IN PARTNERSHIP WITH:
Institut polytechnique de Grenoble

Université Joseph Fourier (Grenoble)

Activity Report 2014

Project-Team CONVECS

Construction of verified concurrent systems

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

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Embedded and Real-time Systems

Table of contents

Project-Team CONVECS

Keywords: Formal Methods, Model-checking, Verification, Distributed Algorithms, Markovian Model

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

1. Members

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 [Grenoble INP, Associate Professor, HdR]

Engineers
Eric Léo [Inria, granted by Conseil Régional Rhône-Alpes]
Soraya Arias [Inria, Engineer]

PhD Students
Rim Abid [Univ. Grenoble I]
Hugues Evrard [Inria, granted by Caisse des Dépôts et Consignations]
Fatma Jebali [Inria, granted by Conseil Général de l'Isère]
Abderahman Kriouile [CIFRE with STMicroelectronics]
Raquel Oliveira [Univ. Grenoble I]

Post-Doctoral Fellows
Jingyan Jourdan-Lu [Inria, granted by Conseil Régional Rhône-Alpes]
Jose Ignacio Requeno [Inria, from Sep 2014]
Lina Ye [Inria, until Jul 2014, granted by Caisse des Dépôts et Consignations]

Administrative Assistants
Myriam Etienne [Inria]
Isabelle Allegre [Inria, from Nov 2014]

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 [35] 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*) [56], 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 [42], provides an "opaque" representation of states and actions (transitions labels): states are represented as memory areas of fixed size and actions are character strings. Although appropriate to the pure process algebraic setting, this representation must be generalized to provide additional information supporting an efficient construction of advanced verification features, such as: handling of the types, functions, data values, and parallel structure of the source program under verification, independence of transitions in the LTS, quantitative (timed/probabilistic/stochastic) information, etc.
- *Compositional framework for on-the-fly LTS analysis*. On-the-fly model checkers and equivalence checkers usually perform several operations on graph models (LTSs, Boolean graphs, etc.), such as exploration, parallel composition, partial order reduction, encoding of model checking and equivalence checking in terms of Boolean equation systems, resolution and diagnostic generation for Boolean equation systems, etc. To facilitate the design, implementation, and usage of these functionalities, it is necessary to encapsulate them in software components that could be freely combined and replaced. Such components would act as graph transformers, that would execute (on a sequential machine) in a way similar to coroutines and to the composition of lazy functions in functional programming languages. Besides its obvious benefits in modularity, such a component-based architecture will also make it possible to take advantage of multi-core processors.

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

3.5. Real-Life Applications and Case Studies

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

4. Application Domains

4.1. Application Domains

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

- *Bioinformatics:* genetic regulatory networks, nutritional stress response, metabolic pathways,
- *Component-based systems:* Web services, peer-to-peer networks,
- *Databases:* transaction protocols, distributed knowledge bases, stock management,
- *Distributed systems:* virtual shared memory, dynamic reconfiguration algorithms, fault tolerance algorithms, cloud computing,
- *Embedded systems:* air traffic control, avionic systems, medical devices,
- *Hardware architectures:* multiprocessor architectures, systems on chip, cache coherency protocols, hardware/software codesign,
- *Human-machine interaction:* graphical interfaces, biomedical data visualization, plasticity,
- *Security protocols:* authentication, electronic transactions, cryptographic key distribution,
- *Telecommunications:* high-speed networks, network management, mobile telephony, feature interaction detection.

5. New Software and Platforms

5.1. The CADP Toolbox

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

We maintain and enhance CADP (*Construction and Analysis of Distributed Processes* – formerly known as *CAESAR/ALDEBARAN Development Package*) [1], a toolbox for protocols and distributed systems engineering [1]. In this toolbox, we develop and maintain the following tools:

- CAESAR.ADT [41] 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 [47], [46] 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 [42] 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,

[1] http://cadp.inria.fr

  - 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 [51], CTL [37], ACTL [39], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [45] 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 [55] 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 the VERTECS project-team at Inria Rennes – Bretagne-Atlantique.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [43] 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.

5.2. The TRAIAN Compiler

Participants: Hubert Garavel [correspondent], Frédéric Lang, Wendelin Serwe.

We develop a compiler named TRAIAN 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 [44], being used in all recent compilers developed by CONVECS.

The TRAIAN compiler can be freely downloaded from the CONVECS Web site [2].

5.3. The PIC2LNT Translator

Participants: Radu Mateescu, Gwen Salaün [correspondent].

We develop a translator named PIC2LNT from an applied π-calculus (see § 6.1) to LNT, which enables the analysis of concurrent value-passing mobile systems using CADP.

PIC2LNT is developed by using the SYNTAX tool (developed at Inria Paris-Rocquencourt) for lexical and syntactic analysis together with LOTOS NT for semantical aspects, in particular the definition, construction, and traversal of abstract trees.

The PIC2LNT translator can be freely downloaded from the CONVECS Web site [3].

5.4. The PMC Partial Model Checker

Participants: Radu Mateescu, Frédéric Lang.

We develop a tool named PMC (*Partial Model Checker*, see § 6.4), which performs the compositional model checking of dataless MCL formulas on networks of communicating automata described in the EXP language.

PMC can be freely downloaded from the CONVECS Web site [4].

6. New Results

6.1. New Formal Languages and their Implementations

LNT is a next generation formal description language for asynchronous concurrent systems, which attempts to combine the best features of imperative programming languages and value-passing process algebras. LNT is increasingly used by CONVECS for industrial case studies and applications (see § 6.5) and serves also in university courses on concurrency, in particular at ENSIMAG (Grenoble) and at Saarland University.

6.1.1. Translation from LNT to LOTOS

Participants: Hubert Garavel, Frédéric Lang, Wendelin Serwe.

In 2014, the translator from LNT to LOTOS was further improved. In addition to bug fixes and removal of incorrect warnings emitted by the translator itself or by the C compiler on the generated code, the following enhancements have been brought: the LNT language was extended with a "`!representedby`" pragma for processes, and a "`only if`" statement to concisely express guarded commands; the translator now performs static analysis and warns about unused variables, unused "`in`" or "`in out`" parameters, useless (deterministic or nondeterministic) assignments to variables, "`in out`" parameters that are never assigned, and dubious synchronizations between processes; checks for underflow/overflow on natural and integer numbers are now activated by default. The translator also generates better LOTOS code, and the LNT reference manual was shortened and updated in many places.

6.1.2. Translation from LOTOS to Petri nets and C

Participants: Hubert Garavel, Wendelin Serwe.

[2] http://convecs.inria.fr/software/traian
[3] http://convecs.inria.fr/software/pic2lnt
[4] http://convecs.inria.fr/software/pmc

The LOTOS compilers CAESAR and CAESAR.ADT, which were once the flagship of CADP, now play a more discrete role since LNT (rather than LOTOS) has become the recommended specification language of CADP. Thus, CAESAR and CAESAR.ADT are mostly used as back-end translators for LOTOS programs automatically generated from LNT or other formalisms such as Fiacre, and are only modified when this appears to be strictly necessary.

In 2014, the CAESAR compiler has been modified to tolerate LOTOS specifications that would be normally rejected under the ISO/IEC 8807 standard definition of LOTOS. The first change extends the visibility scope of local definitions when the global definitions are empty. The second change uses the type information of process definitions to better resolve overloading ambiguities in expressions passed as actual parameters to process calls.

