RESEARCH CENTRE

Inria Center at Université Grenoble Alpes

IN PARTNERSHIP WITH:

Université de Grenoble Alpes

2022

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 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

B6.5. – Information systems

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]
- Frederic Lang [INRIA, Researcher]
- Wendelin Serwe [INRIA, Researcher]

Faculty Member

- Gwen Salaün [UGA, Professor, HDR]

Post-Doctoral Fellow

- Luca Di Stefano [INRIA, until Apr 2022]

PhD Students

- Pierre Bouvier [UGA]
- Irman Faqrizal [UGA]
- Florian Gallay [UGA, from Oct 2022]
- Jean-Baptiste Horel [INRIA]
- Philippe Ledent [ST Microelectronics]
- Lucie Muller [INRIA]
- Quentin Nivon [UGA, from Oct 2022]
- Ahang Zuo [UGA]

Technical Staff

- Dasarada Ramu Munnangi [INRIA, Engineer, from Feb 2022]
- Ajay Krishna Muroor-Nadumane [INRIA, Engineer, from Feb 2022]

Interns and Apprentices

- Almo Cuci [INRIA and EURIS, until Jun 2022]
- Koffi Marck-Edward Kemeh [UGA, from Feb 2022 until Jul 2022]
- Quentin Nivon [UGA, from Feb 2022 until Jul 2022]
- Carlos Uriel Vargas Lopez [INRIA Startup Studio, from Oct 2022]

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

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

3.2 Parallel and Distributed Verification

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

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

3.3 Timed, Probabilistic, and Stochastic Extensions

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

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

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

To address these issues, CONVECS follows two research directions:

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

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

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

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

- *New interfaces for on-the-fly LTS manipulation.* The current application programming interface (API) for on-the-fly graph manipulation, named OPEN/CAESAR [35], provides an “opaque” representation of states and actions (transitions labels): states are represented as memory areas of fixed size and actions are character strings. Although appropriate to the pure process algebraic setting, this representation must be generalized to provide additional information supporting an efficient construction of advanced verification features, such as: handling of the types, functions, data values, and parallel structure of the source program under verification, independence of transitions in the LTS, quantitative (timed/probabilistic/stochastic) information, etc.

- *Compositional framework for on-the-fly LTS analysis.* On-the-fly model checkers and equivalence checkers usually perform several operations on graph models (LTSs, Boolean graphs, etc.), such as exploration, parallel composition, partial order reduction, encoding of model checking and equivalence checking in terms of Boolean equation systems, resolution and diagnostic generation for Boolean equation systems, etc. To facilitate the design, implementation, and usage of these functionalities, it is necessary to encapsulate them in software components that could be freely combined and replaced. Such components would act as graph transformers, that would execute (on a sequential machine) in a way similar to coroutines and to the composition of lazy functions in functional programming languages. Besides its obvious benefits in modularity, such a component-based architecture will also make it possible to take advantage of multi-core processors.

- *New generic components for on-the-fly verification.* The quest for new on-the-fly components for LTS analysis must be pursued, with the goal of obtaining a rich catalog of interoperable components serving as building blocks for new analysis features. A long-term goal of this approach is to provide an increasingly large catalog of interoperable components covering all verification and analysis functionalities that appear to be useful in practice. It is worth noticing that some components can be very complex pieces of software (e.g., the encapsulation of an on-the-fly model checker for a rich temporal logic). Ideally, it should be possible to build a novel verification or analysis tool by assembling on-the-fly graph manipulation components taken from the catalog. This would provide a flexible means of building new verification and analysis tools by reusing generic, interoperable model manipulation components.

3.5 Real-Life Applications and Case Studies

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

4 Application domains

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

- *Bioinformatics:* genetic regulatory networks, nutritional stress response, metabolic pathways,

- *Component-based systems:* Web services, peer-to-peer networks,

- *Cloud computing:* self-deployment protocols, dynamic reconfiguration protocols,

- *Databases:* transaction protocols, distributed knowledge bases, stock management,

- *Distributed systems:* virtual shared memory, dynamic reconfiguration algorithms, fault tolerance algorithms, multi-agent systems,

- *Embedded systems:* air traffic control, autonomous vehicles, avionic systems, train supervision systems, medical devices,

- *Enterprise systems:* business processes, information systems, manufacturing,

- *Fog and IoT:* stateful IoT applications in the fog,

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

5 New software and platforms

5.1 New software

5.1.1 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 [34] is a compiler that translates LOTOS abstract data types into C types and C functions. The translation involves pattern-matching compiling techniques and automatic recognition of usual types (integers, enumerations, tuples, etc.), which are implemented optimally.
- CAESAR [40, 39] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purposes). The translation is done using several intermediate steps, among which the construction of a Petri net extended with typed variables, data handling features, and atomic transitions.
- OPEN/CAESAR [35] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently of any particular high level language. In this respect, OPEN/CAESAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CAESAR consists of a set of 16 code libraries with their programming interfaces, such as:
  - CAESAR_GRAPH, which provides the programming interface for graph exploration,
  - CAESAR_HASH, which contains several hash functions,
  - CAESAR_SOLVE, which resolves Boolean equation systems on the fly,
  - CAESAR_STACK, which implements stacks for depth-first search exploration, and
  - CAESAR_TABLE, which handles tables of states, transitions, labels, etc.

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

  - EXECUTOR, which performs random execution,
  - EXHIBITOR, which searches for execution sequences matching a given regular expression,
  - GENERATOR, which constructs the graph of reachable states,
  - PROJECTOR, which computes abstractions of communicating systems,
  - REDUCTOR, which constructs and minimizes the graph of reachable states modulo various equivalence relations,
  - SIMULATOR, XSIMULATOR, and OCIS, which enable interactive simulation, and
  - TERMINATOR, which searches for deadlock states.

- BCG (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. BCG also plays a key role in CADP as many tools rely on this format for their inputs/outputs. The BCG environment consists of various libraries with their programming interfaces, and of several tools, such as:

  - BCG_CMP, which compares two graphs,
  - BCG_DRAW, which builds a two-dimensional view of a graph,
  - BCG_EDIT, which allows the graph layout produced by BCG_DRAW to be modified interactively,
  - BCG_GRAPH, which generates various forms of practically useful graphs,
  - BCG_INFO, which displays various statistical information about a graph,
  - BCG_IO, which performs conversions between BCG and many other graph formats,
  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph,
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc.
    For instance, one can define recursive functions on sets of states, which allow evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [41], CTL[30], ACTL[32], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [38] and providing a unified access to a graph partitioned in fragments distributed over a set of remote machines, possibly located in different countries. The PBG format is supported by several tools, such as:

  - PBG_CP, PBG_MV, and PBG_RM, which facilitate standard operations (copying, moving, and removing) on PBG files, maintaining consistency during these operations,
  - PBG_MERGE (formerly known as BCG_MERGE), which transforms a distributed graph into a monolithic one represented in BCG format,
  - PBG_INFO, which displays various statistical information about a distributed graph.

- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CAESAR-compliant compilers, e.g.:

  - BCG_OPEN, for models represented as BCG graphs,
  - CAESAR.OPEN, for models expressed as LOTOS descriptions,
  - EXP.OPEN, for models expressed as communicating automata,

- FSP.OPEN, for models expressed as FSP [42] descriptions,
- LNT.OPEN, for models expressed as LNT descriptions, and
- SEQ.OPEN, for models represented as sets of execution traces.

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

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

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

Contact: Hubert Garavel

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

5.1.2 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 [37], being used in all recent compilers developed by CONVECS.

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

Contact: Hubert Garavel

Participants: Hubert Garavel, Frederic Lang, Wendelin Serwe

6 New results

6.1 New Formal Languages and their Implementations

6.1.1 LNT Specification Language

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

LNT [6] [29] is a next-generation formal description language for asynchronous concurrent systems. The design of LNT at CONVECS is the continuation of the efforts undertaken in the 80s to define sound languages for concurrency theory and, indeed, LNT is derived from the ISO standards LOTOS (1989) and E-LOTOS (2001). In a nutshell, LNT attempts to combine the best features of imperative programming languages, functional languages, and value-passing process calculi.

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

In 2022, the LNT language has continued to evolve, mainly to become a better language and to achieve convergence with the LOTOS NT language supported by the TRAIAN compiler (see § 6.1.2):

- The syntax of natural numbers (binary, octal, hexadecimal) and floating-point numbers was brought closer to mainstream programming languages. The "while" and "for" loops were equipped with an optional label that can be used in "break" statements. Function calls were simplified by making the "eval" keyword optional, and output offers in synchronizations were simplified by forbidding the "!" symbol. Set types were equipped with a predefined operation "subset" checking for set inclusion, and the usage of the "`{}`" constructor was made clearer. The pragmas related to the translation of LNT types and functions in the C language were improved by enforcing the usage of valid C identifiers, and two new pragmas "!pointer" and "!nopointer" were introduced to indicate whether the designated LNT types are implemented as pointer types in C or not.

- The LNT2LOTOS Reference Manual and the LNT tools have been updated to document and implement these changes of LNT, and the CADP demo examples have been upgraded to the latest version of LNT. Four new CADP demo examples have been translated from LOTOS to LNT. The "upc" shell script was extended to ease the migration of LNT programs, in particular the over 15,000 LNT examples used for non-regression testing of LNT2LOTOS. A new tool named LNT_MERGE was added, which transforms a (possibly multi-module) LNT program into a single-module LNT program (also merging the external C code, if any). This tool enables to package multi-module LNT programs into non-regression test suites.

- Also, in addition to seven bug fixes, the LNT2LOTOS compiler was enhanced in several manners. Various sanity checks on pragmas were added, as well as warnings regarding dead code and unsafe usage of process parameters in infinite loops. The generation of LOTOS code was improved to avoid warnings of GCC for certain exhaustive LNT patterns, and warnings about deprecated LNT syntactic features were enhanced with source-code line numbers.

6.1.2 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. Since TRAIAN 3.0, the lexer and parser of TRAIAN are built using the SYNTAX compiler-generation system developed at Inria Paris and 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 are implemented in LOTOS NT itself, so that TRAIAN is capable of bootstrapping itself.

In 2022, our efforts led to the release of three new major versions of TRAIAN, which introduce various changes to the LOTOS NT language in order to further align it with the LNT language supported by the CADP toolbox:

- TRAIAN 3.6 brings twenty-four language changes, mostly to extend the syntax of types, processes, expressions, and patterns (including their type-checking), as well as twelve library and code-generation changes that improve the predefined LOTOS NT libraries and the generated C code. Five bugs were fixed.

- TRAIAN 3.7 brings four language changes introducing array manipulation features and improving pragmas, four library changes notably simplifying exception handling, ten static-semantics changes, notably introducing dataflow analysis for processes, and five code-generation changes accomodating the new features. Nineteen bugs were fixed.

- TRAIAN 3.8 brings thirteen language changes that improve the handling of exceptions and function parameters, as well as the syntax of processes, synchronizations, and modules, nine static-semantics changes improving the type-checking of arrays, loops, and function parameters, two compiler changes to fine-tune the code generation, and six release changes for porting TRAIAN on 64-bit Windows and improving the demo examples. Six bugs were fixed.

In all versions of TRAIAN, the “traian_upc” conversion tool was extended to ease the upgrade of existing LOTOS NT source code whenever possible, and the user manual of TRAIAN, as well as the demo examples, were constantly updated.

6.1.3 Formal Modeling and Analysis of BPMN

Participants: Angel Contreras, Quentin Nivon, Gwen Salaün *(correspondent)*, Ahang Zuo.

Business Process Model and Notation (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 at ease with BPMN. Also, optimizing a BPMN process is difficult, especially for non-expert users, due to the lack of design time support for BPMN.

In 2022, we considered the modeling and analysis of BPMN processes along three research directions:

Semi-automation of BPMN modeling. In the context of the MOAP project (see § 8.5.1), in collaboration with Yliès Falcone (CORSE project-team), we continued the work on our approach for modelling business processes in a semi-automated way [33]. The approach focuses on a timed version of BPMN, where each task is annotated with its minimum and maximum execution duration. The user starts by defining the tasks involved in the process, possibly indicating a partial order between some of these tasks. The approach proceeds in three steps: (i) generation of an abstract graph (simplified version of the process) from the task definitions; (ii) computation of the minimum and maximum time for executing the whole graph, followed by a manual refinement of the graph based on this information; (iii) synthesis of the final BPMN process from that graph. We implemented this approach in the WEASY lightweight modelling tool, which is available as an open-source Web application and led to a publication in an international conference [12].

Debugging of BPMN Processes. Designing business processes using BPMN can be error-prone. Recent works have proposed verification techniques for analyzing processes and for detecting possible issues. In particular, model checking is an established technique for automatically verifying that a model (e.g., a BPMN process) satisfies a given temporal property. When the model does not satisfy a safety property, the model checker returns as counterexample a sequence of actions leading to a state where the property does not hold. Understanding this counterexample for debugging the BPMN process is not an easy task, especially if the counterexample is not expressed using the BPMN notation. To simplify the debugging, we propose an approach for transforming counterexamples back to the original BPMN process by means of coloring techniques. The approach, which involves three steps (matching analysis, unfolding, and folding), was fully automated using several tools in conjunction with CADP, and was validated on many examples. This work led to a publication in an international conference [22].

Optimization of BPMN processes. Business process optimization is a strategic activity in organizations because of its potential to increase profit margins and reduce operational costs. One of the main challenges in this activity is concerned with the problem of optimizing allocation and sharing of resources. In collaboration with Francisco Durán (University of Málaga, Spain) and Camilo Rocha (Universidad Pontificia Javeriana, Cali, Colombia), we proposed two approaches in terms of moving from static to dynamic analysis and allocation of resources for BPMN processes. We considered the BPMN notation extended with an explicit description of execution time and resources associated with tasks, that can be concurrently executed multiple times. First, we proposed a simulation-based approach for computing certain metrics of interest, such as average execution time or resource usage. This approach applies off-line and is static in the sense that the number of resources does not evolve during simulation. Secondly, we proposed an alternative approach, which works online, thus requiring the instrumentation of an existing platform to retrieve relevant information during the execution of processes. This second approach is dynamic, because the number of resource replicas is updated during execution. We studied both approaches, stressing their advantages and drawbacks, and showing their complementarity. This work led to a publication in an international workshop [14].

