward compatible with the previous one (except for slight changes in some error messages), it corrects a bug and brings some optimizations in the C code generated. Two new manual pages "evaluator5" and "mcl5" have been added.
- For certain probabilistic formulas (e.g., expressing the step-bounded reachability of events), the on-the-fly model checking procedure can be optimized by taking advantage of the possible *query containments*, i.e., implications between instances of the formula with different data parameters. We studied query containment in DHMLR (*Data-based Hennessy-Milner Logic with Recursion*), a parameterized equational formalism used as intermediate language for model checking MCL formulas. Our method consists in detecting, by static analysis, the containment orders present in the DHMLR representation of an MCL formula, and using the information about parameterized Boolean variable implications to improve the convergence of the BES resolution algorithms. We implemented the method in a prototype extension of EVALUATOR 5 and of the CAESAR_SOLVE library for BES resolution, and applied it for verifying probabilistic and also functional properties (e.g., bounded inevitability). The experiments we carried out on self-stabilizing protocols and communication protocols over unreliable channels showed reductions of up to 50% in memory and up to 33% in execution time. This work led to a paper submitted to an international conference.

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

7.4.1. Compositional Verification

Participants: Frédéric Lang, Radu Mateescu.

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 2019, in addition to small bug corrections, we updated SVL to support version 5 of EVALUATOR, and we corrected a semantic bug in the expansion of meta-operators of SVL.

In collaboration with Franco Mazzanti (ISTI-CNR, Pisa, Italy), we also used the compositional verification tools of CADP in the framework of the RERS'2019 challenge [2], which consisted in verifying 180 LTL properties and 180 CTL properties on large models of concurrent systems having up to 70 concurrent processes and 234 synchronization actions.

We applied to these examples the *maximal hiding* technique [48], which consists in hiding in the model all actions that are not necessary to verify the property. We combined this technique with compositional minimization (using the smart reduction heuristic implemented in SVL) as follows:

- In a first attempt, we used the technique consisting in applying minimization modulo either strong bisimulation or divbranching (divergence-preserving branching) bisimulation, depending on the fragment of the modal μ-calculus to which the formula belongs, as proposed in [48]. This was more efficient than non-compositional verification on large models, but not sufficient to verify all RERS problems successfully.
- We then proposed a refinement of this approach, which consists in (1) partitioning the actions of the system to be verified into so-called strong and weak actions, depending on the formula, and (2) minimizing modulo divbranching bisimulation all processes and process compositions containing weak actions only. This is an improvement over the previous technique, since divbranching bisimulation can be used to minimize some processes of the system even though the formula does not belong to the fragment of the μ-calculus adequate with divbranching bisimulation (which corresponds to formulas with an empty set of strong actions). This new technique allowed us to verify a lot more problems successfully, but still letting a few of the largest RERS problems unresolved. We published a paper describing the approach in an international conference [23].
- At last, we designed a new bisimulation relation, named *sharp bisimulation*, parameterized by the strong actions of the system, and we implemented a prototype tool that reduces a behavior graph modulo this relation. Sharp bisimulation parameterized by a set S of strong actions is weaker than strong bisimulation, stronger than divbranching bisimulation, and adequate with formulas whose strong actions are included in S. Such a fine-tuning of the bisimulation relation by strong actions allowed us to verify all RERS problems successfully and to win the 2019 challenge. A paper describing the approach was accepted for publication in an international conference.

7.4.2. Other Component Developments

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

In 2019, several components of CADP have been improved as follows:

- We enhanced the TESTOR tool by adding the possibility to interact with an SUT (*System Under Test*) using its standard input and output.
- We enhanced the XTL compiler with a function converting a transition label into a string (useful for handling the entire content of the label), and we also corrected three bugs.
- We enhanced MCL_EXPAND 5 with a better detection of nondeterminism in probabilistic formulas and a vacuity check for infinite looping operators, and we also corrected a semantic bug.

[2] http://rers-challenge.org/2019

- We enhanced EVALUATOR 5 with more explanative messages about the assignment of probabilities to transitions, and we corrected two bugs in each of the tools EVALUATOR 4 and 5.
- The C code generated by CAESAR has been modified to suppress GCC 6.5 warnings.
- Several changes have been brought to CADP to enable its use on new platforms, including macOS 10.15 "Catalina" and the forthcoming Debian 10.0 Linux distribution. Various bugs specific to Linux and SunOS systems (Solaris or Illumos/OpenIndiana) have been fixed.

