RESEARCH CENTRE

Grenoble - Rhône-Alpes

IN PARTNERSHIP WITH:

Institut polytechnique de Grenoble, Université Joseph Fourier (Grenoble)

2020

ACTIVITY REPORT

Project-Team CONVECS

Construction of verified concurrent systems

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

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Contents

Project-Team CONVECS

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

Keywords

Computer sciences and digital sciences

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

A6.1.3. – Discrete Modeling (multi-agent, people centered)

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 members, 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 [Univ Grenoble Alpes, Professor, HDR]

Post-Doctoral Fellow

- Luca Di Stefano [Inria, from Nov 2020]

PhD Students

- Pierre Bouvier [Univ Grenoble Alpes]
- Philippe Ledent [STMicroelectronics, from Oct 2020]
- Lucie Muller [Inria, from Sep 2020]
- Ajay Muroor-Nadumane [Inria - Nokia Bell Labs]
- Ahang Zuo [Univ Grenoble Alpes, from Oct 2020]

Technical Staff

- Armen Inants [Inria, Engineer, until Jun 2020]

Interns and Apprentices

- Nicolas Amat [Univ Grenoble Alpes, from Feb 2020 until Aug 2020]
- Irman Faqrizal [Inria, from Feb 2020 until Jun 2020]
- Pedro Perez Torres [Inria, until Jul 2020]
- Imane Serdani [Inria, from Feb 2020 until Jun 2020]

Administrative Assistant

- Myriam Etienne [Inria]

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*) [45], 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 [32], 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

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 § 7.4) 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, multi-agent systems,
- *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

Frédéric Lang and Wendelin Serwe, together with Franco Mazzanti from CNR-ISTI/FMT (Pisa, Italy), won the gold medals for the "Parallel CTL" track of the RERS'2020 (Rigorous Evaluation of Reactive Systems) challenge[1]. The goal of this track was to verify 90 properties expressed in the branching-time temporal logic CTL on several complex systems, having up to 16 concurrent processes and 75 synchronization actions. These difficult verification problems were solved by exploiting new results in compositional verification (see § 7.3.1) combined with the on-the-fly and partial model checking techniques of CADP.

6 New software and platforms

6.1 New software

6.1.1 CADP

Name: Construction and Analysis of Distributed Processes

Keywords: Formal methods, Verification

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

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

- CAESAR.ADT [31] 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 [37, 36] 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 [32] 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:

[1] http://rers-challenge.org/2020

  - 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 [40], CTL[28], ACTL[29], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [35] 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 [43] 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 [33] 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.

URL: `http://cadp.inria.fr/`

Authors: Hubert Garavel, Radu Mateescu, Frédéric Lang, Wendelin Serwe, David Champelovier, Damien Thivolle

Contacts: Radu Mateescu, Hubert Garavel

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

6.1.2 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 [34], being used in all recent compilers developed by CONVECS.

URL: `http://convecs.inria.fr/software/traian/`

Authors: Hubert Garavel, Frédéric Lang, Wendelin Serwe

Contacts: Hubert Garavel, Frédéric Lang, Wendelin Serwe

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe

7 New results

7.1 New Formal Languages and their Implementations

7.1.1 LNT Specification Language

Participants Hubert Garavel, Frédéric Lang, Wendelin Serwe.

LNT [6] [27] 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 § 7.4). 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 2020, the LNT language has progressively evolved, breaking away from some design decisions inherited from LOTOS in the 80s to become a better language and to achieve convergence with the LOTOS NT language supported by the TRAIAN compiler (see § 7.1.2):

- From now on, LNT keywords and pragmas must be written in lower case only. Six keywords denoting new constructs (“access”, “alt”, “ensure”, “require”, “result”, and “trap”) have been introduced, as well as six other keywords for arithmetic and logical operations, now available for overloaded redefinitions. Predefined comparison operators and some Boolean operators were renamed in a form closer to their mathematical notation. The symbol “=>” was replaced with “->” in field updates and named-styles actual parameters of procedures and functions.

- A new instruction “access E_1, ..., E_n” has been introduced, which is similar to “use X_1, ..., X_n”, but operates on events instead of variables.

- Two new clauses “require” and “ensure” have been introduced to express preconditions and post-conditions for LNT functions and processes. Preconditions are Boolean expressions on the input values of “in”, “in var”, and “in out” parameters of a function or process. Postconditions use the new “result” keyword denoting the value returned by a function and the new “.in” and “.out” qualifiers denoting the values of "in out" parameters when entering and leaving a function or process. Preconditions (resp. postconditions) are checked upon function or process call (resp. return), an exception being raised when a precondition or postcondition is false.

- The LNT2LOTOS Reference Manual has been updated to document the changes of LNT, and the CADP demo examples have been upgraded to the latest version of LNT. The “upc” shell script was extended to ease the migration of LNT programs.

- Also, in addition to eleven bug fixes, the LNT2LOTOS translator was enhanced in many respects. It can now handle larger LNT programs, it generates better LOTOS code (without useless parameters in auxiliary processes) and, due to a refined data-flow analysis, it emits more warnings about unused parameters, useless assignments, dead “case” branches, and synchronized events that are not accessed in some parallel branches.

7.1.2 LOTOS NT Specification Language

Participants Hubert Garavel, Frédéric Lang, Wendelin Serwe.

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 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 2020, our efforts led to the release of three new major versions of TRAIAN:

- TRAIAN 3.0 is the first version departing from TRAIAN 2.9, with a totally different internal architecture, although with a source code of nearly the same size.

  The lexer and parser of TRAIAN 3.0 are built using the SYNTAX compiler-generation system developed at Inria Paris, which triggered various enhancements of the programming interfaces of SYNTAX. The abstract syntax tree of LOTOS NT, the library of predefined LOTOS NT types and functions, the static semantics checking (identifier binding, type checking, dataflow analysis, etc.), and the C code generation have been entirely redesigned in LOTOS NT.

  TRAIAN 3.0 brings three major enhancements compared to TRAIAN 2.9:

  – Compiler performance greatly improved, TRAIAN 3.0 being 6-7 times faster and using 120-150 times less memory than TRAIAN 2.9, as measured when compiling two large programs (LNT2LOTOS and TRAIAN 3.0 itself).
  – Native 64-bit binaries of TRAIAN 3.0 are now available for Linux, macOS, OpenIndiana, and Solaris 11. Together with the reduction of memory consumption, this fully solves the former 4 GB memory limitation that existed when running TRAIAN 2.9 in 64-bit compatibility mode.
  – Error and warning messages produced by TRAIAN 3.0 are clearer and more concise.