Conversely, CAESAR was made stricter by rejecting at compile-time LOTOS specifications containing out-of-bound constants, even if such constants are never used.

Performance has been increased by adding or strengthening a number of optimizations concerning, e.g., internal data structures, Boolean guards that can be statically evaluated, values belonging to singleton sorts, disconnected or otherwise unreachable Petri net places and transitions, etc.

The CAESAR.BDD tool of CADP, which analyzes hierarchical Petri nets generated from higher-level specifications (e.g., LOTOS or LNT) has been significantly enhanced. The semantic model accepted by CAESAR.BDD has been made more general and given the new name of NUPN (*Nested-Units Petri Nets*). The definition and theoretical properties of NUPN have been formalized.

The textual syntax for NUPN has been extended with pragmas intended to retain useful properties of non-ordinary and/or non-safe Petri nets translated to NUPN. An XML syntax for NUPN (compatible with the ISO standard PNML for the representation of Petri nets) has been defined and adopted by the Model Checking Contest [5]. A translator from PNML to NUPN has been developed at LIP6 (Paris, France).

The CAESAR.BDD tool has been updated accordingly, and extended to perform stricter checks and compute more structural and behavioral properties of NUPN models. CAESAR.BDD has been intensively used to correct the descriptions of the Model Checking Contest benchmarks: a first campaign (January-February 2014) detected 9 errors in structural properties and 8 errors in behavioral properties, and a second campaign (April 2014) revealed 23 more errors. CAESAR.BDD has also been used to automatically generate new benchmarks, together with their descriptions.

6.1.3. Translation from GRL to LNT

Participants: Fatma Jebali, Frédéric Lang, Eric Léo, Radu Mateescu.

In the context of the Bluesky project (see § 8.1.2.1), we study the formal modeling of GALS (*Globally Asynchronous, Locally Synchronous*) systems, which are composed of several synchronous subsystems evolving cyclically, each at its own pace, and communicating with each other asynchronously. Designing GALS systems is challenging due to both the high level of (synchronous and asynchronous) concurrency and the heterogeneity of computations (deterministic and nondeterministic). To bring our formal verification techniques and tools closer to the GALS paradigm, we designed a new formal language named GRL (*GALS Representation Language*), as an intermediate format between GALS models and purely asynchronous concurrent models. GRL combines the main features of synchronous dataflow programming and asynchronous process calculi into one unified language, while keeping the syntax homogeneous for better acceptance by industrial GALS designers. GRL allows a modular composition of synchronous systems (blocks), environmental constraints (environments), and asynchronous communication mechanisms (mediums), to be described at a level of abstraction that is appropriate to verification. GRL also supports external C and LNT code.

In 2014, we have continued to enhance the syntax and the formal semantics of GRL. We have written a detailed research report (82 pages) [25] containing the complete definition of the syntax, static semantics, and dynamic semantics (in the form of structural operational semantics rules), and also illustrating the checking of dynamic

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

semantics rules on several examples of GRL programs. A paper describing GRL has been published in an international conference [14].

To equip GRL with verification features, we formally defined a translation from GRL to LNT. GRL blocks are translated into LNT functions, possibly encapsulated within LNT wrapper processes to enable asynchronous communication, whereas GRL environments and mediums are directly translated into LNT processes. The asynchronous composition of blocks, environments, and mediums is translated to an LNT parallel composition of the corresponding processes.

Using the SYNTAX and LOTOS NT compiler construction technology [44], we have developed a translator named GRL2LNT (about $25,000$ lines of code), allowing an LNT program to be generated from a GRL specification automatically. GRL2LNT performs the lexical and syntactic analysis of GRL programs, together with almost all static semantic checks specified in its formal definition [25]. A stable version of GRL2LNT has been released in 2014. Additionally, we have developed an OPEN/CAESAR-compliant compiler GRL.OPEN (based on GRL2LNT and LNT.OPEN), which makes possible the on-the-fly exploration of the LTS underlying a GRL specification using CADP. We have also built a test base containing about 250 (correct and incorrect) GRL programs, and used it for non-regression testing of GRL2LNT. The correct GRL programs represent about $7,000$ lines of code and produce about $18,000$ lines of LNT code after translation using GRL2LNT.

A paper describing the formal verification of GALS systems using GRL and CADP, with a focus on the translation from GRL to LNT, has been submitted to an international conference [28].

6.1.4. Coverage Analysis for LNT

Participants: Gwen Salaün, Lina Ye.

In the classic verification setting, the designer has a specification of a system in a value-passing process algebra, a set of temporal properties to be verified on the corresponding LTS model, and a data set of examples (test cases) for validation purposes. At this stage, building the set of validation examples and debugging the specification is a complicated task, in particular for non-experts.

We propose a new framework for debugging value-passing process algebra through coverage analysis and we illustrate our approach with LNT. We define several coverage notions before showing how to instrument the specification without affecting original behaviors. Our approach helps one to improve the quality of a data set of examples used for validation purposes, but also to find ill-formed decisions, dead code, and other errors in the specification. We have implemented a tool for automating our debugging approach, and applied it to several real-world case studies in different application areas.

In 2014, a paper has been accepted in an international conference [19].

6.1.5. Other Language Developments

Participants: Hugues Evrard, Hubert Garavel, Frédéric Lang, Eric Léo, Wendelin Serwe.

The ability to compile and verify formal specifications with complex, user-defined operations and data structures is a key feature of the CADP toolbox since its very origins. A long-run effort has been recently undertaken to ensure a uniform treatment of types, values, and functions across all the various CADP tools.

In 2014, convergence between the LOTOS, LNT, BCG, and XTL data-type libraries has been increased by defining common libraries for eight predefined types: Boolean, Natural, Integer, Real, Character, String, Raw, and Gate. These libraries gather in the same place definitions of types, constants, and functions that were previously disseminated across different tools. Additionally, systematic checks for underflows and overflows have been set for the Natural and Integer types. Also, unprintable characters and C-like escape sequences are now uniformly handled by the Character, String, and Raw types.

To support the LNT language in the Emacs/XEmacs, jEdit, and Vim editors, configuration files have been added or updated, which provide for syntax highlighting/coloring, and enable autocompletion in Emacs using YASnippet.

6.2. Parallel and Distributed Verification

6.2.1. Distributed Code Generation for LNT

Participants: Hugues Evrard, Frédéric Lang.

Rigorous development and prototyping of a distributed verification algorithm in LNT involves the automatic generation of a distributed implementation. For the latter, a protocol realizing process synchronization is required. As far as possible, this protocol must itself be distributed, so as to avoid the bottleneck that would inevitably arise if a unique process would have to manage all synchronizations in the system. A particularity of such a protocol is its ability to support *branching synchronizations*, corresponding to situations where a process may offer a choice of synchronizing actions (which themselves may nondeterministically involve several sets of synchronizing processes) instead of a single one. Therefore, a classical barrier protocol is not sufficient and a more elaborate synchronization protocol is needed.

Using a synchronization protocol that we verified formally in 2013, we developed a prototype distributed code generator, named DLC (*Distributed LNT Compiler*), which takes as input the model of a distributed system described as a parallel composition of LNT processes.