7.5. Real-Life Applications and Case Studies

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

Participants: Umar Ozeer, Gwen Salaün.

Recent computing trends have been advocating for more distributed paradigms, namely Fog computing, which extends the capacities of the Cloud at the edge of the network, that is close to end devices and end users in the physical world. The Fog is a key enabler of the Internet of Things (IoT) applications as it resolves some of the needs that the Cloud fails to provide such as low network latencies, privacy, QoS, and geographical requirements. For this reason, the Fog has become increasingly popular and finds application in many fields such as smart homes and cities, agriculture, healthcare, transportation, etc.

The Fog, however, is unstable because it is constituted of billions of heterogeneous devices in a dynamic ecosystem. IoT devices may regularly fail because of bulk production and cheap design. Moreover, the Fog-IoT ecosystem is cyber-physical and thus devices are subjected to external physical world conditions, which increase the occurrence of failures. When failures occur in such an ecosystem, the resulting inconsistencies in the application affect the physical world by inducing hazardous and costly situations.

In the framework of the collaboration with Orange Labs (see § 8.1.1), we proposed an end-to-end autonomic failure management approach for IoT applications deployed in the Fog. The proposed approach recovers from failures in a cyber-physical consistent way. Cyber-physical consistency aims at maintaining a consistent behavior of the application with respect to the physical world, as well as avoiding dangerous and costly circumstances. The approach was validated using model checking techniques to verify important correctness properties. It was then implemented as a framework called F3ARIoT. This framework was evaluated on a smart home application. The results showed the feasibility of deploying F3ARIoT on real Fog-IoT applications as well as its good performances in regards to end user experience.

These results have been published in U. Ozeer's PhD thesis [10] and at an international conference [26]. Another paper was submitted to an international journal.

7.5.2. Verified Composition and Deployment of IoT Applications

Participants: Alejandro Martinez Rivero, Radu Mateescu, Ajay Muroor Nadumane, Gwen Salaün.

The Internet of Things (IoT) applications are built by interconnecting everyday objects over internet. As IoT is becoming popular among consumers, the challenge of making IoT applications easy to design and deploy is more relevant than ever. In 2019, we considered this challenge along two perspectives.

- In the framework of the collaboration with Nokia Bell Labs (see § 8.1.2), we focused on helping consumers to easily design IoT applications that are correct, and also support the deployment of these applications. The correctness of the applications is ensured through formal methods and verification techniques.

  Using W3C Web of Things (WoT) specification as the basis of our work, we extended the specification of objects in WoT with a behavioural model. This allows us to describe formally the composition of objects and thus, to verify their behavioural correctness. Typically, an IoT application is defined using Event-Condition-Action (ECA) rules of the type “IF event THEN action”. Our work supports users to specify not only the ECA rules, but also the composition of rules using a simple, yet expressive language. This makes possible the construction of advanced compositions, which would have been hard or sometimes impossible to build using simple ECA rules. Finally, users are provided with an easy-to-deploy solution for these advanced compositions. All these steps were implemented and packaged in a tool named MozART, built on top of Mozilla WebThings platform. LNT is used as the formal specification language, and various tools of CADP are used for verifying the composition. Also, an execution engine based on Mozilla WebThings API was built to support the deployment of advanced compositions. The work has led to the preparation of two conference articles.

- Building IoT applications of added-value from a set of available devices with minimal human intervention is one of the main challenges facing the IoT. This is a difficult task that requires models for specifying objects, in addition to user-friendly and reliable composition techniques which in turn prevent the design of erroneous applications.

  In collaboration with Francisco Durán (University of Málaga, Spain), we tackled this problem by first describing IoT applications using abstract models obtained from existing models of concrete devices. Then, we proposed automated techniques for building compositions of devices using a repository of available devices, and an abstract goal of what the user expects from such compositions. Since the number of possible solutions can be quite high, we used both filtering and ranking techniques to provide the most relevant solutions to users. The provided solutions satisfy the given goal and may be analysed with respect to properties such as deadlock-freeness or unmatched send messages. Finally, the application can be deployed using existing execution engines. This work led to a publication in an international conference [20].

7.5.3. Autonomous Car

Participants: Philippe Ledent, Lina Marsso, Radu Mateescu, Wendelin Serwe.