  TRAIAN 3.0 remains largely compatible with TRAIAN 2.9 concerning its input and (as close as possible to) the C code generated as output. The TRAIAN manual page, the LOTOS NT user manual, the mode files for the various editors, and the demo examples contained in the TRAIAN distribution have been updated to reflect the changes brought to the language and its compiler. Test bases gathering thousands of correct and incorrect LOTOS NT programs (totalling 920,000 lines of code) have been set up and are systematically used for non-regression testing.

- TRAIAN 3.1 is a consolidation release that brings five general enhancements and eight bug fixes, and also a transition release that introduces seventeen language changes (all but one being backward compatible) to LOTOS NT in order to progressively align it with LNT. To ease the upgrade of existing LOTOS NT source code, a conversion tool named "traian_upc" was developed and used to swiftly upgrade seven compilers written using LOTOS NT (namely, EXP2C, FSP2LOTOS, GRL2LNT, MCL_EXPAND, PIC2LNT, SVL, and TRAIAN itself) totalling 168,000 lines of LOTOS NT code, as well as the LOTOS NT test suites totalling 1,940,000 lines of code.

- TRAIAN 3.2 brings, in addition to nine bug fixes, nine language changes to LOTOS NT in order to further align it with LNT, and twelve code-generation changes w.r.t. the former versions 3.0 and 3.1 of TRAIAN in order to simplify and improve, in many respects, its generated C code.

7.1.3 Nested-Unit Petri Nets

Participants Pierre Bouvier, Hubert Garavel.

Nested-Unit Petri Nets (NUPNs) are 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). The complete theory of NUPNs is formalized in a journal article [3] and their PNML representation is described here[2].

In 2020, the NUPN format and its associated software tools were enhanced as follows:

- The definition of the NUPN format was made more precise concerning character strings and the "!multiple_arcs" and "!multiple_initial_tokens" pragmas, for which new static semantics constraints and invariants have been formulated. These changes have been implemented in the CAESAR.BDD tool, which is now stricter.
- In addition to five bug fixes, CAESAR.BDD has undergone deep changes. The internal data structures for representing transitions and arcs have been entirely rewritten, replacing bit matrices with adjacency lists; although both implementations have respective merits, the new implementation handles all NUPN models that the former implementation handled, plus other NUPN models that could not be handled.
- The static exploration algorithm of CAESAR.BDD that computes a subset of dead places and dead transitions was enhanced to exploit the "!unit_safe" pragma. Also, this static algorithm is now invoked twice, before and after the dynamic exploration algorithm based on Binary Decision Diagrams (BDDs), reducing the number of iterations by 37%.
- Several options of CAESAR.BDD were enhanced in terms of performance (increased speed), accuracy (results with fewer unknown values), and scalability (handling of larger NUPN models), and six new options were added. The CAESAR.BDD and NUPN manual pages have been updated accordingly.

Our work on decomposing Petri nets into networks of automata based on the NUPN model led to a publication in an international conference [13]. Besides experimenting our tool chain extensively on a collection of over 12,000 Petri nets, we also applied it to decompose into networks of automata 113 models or instances of Petri nets proposed at the MCC contest.

7.1.4 Formal Modeling and Analysis of BPMN

Participants Francisco Durán, Yliès Falcone, Camilo Rocha, Gwen Salaün, Ahang Zuo.

BPMN is a workflow-based notation that has been published as an ISO standard and has become the main language for business process modeling. However, specifying processes with BPMN is not an easy task for non-experts and this is still an issue to make BPMN widely used in any company around the world. Process mining techniques are helpful to automatically infer processes from execution logs, but this is not a solution to make users more comfortable with BPMN. In the context of the MOAP project (see § 9.4.1), we proposed a different approach supporting the modelling of business processes in a semi-automated way. We focus on a timed version of BPMN, where to each task is associated a range indicating the minimum and maximum duration it takes to execute that task. In a first step, the user defines the tasks involved in the process and possibly gives a partial order between some of these tasks. A first algorithm then generates an abstract graph, which serves as a simplified version of the process being specified. Given such an abstract graph, a second algorithm computes the minimum and maximum time for executing the whole graph. The user can rely on this information for refining the graph. For each version of the graph, these minimum/maximum execution times are computed. Once the user is satisfied with a specific abstract graph, our third algorithm can be used to synthesize the BPMN process from that graph. This approach was implemented in a tool and validated on several case studies.

In collaboration with Francisco Durán (University of Málaga, Spain) and Camilo Rocha (University of Cali, Colombia), we also considered the optimization of business processes, which is a strategic activity in organizations because of its potential to increase profit margins and reduce operational costs. One of the

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

main challenges in this activity is concerned with the problem of optimizing the allocation and sharing of resources. Companies are continuously adjusting their resources to their needs following various dynamic provisioning strategies, which are difficult to compare. In this work, we proposed an automatic analysis technique to evaluate and compare the execution time and resource occupancy of a business process relative to a workload and a provisioning strategy. Such analysis is performed on models conforming to an extension of BPMN with quantitative information, including resource availability and constraints. Within this framework, the approach is fully mechanized using a formal and executable specification in the rewriting logic framework, which relies on existing techniques and tools for simulating probabilistic and real-time specifications. This work led to a publication in an international workshop [15].

7.2 Parallel and Distributed Verification

7.2.1 Debugging of Concurrent Systems using Counterexample Analysis

Participants Irman Faqrizal, Gwen Salaün.

Designing and developing distributed software has always been a tedious and error-prone task, and the ever increasing software complexity is making matters even worse. Model checking is an established technique for automatically finding bugs by 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 or program is a complicated task because the counterexample gives only a partial view of the source of the problem, and because there is usually little support beyond that counterexample to identify the source of the problem.

In 2020, we proposed a few techniques for simplifying the debugging of erroneous behavioural models represented as Labelled Transition Systems. We first focused on the erroneous part of the model, for which we detect specific states (called faulty states) where a choice is possible between executing a correct behaviour or falling into an erroneous part of the model. Our goal was to group these faulty states into clusters, which help the user to identify the source of the bug, since each cluster of states provides some information about the bug. We implemented this technique into a tool, which allows the visualization of the faulty model and the computation of clusters. This work led to a publication in an international conference [16].

7.2.2 Identifying Timing Interferences on Multicore Processors

Participants Frédéric Lang, Radu Mateescu, Wendelin Serwe.

Multicore platforms provide the computing capabilities and the power efficiency required by the complex applications embedded in aeronautical, spatial, and automotive systems. Some of the hardware resources provided by the platform — including buses, caches, IPs — are shared between concurrent tasks executing in parallel on different cores. This sharing may lead tasks to interfere with each other. Therefore, crucial design activities are to identify interferences, and bound the penalty induced by those interferences, as part of the demonstration of compliance of applications to their temporal requirements.