We also proposed another kind of optimization, which operates on the structure of a BPMN process involving time and resources. This optimization consists of a refactoring procedure whose final goal is to reduce the total execution time of the BPMN process given as input. This procedure relies on refactoring operations that reorganize the tasks in the process by taking into account the resources used by those tasks. This process refactoring technique was fully automated by the ROP tool that we implemented and applied on several examples for validation purposes. This work led to a publication in an international conference [15].

6.1.4 Other language developments

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

In 2022, various compilers of CADP have been improved as follows:

- In addition to a bug fix in the X_BIT library, the CAESAR and CAESAR.ADT compilers for LOTOS have been enhanced to generate C code with less warnings from GCC and from the Sparse and Splint code checkers.
- All the CADP tools are now built using the latest version of the SYNTAX compiler generator and the most recent version of TRAIAN (see § 6.1.2). Also, the garbage collector invoked by CAESAR was upgraded from version 7.6.4 to version 8.2.2.

6.2 Parallel and Distributed Verification

6.2.1 Counting Bugs in Behavioural Models 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 (e.g., a Labelled Transition System obtained from a higher-level specification) satisfies a given temporal property. When the model violates the property, the model checker returns a counterexample, which is (for safety properties) 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 2022, we proposed some techniques for simplifying the debugging of these models. These techniques first extract the part of the behavioural model that does not satisfy the given property. In that model, we then 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. Based on this model, we proposed some techniques to count the number of bugs in the original specification. The core idea of the approach is to change the specification for some specific actions that may cause the property violation, and compare the model before and after modification to detect whether this potential bug is real or not. This approach is fully automated by a tool we implemented and applied on several examples for validation purposes. This work led to a publication in an international conference [19].

6.2.2 Verification of Emergent Properties in Multi-agent Systems

Participants: Frédéric Lang, Luca Di Stefano.

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 2022, we adapted the existing translation procedure from LAbS specifications [47] to LNT programs, so that agents are now represented by networks of LNT processes combined using parallel composition operators, whereas the previous procedure emulated the concurrent execution of agents using only sequential operators (basically sequential composition and choice). This allowed us to apply compositional verification to verify properties on the generated LNT programs. We combined compositional verification with a static value analysis to prune the state space of individual agents and demonstrated the effectiveness of our approach by verifying a collection of representative systems. These results led to a paper, which has been accepted for publication in an international conference [13].

6.3 Timed, Probabilistic, and Stochastic Extensions

6.3.1 Probabilistic Analysis of Industrial IoT Applications

Participants: Irman Faqrizal, Gwen Salaün.

Industrial automation is a complex process involving various stakeholders. The international standard IEC 61499 helps to specify distributed automation using a generic architectural model, targeting the technical development of the automation. However, analysing the correctness of IEC 61499 models remains challenging because of their informal semantics and distributed logic.

In 2022, in the context of the DIIoT project (see § 8.5.2), in collaboration with Yliès Falcone (CORSE project-team), we introduced new verification techniques for IEC 61499 applications that combine a design-time analysis for computing a formal model of the application with a runtime analysis for extracting additional information during the execution. Firstly, at design time, all possible executions of the application are captured using a Labelled Transition System (LTS), obtained by translating the IEC 61499 application into an LNT specification and applying the LNT compilers of CADP. Next, the application is instrumented by inserting a monitor function block, which allows executing a monitored version of the verified system with the existing IEC 61499 runtime environments. Then, the LTS model is enriched with probabilistic values computed from the execution traces collected by the monitor. Finally, the probabilistic version of the EVALUATOR model checker is used to verify certain requirements of the application, described as probabilistic temporal logic properties. The approach is automated using several tools and was validated on a realistic IEC 61499 application. This work led to a publication in an international conference [18].

6.3.2 Probabilistic Model Checking of BPMN Processes

Participants: Gwen Salaün, Ahang Zuo.

Business Process Model and Notation (BPMN) is a standard business process modelling language that allows users to describe a set of structured tasks, which results in a service or product. Before running a BPMN process, the user often has no clear idea of the probability of executing some task or specific combination of tasks. This is, however, of prime importance for adjusting resources associated with tasks and thus optimizing costs.

In 2022, in the context of the MOAP project (see § 8.5.1), in collaboration with Yliès Falcone (CORSE project-team), we defined an approach to perform probabilistic model checking of BPMN models at runtime. Our approach consists of three steps: (i) transformation of the BPMN model into a Labelled Transition System (LTS) by translating the BPMN model into an LNT specification and applying the LNT compilers of CADP; (ii) conversion of the LTS into a Probabilistic Transition System (PTS) by running multiple instances of the process and analyzing the execution traces; (iii) probabilistic model checking for verifying that the PTS model satisfies a given probabilistic property. The steps (ii) and (iii) are applied periodically to update the results according to the execution of the process instances. The overall approach was implemented in a tool chain, which was applied successfully to several realistic BPMN processes. This work led to a publication in an international conference [18].

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

6.4.1 Compositional Verification

Participants: Frédéric Lang, Luca Di Stefano.

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 2022, we published a research report [25], where:

- We describe the signature-based algorithm implemented in BCG_MIN and BCG_CMP to compute equivalence with respect to sharp bisimulation.
- We show that, under some conditions on strong actions, sharp bisimulation equivalence is a congruence for the priority operator implemented in EXP.OPEN. This makes sharp bisimulation appropriate to verify priority systems compositionally.
- We compare sharp bisimulation with orthogonal bisimulation [28], whose equivalence is also a congruence for priority. We show that compositional minimization using sharp bisimulation may yield LTSs that are several orders of magnitude smaller than using orthogonal bisimulation.

These results have also been submitted to an international journal.

6.4.2 On-the-fly Resolution of Boolean Equation Systems

Participants: Radu Mateescu.

OPEN/CAESAR is an extensible, modular, language-independent software framework for exploring implicit graphs (i.e., defined by their *post* function, which enumerates the transitions going out of each vertex). This key component of CADP is used to build simulation, execution, verification, and test generation tools.

CAESAR_SOLVE_1 is a generic software library, based on OPEN/CAESAR, for solving Boolean equation systems of alternation depth 1 (i.e., without mutual recursion between minimal and maximal fixed point equations) on the fly. This library is at the core of several CADP verification tools, namely the equivalence checker BISIMULATOR, the minimization tool REDUCTOR, and the model checker EVALUATOR. The resolution method is based on boolean graphs, which provide an intuitive representation of dependencies between boolean variables, and which are handled implicitly, in a way similar to the OPEN/CAESAR interface [35].