Autonomous vehicles are complex cyber-physical systems that must satisfy critical correctness requirements to increase the safety of road traffic. The validation of autonomous driving is a challenging field because of the complexity of its key components (perception of the environment, scene interpretation, decision making and undertaking of actions) and the intertwinning of physical and software components. In 2019, we considered this challenge along two lines of work.

- From the embedded software perspective, autonomous cars can be considered as GALS systems, which integrate reactive synchronous components that interact asynchronously. The complexity induced by combining synchronous and asynchronous aspects makes GALS systems difficult to develop and debug.

  In the framework of the ARC6 collaboration (see § 9.1.1), we proposed a testing methodology for GALS systems that leverages conformance test generation for asynchronous systems to automatically derive realistic scenarios (inputs constraints and oracles), which are necessary ingredients for the unit testing of individual synchronous components, and are difficult and error-prone to design manually. The methodology consists of several steps (derivation of asynchronous test cases from a GALS model and a test purpose, projection of the complete test graph on a synchronous component, extraction and execution of test scenarios) and was illustrated on a simple, but relevant example inspired by autonomous cars. These results were published in L. Marsso's PhD thesis [9] and at an international conference [25].

- In collaboration with Christian Laugier, Anshul Paigwar, and Alessandro Renzaglia (CHROMA project-team), we proposed a new approach where formal verification is employed to validate systems with probabilistic predictions. We focused on the risk assessment generated by CMCDOT (*Conditional Monte Carlo Dense Occupancy Tracker*), a probabilistic perception system for autonomous cars. CMCDOT provides an environment representation through Bayesian probabilistic occupancy grids and estimates Time-to-Collision probabilities for every static and dynamic part of the grid in the near future. To validate the probabilistic collision risk estimation, we used the CARLA simulator to generate a large number of realistic intersection crossing scenarios with two vehicles. The set of scenarios is then validated using the XTL model checker, by defining appropriate KPIs (*Key Performance Indicators*) as temporal logic formulas and also performing a quantitative analysis. This work led to a publication at an international conference [24].

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Grants with Industry

8.1.1. 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 § 7.5.1), under the supervision of Loïc Letondeur (Orange Labs), Gwen Salaün (CONVECS), François Gaël Ottogalli (Orange Labs), and Jean-Marc Vincent (POLARIS project-team).

8.1.2. 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 (see § 7.5.2) supported by formal methods, under the supervision of Gwen Salaün (CONVECS), Radu Mateescu (CONVECS), Ludovic Noirie, and Michel Le Pallec (Nokia Bell Labs).

9. Partnerships and Cooperations

9.1. Regional Initiatives

9.1.1. 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), and Ioannis Parissis (LCIS, Valence).

9.2. National Initiatives

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

9.2.1.1. 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.

9.2.2. Competitivity Clusters

9.2.2.1. SECURIOT-2

Participants: Hubert Garavel [correspondent], Armen Inants, 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.

9.2.3. 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),
- Michel Le Pallec (Nokia Bell Labs, Nozay),
- Chu-Min Li (University of Picardie Jules Verne),
- Ioannis Parissis and Oum-El-Kheir Aktouf (LCIS, Valence),
- Pascal Poizat (LIP6, Paris).

9.3. European Initiatives

9.3.1. Collaborations with Major European Organizations

The CONVECS project-team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM [3]. H. Garavel and R. Mateescu are members of the FMICS board, H. Garavel being in charge of dissemination actions.

9.4. 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.

9.4.1. Inria International Partners

9.4.1.1. 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.

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

9.4.2. Other International Collaborations

In 2019, 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),
- FBK, Trento, Italy (Enrico Magnano),
- Aalto University, Finland and Northeastern University, Boston, Massachusetts (Stavros Tripakis),
- Saarland University, Germany (Holger Hermanns),
- Eindhoven University of Technology, The Netherlands (Anton Wijs and Sander de Putter),
- University of Zielona Gora, Poland (Remigiusz Wisniewski).

9.5. International Research Visitors

9.5.1. Visits of International Scientists

- H. Garavel is an invited professor at Saarland University (Germany) as a holder of the Gay-Lussac Humboldt Prize.
- Hernan Ponce de Leon (Fortiss, Munich, Germany) visited us on June 25–26, 2019. He gave a lecture entitled "*BMC with Weak Memory Models*".
- Hugues Evrard (Google, London, UK) visited us on October 21, 2019. He gave a lecture entitled "*GPU Schedulers: How Fair is Fair Enough?*".
- Karoliina Lehtinen (University of Liverpool, UK) visited us on October 23, 2019. She gave a lecture entitled "*Quasi-Polynomial Techniques for Parity Games and Other Problems*".
- Peter Csaba Ölveczky (University of Oslo, Norway) visited us on November 25, 2019. He gave a lecture entitled "*Formal Specification and Analysis of Real-Time Systems in Real-Time Maude*".