In the framework of the CAPHCA project (see § 9.3.1), in collaboration with Eric Jenn and Viet Anh Nguyen (IRT Saint-Exupéry, Toulouse), we studied the detection of interferences in concurrent applications using formal methods. A first and conservative approach is to consider that every access to a shared resource leads to an interference. This safe approach is usually too pessimistic to be useful. Therefore, we proposed a less pessimistic approach, which takes into account the actual behavior of the hardware and application to filter out situations where interferences cannot occur. Our method relies on (i) the behavioral modeling of applications and of their execution platform, using the LNT formal language, (ii) the definition of interferences using temporal properties, and (iii) the exploitation of the behavioral model and of the temporal properties, using the CADP toolbox. This method was applied to the Infineon AURIX TC275 system-on-chip, and the experimental results indicated that our approach is not only safe, but also prevents reporting spurious interferences compared to a purely structural analysis. This work led to a publication in an international conference [22].

7.2.3 Verification of Emergent Properties in Multi-agent Systems

Participants Luca Di Stefano, Frédéric Lang, Wendelin Serwe.

Multi-agent systems are collections of autonomous components that interact with each other and with their shared environment. These systems may display collective properties that arise from the interplay between agents. Reasoning about these properties turns out to be hard, due to the very large state space that these systems usually exhibit. Therefore, automatic procedures to formally guarantee the emergence of such properties may prove helpful in the design of reliable artificial multi-agent systems.

In 2020, we contributed to this topic as follows:

- We developed a fully-automated translation from a domain-specific language called LAbS [50] to the LNT formal description language. After translating a system specification into LNT, we can use CADP to either verify that the system displays emergent properties, or generate random execution traces which could be used, e.g., for testing. The procedure was described in L. Di Stefano's PhD thesis [51] and resulted in a paper which is currently under review by an international journal.
- The aforementioned translation was implemented as part of the SLiVER tool [51]. This work led to a publication in an international conference [14].

7.2.4 Other Developments

Participants Pierre Bouvier, Hubert Garavel.

We built the VLSAT-1 benchmark suite [24] (where "VL" stands for "Very Large"), a collection of 100 SAT formulas to be used as benchmarks in scientific experiments and software competitions. These SAT formulas have been obtained from the automatic conversion [13] into NUPNs of a large collection of Petri nets modelling real-life problems, such as communication protocols and concurrent systems. The VLSAT-1 benchmark suite is available via the CADP web site[3] or via its DOI[4].

7.3 Component-Based Architectures for On-the-Fly Verification

7.3.1 Compositional Verification

Participants Frédéric Lang, Radu Mateescu, Wendelin Serwe.

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 2020, we implemented a reduction preserving sharp bisimulation (see below) in BCG_MIN and SVL.

[3]`http://cadp.inria.fr/resources/vlsat`
[4]`http://dx.doi.org/10.18709/perscido.2020.03.ds300`

In collaboration with Franco Mazzanti (ISTI-CNR, Pisa, Italy), we used the compositional verification tools of CADP in the framework of the RERS'2020 challenge, which consisted in verifying 90 CTL properties on varying-size models of concurrent systems.

We applied to these examples a combination of techniques, namely:

- maximal hiding [46], which hides in the model all actions that are not necessary to verify the property;
- sharp bisimulation [18], which identifies in the property a set of so-called weak and strong actions, and uses this knowledge to (sometimes noticeably) enhance the reduction that can be applied to the model, while preserving the truth value of the formula;
- partial model checking [42], a compositional verification technique originally proposed by Andersen in the 90's and for which we developed an implementation on top of CADP.

This combination of techniques and tools allowed us to verify 79 out of the 90 CTL formulas and to win the RERS'2020 challenge. The 11 remaining formulas could not be solved due to state space explosion.

This work led to one publication in an international conference [18] and another publication accepted in an international journal to be published in 2021. Another work on compositional verification, carried out in collaboration with Sander de Putter and Anton Wijs (Eindhoven University of Technology, The Netherlands), led to a publication in an international journal [11].

7.3.2 On-the-fly Test Case Extraction

Participants Radu Mateescu, Wendelin Serwe.

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

To address these limitations, we developed the prototype tool TESTOR[5] to extract test cases completely on the fly. TESTOR presents several advantages w.r.t. TGV: (i) it has a more modular architecture, based on generic graph transformation components taken from the OPEN/CAESAR libraries (τ-compression, τ-confluence, τ-closure, determinization, resolution of Boolean equation systems); (ii) it is capable of extracting a test case entirely on the fly, by exploiting the diagnostic generation features of the Boolean equation system resolution algorithms; (iii) it enables a more flexible expression of test purposes, taking advantage of the multiway rendezvous, a primitive to express communication and synchronization among a set of distributed processes.

In 2020, in collaboration with Lina Marsso (University of Toronto, Canada), we proposed an automatic approach to generate a test plan (set of test purposes) with its associated test suite (set of test cases) covering all transitions of the IOLTS model of the system. The approach can also be applied to improve an existing test plan, by both completing the coverage and eliminating all redundancies. We implemented our approach on top of CADP and experimented it on several examples of concurrent systems. This enabled us to identify and evaluate possible variants and heuristics to fine-tune the overall performance of the approach, as well as the quality of the computed test plan. This work led to a publication in an international conference [20] and a new version 3.3 of the TESTOR tool.

7.3.3 Other Component Developments

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

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

In 2020, several components of CADP have been improved as follows:

- In addition to seven bug fixes, the MCL v4 language was extended with a function “card” to compute the number of elements in a set of natural numbers, which is a useful feature for specifying certain temporal operators on the past.
- In addition to eight bug fixes and various other enhancements, all the CADP tools built using Syntax and TRAIAN (namely, EXP2C, FSP2LOTOS, LNT2LOTOS, MCL_EXPAND, and SVL) have been recompiled using the new versions 3.0–3.2 of TRAIAN. The C compilers used to build the CADP tools have been upgraded to more recent versions, and all the C code (either source code of the CADP tools, or code generated by these tools) was modified to remove all compiler warnings.
- The binaries of CADP have been ported on SunOS 5.11 OpenIndiana, Debian 10.0, and macOS 10.15 “Catalina”, respectively.

7.4 Real-Life Applications and Case Studies

7.4.1 Failure Management Protocol for Stateful IoT Applications in the Fog

Participants 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 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, so that 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 may affect the physical world by inducing hazardous and costly situations. Together with Orange Labs, we proposed an end-to-end autonomic failure management approach for IoT applications deployed in the Fog, ensuring that failures are recovered in a cyber-physical consistent way [10]. Designing such highly distributed management protocols is a difficult and error-prone task.