In 2014, we continued the development of DLC. We improved the performances of DLC generated code by reducing the number of protocol messages when one or several processes are ready on a single gate. We experimented this optimization on a set of processes running on different computers and synchronizing all together on a single barrier interaction (i.e., all processes are ready on a single gate). In this situation, DLC now generates code that is faster than Java or Erlang.

The distributed program generated by DLC would be of little interest if it could not interact with its environment (e.g., users through human-computer interfaces, or other systems, such as databases, Web services, etc.). Therefore, we designed a mechanism to embed user-defined C functions, called *hook functions*, into the code generated by DLC. Hook functions are triggered on events related to actions in the system. This allows system actions to be, e.g., monitored by the user or controlled by external conditions. Using hook functions, the code generated by DLC can thus both take an account of and have an effect on its environment.

In order to demonstrate DLC on a real-world example, we applied it to the recent Raft [6] consensus algorithm [60]. We wrote an LNT specification of a simple key-value store made fault tolerant by replication of commands using the Raft consensus algorithm. During the modeling phase, we found a missing transition in the TLA+ specification of the protocol. We signaled it to the authors [7], who corrected the TLA+ specification. We used hook functions to implement interaction with the replicated store from external clients. The distributed implementation generated by DLC was successfully tested on clusters of the Grid5000 platform. We presented an overview of DLC, the hook functions and the Raft experiment in an article that has been accepted for publication in an international conference [12].

6.3. Timed, Probabilistic, and Stochastic Extensions

6.3.1. Model Checking for Extended PCTL

Participants: Hubert Garavel, Radu Mateescu, Jose Ignacio Requeno.

In the context of the SENSATION project (see § 8.2.1.1), we study the specification and verification of quantitative properties of concurrent systems.

In 2014, we defined an extension of PCTL (*Probabilistic Computation Tree Logic*) [49] with the manipulation of data values and actions. This logic is interpreted on extended DTMCs (*Discrete-Time Markov Chains*) containing visible transitions, labeled with channel names and data values, in addition to probabilistic transitions. Extended PCTL makes possible the specification of temporal properties involving discrete time, probabilities, and data values.

[6] http://raftconsensus.github.io
[7] https://groups.google.com/forum/#!topic/raft-dev/yu-wOUx-gnA

We devised a prototype model checker for extended PCTL in the form of an XTL library describing the denotational semantics of all PCTL operators (both primitive and derived ones), accompanied by external C code implementing the algorithms for LTS exploration and numerical computation of probabilities. The high-level programming language constructs of XTL (iterators, sets in comprehension, parameterized macro-definitions) allowed us to easily implement the advanced features (filters on arithmetic and logical operators, computation of probabilities, experiments over data series, etc.) of established probabilistic model checkers, such as PRISM [54]. Also, the manipulation of data values in XTL allows one to specify properties in which probabilities and discrete time deadlines depend on the values of state variables, a feature currently not provided by PRISM.

To experiment and cross-check our extended PCTL library w.r.t. PRISM, we developed an automated translator from the (state-based) DTMCs used by PRISM into the (action-based) DTMCs in BCG format used by CADP. State information is represented by means of special self-looping transitions containing the values of state variables, which are properly handled during the evaluation of probabilistic temporal operators.

The experiments we performed with our extended PCTL library on various examples of DTMCs (produced from communication protocols, chemical reactions, hazard games, etc.) showed a performance comparable to (explicit-state) PRISM for pure PCTL formulas.

Furthermore, in addition to many bug fixes, the XTL compiler and its XTL_EXPAND preprocessor have been strengthened to better detect and report potential mistakes in source XTL specifications. In particular, vacuity checks have been introduced, which warn the user when no label in a BCG graph has the right number of fields or the appropriate field types to satisfy an XTL label match expression (previously, this expression would silently evaluate to false).

The type checking system of XTL and its list of predefined functions have been extended to support the new Natural and Raw types of the BCG format, and to properly distinguish between Natural and Integer values, and Raw and String values, while achieving a high degree of backward compatibility. In particular, XTL now uses type information from the BCG labels to better solve overloading in label offers, so that certain XTL programs that were formerly invalid are now accepted. Finally, it is now possible to use the predefined types and functions of XTL when defining temporal operators directly using external C code.

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

6.4.1. Property-Dependent Reductions for the Modal Mu-Calculus

Participant: Radu Mateescu.

In collaboration with Anton Wijs (Technical University of Eindhoven), we proposed a new method for enhancing the performance of model checking a temporal formula on an LTS by reducing the LTS as much as possible depending on the formula prior to (or simultaneously with) the verification. Given an LTS and a formula, the method consists of two steps:

- The maximal set of actions that one can hide (i.e., rename into the internal action τ) in the LTS without disturbing the interpretation of the formula is computed, and those actions are hidden in the LTS. This works for any formula of the full modal μ-calculus (i.e., of arbitrary alternation depth) and provides the highest potential for reducing the LTS, and hence for improving verification performance, w.r.t. that formula.
- The LTS is reduced modulo an equivalence relation preserving the formula. The reduction can be done before verification, either by constructing the LTS explicitly and using the direct minimization features provided by the BCG_MIN tool, or by constructing the minimized LTS incrementally using the compositional verification features provided by EXP.OPEN and SVL. The reduction can be also done simultaneously during verification, by detecting τ-confluent transitions and prioritizing them against their neighbors.

We defined a μ-calculus fragment, named Lμ-dsbr, and shown its adequacy w.r.t. divergence-sensitive branching bisimulation (divbranching for short). We also shown that Lμ-dsbr is equally expressive to the μ-ACTL$\setminus$X logic, an extension of ACTL$\setminus$X (Action-based CTL without the next time operator) with fixed point operators [39], [40]. This result also implies the adequacy w.r.t. divbranching of μ-ACTL$\setminus$X, which was previously shown to be adequate w.r.t. strong bisimulation.

We experimented our method using the EVALUATOR model checker on various examples of protocols and distributed systems, by specifying the temporal properties in MCL and reducing the LTSs modulo strong and divbranching bisimulation. The experiments showed performance enhancements both in execution time (reduction by a factor 4 for strong bisimulation and 20 for divbranching) and memory consumption (reduction by a factor 2 for strong bisimulation and 5 for divbranching).

We also built a prototype MCL library regrouping the temporal operators of ACTL$\setminus$X (which were already present in CADP) and the modal and temporal operators of Lμ-dsbr (which were newly added). Used in conjunction with the Boolean and fixed point operators of MCL, the operators of this library can be used to specify temporal formulas adequate w.r.t. divbranching, which allows one to reduce the LTS modulo this equivalence (after applying maximal hiding) and to increase the performance of verification accordingly. An article has been published in an international journal [8].

6.4.2. Compositional Verification

Participants: Hubert Garavel, Frédéric Lang.