The annual CONVECS seminar was held in Villard-de-Lans (France) on July 1-3, 2019. The following invited scientists attended the seminar:

- Loïc Letondeur (Orange Labs) gave on July 2, 2019 a talk entitled "*Artificial Intelligence and Edge Computing*".
- Eric Jenn (IRT Saint-Exupéry / Thales Avionics) gave on July 3, 2019 a talk entitled "*Recent Achievements of the CAPHCA Project*".
- Viet Anh Nguyen (IRT Saint-Exupéry) gave on July 3, 2019 a talk entitled "*Using Model Checking to Identify Timing Interferences on Multicore Processors*".

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Organisation

10.1.1.1. General Chair, Scientific Chair

- H. Garavel is a member of the model board [4] of MCC (*Model Checking Contest*). In 2019, 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'2019 have been published online [44].

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

- 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 [5]. This repository currently contains 21 models, among which 5 were deposited 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.

10.1.1.2. Member of the Organizing Committees

- L. Marsso and A. Muroor Nadumane were publicity chairs (also in charge of the conference Web site and social media feeds) for SEFM'2019 (*17th International Conference on Software Engineering and Formal Methods*), Oslo, Norway, September 16–20, 2019.

10.1.2. Scientific Events: Selection

10.1.2.1. Chair of Conference Program Committees

- G. Salaün was co-chair of SEFM'2019.

10.1.2.2. Member of the Conference Program Committees

- H. Garavel was program committee member of FMICS'2019 (*24rd International Conference on Formal Methods for Industrial Critical Systems*), Amsterdam, The Netherlands, August 30–31, 2019.
- R. Mateescu was program committee member of IFIP-ICTSS'2019 (*31st IFIP International Conference on Testing Software and Systems*), Paris, France, October 15–17, 2019.
- G. Salaün was program committee member of SAC-SVT'2019 (*34th ACM Symposium on Applied Computing - Software Verification and Testing Track*), Limassol, Cyprus, April 8–12, 2019.
- G. Salaün was program committee member of FSEN'2019 (*8th IPM International Conference on Fundamentals of Software Engineering*), Tehran, Iran, May 1–3, 2019.
- G. Salaün was program committee member of COORDINATION'2019 (*21st International Conference on Coordination Models and Languages*), Lyngby, Denmark, June 18–21, 2019.
- G. Salaün was program committee member of Microservices'2019, Dortmund, Germany, February 19–21, 2019.
- G. Salaün was program committee member of COMPSAC-ITIP'2019 and COMPSAC-SETA'2019 (*IEEE International Conference on Computers, Software, and Applications – IT in Practice, and Software Engineering Technologies and Applications*), Wisconsin, USA, July 15–19, 2019.
- G. Salaün was program committee member of HPCS-4PAD'2019 and HPCS-SERCO'2019 (*6th International Symposium on Formal Approaches to Parallel and Distributed Systems, and 3rd Special Session on High Performance Services Computing and Internet Technologies*), Dublin, Ireland, July 15–19, 2019.
- G. Salaün was program committee member of FACS'2019 (*16th International Conference on Formal Aspects of Component Software*), Amsterdam, The Netherlands, October 23–25, 2019.
- G. Salaün was program committee member of FOCLASA'2019 (*17th International Workshop on Coordination and Self-Adaptativeness of Software Applications*), Oslo, Norway, September 17, 2019.
- G. Salaün was program committee member of DATAMOD'2019 (*8th International Symposium "From Data to Models and Back"*), Porto, Portugal, October 7–8, 2019.

[5] http://www.mars-workshop.org/

10.1.2.3. Reviewer

- R. Mateescu was a reviewer for the Festschrift in honor of Stefania Gnesi.
- A. Muroor Nadumane was a reviewer for SAC-SVT'2020, FACS'2019, SEFM'2019, COMPSAC-SETA'2019.