In 2020, in collaboration with Loïc Letondeur (Orange Labs, Meylan), we undertook the formal specification and verification of this failure management protocol. We started by devising a formal specification of the protocol in LNT, encompassing the behaviour of the various manager entities involved (stable storage, global manager, and Fog agents). A part of the LNT model was generated automatically from an abstract description of the input application (Fog nodes, software elements, appliances, and their dependencies). This LNT model was used for extensive analysis to ensure that the protocol satisfies the desired properties. We identified twelve properties that must be respected by the protocol (architectural invariants, final objective, and functional properties), specified them formally in MCL, and verified them on the LNT model using the EVALUATOR model checker. This verification allowed us to detect several issues in the protocol and to correct them in its implementation developed by Orange Labs. This work led to a publication in an international conference [23].

7.4.2 Rigorous Design, Reconfiguration, and Deployment of IoT Applications

Participants Radu Mateescu, Ajay Muroor Nadumane, Gwen Salaün.

The Internet of Things (IoT) applications are built by interconnecting everyday objects over the 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 2020, we considered this challenge along two perspectives:

- In the framework of our collaboration with Nokia Bell Labs (see § 8.1.1), we continued our work on helping consumers to easily design correct IoT applications and also support the deployment of these applications. One popular way to build IoT applications in the consumer domain is by combining different objects using Event-Condition-Action (ECA) rules of the form "IF event THEN action". Our broad objective is to leverage formal methods to provide end-users of IoT applications design-time guarantees that the designed application will behave, upon deployment, as intended. In this context, we proposed a formal development framework based on the Web of Things (WoT). The objects are described using a behavioural model derived from the Thing Description specification of WoT. Then, the applications are designed not only by specifying individual ECA rules, but also by composing these rules using a simple, yet versatile composition language. The description of the objects and their composition are translated automatically into an LNT model, on which a set of generic and application-specific properties are verified using the CADP tools before the deployment of the application. All these steps are implemented and packaged in a tool named MozART, built on top of Mozilla WebThings platform. These results have been published in A. Muroor Nadumane's PhD thesis [49] and at an international conference [17].
- In collaboration with Francisco Durán (University of Málaga, Spain), we studied the reconfiguration of the deployed IoT applications during their lifecycle. The approach proposed relies on specifying reconfiguration properties that enable one to qualitatively compare the behaviour of the new configuration against the original configuration. The reconfiguration analysis is based on a specification in rewriting logic using Maude, and was implemented in the R-MozART tool built on top of the WebThings platform. This work led to a publication accepted in an international conference.

7.4.3 Asynchronous Circuit for the Protection against Physical Attacks

Participants Radu Mateescu, Wendelin Serwe.

In the context of the Securiot-2 project (see § 9.3.2), we experimented with modeling, at various abstraction levels, an asynchronous circuit patented by Tiempo Secure for detecting physical attacks, such as cutting wires, setting a wire to a constant voltage, or producing short-circuits. We considered the modeling and analysis of this circuit, called *shield*, at two abstraction levels. First, at circuit level, we took into account only the components (sequencers) of the circuit and their interconnection, without modeling the implementation details of these components. This level is appropriate for reasoning about the desired properties of the shield, namely the detection of physical attacks. The regular structure of the shield (serial pipeline or sequencers) enables inductive arguments that reduce all possible attack configurations to a finite set, which we analyzed exhaustively. Our analysis confirmed that all physical attacks, except two kinds of short-circuit, are detected by the shield. However, even if these short-circuits are theoretically possible, they are forbidden in practice by the physical layout of the shield, which therefore ensures full protection. Next, we undertook a gate-level modeling, focusing on the implementation of a sequencer in terms of logical gates. Here, we explored a range of different modeling variants for gates, electric wires, and forks (isochronic or not), and analyzed their respective impact on the faithfulness of the global circuit model, the size of the underlying state spaces, the expression of correctness properties, and the overall ease of verification. We also pointed out that certain modeling variants lead to deadlocks in the circuit.

This work led to a publication in an international workshop [21]. All models and verification scripts are available from the MARS model repository[6].

8 Bilateral contracts and grants with industry

8.1 Bilateral grants with industry

8.1.1 Nokia Bell Labs

Participants Radu Mateescu, Ajay Muroor Nadumane, Gwen Salaün *(correspondent)*.

Ajay Muroor Nadumane is supported by a PhD grant (from October 2017 to December 2020) from Nokia Bell Labs (Nozay) on IoT service composition (see § 7.4.2) supported by formal methods, under the supervision of Gwen Salaün (CONVECS), Radu Mateescu (CONVECS), and Michel Le Pallec (Nokia Bell Labs).

8.1.2 ST Microelectronics

Participants Philippe Ledent, Radu Mateescu *(correspondent)*, Wendelin Serwe.

Philippe Ledent is supported by a CIFRE PhD grant (from October 2020 to September 2023) from ST Microelectronics (Grenoble) on the formal validation of security requirements for Systems-on-Chip, under the supervision of Olivier Haller (ST Microelectronics), Radu Mateescu (CONVECS), and Wendelin Serwe (CONVECS).

9 Partnerships and cooperations

9.1 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.

9.1.1 Inria international partners

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.

9.1.2 Other International Collaborations

In 2020, we had scientific relations with several universities and institutes abroad, including:

- University of Málaga, Spain (Francisco Durán),
- University of Cali, Colombia (Camilo Rocha),
- ISTI/CNR, Pisa, Italy (Franco Mazzanti),
- Eindhoven University of Technology, The Netherlands (Anton Wijs and Sander de Putter),
- IMT School for Advanced Studies, Lucca, Italy (Rocco De Nicola),

[6] http://mars-workshop.org/repository/022-Shield.html

- Gran Sasso Science Institute, L'Aquila, Italy (Omar Inverso),
- University of Toronto, Canada (Lina Marsso).

9.2 European initiatives

9.2.1 FP7 & H2020 Projects

ArchitectECA2030

Participants Radu Mateescu *(correspondent)*, Lucie Muller, Wendelin Serwe.

Title: Trustable architectures with acceptable residual risk for the electric, connected and automated cars

Duration: *July 2020 - June 2023*

Coordinator: *Infineon Technologies AG (Germany)*

Partners:

- AVL List GMBH (Austria)
- Datasoft Embedded GMBH (Austria)
- Infineon Technologies AG (Austria)
- SBA Research GMBH (Austria)
- Virtual Vehicle Research GMBH (Austria)
- TU GRAZ (Austria)
- IMA (Czech Republic)
- Brno University of Technology (Czech Republic)
- Inria (France)
- Infineon Technologies AG (Germany)
- SafeTRANS e.V. (Germany)
- Volkswagen AG (Germany)
- TU Dresden (Germany)
- TeraGlobus (Lithuania)
- Nxtech (Norway)
- Sintef (Norway)
- TracSense (Norway)
- NXP (The Netherlands)
- TU Delft (The Netherlands)
- University of Nevada, Reno (USA)

Inria contact: *Radu Mateescu*

Summary: *Independent validation is fundamental for assessing the capability and safety of solutions in electric, connected and automated (ECA) vehicles. The project aims at designing electronic components and systems (ECS) in a robust, traceable, mission-validated way, quantifying the accepted residual risk of ECS for ECA vehicles, and increasing the end-user acceptance due to more reliable and robust ECS. The main contributions of CONVECS in the project are the formal modeling and validation of components embedded in autonomous vehicles.*

9.2.2 Collaborations with major European organizations

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

9.3 National initiatives

9.3.1 PIA (Programme d'Investissements d'Avenir)

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 and ended in September 2020. The main contributions of CONVECS to CAPHCA were the detection of concurrency errors and timing interferences in parallel applications by means of formal methods and verification techniques.