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

In 2014, we corrected 2 bugs in EXP.OPEN, 6 bugs in BCG_MIN and BCG_CMP, and 5 bugs in SVL. We also enhanced these tools as follows:

- We corrected the diagnostic generation algorithm of BCG_CMP, which sometimes generated irrelevant diagnostics.
- We improved the messages displayed by SVL and EXP.OPEN when generating an LTS from a composition expression using the *smart reduction* strategy [38], so that the user can follow more easily the selected composition order.
- Following the recent progress made on the development of the language LNT (see § 6.1), the syntax of the SVL and EXP languages for comments, gate typing, and the "`par`", "`hide`", "`rename`", "`cut`", and "`prio`" operators was extended to be compatible with the syntax of LNT. This enables composition expressions (including comments, channel typing, etc.) copied from LNT programs to be pasted in SVL scripts while requiring as few syntactic changes as possible.
- The "`verify`" operator has been generalized to give access to all three model checkers of CADP (EVALUATOR 3, EVALUATOR 4, and XTL). A new statement "`|=`" has been added to SVL, which enables MCL and XTL formulas to be directly written in an SVL script, rather than being stored in external files.
- To provide for requirements expression and traceability in SVL, we introduced two new statements, "`property`" and "`check`", which increase the readability and good structure of SVL scripts by allowing to define and verify properties, each of which is given a name, instantiable parameters, an informal textual description, and (optionally) an expected truth value.
- We updated several demo examples of CADP in order to illustrate the above extensions.

6.4.3. On-the-Fly Test Generation

Participants: Hubert Garavel, Radu Mateescu, Wendelin Serwe.

In the context of the collaboration with STMicroelectronics, we study techniques for testing if a (hardware) implementation is conform to a formal model described in LNT. Our approach is inspired by the theory of conformance testing [62], as implemented for instance in TGV [53] and JTorX [33]. We have developed two prototype tools to support this approach. The first tool implements a dedicated OPEN/CAESAR-compliant compiler for the particular asymmetric synchronous product between the model and the test purpose. The second tool, based on slightly extended generic components for graph manipulation (τ-compression, τ-confluence reduction, determinization) and resolution of Boolean equation systems, generates the complete test graph (CTG), which can be used to extract concrete test cases or to drive the test of the implementation. The principal advantage of our approach compared to existing tools is the use of LNT for describing test purposes, which facilitates the manipulation of data values.

In 2014, we developed a third prototype tool that takes as input a CTG and extracts either a single test case (randomly chosen or the first encountered one), or the set of *all* test cases. This prototype tool was used in the case study with STMicroelectronics (see § 6.5.1).

The test-generation tool TGV has been streamlined by removing some obsolete options and replacing a large part of its code by calls to the standard CADP libraries. TGV has been made faster, it now supports the latest version of the AUT format, and ensures that test purposes provided in the BCG format are deterministic. The manual page has been updated and completed.

6.4.4. Other Component Developments

Participants: Soraya Arias, Hubert Garavel, Frédéric Lang, Radu Mateescu.

The AUT textual format for CADP for storing LTSs was extended to support recent languages (such as LNT and the PseuCo language developed at Saarland University) that manipulate character-string values. The AUT format, which was defined in the late 80s, did not support such values. A new version 2014 of the AUT format has been defined, which solves this problem and maintains backward compatibility. All the CADP tools that read or write AUT files have been updated accordingly.

The BCG format of CADP for storing LTSs has been upgraded with the advent of a new version 1.2, which replaces version 1.1 released in 2009. New predefined types have been added to BCG to express the difference between unsigned and signed integers, and between character strings and untyped raw-data values. The new version of the BCG format is also more compact and now uses variable-length encoding for strings. The rules for label parsing of the BCG_WRITE interface have been extended, and BCG_IO now supports version 2014 of the AUT format. The intrinsic difficulty of these changes was to preserve the backward compatibility with the BCG files generated over the last twenty years.

To simplify the installation of CADP on Windows systems, we studied an alternative execution environment based on Gnuwin32 and MinGW/Msys rather than Cygwin. Preliminary changes have been brought to CADP scripts to undertake such a migration.

6.5. Real-Life Applications and Case Studies

6.5.1. ACE Cache Coherency Protocol

Participants: Abderahman Kriouile, Radu Mateescu, Wendelin Serwe.

In the context of a CIFRE convention with STMicroelectronics, we study system-level cache coherency, a major challenge faced in the current System-on-Chip architectures. Because of their increasing complexity (mainly due to the significant number of computing units), the validation effort using current simulation-based techniques grows exponentially. As an alternative, we study formal verification.

We focused on the ACE (AXI Coherency Extensions) cache coherency protocol, a system-level coherency protocol proposed by ARM [29]. In previous years, we developed a formal LNT model (about $3,400$ lines of LNT) of a system consisting of an ACE-based cache coherent interconnect, processors, and a main memory. The model is parametric and can be instantiated with different configurations (number of processors, number of cache lines, number of memory lines) and different sets of supported elementary ACE operations (currently, a representative subset of 15 operations), including an abstract operation that represents any other ACE operation. We handled the global requirements of the ACE specification using a constraint oriented programming style, i.e., by representing each global requirement as a dedicated process observing the global behavior and inhibiting incorrect executions. We also specified temporal properties expressing cache coherence, data integrity, and successful completion of each transaction. Note that the former property required to transform state-based properties into action-based ones, by adding information about the cache state to the actions executed by the cache.

In 2014, we exploited the formal model to improve the validation of the architecture under design at STMicroelectronics. In a first step, we studied the sanity (soundness and completeness) of an industrial interface verification unit, consisting of a list of so-called *formal checks*. After modeling each check in LNT, we used the BISIMULATOR tool to verify that each check is an overapproximation of the corresponding projection of the formal model. When we tried to establish that the parallel composition of all checks is an overapproximation of the projection of the formal model, we discovered a missing check (a particular channel did not occur in any of the checks).

In a second step, we studied the derivation of system level test cases, using a two-phase approach:

- In the first phase, abstract test cases were extracted automatically from the formal model using a prototype tool (see § 6.4). To circumvent the complexity of extracting test cases from the complete model, we proposed an iterative approach based on the automatic selection of a comprehensive set of interesting scenarios leading to LTSs of tractable size. The selection of the interesting scenarios relies on the counterexamples provided by the EVALUATOR model checker for the properties of coherence and data integrity.
- In the second phase, the abstract test cases were translated into the input format of an industrial test bench in charge of refining them into concrete test cases to be executed on the RTL (*Register Transfer Level*) description of the architecture under study. Experiments with manually translated abstract test cases led to the early discovery of bugs in commercial verification blocks, which could therefore be corrected before their use became critical in the development process.

The tests derived from the formal model increased the coverage of problematic features of some blocks used in the architecture. In particular, our approach was able to detect a limitation concerning data integrity 20 months before it was confirmed by classical methods, and our methodology provides all the scenarios triggering the limitation.

This work led to a publication accepted in an international conference [15]. Also, a large Petri net derived from our LNT model was provided as benchmark example for the Model Checking Contest.

6.5.2. Formal Verification of BPMN Processes

Participants: Radu Mateescu, Gwen Salaün, Lina Ye.

A business process is a set of structured, related activities that aims at fulfilling a specific organizational goal for a customer or market. An important metric when developing a business process is its degree of parallelism, i.e., the maximum number of tasks that are executable in parallel in that process. The degree of parallelism determines the peak demand on tasks, providing a valuable guide for the problem of resource allocation in business processes.

