RESEARCH CENTRE

Grenoble - Rhône-Alpes

IN PARTNERSHIP WITH:

Institut polytechnique de Grenoble, Université Joseph Fourier (Grenoble)

2021
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

B6.6. – 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 Fellows

- Luca Di Stefano [Inria]
- Ajay Muroor Nadumane [Inria, until Aug 2021]

PhD Students

- Pierre Bouvier [Univ Grenoble Alpes]
- Irman Faqrizal [Univ Grenoble Alpes, from Oct 2021]
- Jean-Baptiste Horel [Inria, from Apr 2021]
- Philippe Ledent [ST Microelectronics]
- Lucie Muller [Inria]
- Ahang Zuo [Univ Grenoble Alpes]

Interns and Apprentices

- Angel Contreras Duenas [Inria, from Feb 2021 until Jun 2021]
- Irman Faqrizal [Inria, from Feb 2021 until Jul 2021]
- Esau Martinez Lopez [Inria, from Feb 2021 until Jun 2021]
- Nathan Miscopein [Inria, from Jul 2021 until Aug 2021]

Administrative Assistant

- Myriam Etienne [Inria]

External Collaborator

- Ajay Muroor Nadumane [Free lance, from Sep 2021]

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*) [49], 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 [40], 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, autonomous vehicles, avionic systems, train supervision systems, medical devices,
- *Hardware architectures:* multiprocessor architectures, systems on chip, cache coherency protocols, hardware/software codesign,

- *Human-machine interaction:* graphical interfaces, biomedical data visualization, plasticity,
- *Security protocols:* authentication, electronic transactions, cryptographic key distribution,
- *Telecommunications:* high-speed networks, network management, mobile telephony, feature interaction detection.

5 Highlights of the year

5.1 Awards

H. Garavel, R. Mateescu, F. Lang, and W. Serwe received the Inria - Académie des Sciences - Dassault Systèmes Innovation Prize (2021).

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 [39] 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 [45, 44] 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 [40] 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 [46], CTL[34], ACTL[35], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [43] 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 [47] 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 [41] 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

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 [42], 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

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] [33] 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 2021, 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 § 7.1.2):

- The syntax of LNT was brought closer to classical programming languages. The priority rules for infix operators were changed to make LNT expressions more in line with mathematical conventions, reducing the need of parentheses wherever there should be no ambiguity, similarly to the Ada language. The predefined LNT operator for string concatenation (formerly noted "+") was renamed to "&", which is the notation used in LOTOS NT, but also in Ada, AppleScript, and VisualBasic. The syntax of the "case" instruction changed from "case ... in var ... in ... " to "case ... var ... in ... " to avoid the sequence of two consecutive keywords "in var". The pragmas related to the translation of LNT types and functions in the C language were improved to make clear that the target identifiers respect the rules of C.

- The LNT2LOTOS Reference Manual has been updated to document these changes of LNT, and the CADP demo examples have been upgraded to the latest version of LNT. Six 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.

- Also, in addition to seven bug fixes, the LNT2LOTOS compiler was enhanced in several manners. It emits more warnings about potentially erroneous use of the new priorities of infix operators and deprecated use of LNT constructs in a way that does not exist in LOTOS NT (e.g., channel definitions without field names, "any T" used in a pattern on the left-hand side of an infix operator and without parentheses, etc.). It now takes into account the existence of pre-conditions and post-conditions in functions and processes in its data-flow analysis to warn about unused parameters.

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. 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 2021, our efforts led to the release of three new major versions of TRAIAN:

- TRAIAN 3.3 brings five language changes to LOTOS NT in order to further align it with LNT, six changes in the dataflow analysis to prevent users from making common programming errors, and five code generation changes in order to improve the C code generated by LOTOS NT. Two bugs were fixed and one demo was updated to get it accepted by TRAIAN 3.3.

- TRAIAN 3.4 brings nine language changes to LOTOS NT, mostly to extend the syntax of processes, which was so far limited w.r.t. the LNT language, and to introduce preconditions and postconditions for both functions and processes. It also brings six code generation changes, in order to make the generated C code smaller, more structured, more readable, and safer. Four bugs were fixed.

- TRAIAN 3.5 brings two language changes and four code generation changes to LOTOS NT. Above all, it brings 24 changes in the static semantics analysis, in order to provide users with more relevant warnings and error messages. Ten bugs were fixed, mostly in the dataflow analysis.

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 was constantly updated.

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.