9.3.2 Competitivity Clusters

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 and ended in June 2020. The main contributions of CONVECS to SECURIOT-2 (see § 7.4.3) were the formal modeling and verification of the asynchronous hardware implementing the secure elements developed by the project partners.

9.3.3 Other National Collaborations

We had sustained scientific relations with the following researchers:

- Loïc Letondeur (Orange Labs, Meylan),
- Fabrice Kordon and Lom Messan Hillah (LIP6, Paris),
- Michel Le Pallec (Nokia Bell Labs, Nozay).

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

9.4 Regional initiatives

9.4.1 Pack Ambition Recherche Région Auvergne-Rhône-Alpes

MOAP

Participants Gwen Salaün *(correspondent)*, Ahang Zuo.

MOAP is a project funded by the Auvergne-Rhône-Alpes region within the *Pack Ambition Recherche* programme. The project involves the project-teams CONVECS and CORSE, and the SOITEC company. MOAP aims at providing modelling and automated analysis techniques for enabling companies to master the complexity of their internal processes and for optimizing those processes with the final goal of improving the quality and productivity of their businesses.

MOAP started in October 2020 for three years. The main contributions of CONVECS to MOAP are the formal modeling and automated verification of BPMN processes.

10 Dissemination

10.1 Promoting scientific activities

General chair, scientific chair

- H. Garavel is a member of the model board[8] of MCC (*Model Checking Contest*). In 2020, 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'2020 have been published online [41].

- Together with Peter Höfner (Data61, CSIRO, Sydney, Australia), H. Garavel set up a model repository (hosted on the Gforge of Inria) to collect and archive formal models of real systems; this infrastructure is used by the series of MARS workshops[9]. This repository currently contains 21 models, among which 7 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 Scientific events: selection

Chair of conference program committees

- Ansgar Fehnker (University of Twente, The Netherlands) and H. Garavel were co-chairs for MARS'2020 (*4th Workshop on Models for Formal Analysis of Real Systems*) held as part of ETAPS'2020, Dublin, Ireland, April 26, 2020.

Member of the conference program committees

- F. Lang was a programme committee member of ETR'2020 (*Ecole d'été Temps-Réel*). Due to the Covid-19 pandemic, the event has been postponed to 2021.

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

- R. Mateescu was a programme committee member of FMICS'2020 (*25th International Conference on Formal Methods for Industrial Critical Systems*), Vienna, Austria, September 2-3, 2020.
- R. Mateescu was a programme committee member of IFIP-ICTSS'2020 (*32nd IFIP International Conference on Testing Software and Systems*), Napoli, Italy, October 6-8, 2020.
- G. Salaün was program committee member of SAC-SVT'2020 (*35th ACM/SIGAPP Symposium on Applied Computing - Software Verification and Testing Track*), Brno, Czech Republic, March 30-April 3, 2020.
- G. Salaün was a programme committee member of COORDINATION'2020 (*22nd International Conference on Coordination Models and Languages*), La Valletta, Malta, June 15-19, 2020.
- G. Salaün was a programme committee member of COMPSAC-SETA'2020 (*IEEE International Conference on Computers, Software, and Applications - Software Engineering Technologies and Applications*), Madrid, Spain, July 13-17, 2020.
- G. Salaün was a programme committee member of FormaliSE'2020 (*8th International Conference on Formal Methods in Software Engineering*), virtual event, July 13, 2020.
- G. Salaün was a programme committee member of SEFM'2020 (*18th International Conference on Software Engineering and Formal Methods*), Amsterdam, The Netherlands, September 14-18, 2020.
- G. Salaün was a programme committee member of FOCLASA'2020 (*18th International Workshop on Coordination and Self-Adaptativeness of Software Applications*), Amsterdam, The Netherlands, September 15, 2020.
- G. Salaün was a programme committee member of HPCS-SERCO'2020 (*4th Special Session on High Performance Services Computing and Internet Technologies*), virtual event, January 25-29, 2021.

Reviewer

- F. Lang was a reviewer for SAC-SVT'2020 and MARS'2020.
- A. Muroor Nadumane was a reviewer for SAC-SVT'2020, SEFM'2020, COORDINATION'2020, COMPSAC-SETA'2020, and FormaliSE'2020.
- W. Serwe was a reviewer for SEFM'2020.

10.1.2 Journal

Member of the editorial boards

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

Reviewer - reviewing activities

- F. Lang was a reviewer for JSS (*Journal of Systems & Software*).
- R. Mateescu was a reviewer for ISSE (*Innovations in Systems and Software Engineering*) and SQJ (*Software Quality Journal*).
- G. Salaün was a reviewer for JUCS (*Journal of Universal Computer Science*), LMCS (*Logical Methods in Computer Science*), SCP (*Science of Computer Programming*), SOCA (*Service Oriented Computing and Applications*), SOSYM (*Software and Systems Modeling*), SQJ, and TSE (*IEEE Transactions on Software Engineering*).

10.1.3 Software Dissemination and Internet Visibility

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

In 2020, the main facts are the following:

- We prepared and distributed twelve successive versions (2020-a to 2020-l) of CADP.
- We were requested to grant CADP licenses for 222 different computers, located in 47 different institutions in the world.

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

By the end of December 2020, the CADP forum[11], opened in 2007 for discussions regarding the CADP toolbox, had over 455 registered users and over 1957 messages had been exchanged.

Also, for the 2020 edition of the Model Checking Contest, we provided 7 families of models (totalling 141 Nested-Unit Petri Nets) derived from our LNT models.

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

- The OCARINA tool for Analyzing AADL Descriptions [47, 48]
- Verification of Formal Requirements in the Context of ISO 26262 [44]

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

- Modeling the Raft Distributed Consensus Protocol in LNT [30]
- Detection of Android Malware using Model Checking and Machine Learning [38, 39]

10.1.4 Invited talks

- A. Muroor Nadumane gave a talk entitled "*Verification Guided Design and Deployment of IoT Applications*" at the Inria-Nokia Bell Labs seminar held as a virtual event on November 5, 2020.