In 2014, we investigated how to automatically measure the degree of parallelism for business processes, described using the BPMN standard notation. To this aim, we defined a formal model for BPMN processes in terms of LTSs, which are obtained through an encoding in LNT. We then proposed an approach for automatically computing the degree of parallelism by using model checking of parameterized MCL formulas

and dichotomic search. We developed a prototype tool for automating this check and we applied it successfully to more than one hundred BPMN processes.

This work led to a publication in an international conference [16].

6.5.3. Stability of Asynchronously Communicating Systems

Participants: Gwen Salaün, Lina Ye.

Analyzing communicating systems that interact asynchronously via reliable FIFO buffers is an undecidable problem. A typical approach is to check whether the system is bounded, and if not, the corresponding state space can be made finite by limiting the presence of communication cycles in behavioral models or by fixing buffer sizes.

We followed a different approach, which aims at analyzing communicating systems without restricting them by imposing any arbitrary bounds. These systems are likely to be unbounded and therefore result in infinite state spaces. We introduce a notion of stability and prove that once the system is stable for a specific buffer bound (called stability bound), it remains stable whatever larger bounds are chosen for the buffers. This enables us to check certain properties on the (finite-state) system obtained for the stability bound and to ensure that the system will preserve them whatever larger bounds are used for buffers.

We have also proven that computing the stability bound is in general undecidable, and we proposed a semi-algorithm that successfully computes the stability bounds for many typical examples of communicating systems using heuristics and equivalence checking. This work is described in a research report [27].

6.5.4. Deployment and Reconfiguration Protocols for Cloud Applications

Participants: Rim Abid, Gwen Salaün.

In the context of the OpenCloudware project (see § 8.1.1.1), we collaborate with Noël de Palma and Fabienne Boyer (University Joseph Fourier), Xavier Etchevers and Thierry Coupaye (Orange Labs) in the field of cloud computing applications, which are complex distributed applications composed of interconnected software components running on distinct virtual machines (VMs). Setting up, (re)configuring, and monitoring these applications involve intricate management protocols, which fully automate these tasks while preserving application consistency as well as some key architectural invariants.

In 2014, we extended the specification of the self-deployment protocol to support VM failures. This led to a publication in an international conference [11], of which an extended version is under preparation for submission to an international journal.

We also worked on the dynamic reconfiguration of cloud applications. As a first attempt, we proposed to design this protocol using a publish-subscribe communication model [32]. In 2014, we improved the protocol to also support VM failures, and drastically validated the corresponding LNT specification using model checking. A paper presenting these results was submitted to an international journal. In parallel, we studied a version of this protocol where the different participants interact asynchronously via FIFO buffers. This led to a publication in an international conference [10].

As a new line of work, we undertook the study of controller synthesis techniques for the coordination of autonomic managers in asynchronous environments. Our approach relies on an encoding into LNT and on the application of several operations on automata (synchronous products, hiding, reduction) for synthesizing the corresponding controller using CADP tools. We also proposed automated techniques for generating Java code from an abstract model of the controller. For validation purposes, we applied our approach to real-world three-tier Web applications and showed that the introduction of a controller allows one to avoid erroneous situations due to the absence of coordination between autonomic managers.

6.5.5. Networks of Programmable Logic Controllers

Participants: Hubert Garavel, Fatma Jebali, Jingyan Jourdan-Lu, Frédéric Lang, Eric Léo, Radu Mateescu.

In the context of the Bluesky project (see § 8.1.2.1), we study the software applications embedded on the PLCs (*Programmable Logic Controllers*) manufactured by Crouzet Automatismes. One of the objectives of Bluesky is to enable the rigorous design of complex control applications running on several PLCs connected by a network. Such applications are instances of GALS (*Globally Asynchronous, Locally Synchronous*) systems composed of several synchronous automata embedded on individual PLCs, which interact asynchronously by exchanging messages. A formal analysis of these systems can be naturally achieved by using the formal languages and verification techniques developed in the field of asynchronous concurrency.

For describing the applications embedded on individual PLCs, Crouzet provides a dataflow language with graphical syntax and synchronous semantics, equipped with an ergonomic user-interface that facilitates the learning and use of the language by non-experts. To equip the PLC language of Crouzet with functionalities for automated verification, the solution adopted in Bluesky was to translate it into GRL (see § 6.1.3), which enables the connection to testing and verification tools covering the synchronous and asynchronous aspects.

In 2014, we have developed a set of GRL libraries implementing about 40 of the function blocks present in the PLC programming tool of Crouzet, to facilitate the integration of GRL in the PLC software design process. These function blocks include (among others) logic and comparison functions, timers, triggers, and counters. These GRL libraries have been used to model large applications provided by Crouzet. The GRL2LNT and GRL.OPEN tools (see § 6.1.3) provide a direct connection to all verification functionalities of CADP, in particular model checking and equivalence checking.

Regarding model checking, we have studied existing work in the verification of synchronous systems and GALS systems. We have identified a set of typical patterns of temporal properties (e.g., deadlocks, safety, liveness) relevant for GALS systems. These property patterns have been specified using MCL and checked on a set of feature-rich GRL examples using GRL.OPEN and EVALUATOR.

Regarding equivalence checking, the purpose is to compare the behavior of a GALS system with its *service*, which represents its desired observable behavior, modulo a suitable equivalence relation. We have studied existing work in equivalence checking for GALS systems and we have investigated how to formally define the expected service of a GALS system at the appropriate level of expressiveness and abstraction, which requires a careful identification of the observable actions corresponding to the interactions between the GALS system and its physical environment. We have modeled several examples of GALS systems in GRL, and experimented the definition of appropriate services and their usage for equivalence checking by means of GRL.OPEN and BISIMULATOR.

The validation approach we promote, together with our colleagues from the LCIS laboratory (Valence) in the Bluesky project, led to a common publication in a national conference [21].

6.5.6. *EnergyBus Standard for Connecting Electric Components*

Participants: Hubert Garavel, Wendelin Serwe.

The EnergyBus [8] is an upcoming industrial standard for electric power transmission and management, based on the CANopen field bus. It is developed by a consortium assembling all major industrial players (such as Bosch, Panasonic, and Emtas) in the area of light electric vehicles (LEV); their intention is to ensure interoperability between all electric LEV components. At the core of this initiative is a universal plug integrating a CAN-Bus [9] with switchable power lines. The central and innovative role of the EnergyBus is to manage the safe electricity access and distribution inside an EnergyBus network.

In the framework of the European FP7 project SENSATION (see § 8.2.1.1) a formal specification in LNT of the main EnergyBus protocols is being developed by Alexander Graf-Brill and Holger Hermanns at Saarland University [48], with the active collaboration of CONVECS.

In 2014, our joint work with Saarland University on the modeling, verification, and test case generation for the EnergyBus standard led to a common publication [13].

[8] http://www.energybus.org
[9] http://www.can-cia.org

6.5.7. Graphical User-Interfaces and Plasticity

Participants: Hubert Garavel, Frédéric Lang, Raquel Oliveira.