Our prior work on the computation of concurrent places in Petri nets led to a publication in an international conference [15], and enabled us to decompose [31] all the ordinary, safe Petri Nets of the MCC contest, including the largest ones, which could not be processed previously.

In 2021, the NUPN format and its associated software tools were enhanced as follows:

- In addition to three bug fixes, the CAESAR.BDD tool was enhanced in multiple ways. New internal data structures have been introduced to store the transitions going in and out of each place, which makes the tool (usually two times) faster on large NUPNs.
- The BDD exploration of reachable markings was made more efficient by no longer attempting to fire transitions known to be dead. A new, more efficient strategy for dynamic reordering of BDDs has been devised, which better reflects the structure and the intrinsic complexity of a NUPN. These changes, combined to other enhancements, increased the performance of many CAESAR.BDD options, which have been made faster and, possibly, more precise.
- CAESAR.BDD now accepts NUPNs with an empty initial marking. It has been also extended with three new options intended to characterize NUPNs that are equivalent modulo net isomorphism. These options have been continuously improved during year 2021 in order to better detect symmetries.
- In addition to three bug fixes, the NUPN_INFO tool was enhanced in various ways. Its code was made simpler and more modular, and the tool is now able to put NUPNs under a canonical form that helps detecting duplicate models in large collections of Petri nets and NUPNs. Eleven new options have also been added, whose combined use enabled to strengthen the behaviour of the “-canonical-nupn” option of NUPN_INFO.
- The manual pages for CAESAR.BDD, NUPN_INFO, and the NUPN format have been updated accordingly and modified at various places to resolve ambiguities.

We also studied two alternative approaches for the detection of isomorphic NUPNs: one approach reduces this problem to graph isomorphism and uses software tools dedicated to this problem; another approach encodes the isomorphism of two NUPNs in a quantifier-free integer-difference logic formula that can be passed to an SMT solver. Finally, we implemented an approach for translation of NUPNs into a process calculus such as LNT.

7.1.4 Formal Modeling and Analysis of BPMN

Participants: Gwen Salaün *(correspondent)*, 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 at ease with BPMN.

In the context of the MOAP project (see § 9.5.1), in collaboration with Yliès Falcone (CORSE project-team), 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, s/he can use our third algorithm to synthesize the BPMN process from that graph. This approach was implemented in a tool, validated on several case studies, and led to a publication in an international conference [20].

In 2021, we also improved this work by developing probabilistic model checking techniques of BPMN processes at runtime. Before running a BPMN process, the user has no clue 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 this work, we first transform the BPMN model into an LTS (Labelled Transition System). Then, by analyzing the execution traces obtained when running multiple instances of the process, we can compute the probability of executing each transition in the LTS model, and thus generate a Probabilistic Transition System (PTS). Finally, we perform probabilistic model checking for verifying that the PTS model satisfies a given probabilistic property. This verification loop is applied periodically to update the results according to the execution of the process instances. All these ideas are implemented in a tool chain, which was applied successfully to several real-world 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. Companies are continuously adjusting their resources to their needs following different strategies. However, the dynamic provisioning strategies are hard to compare. In collaboration with Francisco Durán (University of Málaga, Spain) and Camilo Rocha (Universidad Pontificia Javeriana, Cali, Colombia), 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. Four different strategies are presented, which are guided—respectively—by recent resource usage, recent resource request, predicted behavior, and a combination of available strategies. 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 Maude 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 journal [11].

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 2021, we proposed some techniques for simplifying the debugging of concurrent system specifications by exploiting their underlying behavioural model, represented as an LTS (Labeled Transition System). We start by extracting from the whole LTS the part which does not satisfy the given property. Then, in that LTS part, 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 LTS. Based on this annotated LTS part, we devised an approach for counting 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 LTS model before and after modification to detect whether this potential bug is a real bug or not. A tool was implemented for automating all the steps of the approach and applied on realistic examples for validation purposes.

7.2.2 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 2021, we contributed to this topic as follows:

- We improved an existing, fully-automated translation procedure [37] from the LAbS specification language [50] to the LNT formal description language. These improvements extend the range of supported temporal properties, reduce verification times, and allow us to use compositional verification to achieve further performance gains. These results led to a publication in an international conference [16].
- We described the above translation procedure in another paper, which has been accepted for publication in an international journal [38].
- We described the implementation and usage of SLiVER-CADP, i.e., the CADP-based automated verification tool for multi-agent systems, in another manuscript which is currently under review by an international journal.

7.2.3 Other Developments

Participants: Pierre Bouvier, Hubert Garavel.