In 2022, we enhanced the CAESAR_SOLVE_1 library with two new on-the-fly resolution algorithms, named A10 and A11. These algorithms are based on a nested depth-first search (DFS) of the dependency graph between boolean variables, similar to the algorithms used for detecting accepting cycles in Büchi automata [31]. Algorithm A10 (resp. A11) is specialized for solving disjunctive (resp. conjunctive) equation blocks of minimal (resp. maximal) fixed point sign, in the case where a single invocation of the algorithm is requested on the block. Algorithms A10 and A11 consume less memory then their counterparts A3 and A4, which are based on a DFS traversal with computation of strongly connected components and allow multiple invocations on the same equation block. The CAESAR_SOLVE_1 manual page, the OPEN/CAESAR Reference Manual, and the BES_SOLVE tool were updated to take into account the two new algorithms.

6.4.3 Runtime Enforcement for IEC 61499 Applications

Participants: Irman Faqrizal, Gwen Salaün.

The ever-increasing demands from stakeholders in the industry for frameworks that can help maximizing the efficiency and productivity encourage innovations in industrial automation. As a result, the international standard IEC 61499 was proposed by researchers in this domain. The standard has been gaining popularity in the past few years particularly because of its distributed paradigm and vendor independent format. The technicality of IEC 61499, such as its syntax, is well-defined; however, it comes only with informal semantics described using natural language. In this regard, the verification is not straightforward, while it is known that the correctness of high-stakes industrial systems is crucial.

In 2022, in the context of the DIIoT project (see § 8.5.2), in collaboration with Yliès Falcone (CORSE project-team), we proposed new verification techniques for IEC 61499 applications. These techniques rely on the concept of runtime enforcement, which can be applied to systems for preventing bad behaviours from happening. The main idea of our approach is to integrate an enforcer in the application for allowing it to respect specific properties when executing. The techniques begin with the definition of a property. The language of this property supports features such as discarding and replacing events. Next, this property is used to synthesize an enforcer in the form of a function block. Finally, the synthesized enforcer is integrated into the application. Our approach was validated on a realistic example and fully automated. This work led to a publication in an international conference [17].

6.4.4 Other Component Developments

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

In 2022, several components of CADP have been improved as follows:

- Besides a bug fix in the BCG_READ manual page, the vocabulary and option names concerning divergence-preserving branching bisimulation and sharp bisimulation in BCG_CMP, BCG_MIN, and SVL were clarified.

- The CAESAR.BDD tool was upgraded to version 3.1.0 of the CUDD library, and extended with four new options: "-arcs-pt", "-arcs-tp", "-signature", and "-signature-multiple" (the two latter options display a NUPN checksum that is invariant by any permutation of places, transitions, and/or units). The existing options "-place-order" and "-unit-order" were enhanced to be faster and produce shorter, yet more discriminative results, and the "-concurrent-places" option was made faster.

- The ergonomy of CADP was enhanced by pre-selecting the MAIN process of LNT files in the EUCALYPTUS graphical user interface (thereby saving user clicks). Also, the style files for GNOME text editors (Gedit, Pluma, etc.) were upgraded to Gtk5, and new style files for the Sublime Text editor have been added for the various languages and file formats supported by CADP.

- In addition to six bug fixes, the CADP tools received specific enhancements targeted to various processors and operating systems. CADP was ported to macOS 13 "Ventura", and was adapted to support Apple machines with an ARM processor. Support for the 32-bit executables on Solaris and OpenIndiana has been discontinued, since the 64-bit executables are deemed to be sufficient. The installation and usage of CADP on Windows/Cygwin were simplified, and preliminary changes to support 64-bit Windows executables have been undertaken.

6.5 Real-Life Applications and Case Studies

6.5.1 Autonomous Car

Participants: Jean-Baptiste Horel, Radu Mateescu *(correspondent)*, Philippe Ledent, Lucie Muller, Wendelin Serwe.

A common practice to evaluate autonomous vehicles is simulation, which requires to specify realistic scenarios, in particular critical ones, occurring rarely and potentially dangerous to reproduce on the road. Such scenarios may be either generated randomly, or specified manually. Randomly generating scenarios is easy, but their relevance might be difficult to assess. Manually specified scenarios can focus on a given feature, but their design might be difficult and time-consuming, especially to achieve satisfactory coverage.

In the framework of the ArchitectECA2030 (see § 8.3.1) and PRISSMA (see § 8.4.1) projects and in collaboration with Lina Marsso (University of Toronto, Canada), Christian Laugier, Anshul Paigwar, and Alessandro Renzaglia (CHROMA project-team), we proposed an automatic approach to generate a large number of relevant critical scenarios for autonomous driving simulators. The approach relies on the generation of behavioral conformance tests using the TESTOR tool [43], from an LNT model (specifying the ground truth configuration with the range of vehicle behaviors) and a test purpose (specifying the critical feature under analysis). The obtained test cases (which cover all possible executions exercising a given feature) are automatically translated into the inputs of autonomous driving simulators. Using this approach, we generated thousands of behavior trees for the CARLA simulator for several realistic configurations. This work led to a publication in an international conference [20].

In 2022, we finalized the LNT model describing the scene configuration and the behaviour of the actors (autonomous car and mobile obstacles), and compared it with a previous model focused on the decision component of the car. The two models were made publicly available and led to a publication in an international workshop [21]. We also extended our scenario generation approach by executing the scenarios in the CARLA simulator to produce execution traces, which we analyzed using a combination of model checking and statistical analysis to assess the collision-risk estimation of the perception component equipping the autonomous car. This work led to an article accepted for publication in an international journal.

In the context of the ArchitectECA2030 project, we considered a thermal controller for a high-voltage car battery. Based on an informal specification using four mode automata and a list of requirements, both provided by AVL, we developed a formal model in LNT (about 1000 lines) and expressed the relevant requirements as 26 MCL formulas. This formalization discovered several small issues, such as inconsistent naming of variables, mix between temperature units (Kelvin and Celsius), and nondeterministic behavior. After ensuring that the LNT model satisfies all requirements, we started investigating the use of the model for runtime monitoring of the thermal controller, by taking real simulation traces provided by AVL and checking whether an appropriate abstraction of these traces (rounding the floating-point temperature values to the discrete intervals considered in the model) is included in the model, enabling to distinguish between correct and faulty traces.

Still within the ArchitectECA2030 project, we also considered a combined hardware/software aging monitor, provided by NXP (hardware part) and DATASOFT (software part). We started the development of a formal LNT model, based on the initial descriptions provided by our partners, illustrating the feasibility of the modelling and validation approach.

7 Bilateral contracts and grants with industry

7.1 Start-up Project

Participants: Gwen Salaün *(correspondent)*, Ajay Muroor-Nadumane.

G. Salaün and A. Muroor-Nadumane initiated the Voyance start-up, which became part of Inria Startup Studio in February 2022. Voyance provides integrated analytics enabling enterprises to improve efficiency and reduce greenhouse gas emissions.

Voyance leverages model-based techniques to analyse different types of processes, ranging from operations workflows to complex processes in retail, manufacturing, logistics, and energy sectors. The analyses enable technical analysts and business leaders to optimize their workflows in terms of resource utilization, greenhouse gas emissions, cost, throughput, and other key performance indicators.