In the context of the Connexion project (see § 8.1.1.2) and in close collaboration with Gaëlle Calvary, Eric Ceret, and Sophie Dupuy-Chessa (IIHM team of the LIG laboratory), we study the formal description and validation of graphical user-interfaces using the most recent features of the CADP toolbox. The case study assigned to LIG in this project is a prototype graphical user-interface [36] designed to provide human operators with an overview of a running nuclear plant. The main goal of the system is to inform the operators about alarms resulting from faults, disturbances, or unexpected events in the plant. Contrary to conventional control rooms, which employ large desks and dedicated hardware panels for supervision, this new-generation interface uses standard computer hardware (i.e., smaller screen(s), keyboard, and mouse), thus raising challenging questions on how to best provide synthetic views of the plant status. Another challenge is to introduce plasticity in such interface, so as to enable several supervision operators, including mobile ones outside of the control room, to get accurate information in real time.

We formally specified this new-generation interface in LNT, encompassing not only the usual components traditionally found in graphical user-interfaces, but also a model of the physical world (namely, a nuclear reactor with various fault scenarios) and a cognitive model of a human operator in charge of supervising the plant. Also, several desirable properties of the interface have been expressed in MCL and verified on the LNT model using CADP. This led to a publication in an international conference [17].

In 2014, we continued our activity along several directions. The LNT specification was matured in various respects. As a result of several interactions with EDF, the specification was enhanced with a more realistic representation of the plant (currently $5,358$ lines of LNT code). Besides, new desirable properties of the user-interfaces emerged with the evolution of the formal model, making a total of seven complex properties formally specified in MCL.

We initiated an integration of our formal model with an industrial control room prototype, provided by a partner in the project. To this aim, several improvements were done in the formal specification, and the integration is currently in progress.

We started to address the introduction of plasticity in the formal specification, a challenge that was identified in 2013. Plasticity is the capacity of a user-interface to withstand variations of its context of use (i.e., platform, user, environment) while preserving usability. We proposed two approaches to introduce plasticity in the analysis. The first one introduces in the formal model a representation of a plasticity engine (responsible for user-interfaces adaptation) and applies model checking to verify its properties. The second approach consists in formally specifying several versions of the user-interfaces, derived from adaptation, and applying equivalence checking to verify similarity relations on the user-interface models.

6.5.8. Fault-Tolerant Routing for Network-on-Chip Architectures

Participant: Wendelin Serwe.

Fault-tolerant architectures provide adaptivity for on-chip communications, but also increase the complexity of the design, so that formal verification techniques are needed to check their correctness. In collaboration with Chris Myers and Zhen Zhang (University of Utah, USA), we studied an extension of the link-fault tolerant Network-on-Chip (NoC) architecture introduced by Wu *et al* [67] that supports multiflit wormhole routing.

To keep the state space manageable, the formal LNT model of the routing algorithm was constructed in several steps, applying different abstractions (structural and related to data). This modeling process led to several insights. First, it led to the discovery of a package leakage path that could lead to the complete loss of a packet and a deadlock. This error in the design of an arbiter was corrected in the subsequent models. Second, a buffering capacity in an arbiter was found to be crucial; this insight also led to a redesign of the arbiters. The resultant changes on the router and arbiter models uncovered interesting symmetries. Finally, we studied how deadlock freedom and tolerance of a single-link fault can be verified for a NoC architecture.

This work led to a publication in an international conference [20].

6.5.8.1. Other Case Studies

The demo examples of CADP, which have been progressively accumulated since the origins of the toolbox, are a showcase for the multiple capabilities of CADP, as well as a test bed to assess the new features of the toolbox. In 2014, the effort to maintain and enhance these demos has been pursued. The progressive migration to LNT has continued, by translating certain demos from LOTOS to LNT. Many demos have been enriched with value-passing temporal formulas that illustrate the conciseness and expressiveness of MCL and the capabilities of the EVALUATOR 4 model checker. Finally, many demos have been shortened and made more readable by using the new features of SVL, especially the "`property`" and "`|=`" statements that allow formulas to be gathered in a single SVL file rather than disseminated in a collection of MCL or XTL files.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Grants with Industry

Participants: Hubert Garavel, Abderahman Kriouile, Radu Mateescu, Wendelin Serwe.

Abderahman Kriouile is supported by a CIFRE PhD grant (from March 2012 to March 2015) from STMicroelectronics (Grenoble) on the verification of cache coherency in systems on chip (see § 6.5.1), under the supervision of Guilhem Barthes (STMicroelectronics), Christophe Chevallaz (STMicroelectronics), Grégory Faux (STMicroelectronics), Radu Mateescu (CONVECS), Wendelin Serwe (CONVECS), and Massimo Zendri (STMicroelectronics).

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. FSN (Fonds national pour la Société Numérique)

8.1.1.1. OpenCloudware

Participants: Rim Abid, Hugues Evrard, Frédéric Lang, Gwen Salaün [correspondent], Lina Ye.

OpenCloudware [10] is a project funded by the FSN. The project is led by France Telecom / Orange Labs (Meylan, France) and involves 18 partners (among which Bull, OW2, Thalès, Inria, etc.). OpenCloudware aims at providing an open software platform enabling the development, deployment and administration of cloud applications. The objective is to provide a set of integrated software components for: (i) modeling distributed applications to be executed on cloud computing infrastructures; (ii) developing and constructing multi-tier virtualized applications; and (iii) deploying and administrating these applications (PaaS platform) possibly on multi-IaaS infrastructures.

OpenCloudware started in January 2012 for three years and nine months. The main contributions of CONVECS to OpenCloudware (see § 6.5.4) are the formal specification of the models, architectures, and protocols (self-deployment, dynamic reconfiguration, self-repair, etc.) underlying the OpenCloudware platform, the automated generation of code from these specifications for rapid prototyping purposes, and the formal verification of the aforementioned protocols.

8.1.1.2. Connexion

Participants: Hubert Garavel [correspondent], Frédéric Lang, Raquel Oliveira.

[10] http://www.opencloudware.org

Connexion [11] (*COntrôle commande Nucléaire Numérique pour l'EXport et la rénovatION*) is a project funded by the FSN, within the second call for projects "*Investissements d'Avenir — Briques génériques du logiciel embarqué*". The project, led by EDF and supported by the *Pôles de compétitivité* Minalogic, Systematic, and *Pôle Nucléaire Bourgogne*, involves many industrial and academic partners, namely All4Tech, Alstom Power, ArevA, Atos Worldgrid, CEA-LIST, CNRS/CRAN, Corys Tess, ENS Cachan, Esterel Technologies, Inria, LIG, Predict, and Rolls-Royce. Connexion aims at proposing and validating an innovative architecture dedicated to the design and implementation of control systems for new nuclear power plants in France and abroad.

Connexion started in April 2012 for four years. In this project, CONVECS will assist another LIG team, IIHM, in specifying human-machine interfaces formally using the LNT language and in verifying them using CADP (see § 6.5.7).

8.1.2. Competitivity Clusters

8.1.2.1. Bluesky for I-Automation

Participants: Hubert Garavel, Fatma Jebali, Jingyan Jourdan-Lu, Frédéric Lang, Eric Léo, Radu Mateescu [correspondent].

Bluesky for I-Automation is a project funded by the FUI (*Fonds Unique Interministériel*) within the *Pôle de Compétitivité* Minalogic. The project, led by Crouzet Automatismes (Valence), involves the SMEs (*Small and Medium Enterprises*) Motwin and VerticalM2M, the LCIS laboratory of Grenoble INP, and CONVECS. Bluesky aims at bringing closer the design of automation applications and the Internet of things by providing an integrated solution consisting of hardware, software, and services enabling a distributed, Internet-based design and development of automation systems. The automation systems targeted by the project are networks of programmable logic controllers, which belong to the class of GALS (*Globally Asynchronous, Locally Synchronous*) systems.