VLSAT ("*Very Large Boolean SATisfiability problems*") is a benchmark suite to be used in scientific experiments and software competitions addressing SAT and SMT (Satisfiability Modulo Theories) solving issues. We obtained these SAT and SMT formulas from the automatic conversion [31] into Nested-Unit Petri Nets (NUPNs) [3] of a large collection of Petri nets modelling real-life problems, such as communication protocols and concurrent systems.

In 2021, we enriched VLSAT with two new benchmarks:

- VLSAT-2 [28][30] contains 100 benchmarks (50 satisfiable and 50 unsatisfiable formulas) of increasing complexity, proposed in DIMACS CNF format under a permissive Creative Commons license (VLSAT-2 is also available via its DOI). 25% of these benchmarks have been used during the 2020 and 2021 editions of the International SAT Competition.
- VLSAT-3 [27] contains 1200 (600 satisfiable and 600 unsatisfiable) quantifier-free first-order logic formulas of increasing complexity, proposed in SMT-LIB format under a permissive Creative Commons license (VLSAT-3 is also available via its DOI). More than 90% of these benchmarks have been used during the 16th International Satisfiability Modulo Theories Competition (SMT-COMP 2021).

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

7.3.1 Compositional Verification

Participants: Frédéric Lang.

The CADP toolbox contains various tools dedicated to compositional verification, among which EXP.OPEN, BCG_MIN, BCG_CMP, and SVL play a central role. EXP.OPEN explores on the fly the graph corresponding to a network of communicating automata (represented as a set of BCG files). BCG_MIN and BCG_CMP respectively minimize and compare behavior graphs modulo strong or branching bisimulation and their stochastic extensions. SVL (*Script Verification Language*) is both a high-level language for expressing complex verification scenarios and a compiler dedicated to this language.

In 2021, we corrected one bug in SVL, one bug in EXP.OPEN, and one bug in the ALDEBARAN shell script (which calls BCG_MIN and BCG_CMP). In addition, we implemented minimisation with respect to sharp bisimulation in BCG_MIN, using a new signature-based partition-refinement algorithm that replaces the partial reduction algorithm implemented in 2020. This new algorithm computes the sharp equivalence classes of an LTS, which allows the smallest LTS of its sharp equivalence class to be obtained. Also, this algorithm allowed us to extend the BCG_CMP tool to compare two LTSs modulo sharp bisimulation. The SVL language and compiler were extended to support LTS comparison modulo sharp bisimulation. A paper about sharp bisimulation was published in an international journal [13].

7.3.2 On-the-fly Test Case Extraction

Participants: Radu Mateescu, Nathan Miscopein, 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 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 2021, we rerun the benchmarks used to compare the original version of TESTOR to TGV, so as to thouroughly evaluate the impact the recent improvements brought to TESTOR. The results show that for the generation of a CTG, the new version of TESTOR requires on average almost three times less memory than TGV, with, on average, comparable execution times. This is a significant improvement with respect to the previous version of TESTOR, which was for some examples several orders of magnitude slower than TGV.

7.3.3 Runtime Enforcement

Participants: Gwen Salaün.

Runtime enforcement is a technique that analyzes a trace of actions (events) produced by the execution of a concurrent system, detects when this execution deviates from its expected behaviour w.r.t. a given property, and corrects the trace to make it satisfy the property. In collaboration with Yliès Falcone (CORSE project-team), we devised new runtime enforcement techniques that reorder actions in the trace when necessary, inject actions to the application to ensure progress of the property, and discard actions to avoid storing too many unnecessary actions. At any step of the enforcement, we provide a verdict, called enforcement trend, which takes its value in a 4-valued truth domain. Our approach has been implemented in a tool and validated on several application examples. Experimental results show that our techniques better preserve the application actions, hence ensuring better service continuity. This work led to a publication in an international conference [19].

7.3.4 Other Component Developments

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

In 2021, several components of CADP have been improved as follows:

- The FSP2LOTOS compiler now accepts the definition of local processes named ERROR or STOP, but only if they have a non-empty set of parameters. The CAESAR, CAESAR.ADT, and XTL compilers now reject C identifiers that collide with the reserved keywords of the C99 and C11 versions of the C language.
- In addition to nine bug fixes, the CADP tools have been updated to use, whenever possible, the HTTPS protocol instead of the FTP and HTTP protocols, and to use “gzip” and “xz” compression (instead of “compress” and “.Z” files). The installation tools of CADP now support the Oarsh/Oarcp commands for deployment on large clusters, such as Grid’5000.
- CADP was ported to macOS 11 "Big Sur" and macOS 12 "Monterey", and was adapted to support recent Linux distributions based on Glibc 2.32, as well as Linux distributions available from the Microsoft Store and running on top of Windows/WSL2 (Windows Subsystem for Linux).