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

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

10.1.3.2. Reviewer - Reviewing Activities

- F. Lang and A. Muroor Nadumane were reviewers for STTT.
- R. Mateescu was a reviewer for ISSE (*Innovations in Systems and Software Engineering*), IST (*Information and Software Technology*), LMCS (*Logical Methods in Computer Science*), and SCP (*Science of Computer Programming*).
- G. Salaün was a reviewer for JISA (*Journal of Internet Services and Applications*), JLAMP (*Journal of Logical and Algebraic Methods in Programming*), JSME (*Journal of Software: Evolution and Process*), LMCS, and SCP.

10.1.4. Software Dissemination and Internet Visibility

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

In 2019, the main facts are the following:

- We prepared and distributed twelve successive versions (2019-a to 2019-l) of CADP.
- We were requested to grant CADP licenses for 340 different computers in the world.

The CONVECS Web site [6] was updated with scientific contents, announcements, publications, etc.

By the end of December 2019, the CADP forum [7], opened in 2007 for discussions regarding the CADP toolbox, had over 435 registered users and over 1940 messages had been exchanged.

Also, for the 2019 edition of the Model Checking Contest, one family of models generated using CADP (totalling 16 Nested-Unit Petri Nets) was provided.

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

- The OCARINA Tool and its Extension AADL2LNT for Analysing AADL Descriptions [49]
- The LFD-MPI Tool for Lightweight Formal Development of MPI Applications [52], [53]
- The FTRES Tool for Rare Event Simulation in Dynamic Fault Trees [54]
- Formal Analysis of Security Guidelines for Program Certification [55]

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

- Applying Automata Learning to Embedded Control Software [50]
- Model Checking Based Approach for Compliance Checking [46]
- Finding Conservative Schema Evolutions by Analysing API Changes [51]
- Selection of Model Checking Strategies using Machine Learning [42]
- Verifying Complex Software Control Systems from Test Objectives [29]
- Designing Safe Synchronous Reactive Systems [30]

[6] http://convecs.inria.fr
[7] http://cadp.inria.fr/forum.html

10.1.5. Invited Talks

- H. Garavel participated in the workshop "*Advancing Verification Competitions as a Scientific Method*" organized by the Lorentz Center (Leiden, The Netherlands) on February 18–22, 2019. He gave a lecture entitled "*Managing Large Collections of Benchmarks: An Experiment Report*".
- H. Garavel presented a retrospective of the Rewrite Engines Competitions at the TOOLympics event at ETAPS 2019, Prague, Czech Republic, April 6–7, 2019.
- H. Garavel was one of the six invited speakers at the panel discussion "*Moore's Law, and More?*" organized at the occasion of the 25th Anniversary of the TACAS conference, Prague, Czech Republic, April 7, 2019.
- H. Garavel gave a highlight talk entitled "*From Safe Petri Nets to NUPNs*" at the 8th IFIP WG 1.8 Workshop on Trends in Concurrency Theory, Amsterdam, The Netherlands, August 31, 2019.
- H. Garavel gave a seminar entitled "*Converting Safe Petri Nets to NUPNs*" at Saarland University, Germany, on September 26, 2019.
- R. Mateescu gave a talk entitled "*The MCL Language for Temporal and Probabilistic Analysis of Concurrent Systems*" at the Languages, Semantics, and Compilation seminar of the LIP laboratory, Lyon, September 19, 2019.
- R. Mateescu visited the Gran Sasso Science Institute (L'Aquila, Italy) on November 4–7, 2019. He gave a talk entitled "*An Action-based Model Checking Language for Concurrent Systems*".
- A. Muroor Nadumane gave a talk entitled "*Advanced Rule-based Composition and Deployment for Web of Things*" at the Inria-Nokia Bell Labs meeting held at Inria Rennes on June 11, 2019.

10.1.6. 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, and giving an advice on the recruitment of temporary engineers.
- 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 2019 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.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

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

In 2019, we carried out the following teaching activities:

P. Bouvier supervised a group of seven two-person teams of 2nd year students in the context of the "*projet informatique*" (31 hours "*équivalent TD*", consisting of lab exercises and supervision) at PHELMA.

H. Garavel was a jury member for the MOSIG master defenses in June and September 2019.

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.

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.

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 250 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 (third year of university) at IUT1/UGA.

W. Serwe gave the part on "*Behavioural Testing*" (9 hours "*équivalent TD*") of the course "*Verification and Test Theories*" to second year students of the MOSIG.