Bluesky started in September 2012 for three years. The main contributions of CONVECS to Bluesky (see § 6.1.3 and § 6.5.5) are the definition of GRL, the formal pivot language for describing the asynchronous behavior of logic controller networks, and the automated verification of the behavior using compositional model checking and equivalence checking techniques.

8.1.3. Other National Collaborations

Additionally, we collaborated in 2014 with the following Inria project-teams:

- OASIS (Inria Sophia-Antipolis – Méditerranée): Eric Madelaine and Ludovic Henrio,
- ESTASYS (Inria Rennes – Bretagne Atlantique): Kevin Corre and Axel Legay,
- MEXICO (Inria Saclay – Île-de-France): Alban Linard.

Beyond Inria, we had sustained scientific relations with the following researchers:

- Gaëlle Calvary and Sophie Dupuy-Chessa (LIG, Grenoble),
- Fabrice Kordon and Lom Messan Hillah (LIP6, Paris),
- Alexandre Hamez (ISAE, Toulouse),
- Noël De Palma and Fabienne Boyer (LIG, Grenoble),
- Xavier Etchevers (Orange Labs, Meylan),
- Matthias Güdemann (Systerel, Aix-en-Provence),
- Meriem Ouederni (IRIT, Toulouse),
- Christophe Deleuze, Ioannis Parissis, and Mouna Tka Mnad (LCIS, Valence),
- Pascal Poizat (LIP6, Paris).

[11] http://www.cluster-connexion.fr

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. SENSATION

Participants: Hubert Garavel [correspondent], Radu Mateescu, Jose Ignacio Requeno, Wendelin Serwe.

SENSATION [12] (*Self ENergy-Supporting Autonomous computaTION*) is a European project no. 318490 funded by the FP7-ICT-11-8 programme. It gathers 9 participants: Inria (ESTASYS and CONVECS project-teams), Aalborg University (Denmark), RWTH Aachen and Saarland University (Germany), University of Twente (The Netherlands), GomSpace (Denmark), and Recore Systems (The Netherlands). The main goal of SENSATION is to increase the scale of systems that are self-supporting by balancing energy harvesting and consumption up to the level of complete products. In order to build such Energy Centric Systems, embedded system designers face the quest for optimal performance within acceptable reliability and tight energy bounds. Programming systems that reconfigure themselves in view of changing tasks, resources, errors, and available energy is a demanding challenge.

SENSATION started on October 1st, 2012 for three years. CONVECS contributes to the project regarding the extension of formal languages with quantitative aspects (see § 6.3.1), studying common semantic models for quantitative analysis, and applying formal modeling and analysis to the case studies provided by the industrial partners (see § 6.5.6).

8.2.2. Collaborations with Major European Organizations

The CONVECS project-team is member of the FMICS (*Formal Methods for Industrial Critical Systems*) working group of ERCIM [13]. R. Mateescu was the chairman of the FMICS working group until November 1st, 2014. H. Garavel is member of the FMICS board, in charge of dissemination actions.

H. Garavel was appointed to a new Working Group within Informatics Europe: "*Parallel Computing (Supercomputing) Education in Europe: State-of-Art*". This is a relatively small working group (about 10 people) with the following missions: to show the need for urgent changes in higher education in the area of computational sciences, to compose a survey of the current landscape of parallel computing and supercomputing education in Europe with respect to different universities and countries, and to prepare a set of recommendations on how to bring ideas of parallel computing and supercomputing into higher educational systems of European countries.

8.2.3. Other European Collaborations

In addition to our partners in aforementioned contractual collaborations, we had scientific relations in 2014 with several European universities and research centers, including:

- Saarland University (Alexander Graf-Brill, Holger Hermanns, and Felix Freiberger),
- RWTH Aachen (Joost-Pieter Katoen and Xiaoxiao Yang),
- Oxford University (Ernst-Moritz Hahn and Marta Kwiatkowska),
- University of Birmingham (Dave Parker),
- Technical University of Eindhoven (Anton Wijs),
- University of Twente (Marieke Huisman and Jaco van de Pol),
- University of Málaga (Carlos Canal, Francisco Duran and Ernesto Pimentel), and
- Brandenburg University of Technology Cottbus - Senftenberg (Monika Heiner).

Our partnership with Saarland University was sustained by the Humboldt Forschungspreis received by H. Garavel, who continued his regular visits to Saarland University.

[12] http://sensation-project.eu/
[13] http://fmics.inria.fr

8.3. International Initiatives

8.3.1. Inria International Labs

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.

8.3.2. Other International Collaborations

In 2014, we had scientific relations with several universities abroad, including:

- University of California at Santa Barbara, USA (Tevfik Bultan),
- University of Utah, USA (Chris Myers and Zhen Zhang), and
- Universidad Nacional de Cordoba, Argentina (Pedro d'Argenio).

8.4. International Research Visitors

8.4.1. Visits of International Scientists

- Alexandre Hamez (ISAE, Toulouse) visited us on March 26-28, 2014. He gave a seminar entitled "*Symbolic Model Checking and Hierarchical Set Decision Diagrams*".
- Chris Myers (University of Utah, USA) visited us from July 7–11, 2014. He gave a talk entitled "*Genetic Design Automation*" on July 8, 2014.
- The annual CONVECS seminar was held in Herbelon (France) on June 23-25, 2014. The following invited scientists attended the seminar:
  - Laurence Pierre (TIMA, Grenoble, France) gave on June 23, 2014 a talk entitled "*Verification of Correctness and Safety Requirements for SoC Models*".
  - Matthias Güdemann (Systerel, Aix-en-Provence) gave on June 24, 2014 a talk entitled "*Industrial Formal Methods*".
  - Lom Messan Hillah (LIP6, Paris) gave on June 25, 2014 a talk entitled "*Formal Methods in Model-Driven Development and Model-Driven Development in Formal Methods: Practice Makes a Better Bridge*".

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. General chair, scientific chair

- G. Salaün was general chair for SEFM'2014 (*12th International Conference on Software Engineering and Formal Methods*), Grenoble, France, September 1–5, 2014.

9.1.1.2. Member of the organizing committee

- L. Ye was publicity chair for SEFM'2014. R. Abid and H. Evrard were members of the organizing committee for SEFM'2014.
- W. Serwe was publicity chair for FMICS'2014 (*19th International Workshop on Formal Methods for Industrial Critical Systems*), Florence, Italy, September 11–12, 2014.

9.1.2. Scientific events selection

9.1.2.1. Chair of conference program committee

- G. Salaün was programme committee chair for SEFM'2014.

- F. Lang was programme committee chair for FMICS'2014.