7.4 Real-Life Applications and Case Studies

7.4.1 Rigorous Design, Reconfiguration, and Deployment of (I)IoT Applications

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

The Internet of Things (IoT) consists of devices and software interacting altogether in order to build powerful and added-value services. One of the main challenges in this context is to support end-users with simple, user-friendly, and automated techniques to design such applications. In 2021, we continued our work on this challenge in the framework of the Nokia Bell Labs collaboration (see § 8.1.1):

- Given the dynamicity of IoT applications, it is necessary consider at design time that often these applications are not built once and for all, but they can evolve over time and objects may be added or removed for various reasons (replacement, loss of connectivity, upgrade, failure, etc.). In collaboration with Francisco Durán (University of Málaga, Spain), we proposed new techniques for supporting the reconfiguration of running IoT applications, by comparing two versions of the application (before and after reconfiguration) to check if several properties related to reconfiguration are preserved. The analysis techniques have been implemented using the Maude framework and integrated into the WebThings platform. This work led to two publications in international conferences [17, 18].
- Industrial automation is a complex process involving various stakeholders. The two important aspects to consider during the automation system development are its business production goals and its technical implementation. The international standard IEC 61499 helps to specify distributed automation using a generic architectural model, targeting the technical development of the automation. However, it is not easy to analyse whether these IEC 61499 models satisfy production goals due to their informal semantics and inherent complexities of distributed logic. We devised a transformation from an IEC 61499 specification to a BPMN (Business Process Model Notation) model, which presents the automation from a business point of view, and also enables quantitative analysis of process models. Specifically, it allows business process designers to analyze the automation w.r.t. cost, resource allocation, and time. This analysis is achieved by transforming the business processes to formal models in the Maude rewriting logic. This work led to a publication in an international conference [23].

7.4.2 Autonomous Car

Participants: Jean-Baptiste Horel, Radu Mateescu *(correspondent)*, 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 § 9.3.1) and PRISSMA (see § 9.4.1) 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 is based on the generation of behavioral conformance tests, using the TESTOR tool (see § 7.3.2) from an LNT model (specifying the ground truth configuration with the range of vehicle behaviors) and a test purpose (specifying the critical feature to focus on). The obtained abstract test cases cover, by construction, all possible executions exercising a given feature, and can be automatically translated into the inputs of autonomous driving simulators. We illustrated our approach by generating thousands of behavior trees for the CARLA simulator for several realistic configurations. This work led to a paper accepted for publication in an international conference [22].

8 Bilateral contracts and grants with industry

8.1 Bilateral grants with industry

8.1.1 Nokia Bell Labs

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

A. Muroor Nadumane was supported by a post-doctorate grant (from January 2021 to August 2021) from Nokia Bell Labs (Nozay) on the formal modeling and analysis of industrial IoT (see § 7.4.1).

8.1.2 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).

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 Proof of Concept Grant ("LEOpowver") in 2021.

9.2 International research visitors

9.2.1 Visits of international scientists

- Nicolas Amat (LAAS-CNRS, Toulouse) visited us on September 27-29, 2021.
- Hugues Evrard (Google) visited us on November 15, 2021. He gave a seminar entitled "*Specifying and Testing GPU Workgroup Progress Models*".

9.2.2 Other international collaborations

In 2021, we had scientific relations with several universities and institutes abroad, including:

- University of Málaga, Spain (Francisco Durán),
- University of Cali, Colombia (Camilo Rocha),
- University of Toronto, Canada (Lina Marsso),
- Saarland University, Germany (Holger Hermanns),
- ISTI/CNR, Pisa, Italy (Franco Mazzanti),
- IMT Lucca, Italy (Rocco De Nicola),
- GSSI, L'Aquila, Italy (Omar Inverso).

9.3 European initiatives

9.3.1 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)
- Web site: https://autoc3rt.automotive.oth-aw.de/
- 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.

9.3.2 Collaborations with major European organizations

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

9.4 National initiatives

9.4.1 Fonds pour l'innovation et l'industrie

PRISSMA

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

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, LNE, RATP, and VALEO), as well as Université Gustave-Eiffel 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.