7.2 Bilateral grants with industry

7.2.1 Nokia Bell Labs

Participants: Gwen Salaün *(correspondent)*.

The collaboration with Nokia Bell Labs (Nozay) in the framework of the ADR Sapiens, started in 2017, ended in 2022. The CONVECS project-team was involved in the formal modeling and analysis of IoT applications. The latest results of the collaboration are presented in two publications in international journals [10, 11].

7.2.2 ST Microelectronics

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

Ph. 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 Hajer Ferjani (ST Microelectronics), Radu Mateescu (CONVECS), and Wendelin Serwe (CONVECS).

8 Partnerships and cooperations

8.1 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.

8.1.1 Other international collaborations

In 2022, we had scientific relations with several universities and institutes abroad, including:

- University of Málaga, Spain (Francisco Durán),
- Saarland University, Germany (Holger Hermanns),
- University of Cali, Colombia (Camilo Rocha),

- University of Toronto, Canada (Lina Marsso),
- Chalmers University, Sweden (Luca Di Stefano).

8.2 International research visitors

8.2.1 Visits of international scientists

- Francisco Durán (University of Málaga, Spain) visited us from May 4 to May 6, 2022. He participated at the workshop on formal modelling, analysis, and optimization of BPMN processes organized by G. Salaün at Inria Grenoble on May 4, 2022. He gave a seminar entitled "*Machine Learning for the Dynamic Provisioning of Resources*".

8.2.2 Visits to international teams

Research stays abroad

- G. Salaün visited the University of Málaga (Spain) from February 13 to March 9, 2022.

8.3 European initiatives

8.3.1 H2020 projects

ArchitectECA2030

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

ArchitectECA2030 project on cordis.europa.eu

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

Duration: From July 1, 2020 to June 30, 2023

Partners:

- UAB TERAGLOBUS, Lithuania
- INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (INRIA), France
- TECHNISCHE UNIVERSITAET GRAZ (TU GRAZ), Austria
- INFINEON TECHNOLOGIES AUSTRIA AG (IFAT), Austria
- BOARD OF REGENTS OF NEVADA SYSTEM OF HIGHER EDUCATION, United States
- SafeTRANS e.V. (SafeTRANS), Germany
- TRACSENSE AS, Norway
- NXTECH AS, Norway
- INSTITUT MIKROELEKTRONICKYCH APLIKACI SRO (IMA), Czechia
- INFINEON TECHNOLOGIES AG (IFAG), Germany
- VYSOKE UCENI TECHNICKE V BRNE (BRNO UNIVERSITY OF TECHNOLOGY), Czechia
- SINTEF AS (SINTEF), Norway
- SMARTSOL SIA, Latvia
- VIRTUAL VEHICLE RESEARCH GMBH (VIF), Austria
- NXP SEMICONDUCTORS NETHERLANDS BV, Netherlands

- AVL LIST GMBH (AVL), Austria
- DATASOFT EMBEDDED GMBH (DATASOFT EMBEDDED), Austria
- SBA RESEARCH GEMEINNUTZIGE GMBH (SBA), Austria
- VOLKSWAGEN AKTIENGESELLSCHAFT (VW AG), Germany
- TECHNISCHE UNIVERSITEIT DELFT (TU Delft), Netherlands
- TECHNISCHE UNIVERSITAET DRESDEN (TUD), Germany
- AVL MTC MOTORTESTCENTER AB (AVL MTCAB), Sweden

Website: https://autoc3rt.automotive.oth-aw.de/

Inria contact: Radu Mateescu

Coordinator: Georg Stettinger (IFAG)

Summary: Independent validation is fundamental to emphasise the capability and safety of any solution in the electric, connected and automated (ECA) vehicles space. It is vital that appropriate and audited testing takes place in a controlled environment before any deployment takes place. As the software and hardware components come from multiple vendors and integrate in numerous ways, the various levels of validation required must be fully understood and integration with primary and secondary parts must be considered.

The key targets of ArchitectECA2030 are the robust mission-validated traceable design of electronic components and systems (ECS), the quantification of an accepted residual risk of ECS for ECA vehicles to enable type approval, and an increased end-user acceptance due to more reliable and robust ECS. The proposed methods include automatic built-in safety measures in the electronic circuit design, accelerated testing, residual risk quantification, virtual validation, and multi-physical and stochastic simulations.

The project will implement a unique in-vehicle monitoring device able to measure the health status and degradation of the functional electronics empowering model-based safety prediction, fault diagnosis, and anomaly detection. A validation framework comprised of harmonized methods and tools able to handle quantification of residual risks using data different sources (e.g. monitoring devices, sensor/actuators, fleet observations) is provided to ultimately design safe, secure, and reliable ECA vehicles with a well-defined, quantified, and acceptable residual risk across all ECS levels. The project brings together stakeholders from ECS industry, standardization and certification bodies (e.g. ISO, NIST, TUEV), test field operators, insurance companies, and academia closely interacting with ECSEL lighthouse initiative Mobility.E to align and influence emerging standards and validation procedures for ECA vehicles.

The main contributions of CONVECS in the project are the formal modeling and validation of components embedded in autonomous vehicles.

8.3.2 Other european programs/initiatives

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

8.4 National initiatives

8.4.1 Fonds pour l'innovation et l'industrie

PRISSMA

Participants: Jean-Baptiste Horel, Radu Mateescu *(correspondent)*.

PRISSMA is a project funded by the *Fonds pour l'innovation et l'industrie* within the *Grand défi 2 : sécuriser, certifier et fiabiliser les systèmes fondés sur l'intelligence artificielle* programme. The project involves 19 industrial partners (among which ANSYS, RATP, and VALEO), as well as Université Gustave-Eiffel, LNE, and Inria (project-teams CHROMA and CONVECS). PRISSMA aims at proposing a platform enabling to release the technological locks that hamper the deployment of secure IA-based systems and to integrate all the necessary elements for the homologation activities of autonomous vehicles and their validation in real environments given by use cases.

PRISSMA started in April 2021 for three years. The main contributions of CONVECS to PRISSMA are the formal modeling and validation of perception components of the autonomous vehicle.

8.4.2 Other national collaborations

We had sustained scientific relations with the following researchers:

- Fabrice Kordon and Lom Messan Hillah (LIP6, Paris),
- Nicolas Amat and Silvano Dal Zilio (LAAS-CNRS, Toulouse),
- Michel Le Pallec (Nokia Bell Labs).

8.5 Regional initiatives

8.5.1 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 five years. The main contributions of CONVECS to MOAP are the formal modeling and automated verification of BPMN processes.

8.5.2 Persyval labex

D-IIoT

Participants: Irman Faqrizal, Gwen Salaün *(correspondent)*.

D-IIoT is a project funded by the Persyval Labex via the ANR ("*Agence Nationale de la Recherche*"). The project involves research teams of three local laboratories (CEA, LIG, VERIMAG). D-IIoT aims at studying and proposing new techniques to support the execution of long-running and evolving IIoT (Industrial IoT) applications with dependability guarantees (e.g., security, correctness).