10.1.5 Research administration

- 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 was (until September 2020) a member of the "*Bureau*" of the LIG laboratory.
- R. Mateescu was appointed to the Executive Commission in charge of International Relations at COMUE Université Grenoble Alpes.
- 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 2020 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 was (until September 2020) "*chargé de mission*" for the scientific axis *Formal Methods, Models, and Languages* of the LIG laboratory.

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

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 2020, we carried out the following teaching activities:

- P. Bouvier supervised two groups of 1st year students in the context of the course "*Algorithmique*" (33 hours "*équivalent TD*") at ENSIMAG.
- F. Lang gave a course on "*Formal Software Development Methods*" (7.5 hours "*équivalent TD*") in the framework of the "*Software Engineering*" lecture given to first year students of the MOSIG.
- F. Lang and 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.
- G. Salaün taught about 230 hours of classes (algorithmics, Web development, object-oriented 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 supervised a group of six teams in the context of the "*projet Génie Logiciel*" (55 hours "*équivalent TD*", consisting in 13.5 hours of lectures, plus supervision and evaluation), ENSIMAG, January 2020.
- W. Serwe, together with Ioannis Parissis (LCIS), gave a lecture on "*Verification and Test Theories*" (18 hours "*équivalent TD*" on behavioural testing) to second year students of the MOSIG.

10.2.2 Supervision

- PhD: A. Muroor Nadumane, "*Models and Verification for Composition and Reconfiguration of Web of Things Applications*", Université Grenoble Alpes, December 10, 2020, G. Salaün, R. Mateescu, and M. Le Pallec
- PhD in progress: P. Bouvier, "*Implémentation et vérification des langages concurrents de nouvelle génération*", Université Grenoble Alpes, since October 2019, H. Garavel and R. Mateescu
- PhD in progress: P. Ledent, "*Formal Validation of Security Requirements for a System-on-Chip Architecture*", Université Grenoble Alpes, since October 2020, R. Mateescu, W. Serwe, and O. Haller
- PhD in progress: L. Muller, "*Formal Modelling and Validation for Electric, Connected, and Automated Vehicles*", Université Grenoble Alpes, since September 2020, R. Mateescu and W. Serwe
- PhD in progress: A. Zuo, "*Modelling, Optimization and Predictive Analysis of Business Processes*", Université Grenoble Alpes, since October 2020, G. Salaün and Yliès Falcone

10.2.3 Juries

- R. Mateescu was reviewer of Thomas Neele's PhD thesis, entitled "*Reductions for Parity Games and Model Checking*", defended at Eindhoven University of Technology (The Netherlands) on September 16, 2020.
- R. Mateescu was reviewer of Luca Di Stefano's PhD thesis, entitled "*Modelling and Verification of Multi-Agent Systems via Sequential Emulation*", defended at Gran Sasso Science Institute (Italy) on October 13, 2020.
- R. Mateescu was reviewer of Anders Mariegaard's PhD thesis, entitled "*Quantitative Systems: Efficient Reasoning under Uncertainty*", defended at Aalborg University (Denmark) on November 6, 2020.

- R. Mateescu was reviewer of Hermann Felbinger's PhD thesis, entitled "*Characterizing Quality Assessment and Redundancy Elimination of Test Suites Without Execution*", defended at Graz University of Technology (Austria) on December 22, 2020.
- G. Salaün was reviewer of Samir Chouali's Habilitation thesis, entitled "*Contributions à la conception rigoureuse des systèmes à base de composants exploitant des modèles SysML et des approches formelles*", defended at Université de Franche-Comté on July 10, 2020.

10.3 Popularization

10.3.1 Articles and contents

Participants Hubert Garavel.

At the FMICS'2020 international conference, together with Maurice ter Beek (ISTI-CNR, Pisa, Italy) and Jaco van de Pol (University of Twente, The Netherlands), we organized an expert survey on formal methods to celebrate the 25th anniversary of FMICS. The survey addressed 30 questions on the past, present, and future of formal methods in research, industry, and education. The detailed report of the study [26] presents an analysis of the opinions of 130 renowned experts in formal methods (among which three Turing award winners), as well as thought-provoking position statements on formal methods of 111 of them. The survey is both an exercise in collective thinking and a family picture of key actors in formal methods.

11 Scientific production

11.1 Major publications

[1] X. Etchevers, G. Salaün, F. Boyer, T. Coupaye and N. De Palma. 'Reliable Self-deployment of Distributed Cloud Applications'. In: *Software: Practice and Experience* 47.1 (2017), pp. 3–20. DOI: 10.1002/spe.2400. URL: https://hal.inria.fr/hal-01290465.

[2] H. Evrard and 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* 88 (Mar. 2017), p. 33. DOI: 10.1016/j.jlamp.2016.09.002. URL: https://hal.inria.fr/hal-01412911.

[3] H. Garavel. 'Nested-unit Petri nets'. In: *Journal of Logical and Algebraic Methods in Programming* 104 (Apr. 2019), pp. 60–85. DOI: 10.1016/j.jlamp.2018.11.005. URL: https://hal.inria.fr/hal-02072190.

[4] H. Garavel, F. Lang and R. Mateescu. 'Compositional Verification of Asynchronous Concurrent Systems using CADP'. In: *Acta Informatica* 52.4 (June 2015), p. 56. DOI: 10.1007/s00236-015-0226-1. URL: https://hal.inria.fr/hal-01247507.

[5] H. Garavel, F. Lang, R. Mateescu and W. Serwe. 'CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes'. In: *International Journal on Software Tools for Technology Transfer* 15.2 (2013), pp. 89–107. DOI: 10.1007/s10009-012-0244-z. URL: http://hal.inria.fr/hal-00715056.

[6] H. Garavel, F. Lang and W. Serwe. 'From LOTOS to LNT'. In: *ModelEd, TestEd, TrustEd - Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday*. Ed. by J.-P. Katoen, R. Langerak and A. Rensink. Vol. 10500. Lecture Notes in Computer Science. Springer, Oct. 2017, pp. 3–26. DOI: 10.1007/978-3-319-68270-9_1. URL: https://hal.inria.fr/hal-01621670.

[7] A. Krishna, P. Poizat and G. Salaün. 'Checking Business Process Evolution'. In: *Science of Computer Programming* 170 (Jan. 2019), pp. 1–26. DOI: 10.1016/j.scico.2018.09.007. URL: https://hal.inria.fr/hal-01920273.

[8] R. Mateescu and W. Serwe. 'Model Checking and Performance Evaluation with CADP Illustrated on Shared-Memory Mutual Exclusion Protocols'. In: *Science of Computer Programming* (Feb. 2012). DOI: 10.1016/j.scico.2012.01.003. URL: http://hal.inria.fr/hal-00671321.