9.4.2 Other national collaborations

We had sustained scientific relations with the following researchers:

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

9.5 Regional initiatives

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

9.5.2 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.

10 Dissemination

Participants: Pierre Bouvier, Luca Di Stefano, Hubert Garavel, Frédéric Lang, Philippe Ledent, Radu Mateescu, Lucie Muller, Ajay Muroor Nadumane, Gwen Salaün, Wendelin Serwe, Ahang Zuo.

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

General chair, scientific chair

- 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.
- 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. 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.

Member of the organizing committees

- A. Zuo was publicity chair of FACS'2021 (*17th International Conference on Formal Aspects of Component Software*), virtual event, October 28-29, 2021.

10.1.2 Scientific events: selection

Chair of conference program committees

- F. Lang was tool chair of TACAS'2021 (*27th International Conference on Tools and Algorithms for the Construction of Systems*), Luxembourg, Luxembourg, March 27 - April 1, 2021.
- G. Salaün and Anton Wijs (Eindhoven University of Technology, The Netherlands) were programme committee co-chairs of FACS'2021.

Member of the conference program committees

- P. Bouvier was a PhD student session committee member and F. Lang was a programme committee member of ETR'2021 (*Ecole d'été Temps-Réel*), Poitiers/Futuroscope, France, September 20-24, 2021.
- H. Garavel was a programme committee member of CTS'2021 (*16th IFAC Symposium on Control in Transportation Systems*), Lille, France, June 8-10, 2021.
- H. Garavel was a programme committee member of appFM'2021 (*1st International Workshop on Applicable Formal Methods*), Beijing, China, November 20, 2021.
- H. Garavel and R. Mateescu were programme committee members of FMICS'2021 (*26th International Conference on Formal Methods for Industrial Critical Systems*), Paris, France, August 23-24, 2021.
- R. Mateescu was a programme committee member of TACAS'2021 (*27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*), Luxembourg, Luxembourg, March 27 - April 1, 2021.
- R. Mateescu was a programme committee member of IFIP-ICTSS'2021 (*33rd IFIP International Conference on Testing Software and Systems*), virtual event, November 10-12, 2021.
- R. Mateescu was a programme committee member of FM'2021 (*24th International Symposium on Formal Methods*), Beijing, China, November 20-26, 2021.
- G. Salaün was a programme committee member of SAC-SVT'2021 (*36th ACM/SIGAPP Symposium on Applied Computing - Software Verification and Testing Track*), virtual event, March 22-26, 2021.
- G. Salaün was a programme committee member of DOORS'2021 (*Edge Computing Workshop*), Zhytomyr, Ukraine, April 11, 2021.
- G. Salaün was a programme committee member of ENASE'2021 (*16th International Conference on Evaluation of Novel Approaches to Software Engineering*), virtual event, April 26-27, 2021.
- G. Salaün was a programme committee member of FormaliSE'2021 (*9th International Conference on Formal Methods in Software Engineering*), Madrid, Spain, May 17-21, 2021.
- G. Salaün was a programme committee member of FSEN'2021 (*9th IPM International Conference on Fundamentals of Software Engineering*), Tehran, Iran, May 19-21, 2021.
- G. Salaün was a programme committee member of COORDINATION'2021 (*23rd International Conference on Coordination Models and Languages*), Valetta, Malta, June 14-18, 2021.
- G. Salaün was a programme committee member of ICSOFT'2021 (*16th International Conference on Software Technologies*), virtual event, July 6-8, 2021.
- G. Salaün was a programme committee member of COMPSAC-SETA'2021 (*IEEE International Conference on Computers, Software, and Applications - Software Engineering Technologies and Applications*), virtual event, July 12-16, 2021.
- G. Salaün was a programme committee member of ICTAC'2021 (*18th International Colloquium on Theoretical Aspects of Computing*), virtual event, September 8-10, 2021.
- G. Salaün was a programme committee member of FACS'2021 (*17th International Conference on Formal Aspects of Component Software*), virtual event, October 28-29, 2021.
- G. Salaün was a programme committee member of SEFM'2021 (*19th International Conference on Software Engineering and Formal Methods*), virtual event, December 6-10, 2021.

Reviewer

- L. Di Stefano was a reviewer for ICSOFT'2021 and FM'2021.
- F. Lang was a reviewer for COMPSAC'2021, FM'2021, ICTAC'2021, SAC-SVT'2021, SEFM'2021, and TACAS'2021.
- P. Bouvier, H. Garavel, Ph. Ledent, L. Muller, and W. Serwe were reviewers for TACAS'2021.
- A. Muroor Nadumane was a reviewer for ENASE'2021 and TACAS'2021.