W. Serwe supervised 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 2019.

10.2.2. Supervision

PhD: Lina Marsso, "*Formal Methods for Testing Networks of Controllers*", Université Grenoble Alpes, December 10, 2019, R. Mateescu, W. Serwe, and I. Parissis

PhD: Umar Ozeer, "*Autonomic Resilience of Applications in a Largely Distributed Cloud Environment*", Université Grenoble Alpes, December 11, 2019, L. Letondeur, G. Salaün, F.-G. Ottogalli, and J.-M. Vincent

PhD in progress: Pierre Bouvier, "*Implémentation et vérification des langages concurrents de nouvelle génération*", Université Grenoble Alpes, since October 2019, H. Garavel and R. Mateescu

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

10.2.3. Juries

- F. Lang was PhD committee member of Sander de Putter's PhD thesis, entitled "*Verification of Concurrent Systems in a Model-Driven Engineering Workflow*", defended at Technische Universiteit Eindhoven (The Netherlands) on January 28, 2019.
- R. Mateescu was reviewer of The Anh Pham's PhD thesis, entitled "*Efficient State-Space Exploration for Asynchronous Distributed Programs*", defended at ENS Rennes on December 6, 2019.
- G. Salaün was committee president of Xavier Etchevers's Habilitation thesis, entitled "*Evolution du déploiement automatisé d'applications : des infrastructures cloud aux enjeux de l'agilité et de l'intelligence ambiante*", defended at Université Grenoble Alpes on November 29, 2019.
- G. Salaün was PhD committee president of Raphaël Jackse's PhD thesis, entitled "*Interactive Runtime Verification*", defended at Université Grenoble Alpes on December 18, 2019.

10.3. Popularization

10.3.1. Interventions

- R. Mateescu has participated to the "*Journée Transfo : Plongez dans le monde du numérique*" organized at Inria Montbonnot on January 24, 2019.
- R. Mateescu has participated to the "*Fête de la Science*" organized at Inria Montbonnot on October 12, 2019.

11. 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^{o} 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. *Nested-unit Petri nets*, in "Journal of Logical and Algebraic Methods in Programming", April 2019, vol. 104, pp. 60-85 [*DOI :* 10.1016/J.JLAMP.2018.11.005], https://hal.inria.fr/hal-02072190

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

[5] 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^{o} 2, pp. 89-107 [*DOI :* 10.1007/S10009-012-0244-Z], http://hal.inria.fr/hal-00715056

[6] 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

[7] 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

[8] 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

[9] L. MARSSO. *On Model-based Testing of GALS Systems*, Université Grenoble Alpes, December 2019

[10] U. OZEER. *Autonomic Resilience of Applications in a Largely Distributed Cloud Environment*, Université Grenoble Alpes, December 2019

Articles in International Peer-Reviewed Journals

[11] G. BARBON, V. LEROY, G. SALAÜN. *Debugging of Behavioural Models using Counterexample Analysis*, in "IEEE Transactions on Software Engineering", 2019, pp. 1-14, forthcoming [*DOI :* 10.1109/TSE.2019.2915303], https://hal.inria.fr/hal-02145610

[12] F. DURÁN, C. ROCHA, G. SALAÜN. *A Rewriting Logic Approach to Resource Allocation Analysis in Business Process Models*, in "Science of Computer Programming", November 2019, vol. 183, pp. 1-32 [*DOI :* 10.1016/J.SCICO.2019.102303], https://hal.inria.fr/hal-02345895

[13] H. GARAVEL. *Nested-unit Petri nets*, in "Journal of Logical and Algebraic Methods in Programming", April 2019, vol. 104, pp. 60-85 [*DOI :* 10.1016/J.JLAMP.2018.11.005], https://hal.inria.fr/hal-02072190

[14] 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

International Conferences with Proceedings

[15] G. BARBON, V. LEROY, G. SALAÜN. *Debugging of Behavioural Models with CLEAR*, in "TACAS 2019 - 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems", Prague, Czech Republic, Lecture Notes in Computer Science, Springer, April 2019, vol. 11427, pp. 386-392 [*DOI :* 10.1007/978-3-030-17462-0_ 26], https://hal.archives-ouvertes.fr/hal-02121180