D-IIoT started in October 2021 for three years. The main contributions of CONVECS to D-IIoT are the formal modeling, verification, and reconfiguration of IIoT applications.

9 Dissemination

9.1 Promoting scientific activities

9.1.1 Scientific events: organisation

General chair, scientific chair

- Together with Peter Höfner (Data61, CSIRO, Sydney, Australia), H. Garavel set up a model repository to collect and archive formal models of real systems; this infrastructure is used by the series of MARS workshops. This repository currently contains 32 models, among which 10 were deposited by CONVECS.
- P. Bouvier and H. Garavel are members of the model board of MCC (*Model Checking Contest*).
- H. Garavel is a member of the steering committee of the MARS (*Models for Formal Analysis of Real Systems*) workshop series since 2015.
- H. Garavel is a member of the steering committee of TTC (*Transformation Tool Contest*) since 2021.
- H. Garavel and R. Mateescu are members of the steering committee of the FMICS (*Formal Methods for Industrial Critical Systems*) conference series since 2018.
- G. Salaün is member of the steering committee of the ACM SAC-SVT (*Symposium of Applied Computing – Software Verification and Testing track*) conference series since 2018.
- G. Salaün is member of the steering committee of the SEFM (*International Conference on Software Engineering and Formal Methods*) conference series since 2014.
- G. Salaün is member of the steering committee of the FOCLASA (*International Workshop on Foundations of Coordination Languages and Self-Adaptative Systems*) workshop series since 2011.

9.1.2 Scientific events: selection

Member of the conference program committees

- H. Garavel was a programme committee member of MARS'2022 (*5th International Workshop on Models for Formal Analysis of Real Systems*), Munich, Germany, April 2-3, 2022.
- H. Garavel was a programme committee member of FORTE'2022 (*42nd International Conference on Formal Techniques for Distributed Objects, Components, and Systems*), Lucca, Italy, June 13-17, 2022.
- H. Garavel was a programme committee member of FMICS'2022 (*27th International Conference on Formal Methods for Industrial Critical Systems*), Warsaw, Poland, September 14-16, 2022.
- G. Salaün was a programme committee member of ENASE'2022 (*17th International Conference on Evaluation of Novel Approaches to Software Engineering*), virtual event, April 25-26, 2022.
- G. Salaün was a programme committee member of SEAMS'2022 (*17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*), virtual event, May 18-20, 2022.
- G. Salaün was a programme committee member of FormaliSE'2022 (*10th International Conference on Formal Methods in Software Engineering*), Pittsburgh, PA, USA, May 22-23, 2022.
- G. Salaün was a programme committee member of COMPSAC-SETA'2022 (*IEEE International Conference on Computers, Software, and Applications - Software Engineering Technologies and Applications*), virtual event, June 27-July 1st, 2022.
- G. Salaün was a programme committee member of ICSOFT'2022 (*17th International Conference on Software Technologies*), Lisbon, Portugal, July 11-13, 2022.
- G. Salaün was a programme committee member of SEFM'2022 (*20th International Conference on Software Engineering and Formal Methods*), Berlin, Germany, September 26-30, 2022.
- G. Salaün was a programme committee member of FACS'2022 (*18th International Conference on Formal Aspects of Component Software*), Oslo, Norway, November 10-11, 2022.

Reviewer

- I. Faqrizal was a reviewer for COMPSAC’2022, FormaliSE’2022, ICSOFT’2022, SEFM’2022, FACS’2022, and FASE’2023 (*25th International Conference on Fundamental Approaches to Software Engineering*).
- F. Lang was a reviewer for FORTE’22 and SEFM’22.
- W. Serwe was a reviewer for COMPSAC’2022 and DATE’2022 (*Design, Automation, and Test in Europe*).

9.1.3 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

- R. Mateescu was a reviewer for STTT and TSE (*IEEE Transactions on Software Engineering*).
- G. Salaün was a reviewer for JLAMP (*Journal of Logic and Algebraic Methods in Programming*), JSS (*Journal of Systems and Software*), and TOSEM (*ACM Transactions on Software Engineering and Methodology*).
- W. Serwe was a reviewer for COSE (*Computers and Security*), STTT, and TECS (*ACM Transactions on Embedded Computing Systems*).

9.1.4 Software dissemination and Internet visibility

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

In 2022, the main facts are the following:

- We prepared and distributed twelve successive versions (2022-a to 2022-l) of CADP.
- We granted CADP licenses for 208 different computers in the world.

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

By the end of December 2022, the CADP forum, opened in 2007 for discussions regarding the CADP toolbox, had over 470 registered users and over 1972 messages had been exchanged.

Also, for the 2022 edition of the Model Checking Contest, we provided 3 families of models (totalling 60 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:

- Kong verification tool for Petri nets [27]
- Tomte tool for learning automata [48]
- Alvis verification tool for concurrent systems [49]

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

- 4SECURail case study on train handover protocols [46, 45]
- Integration of ubiquitous specifications in object systems design [26]

9.1.5 Invited talks

- I. Faqrizal gave a talk entitled "*Runtime Enforcement for IEC 61499 Applications*" at the 3rd meeting of the YODA working group of the GDR GPL, held virtually on December 16, 2022.
- Q. Nivon gave a talk entitled "*Debugging of BPMN Processes using Coloring Technique*" at the 3rd YODA meeting, held virtually on December 16, 2022.
- G. Salaün gave a keynote talk entitled "*Modelling and Quantitative Analysis of BPMN Processes using Maude*" at the 14th International Workshop on Rewriting Logic and its Applications (WRLA'2022), Munich, Germany on April 2-3, 2022.
- G. Salaün gave an invited talk entitled "*Modelling, Analysis and Optimization of BPMN Processes*" at the *Journées du GDR GPL*, Vannes on June 7-10, 2022.
- G. Salaün gave an invited talk entitled "*Modelling, Analysis and Optimization of BPMN Processes*" at the 17th International Summer School on Training And Research On Testing (TAROT'2022), Avila, Spain on July 5-8, 2022.
- G. Salaün gave an invited talk entitled "*Models and Verification for Composition and Reconfiguration of Web of Things Applications*" at the 1st French-German Workshop SeReCo, Lyon on May 23-24, 2022.

9.1.6 Research administration

- F. Lang is chair of the "*Commission du développement technologique*", which is in charge of selecting R&D projects for Inria Grenoble, and giving an advice on the recruitment of temporary engineers.
- R. Mateescu is the scientific correspondent of the International Partnerships for Inria Grenoble.
- R. Mateescu is a member of the "*Comité d'Orientation Scientifique*" for Inria Grenoble. In 2022, he also participated to the recruitment jury for CRCN (*Chargé de Recherche de Classe Normale*) and ISFP (Inria Starting Faculty Positions) at Inria Grenoble.
- R. Mateescu is representative of Inria Grenoble at the International Relations and Outreach of Université Grenoble Alpes (UGA).
- R. Mateescu is member of the council of the Mathematics, Information and Communication Sciences (MSTIC) research department of UGA.
- G. Salaün is a member of the administration council of the IUT1/UGA.
- G. Salaün is the head of the *Métiers du Multimédia et de l'Internet* (MMI) department at IUT1/UGA since September 2022.
- G. Salaün is a member of the Scientific Committee of the PCS (*Pervasive Computing Systems*) action of the PERSYVAL Labex.
- G. Salaün is a member of the council of the LIG laboratory.
- W. Serwe is correspondent in charge of the 2022 Inria activity reports at Inria Grenoble.
- W. Serwe is a member of the "*Comité de Centre*" at Inria Grenoble.