11.2 Publications of the year

International journals

[9] A. Inants and J. Euzenat. 'So, what exactly is a qualitative calculus?' In: *Artificial Intelligence* 289 (2020), p. 103385. DOI: 10.1016/j.artint.2020.103385. URL: https://hal.archives-ouvertes.fr/hal-02942947.

[10] U. Ozeer, L. Letondeur, G. Salaün, F.-G. Ottogalli and J.-M. Vincent. 'F 3 ARIoT: A Framework for Autonomic Resilience of IoT Applications in the Fog'. In: *Internet of Things* (1st Dec. 2020), pp. 1–54. DOI: 10.1016/j.iot.2020.100275. URL: https://hal.inria.fr/hal-02933365.

[11] S. de Putter, F. Lang and A. Wijs. 'Compositional model checking with divergence preserving branching bisimilarity is lively'. In: *Science of Computer Programming* 196 (Sept. 2020), p. 102493. DOI: 10.1016/j.scico.2020.102493. URL: https://hal.inria.fr/hal-02890800.

[12] G. Salaün. 'Quantifying the Similarity of Non-bisimilar Labelled Transition Systems'. In: *Science of Computer Programming* 202 (1st Feb. 2021). DOI: 10.1016/j.scico.2020.102580. URL: https://hal.inria.fr/hal-03017666.

International peer-reviewed conferences

[13] P. Bouvier, H. Garavel and H. Ponce de León. 'Automatic Decomposition of Petri Nets into Automata Networks - A Synthetic Account'. In: PETRI NETS 2020 - 41st International Conference on Application and Theory of Petri Nets and Concurrency. Paris, France, 24th June 2020. URL: https://hal.inria.fr/hal-02875957.

[14] L. Di Stefano, F. Lang and W. Serwe. 'Combining SLiVER with CADP to Analyze Multi-agent Systems'. In: *Lecture Notes in Computer Science*. COORDINATION 2020 - 22nd IFIP WG 6.1 International Conference on Coordination Models and Languages. Vol. 12134. La Valetta, Malta, 10th June 2020, pp. 370–385. DOI: 10.1007/978-3-030-50029-0_23. URL: https://hal.inria.fr/hal-02890401.

[15] F. Duran, C. Rocha and G. Salaün. 'Analysis of the Runtime Resource Provisioning of BPMN Processes using Maude'. In: WRLA 2020 - 13th International Workshop on Rewriting Logic and its Applications. Dublin, Ireland, 25th Apr. 2020, pp. 1–16. URL: https://hal.inria.fr/hal-02931077.

[16] I. Faqrizal and G. Salaün. 'Clusters of Faulty States for Debugging Behavioural Models'. In: APSEC 2020 - 27th Asia-Pacific Software Engineering Conference. Singapore, Singapore, 2nd Dec. 2020, pp. 1–9. URL: https://hal.inria.fr/hal-03035539.

[17] A. Krishna, M. Le Pallec, A. Martinez, R. Mateescu and G. Salaün. 'MOZART: Design and Deployment of Advanced IoT Applications'. In: WWW 2020 - International World Wide Web Conference. Taipei, Taiwan, 20th Apr. 2020, pp. 1–4. DOI: 10.1145/3366424.3383532. URL: https://hal.inria.fr/hal-02554029.

[18] F. Lang, R. Mateescu and F. Mazzanti. 'Sharp Congruences Adequate with Temporal Logics Combining Weak and Strong Modalities'. In: TACAS 2020 - Tools and Algorithms for the Construction and Analysis of Systems. Vol. 12079. Lecture Notes in Computer Science. Dublin, Ireland, 17th Apr. 2020, pp. 57–76. DOI: 10.1007/978-3-030-45237-7_4. URL: https://hal.inria.fr/hal-02555692.

[19] L. Marsso. 'Specifying a Cryptographical Protocol in Lustre and SCADE'. In: MARS 2020 - 4th Workshop on Models for Formal Analysis of Real Systems. Vol. 316. Electronic Proceedings in Theoretical Computer Science. Dublin, Ireland, 26th Apr. 2020, pp. 149–199. DOI: 10.4204/EPTCS.316.7. URL: https://hal.inria.fr/hal-02556856.

[20] L. Marsso, R. Mateescu and W. Serwe. 'Automated Transition Coverage in Behavioural Conformance Testing'. In: ICTSS 2020 - 32nd IFIP International Conference on Testing Software and Systems. Napoli, Italy, 2nd Dec. 2020, pp. 219–235. DOI: 10.1007/978-3-030-64881-7_14. URL: https://hal.inria.fr/hal-03038050.

[21] R. Mateescu, W. Serwe, A. Bouzafour and M. Renaudin. 'Modeling an Asynchronous Circuit Dedicated to the Protection Against Physical Attacks'. In: MARS 2020 - 4th Workshop on Models for Formal Analysis of Real Systems. Vol. 316. Electronic Proceedings in Theoretical Computer Science. Dublin, Ireland, 26th Apr. 2020, pp. 200–239. DOI: 10.4204/EPTCS.316.8. URL: https://hal.inria.fr/hal-02559125.

[22] V. a. Nguyen, E. Jenn, W. Serwe, F. Lang and R. Mateescu. 'Using Model Checking to Identify Timing Interferences on Multicore Processors'. In: ERTS 2020 - 10th European Congress on Embedded Real Time Software and Systems. Toulouse, France: http://www.erts2020.org/, 29th Jan. 2020, pp. 1–10. URL: https://hal.inria.fr/hal-02462085.

[23] U. Ozeer, G. Salaün, L. Letondeur, F.-G. Ottogalli and J.-M. Vincent. 'Verification of a Failure Management Protocol for Stateful IoT Applications'. In: Proc. of FMICS'20. Vienne, Austria, 2nd Sept. 2020. DOI: 10.1007/978-3-030-58298-2_12. URL: https://hal.inria.fr/hal-02930872.

Reports & preprints

[24] P. Bouvier and H. Garavel. *The VLSAT-1 Benchmark Suite*. INRIA Grenoble Rhône-Alpes, 16th Nov. 2020, p. 6. URL: https://hal.archives-ouvertes.fr/hal-03007233.

[25] H. Garavel. *Proposal for Adding Useful Features to Petri-Net Model Checkers*. Inria Grenoble - Rhône-Alpes, 23rd Dec. 2020. URL: https://hal.inria.fr/hal-03087421.

11.3 Other

Scientific popularization

[26] H. Garavel, M. ter Beek and J. van de Pol. 'The 2020 Expert Survey on Formal Methods'. In: FMICS 2020: 25th International Conference on Formal Methods for Industrial Critical Systems. Vienna, Austria, 29th Aug. 2020, pp. 3–69. DOI: 10.1007/978-3-030-58298-2_1. URL: https://hal.inria.fr/hal-03082818.