10.1.3 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

- F. Lang was a reviewer for JSS (*Journal of Systems & Software*) and STTT.
- R. Mateescu was a reviewer for STTT, TSE (*IEEE Transactions on Software Engineering*), and TR (*IEEE Transactions on Reliability*).
- G. Salaün was a reviewer for FGCS (*Future Generation Computer Systems*), IoT (*IEEE Internet of Things*), IST (*Information and Software Technology*), JUCS (*Journal of Universal Computer Science*), and TSE.
- W. Serwe was a reviewer for COSE (*Computers and Security*) et CSI (*Computer Standards & Interfaces*).

10.1.4 Software Dissemination and Internet Visibility

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

In 2021, the main facts are the following:

- We prepared and distributed twelve successive versions (2021-a to 2021-l) of CADP.
- We granted CADP licenses for 290 different computers in the world.

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

By the end of December 2021, the CADP forum, opened in 2007 for discussions regarding the CADP toolbox, had over 466 registered users and over 1972 messages had been exchanged.

Also, for the 2021 edition of the Model Checking Contest, we provided one family of models (totalling 17 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:

- Debugging and verification tools for Lingua Franca [36]
- Explaining safety violations in real-time systems [48]

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

- Compositional verification of synchronous reactive systems [32]
- Compatibility checking for asynchronously communicating software [51]
- Specifying and testing GPU workgroup progress models [53]
- Accelerating the computation of dead and concurrent places [29]
- Radiomic features for prostate cancer grade detection [52]

10.1.5 Invited talks

- P. Bouvier gave a talk entitled "*Trois problèmes d'accessibilité dans les réseaux de Petri saufs*" at the MeFoSyLoMa seminar held at LIP6, Paris on June 11, 2021.

10.1.6 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. In 2021, he also participated to the recruitment jury of the new head of Inria Grenoble' SED (*Service d'Expérimentation et de Développement*).
- 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 2021, 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 was appointed as representative of Inria Grenoble at the International Relations and Outreach of Université Grenoble Alpes.
- R. Mateescu was appointed as member of the council of the Mathematics, Information and Communication Sciences (MSTIC) research department of Université Grenoble Alpes.
- 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 2021 Inria activity reports at Inria Grenoble.
- W. Serwe is a member of the "*Comité de Centre*" at Inria Grenoble.

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 2021, we carried out the following teaching activities:

- P. Bouvier gave a course on "*Théorie des langages 1*" (18 hours "*équivalent TD*") to 1st year students of ENSIMAG (started in 2020, completed in 2021).
- L. Di Stefano taught a course on "*Modelling and Verification*" (42 hours "*équivalent TD*") to 5th year "*filière apprentissage*" Master's students in Computer Engineering at Polytech Paris-Saclay, April-May 2021.
- 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.
- G. Salaün taught about 240 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 (3rd 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) at ENSIMAG.

- W. Serwe participated to a course on "*Embedded Systems: From High-Confidence Design to Safe Execution*" (3 hours "*équivalent TD*" on behavioural testing) to 2nd year students of the MOSIG.
- A. Zuo gave a course on "*Introduction to Java/Android programming*" (34 hours "*équivalent TD*") to L2 students of IUT1/UGA and on "*Computer sciences applied to life sciences*" (27 hours "*équivalent TD*") to L1 students of UGA.

10.2.2 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: 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: A. Zuo, "*Modelling, Optimization and Predictive Analysis of Business Processes*", Université Grenoble Alpes, since October 2020, G. Salaün and Y. Falcone

10.2.3 Juries

- G. Salaün was reviewer of Jose Manuel Carrasco Mora's PhD thesis, entitled "*Unified Management of Applications on Heterogeneous Clouds*", defended at University of Málaga (Spain) on May 27, 2021.
- G. Salaün was reviewer of Erwan Mahé's PhD thesis, entitled "*An Operational Semantics of Interactions for Verifying Partially Observed Executions of Distributed Systems*", defended at Université Paris-Saclay on July 15, 2021.
- G. Salaün was reviewer of Laetitia Laversa's PhD thesis, entitled "*La synchronisabilité pour les systèmes distribués*", defended at Université Côte d'Azur on December 14, 2021.
- G. Salaün was reviewer of Sara Houhou's PhD thesis, entitled "*Parameterised Verification from Formal Specifications of Information Systems*", defended at Sorbonne Université on December 22, 2021.