[16] G. BARBON, V. LEROY, G. SALAÜN, E. YAH. *Visual Debugging of Behavioural Models*, in "ICSE 2019 - IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings", Montreal, Canada, IEEE, May 2019, pp. 107-110 [*DOI :* 10.1109/ICSE-COMPANION.2019.00050], https://hal.inria.fr/hal-02145535

[17] E. BARTOCCI, D. BEYER, P. BLACK, G. FEDYUKOVICH, H. GARAVEL, A. HARTMANNS, M. HUISMAN, F. KORDON, J. NAGELE, M. SIGHIREANU, B. STEFFEN, M. SUDA, G. SUTCLIFFE, T. WEBER, A. YAMADA. *TOOLympics 2019: An Overview of Competitions in Formal Methods*, in "25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019", Prague, Czech Republic, April 2019, pp. 3-24 [*DOI :* 10.1007/978-3-030-17502-3_1], https://hal.sorbonne-universite.fr/hal-02094030

[18] F. DURÁN, H. GARAVEL. *The Rewrite Engines Competitions: A RECtrospective*, in "Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19), Part III: TOOLympics", Prague, Czech Republic, Lecture Notes in Computer Science, Springer, April 2019, vol. 11429, pp. 1-9 [*DOI :* 10.1007/978-3-030-17502-3], https://hal.inria.fr/hal-02133649

[19] F. DURÁN, C. ROCHA, G. SALAÜN. *Analysis of Resource Allocation of BPMN Processes*, in "ICSOC 2019 - 17th International Conference on Service-Oriented Computing", Toulouse, France, Springer, October 2019, pp. 452-457 [*DOI :* 10.1007/978-3-030-33702-5_35], https://hal.inria.fr/hal-02345879

[20] F. DURÁN, G. SALAÜN, A. KRISHNA. *Automated Composition, Analysis and Deployment of IoT Applications*, in "TOOLS 2019 - 51st International Conference on Software Technology: Methods and Tools", Innopolis, Russia, Springer, October 2019, pp. 252-268 [*DOI :* 10.1007/978-3-030-29852-4_21], https://hal.inria.fr/hal-02345865

[21] A. KRISHNA, M. LE PALLEC, R. MATEESCU, L. NOIRIE, G. SALAÜN. *IoT Composer: Composition and Deployment of IoT Applications*, in "ICSE 2019 - IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings", Montreal, Canada, IEEE, May 2019, pp. 19-22 [*DOI :* 10.1109/ICSE-COMPANION.2019.00028], https://hal.inria.fr/hal-02146569

[22] A. KRISHNA, M. LE PALLEC, R. MATEESCU, L. NOIRIE, G. SALAÜN. *Rigorous Design and Deployment of IoT Applications*, in "FormaliSE 2019 - 7th International Conference on Formal Methods in Software Engineering", Montreal, Canada, ACM, May 2019, pp. 21-30 [*DOI :* 10.1109/FORMALISE.2019.00011], https://hal.inria.fr/hal-02146553

[23] F. LANG, R. MATEESCU, F. MAZZANTI. *Compositional Verification of Concurrent Systems by Combining Bisimulations*, in "FM 2019 - 23rd International Conference on Formal Methods", Porto, Portugal, Lecture Notes in Computer Science, Springer Verlag, October 2019, vol. 11800, pp. 196-213 [*DOI :* 10.1007/978-3-030-30942-8_13], https://hal.inria.fr/hal-02295459

[24] P. LEDENT, A. PAIGWAR, A. RENZAGLIA, R. MATEESCU, C. LAUGIER. *Formal Validation of Probabilistic Collision Risk Estimation for Autonomous Driving*, in "CIS-RAM 2019 - 9th IEEE International Conference on Cybernetics and Intelligent Systems (CIS) Robotics, Automation and Mechatronics (RAM)", Bangkok, Thailand, IEEE, November 2019, pp. 1-6, https://hal.inria.fr/hal-02355551

[25] L. MARSSO, R. MATEESCU, I. PARISSIS, W. SERWE. *Asynchronous Testing of Synchronous Components in GALS Systems*, in "IFM'2019 - 15th International Conference on Integrated Formal Methods", Bergen, Norway, Lecture Notes in Computer Science, Springer Verlag, November 2019, vol. 11918, pp. 360-378 [*DOI :* 10.1007/978-3-030-34968-4_20], https://hal.archives-ouvertes.fr/hal-02394989

[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 - 22nd Conference on Innovation in Clouds, Internet and Networks", Paris, France, IEEE, February 2019, pp. 230-232 [*DOI :* 10.1109/ICIN.2019.8685909], https://hal.archives-ouvertes.fr/hal-01979686