9.2 Teaching - Supervision - Juries

9.2.1 Teaching

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

In 2022, we carried out the following teaching activities:

- I. Faqrizal gave a course on “*Introduction to Node.js*” (32 hours “*équivalent TD*”) to L3 students of IUT1/UGA.
- F. Gallay taught a course on “*Langages et automates*” (27 hours “*équivalent TD*”) to L2 (Licence MIN) students of UGA.
- H. Garavel participated to a course on “*Embedded Systems: From High-Confidence Design to Safe Execution*” (12 hours “*équivalent TD*” on probabilistic/stochastic models and fault trees) to second year students of the MOSIG.
- F. Lang gave a course on “*Formal Software Development Methods*” (7.5 hours “*équivalent TD*”) in the framework of the “*Software Engineering*” lecture given to 1st 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 3rd year students of ENSIMAG and second year students of the MOSIG.
- L. Muller gave a course on “*Système et environnement de programmation*” (32 hours “*équivalent TD*”) to L1 students of UGA.
- G. Salaün taught about 240 hours of classes (algorithmics, Web development, object-oriented programming) at the MMI department of IUT1/UGA. He is also headmaster of the “*Services Mobiles et Interface Nomade*” (SMIN) professional licence (third year of university) at IUT1/UGA, and co-headmaster of the “*alternance*” (apprenticeship) at the MMI department.
- 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 2022.
- A. Zuo gave a course on “*Introduction to Java/Android programming*” (30 hours “*équivalent TD*”) to L2 students of IUT1/UGA during the spring 2022.
- A. Zuo gave a course on “*Data Extraction*” (21 hours “*équivalent TD*”) to L3 (Licence Pro) students of IUT2/UGA during the winter 2022.

9.2.2 Supervision

- 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: I. Faqrizal, “*Monitoring and Deployment of IIoT Applications*”, Université Grenoble Alpes, since October 2021, G. Salaün and Yliès Falcone
- PhD in progress: F. Gallay, “*Decentralized Runtime Enforcement of Timed Properties*”, Université Grenoble Alpes, since October 2022, R. Mateescu and Y. Falcone
- PhD in progress: J-B. Horel, “*Validation des composants de perception basés sur l'IA dans les véhicules autonomes*”, Université Grenoble Alpes, since April 2021, R. Mateescu, Alessandro Renzaglia, and Christian Laugier
- 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 Hajer Ferjani
- 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: Q. Nivon, “*Analyse, optimisation et debugging de processus BPMN*”, Université Grenoble Alpes, since October 2022, G. Salaün

- PhD in progress: Choukri Soueidi, "*Instrumentation expressive et correcte de programmes distribués et vérification à l'exécution*", Université Grenoble Alpes, since October 2020, G. Salaün and Y. Falcone
- PhD in progress: A. Zuo, "*Modelling, Optimization and Predictive Analysis of Business Processes*", Université Grenoble Alpes, since October 2020, G. Salaün and Y. Falcone

9.2.3 Juries

- R. Mateescu was reviewer of Soren Enevoldsen's PhD thesis, entitled "*Abstract Dependency Graphs for Model Verification*", defended at University of Aalborg (Denmark) on December 15, 2022.
- G. Salaün was reviewer of Abdul Majith Noordheen's PhD thesis, entitled "*Automated Verification and Synthesis of Distributed Systems*", defended at Université de Rennes on June 7, 2022.

10 Scientific production

10.1 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.

10.2 Publications of the year

International journals

[9] L. Di Stefano, R. de Nicola and O. Inverso. 'Verification of Distributed Systems via Sequential Emulation'. In: *ACM Transactions on Software Engineering and Methodology* 31.3 (2022), pp. 1–41. DOI: 10.1145/3490387. URL: https://hal.inria.fr/hal-03549925.

[10] F. Durán, A. Krishna, M. Le Pallec, R. Mateescu and G. Salaün. 'Models and analysis for user-driven reconfiguration of rule-based IoT applications'. In: *Internet of Things* 19 (Aug. 2022), p. 100515. DOI: 10.1016/j.iot.2022.100515. URL: https://hal.inria.fr/hal-03781473.

[11] A. Krishna, M. Le Pallec, R. Mateescu and G. Salaün. 'Design and Deployment of Expressive and Correct Web of Things Applications'. In: *ACM Transactions on Internet of Things* 3 (28th Feb. 2022), pp. 1–30. DOI: 10.1145/3475964. URL: https://hal.inria.fr/hal-03495593.

International peer-reviewed conferences

[12] A. Contreras, Y. Falcone, G. Salaün and A. Zuo. 'WEASY: A Tool for Modelling Optimised BPMN Processes'. In: FACS 2022 - 18th International Conference on Formal Aspects of Component Software. Oslo / Online, Norway, 10th Nov. 2022. DOI: 10.1007/978-3-031-20872-0_7. URL: https://hal.inria.fr/hal-03848350.

[13] L. Di Stefano and F. Lang. 'Compositional Verification of Stigmergic Collective Systems'. In: 24th International Conference on Verification, Model Checking , and Abstract Interpretation (VMCAI'2023). Boston, United States, 16th Jan. 2023. URL: https://hal.inria.fr/hal-03869922.

[14] F. Durán, Y. Falcone, C. Rocha, G. Salaün and A. Zuo. 'From Static to Dynamic Analysis and Allocation of Resources for BPMN Processes'. In: WRLA 2022 - 14th International Workshop on Rewriting Logic and its Applications. Munich, Germany, 2nd Apr. 2022, pp. 1–18. DOI: 10.1007/978-3-031-12441-9_1. URL: https://hal.inria.fr/hal-03766148.

[15] F. Durán and G. Salaün. 'Optimization of BPMN Processes via Automated Refactoring'. In: ICSOC 2022 - 20th International Conference on Service-Oriented Computing. Sevilla, Spain, 29th Nov. 2022, pp. 1–15. DOI: 10.1007/978-3-031-20984-0_1. URL: https://hal.inria.fr/hal-03883829.

[16] Y. Falcone, I. Faqrizal and G. Salaün. 'Probabilistic Analysis of Industrial IoT Applications'. In: IoT 2022 -The 12th International Conference on the Internet of Things. Delft, Netherlands, 7th Nov. 2022. URL: https://hal.inria.fr/hal-03848674.

[17] Y. Falcone, I. Faqrizal and G. Salaün. 'Runtime Enforcement for IEC 61499 Applications'. In: SEFM 2022 - 20th International Conference on Software Engineering and Formal Methods. Berlin, Germany, 28th Sept. 2022, pp. 1–17. DOI: 10.1007/978-3-031-17108-6_22. URL: https://hal.inria.fr/hal-03766095.