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] G. Barbon, V. Leroy and G. Salaün. 'Debugging of Behavioural Models using Counterexample Analysis'. In: *IEEE Transactions on Software Engineering* 47.6 (June 2021), pp. 1184–1197. DOI: 10.1109/TSE.2019.2915303. URL: https://hal.inria.fr/hal-02145610.

[10] L. Di Stefano, R. De Nicola and O. Inverso. 'Verification of Distributed Systems via Sequential Emulation'. In: *ACM Transactions on Software Engineering and Methodology* (2022). URL: https://hal.inria.fr/hal-03549925.

[11] F. Durán, C. Rocha and G. Salaün. 'Resource Provisioning Strategies for BPMN Processes: Specification and Analysis using Maude'. In: *Journal of Logical and Algebraic Methods in Programming* (1st Nov. 2021), pp. 1–50. DOI: 10.1016/j.jlamp.2021.100711. URL: https://hal.inria.fr/hal-03487960.

[12] 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.

[13] F. Lang, R. Mateescu and F. Mazzanti. 'Compositional Verification of Concurrent Systems by Combining Bisimulations'. In: *Formal Methods in System Design* (19th Feb. 2021). DOI: 10.1007/s10703-021-00360-w. URL: https://hal.inria.fr/hal-03159616.

[14] 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

[15] P. Bouvier and H. Garavel. 'Efficient Algorithms for Three Reachability Problems in Safe Petri Nets'. In: *Proceedings of the 42nd International Conference on Applications and Theory of Petri Nets and Concurrency (PETRI NETS 2021)*. PETRI NETS 2021 - 42nd International Conference on Application and Theory of Petri Nets and Concurrency. Paris, France: Springer, 16th June 2021, pp. 339–359. DOI: 10.1007/978-3-030-76983-3_17. URL: https://hal.inria.fr/hal-03286069.

[16] L. Di Stefano and F. Lang. 'Verifying Temporal Properties of Stigmergic Collective Systems Using CADP'. In: ISoLA 2021 - 10th International Symposium on Leveraging Applications of Formal Methods. Vol. 13036. Lecture Notes in Computer Science. Rhodes, Greece: Springer International Publishing, 12th Oct. 2021, pp. 473–489. DOI: 10.1007/978-3-030-89159-6_29. URL: https://hal.inria.fr/hal-03385131.

[17] F. Durán, A. Krishna, M. Le Pallec, R. Mateescu and G. Salaün. 'R-MOZART: A Reconfiguration Tool for WebThings Applications'. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). Madrid / Virtual, Spain: IEEE, 25th May 2021, pp. 41–44. DOI: 10.1109/ICSE-Companion52605.2021.00031. URL: https://hal.inria.fr/hal-03157158.

[18] F. Durán, A. Krishna, M. Le Pallec, R. Mateescu and G. Salaün. 'Seamless Reconfiguration of Rule-Based IoT Applications'. In: SEAMS 2021 - 16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. Vol. 1. Madrid / Virtual, Spain, 18th May 2021, pp. 142–148. URL: https://hal.inria.fr/hal-03254192.

[19] Y. Falcone and G. Salaün. 'Runtime Enforcement with Reordering, Healing, and Suppression'. In: SEFM 2021 - 19th IEEE International Conference on Software Engineering and Formal Methods. Virtual, United Kingdom: IEEE, 6th Dec. 2021, pp. 1–20. URL: https://hal.inria.fr/hal-03484045.

[20] 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, 5th Sept. 2021, pp. 1–6. URL: https://hal.inria.fr/hal-03330330.

[21] H. Garavel, F. Lang, R. Mateescu and W. Serwe. 'Is CADP an Applicable Formal Method?' In: AppFM 2021 - 1st International Workshop on Applicable Formal Methods. Bejing, China, 20th Nov. 2021. URL: https://hal.inria.fr/hal-03485114.

[22] 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. URL: https://hal.inria.fr/hal-03516799.

[23] A. Krishna and G. Salaün. 'Business Process Models for Analysis of Industrial IoT Applications'. In: IoT 2021 - 11th International Conference on the Internet of Things. St. Gallen, Switzerland, 8th Nov. 2021, pp. 1–8. DOI: 10.1145/3494322.3494336. URL: https://hal.inria.fr/hal-03484052.

[24] G. Salaün. 'Consistent Substitution of Object in Rule-based IoT Applications'. In: COMPSAC 2021 - Computer Software and Applications Conference. Virtual, United States, 12th July 2021, pp. 1–9. URL: https://hal.inria.fr/hal-03484028.