11.4 Cited publications

[27] D. Champelovier, X. Clerc, H. Garavel, Y. Guerte, C. McKinty, V. Powazny, F. Lang, W. Serwe and G. Smeding. 'Reference Manual of the LNT to LOTOS Translator (Version 6.8)'. INRIA, Grenoble, France. Jan. 2019.

[28] E. M. Clarke, E. A. Emerson and A. P. Sistla. 'Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications'. In: *ACM Transactions on Programming Languages and Systems* 8.2 (Apr. 1986), pp. 244–263.

[29] R. De Nicola and F. W. Vaandrager. 'Action versus State Based Logics for Transition Systems'. In: *Semantics of Concurrency*. Vol. 469. Lecture Notes in Computer Science. Springer Verlag, 1990, pp. 407–419.

[30] H. Evrard. 'Modeling the Raft Distributed Consensus Protocol in LNT'. In: *Proceedings of the 4th Workshop on Models for Formal Analysis of Real Systems, MARS@ETAPS'2020 (Dublin, Ireland)*. Ed. by A. Fehnker and H. Garavel. Vol. 316. EPTCS. Apr. 2020, pp. 15–39.

[31] 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)*. Ed. by S. T. Vuong. North Holland, Dec. 1989, pp. 147–162.

[32] 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)*. Ed. by B. Steffen. Vol. 1384. Lecture Notes in Computer Science. Full version available as INRIA Research Report RR-3352. Berlin: Springer Verlag, Mar. 1998, pp. 68–84.

[33] H. Garavel and 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)*. Ed. by M. Kim, B. Chin, S. Kang and D. Lee. Full version available as INRIA Research Report RR-4223. IFIP. Kluwer Academic Publishers, Aug. 2001, pp. 377–392.

[34] H. Garavel, F. Lang and R. Mateescu. ‘Compiler Construction using LOTOS NT’. In: *Proceedings of the 11th International Conference on Compiler Construction CC 2002 (Grenoble, France)*. Ed. by N. Horspool. Vol. 2304. Lecture Notes in Computer Science. Springer Verlag, Apr. 2002, pp. 9–13.

[35] H. Garavel, R. Mateescu and 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)*. Ed. by M. B. Dwyer. Vol. 2057. Lecture Notes in Computer Science. Revised version available as INRIA Research Report RR-4341 (December 2001). Berlin: Springer Verlag, May 2001, pp. 217–234.

[36] H. Garavel and W. Serwe. ‘State Space Reduction for Process Algebra Specifications’. In: *Theoretical Computer Science* 351.2 (Feb. 2006), pp. 131–145.

[37] H. Garavel and J. Sifakis. ‘Compilation and Verification of LOTOS Specifications’. In: *Proceedings of the 10th International Symposium on Protocol Specification, Testing and Verification (Ottawa, Canada)*. Ed. by L. Logrippo, R. L. Probert and H. Ural. IFIP. North Holland, June 1990, pp. 379–394.

[38] S. E. Hatib. ‘Une approche sémantique de détection de maliciel Android basée sur la vérification de modèles et l’apprentissage automatique’. Master thesis. Québec, Canada: Laval University, 2020.

[39] S. E. Hatib, L. Ricaud, J. Desharnais and N. Tawbi. ‘Toward Semantic-Based Android Malware Detection Using Model Checking and Machine Learning’. In: *Risks and Security of Internet and Systems - 15th International Conference, CRiSIS’2020 (Paris, France)*. Ed. by J. García-Alfaro, J. Leneutre, N. Cuppens and R. Yaich. Vol. 12528. Lecture Notes in Computer Science. Springer, Nov. 2020, pp. 289–307.

[40] M. Hennessy and R. Milner. ‘Algebraic Laws for Nondeterminism and Concurrency’. In: *Journal of the ACM* 32 (1985), pp. 137–161.

[41] F. Kordon, H. Garavel, L. M. Hillah, F. Hulin-Hubard, E. Amparore, B. Berthomieu, S. Biswal, D. Donatelli, F. Galla, G. Ciardo, S. Dal Zilio, P. Jensen, C. He, D. Le Botlan, S. Li, A. Miner, J. Srba and . Thierry-Mieg. *Complete Results for the 2020 Edition of the Model Checking Contest*. http://mcc.lip6.fr/2020/results.php. June 2020. (Visited on 2020).

[42] F. Lang and R. Mateescu. ‘Partial Model Checking using Networks of Labelled Transition Systems and Boolean Equation Systems’. In: *Logical Methods in Computer Science* 9.4 (Oct. 2013). URL: `https://hal.inria.fr/hal-00872181`.

[43] J. Magee and J. Kramer. *Concurrency: State Models and Java Programs*. 2006th ed. Wiley, Apr. 2006.

[44] D. Makartetskiy, G. Marchetto, R. Sisto, F. Valenza, M. Virgilio, D. Leri, P. Denti and R. Finizio. ‘(User-friendly) Formal Requirements Verification in the Context of ISO 26262’. In: *Engineering Science and Technology, an International Journal* 23.3 (2020), pp. 494–506.

[45] R. Mateescu and 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)*. Ed. by J. Cuellar, T. Maibaum and K. Sere. Vol. 5014. Lecture Notes in Computer Science. Springer Verlag, May 2008, pp. 148–164.

[46] R. Mateescu and A. Wijs. ‘Property-Dependent Reductions Adequate with Divergence-Sensitive Branching Bisimilarity’. In: *Science of Computer Programming* (Apr. 2014). DOI: `10.1016/j.scico.2014.04.004`. URL: `https://hal.inria.fr/hal-01016922`.

[47] H. Mkaouar, B. Zalila, J. Hugues and M. Jmaiel. 'A Formal Approach to AADL Model-based Software Engineering'. In: *Springer International Journal on Software Tools for Technology Transfer (STTT)* 22 (2020), pp. 219–247.

[48] H. Mkaouar, B. Zalila, J. Hugues and M. Jmaiel. 'Towards a Formal Specification for an AADL Behavioural Subset using the LNT Language'. In: *International Journal of Business and Systems Research* 14.2 (2020), pp. 162–190.

[49] A. K. M. Nadumane. 'Models and Verification for Composition and Reconfiguration of Web of Things Applications'. PhD Thesis. Université Grenoble Alpes, Dec. 2020.

[50] R. D. Nicola, L. Di Stefano and O. Inverso. 'Multi-agent Systems with Virtual Stigmergy'. In: *Sci. Comput. Program.* 187 (2020), p. 102345.

[51] L. D. Stefano. 'Modelling and Verification of Multi-Agent Systems via Sequential Emulation'. PhD Thesis. Gran Sasso Science Institute, L'Aquila, Italy, Oct. 2020.