[18] Y. Falcone, G. Salaün and A. Zuo. 'Probabilistic Model Checking of BPMN Processes at Runtime'. In: iFM 2022 - International Conference on integrated Formal Methods. Lugano, Switzerland, 7th June 2022, pp. 1–17. DOI: 10.1007/978-3-031-07727-2_11. URL: https://hal.inria.fr/hal-03665305.

[19] I. Faqrizal and G. Salaün. 'Counting Bugs in Behavioural Models using Counterexample Analysis'. In: FormaliSE 2022 - International Conference on Formal Methods in Software Engineering. Pittsburgh, United States, 18th May 2022, pp. 1–11. DOI: 10.1145/3524482.3527647. URL: https://hal.inria.fr/hal-03665317.

[20] J.-B. Horel, C. Laugier, L. Marsso, R. Mateescu, L. Muller, A. Paigwar, A. Renzaglia and W. Serwe. 'Using Formal Conformance Testing to Generate Scenarios for Autonomous Vehicles'. In: DATE/ASD 2022 - Design, Automation and Test in Europe - Autonomous Systems Design. Antwerp, Belgium: IEEE, 14th Mar. 2022, pp. 532–537. DOI: 10.23919/DATE54114.2022.9774581. URL: https://hal.inria.fr/hal-03516799.

[21] L. Marsso, R. Mateescu, L. Muller and W. Serwe. 'Formally Modeling Autonomous Vehicles in LNT for Simulation and Testing'. In: Mars 2022 - 5th Workshop on Models for Formal Analysis of Real Systems. Vol. 355. Electronic Proceedings in Theoretical Computer Science. Munich, Germany, 18th Mar. 2022, pp. 60–117. DOI: 10.4204/EPTCS.355.5. URL: https://hal.inria.fr/hal-03623521.

[22] Q. Nivon and G. Salaün. 'Debugging of BPMN Processes Using Coloring Techniques'. In: *Lecture Notes in Computer Science*. FACS 2022 - 18th International Conference on Formal Aspects of Component Software. Vol. LNCS-13712. Formal Aspects of Component Software. Oslo / Virtual, Norway: Springer International Publishing, 2nd Nov. 2022, pp. 90–109. DOI: 10.1007/978-3-031-20872-0_6. URL: https://hal.inria.fr/hal-03847267.

[23] G. Salaün. 'Quantifying the Similarity of BPMN Processes'. In: APSEC 2022 - 29th Asia-Pacific Software Engineering Conference. Virtual, Japan, 6th Dec. 2022, pp. 1–10. URL: https://hal.inria.fr/hal-03890531.

Scientific book chapters

[24] H. Garavel and F. Lang. 'Equivalence Checking 40 Years After: A Review of Bisimulation Tools'. In: *A Journey from Process Algebra via Timed Automata to Model Learning*. Vol. 13560. Lecture Notes in Computer Science. Springer Nature Switzerland; Springer Nature Switzerland, 7th Sept. 2022, pp. 213–265. DOI: 10.1007/978-3-031-15629-8_13. URL: https://hal.inria.fr/hal-03920338.

Reports & preprints

[25] L. Di Stefano and F. Lang. *Compositional Verification of Priority Systems using Sharp Bisimulation*. INRIA, 13th Apr. 2022, pp. 1–32. URL: https://hal.inria.fr/hal-03640683.

10.3 Cited publications

[26] S. Aimene and I. Rassoul. 'Integration of ubiquitous specifications in the conception of objects system'. In: *Int. J. Comput. Appl. Technol.* 68.1 (2022), pp. 70–81. DOI: 10.1504/IJCAT.2022.123235.

[27] N. Amat and L. Chauvet. 'Kong: A Tool to Squash Concurrent Places'. In: *Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings*. Ed. by L. Bernardinello and L. Petrucci. Vol. 13288. Lecture Notes in Computer Science. Springer, 2022, pp. 115–126. DOI: 10.1007/978-3-031-06653-5_6.

[28] J. A. Bergstra, A. Ponse and M. van der Zwaag. 'Branching time and orthogonal bisimulation equivalence'. In: *Theor. Comput. Sci.* 309.1-3 (2003), pp. 313–355. DOI: 10.1016/S0304-3975(03)00277-9.

[29] 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.

[30] 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.

[31] E. M. Clarke, O. Grumberg and D. A. Peled. *Model Checking*. MIT Press, 2001.

[32] 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.

[33] Y. Falcone, G. Salaün and A. Zuo. 'Semi-automated Modelling of Optimized BPMN Processes'. In: *SCC 2021 - IEEE International Conference on Services Computing*. CHICAGO / Virtual, United States: IEEE, Sept. 2021, pp. 1–6. URL: https://hal.inria.fr/hal-03330330.

[34] H. Garavel. 'Compilation of LOTOS Abstract Data Types'. In: *Proceedings of the 2nd International Conference on Formal Description Techniques FORTE'89 (Vancouver B.C., Canada)*. Ed. by S. T. Vuong. North Holland, Dec. 1989, pp. 147–162.

[35] H. Garavel. 'OPEN/CÆSAR: An Open Software Architecture for Verification, Simulation, and Testing'. In: *Proceedings of the First International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'98 (Lisbon, Portugal)*. 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.

[36] 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.

[37] 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.

[38] 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.

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

[40] 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.

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

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

[43] L. Marsso, R. Mateescu and W. Serwe. 'TESTOR: A Modular Tool for On-the-Fly Conformance Test Case Generation'. In: *TACAS 2018 - 24th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Vol. 10806. Lecture Notes in Computer Science. Thessaloniki, Greece: Springer, Apr. 2018, pp. 211–228. DOI: 10.1007/978-3-319-89963-3_13. URL: https://hal.inria.fr/hal-01777861.

[44] 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.

[45] F. Mazzanti and D. Belli. 'Formal Modeling and Initial Analysis of the 4SECURail Case Study'. In: *Proceedings Fifth Workshop on Models for Formal Analysis of Real Systems, MARS@ETAPS 2022, Munich, Germany, 2nd April 2022*. Ed. by C. Dubslaff and B. Luttik. Vol. 355. EPTCS. 2022, pp. 118–144. DOI: 10.4204/EPTCS.355.6.

[46] F. Mazzanti and D. Belli. 'The 4SECURail Formal Methods Demonstrator'. In: *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification - 4th International Conference, RSSRail 2022, Paris, France, June 1-2, 2022, Proceedings*. Ed. by S. C. Dutilleul, A. E. Haxthausen and T. Lecomte. Vol. 13294. Lecture Notes in Computer Science. Springer, 2022, pp. 149–165. DOI: 10.1007/978-3-031-05814-1_11.

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

[48] G. Roose. 'An Experience Report on Model Learning'. Bachelor thesis. Amsterdam, The Natherlands: Vrije Universiteit Amsterdam, July 2022.

[49] M. Wypych. 'Methods of Generation of Transition Systems for Alvis Language'. PhD thesis. Poland: University of Science and Technology in Krakow, 2021.