Edition (books, proceedings, special issue of a journal)

[25] P. C. C. Ölveczky and G. Salaün. *Preface: Special issue on Software Engineering and Formal Methods.* Vol. 20. 2. Springer Verlag, Apr. 2021, pp. 291–292. DOI: 10.1007/s10270-021-00874-1. URL: https://hal.inria.fr/hal-03507829.

[26] G. Salaün and A. Wijs, eds. *Formal Aspects of Component Software: Proceedings of the 17th International Conference on Formal Aspects of Component Software (FACS 2021).* Vol. 13077. Lecture Notes in Computer Science. Springer International Publishing, 2021. DOI: 10.1007/978-3-030-90636-8. URL: https://hal.inria.fr/hal-03507856.

Reports & preprints

[27] P. Bouvier. *The VLSAT-3 Benchmark Suite.* RT-0516. INRIA Grenoble Rhône-Alpes, 7th Dec. 2021, pp. 1–20. URL: https://hal.inria.fr/hal-03468625.

[28] P. Bouvier and H. Garavel. *The VLSAT-2 Benchmark Suite.* RT-0514. INRIA Grenoble Rhône-Alpes, 7th Sept. 2021, pp. 1–8. URL: https://hal.inria.fr/hal-03337115.

11.3 Cited publications

[29] N. Amat, S. Dal-Zilio and D. L. Botlan. 'Accelerating the Computation of Dead and Concurrent Places Using Reductions'. In: *Proceedings of the 27th International Symposium on Model Checking Software (SPIN'2021), Virtual Event.* Ed. by A. Laarman and A. Sokolova. Vol. 12864. Lecture Notes in Computer Science. Springer Verlag, July 2021, pp. 45–62.

[30] P. Bouvier and H. Garavel. 'SAT-Competition Benchmarks Spawning from Concurrency Theory'. In: *Proceedings of SAT Competition 2021 – Solver and Benchmark Descriptions.* Ed. by T. Balyo, N. Froleyks, M. J. H. Heule, M. Iser, M. Järvisalo and M. Suda. Report B-2021-1, University of Helsinki, Department of Computer Science, 2021, pp. 47–48.

[31] 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, June 2020. URL: `https://hal.inria.fr/hal-02875957`.

[32] S. Chabane, R. Ameur-Boulifa and M. Mohamed. 'Towards Compositional Verification of Synchronous Reactive Systems'. In: *International Journal of Critical Computer-Based Systems* 10.2 (Sept. 2021), pp. 120–142.

[33] 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.

[34] 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.

[35] 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.

[36] J. Deantoni, J. Cambeiro, S. Bateni and S. Lin. 'Debugging and Verification Tools for LINGUA FRANCA in GEMOC Studio'. In: *Proceedings of the 24th Forum on specification & Design Languages (FDL'2021), Antibes, France.* IEEE, Sept. 2021, pp. 1–8.

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

[38] L. Di Stefano, R. D. Nicola and O. Inverso. 'Verification of Distributed Systems via Sequential Emulation'. In: *ACM Trans. Softw. Eng. Methodol.* (2022). To appear.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

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

[45] 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.

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

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

[48] T. Mari, T. Dang and G. Gössler. ‘Explaining Safety Violations in Real-Time Systems’. In: *Proceedings of the 19th International Conference on the Formal Modeling and Analysis of Timed Systems (FORMATS'2021), Paris, France*. Ed. by C. Dima and M. Shirmohammadi. Vol. 12860. Lecture Notes in Computer Science. Springer Verlag, Aug. 2021, pp. 100–116.

[49] 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.

[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] M. Ouederni. ‘Compatibility Checking for Asynchronously Communicating Software’. In: *Science of Computer Programming* 205 (Jan. 2021).

[52] A. Santone, M. C. Brunese, F. Donnarumma, P. Guerriero, F. Mercaldo, A. Reginelli, V. Miele, A. Giovagnoni and L. Brunese. ‘Radiomic Features for Prostate Cancer Grade Detection through Formal Verification’. In: *La radiologia medica* 126 (Jan. 2021), pp. 688–697.

[53] T. Sorensen, L. F. Salvador, H. Raval, H. Evrard, J. Wickerson, M. Martonosi and A. F. Donaldson. ‘Specifying and Testing GPU Workgroup Progress Models’. In: *Proc. ACM Program. Lang.* 5.OOPSLA (Oct. 2021), pp. 1–30.