Scientific Books (or Scientific Book chapters)

[27] H. GARAVEL, R. MATEESCU. *Reflections on Bernhard Steffen's Physics of Software Tools*, in "Models, Mindsets, Meta: The What, the How, and the Why Not?", Springer Verlag, June 2019, pp. 186-207 [*DOI :* 10.1007/978-3-030-22348-9_12], https://hal.inria.fr/hal-02394588

[28] V.-A. NGUYEN, W. SERWE, R. MATEESCU, E. JENN. *Hunting Superfluous Locks with Model Checking*, in "From Software Engineering to Formal Methods and Tools, and Back", Lecture Notes in Computer Science, Springer Verlag, October 2019, vol. 11865, pp. 416-432 [*DOI :* 10.1007/978-3-030-30985-5_24], https://hal.inria.fr/hal-02314088

References in notes

[29] R. AMEUR-BOULIFA, A. CAVALLI, S. MAAG. *Verifying Complex Software Control Systems from Test Objectives: Application to the ETCS System*, in "Proceedings of the 14th International Conference on Software Technologies (ICSOFT'2019), Prague, Czech Republic", SciTePress, July 2019, pp. 397–406

[30] S. CHABANE, R. AMEUR-BOULIFA, M. MOHAMED. *Vers une conception de systèmes réactifs synchrones sûrs*, in "Actes du 12ème Colloque sur la Modélisation des Systèmes Réactifs (MSR'2019), Angers, France", November 2019

[31] 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

[32] 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

[33] 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

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

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

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

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

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

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

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

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

[42] M. HENDRIKS. *Can Machine Learning Automatically Choose your Best Model Checking Strategy?*, EEMCS: Electrical Engineering, Mathematics and Computer Science, University of Twente, 2019

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

[44] 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 2019 Edition of the Model Checking Contest*, April 2019, http://mcc.lip6.fr/2019/results.php

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

[46] F. MARTINELLI, F. MERCALDO, V. NARDONE, A. ORLANDO, A. SANTONE, G. VAGLINI. *Model Checking Based Approach for Compliance Checking*, in "Journal of Information Technology and Control", May 2019, vol. 48, n^{o} 2, pp. 278–298

[47] 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

[48] R. MATEESCU, A. WIJS. *Property-dependent Reductions Adequate with Divergence-sensitive Branching Bisimilarity*, in "Science of Computer Programming", 2014, vol. 96, pp. 354–376

[49] H. MKAOUAR. *A Formal Approach for Real-time Systems Engineering*, University of Sfax, Tunisia, February 2019

[50] J. MOERMAN. *Nominal Techniques and Black Box Testing for Automata Learning*, Radboud University, Nijmegen, The Netherlands, July 2019

[51] L. A. OUBELLI, Y. A. AMEUR, J. BEDOUET, B. CHAUSSERIE-LAPREE, B. LARZUL. *Finding Conservative Schema Evolutions by Analysing API Changes*, in "Proceedings of the 31st International Conference on Software Engineering and Knowledge Engineering (SEKE'2019), Lisbon, Portugal", KSI Research Inc., July 2019, pp. 748–777

[52] N. ROSA, H. KAMAL, A. WAGNER. *A LOTOS-based Lightweight Approach to Formally Verify MPI Applications*, Universidade Federal de Pernambuco, Centro de Informática, 2015

[53] N. ROSA, A. WAGNER, H. KAMAL. *Towards Lightweight Formal Development of MPI Applications*, in "Proceedings of 36th WoTUG Conference on Concurrent and Parallel Programming – Communicating Process Architectures (CPA'2015), Kent, UK", Concurrent Systems Engineering Series, Open Channel Publishing, August 2015, vol. 69, pp. 67–85

[54] E. RUIJTERS, D. REIJSBERGEN, P.-T. DE BOER, M. STOELINGA. *Rare Event Simulation for Dynamic Fault Trees*, in "Reliability Engineering & System Safety", June 2019, vol. 186, pp. 220–231

[55] Z. ZHIOUA, R. AMEUR-BOULIFA, Y. ROUDIER. *Framework for the Formal Specification and Verification of Security Guidelines*, in "Advances in Science, Technology and Engineering Systems Journal", January 2018, vol. 3, n^{o} 1, pp. 38–48