9.1.2.2. Member of the conference program committee

- G. Salaün was programme committee member for GRAPHITE'2014 (*3rd Workshop on Graph Inspection and Traversal Engineering*), Grenoble, France, April 5, 2014.
- G. Salaün was programme committee member for ORCHOR'2014 (*International Workshop on Service Orchestration and Choreography for the Future Internet*), Anchorage, Alaska, June 30, 2014.
- G. Salaün was programme committee member for WWV'2014 (*International Workshop on Automated Specification and Verification of Web Systems*), Vienna, Austria, July 18, 2014.
- H. Garavel was programme committee member for SEFM'2014.
- G. Salaün was programme committee member for FOCLASA'2014 (*13th International Workshop on Foundations of Coordination Languages and Self-Adaptive Systems*), Rome, Italy, September 6, 2014.
- G. Salaün was programme committee member for FACS'2014 (*11th International Symposium on Formal Aspects of Component Software*), Bertinoro, Italy, September 10–12, 2014.
- R. Mateescu and W. Serwe were programme committee members for FMICS'2014.
- W. Serwe was a program committee member for the track on "Prototyping, Validation, Verification, Modeling and Simulation" of VLSI-SOC'2014 (*22nd IFIP/IEEE International Conference on Very Large Scale Integration*), Playa del Carmen, Mexico, October 6–8, 2014.

9.1.2.3. Reviewer

- H. Evrard reviewed articles submitted to FOCLASA'2014 and TACAS'2015 (*21th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*).
- F. Jebali reviewed articles submitted to AVOCS'2014 (*14th International Workshop on Automated Verification of Critical Systems*).
- A. Kriouile reviewed articles submitted to SAC-SVT'2015 (*30th ACM/SIGAPP Symposium on Applied Computing — Software Verification Track*).
- F. Lang reviewed articles submitted to TACAS'2015.
- R. Mateescu reviewed articles submitted to AVOCS'2014, SEFM'2014, and TACAS'2015.
- G. Salaün reviewed articles submitted to WETICE-PASCS'2014 (*23rd IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises — Track on Privacy and Accountability for Software and Cloud Services*).
- W. Serwe reviewed articles submitted to AVOCS'2014, DATE'2014 (*Design, Automation & Test in Europe*), and TACAS'2015.
- L. Ye reviewed articles submitted to SEFM'2014, VALID'2014 (*6th International Conference on Advances in System Testing and Validation Lifecycle*), SAC-SVT'2015, and FSEN'2015 (*6th IPM International Conference on Fundamentals of Software Engineering*).

9.1.3. Journal

9.1.3.1. Member of the editorial board

- H. Garavel is an editorial board member of STTT (*Springer International Journal on Software Tools for Technology Transfer*).
- G. Salaün is an editorial board member of SOCA (*Springer International Journal on Service Oriented Computing and Applications*).

9.1.3.2. Reviewer

- R. Mateescu was reviewer for ACM TECS (*Transactions on Embedded Computing Systems*) and STTT.

- J. I. Requeno was reviewer for IEEE TC (*Transactions on Computers*).
- G. Salaün was reviewer for JLAMP (*Journal of Logical and Algebraic Methods in Programming*), JISA (*Journal of Internet Services and Applications*), IEEE TSE (*Transactions on Software Engineering*), and IJWET (*International Journal of Web Engineering and Technology*).

9.1.4. Software Dissemination and Internet Visibility

The CONVECS project-team distributes several software tools: the CADP toolbox (see § 5.1), the TRAIAN compiler (see § 5.2), the PIC2LNT translator (see § 5.3), and the PMC model checker (see § 5.4).

In 2014, the main facts are the following:

- We prepared and distributed twelve successive versions (2014-a to 2014-l) of CADP.
- We were requested to grant CADP licenses for 425 different computers in the world.

The CONVECS Web site [14] was updated with scientific contents, announcements, publications, etc.

By the end of December 2014, the CADP forum [15], opened in 2007 for discussions regarding the CADP toolbox, had over 320 registered users and over 1570 messages had been exchanged.

Also, five large Petri nets derived from our LNT models were provided as benchmark examples for the 2014 edition of the Model Checking Contest.

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

- The IMCReo tool to reason about QoS in Stochastic Reo connectors [58], [59]
- The Alvis virtual machine to execute formal models in the Alvis language [57]
- The Tomte tool for active learning of Mealy machines [30]
- The Vercors platform for verifying the correct composition of distributed components [52]
- The TLM.open tool connecting SystemC/TLM models to CADP [50]
- The REFINER tool for formal Verification of Model Transformations [65], [66]
- The ELOTON development environment for LOTOS specifications [68]

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

- Model-based specification, implementation and testing of a software bus [61]
- Protocol conformance testing using active learning of Mealy machines [31]
- High-performance fractal coherence [64], [63]
- Verification of models reverse engineered from smart-card readers [34]

9.1.5. Awards and Distinctions

H. Garavel is an invited professor at Saarland University (Germany) as a holder of the Gay-Lussac Humboldt Prize.

[14] http://convecs.inria.fr
[15] http://cadp.inria.fr/forum.html

9.1.6. Lectures and Invited Conferences

- H. Garavel gave an invited talk entitled "*Trois décennies de réussite en méthodes formelles*" at the Journées MFDL/MTV2 (*Méthodes Formelles pour le Développement de Logiciels/Méthodes de Test pour la Vérification et la Validation*) held on January 14, 2014 in Grenoble, France.
- H. Garavel and R. Mateescu gave a talk entitled "*Analyse de traces par model checking / Trace Analysis using Model Checking*" at the PIMLIG seminar "*Traces dans tous leurs états*" held on January 22, 2014 in Grenoble, France.
- G. Salaün gave a talk entitled "*On the Verification of Asynchronously Communicating Systems*" on February 18, 2014 in Málaga, Spain.
- G. Salaün gave a talk entitled "*Verification of Asynchronously Communicating Systems*" on October 2nd, 2014 in Clermont-Ferrand, France.
- R. Mateescu gave an invited talk entitled "*Walking Back and Forth in Labelled Transition Systems*" at the GRAPHITE'2014 workshop held on April 5, 2014, in Grenoble, France.
- H. Garavel gave a talk entitled "*Benchmarks and Benchmarking: The Model Checking Contest*" at the 20 years of TACAS workshop and celebration held on April 8, 2014 in Grenoble, France.
- H. Garavel gave a talk entitled "*Real Time without Clocks*" on June 6, 2014 at Saarland University, Saarbrücken, Germany.
- H. Garavel gave a talk entitled "*Reconciling Concurrency Theory with Other Branches of Computer Science*" at the *IFIP WG 1.8 Research Seminar on Open Problems in Concurrency Theory* held on June 18–21, 2014 in Bertinoro (Forlì), Italy.
- R. Mateescu gave an invited talk entitled "*Mu-Calculus Property-Dependent Reductions for Divergence-Sensitive Branching Bisimilarity*" at the WS-FMDS'2014 workshop held on September 2, 2014 in Grenoble, France.
- R. Mateescu gave a talk entitled "*Two Decades of Formal Methods for Industrial Critical Systems*" at the ERCIM 25th anniversary held on October 23–24, 2014, in Pisa, Italy.
- F. Jebali gave a talk entitled "*A Step Towards Reconciling GALS Industrial Design with Formal Verification*" on September 11, 2014 at the 11th LASER Summer School on Software Engineering held on September 7–13, 2014 in Elba Island, Italy.
- F. Lang gave a talk entitled "*Partial Model Checking Using Labelled Transition Systems and Boolean Equation Systems*" to a panel of 1st year students of Ecole Normale Supérieure de Cachan during their visit to the LIG laboratory on December 9, 2014.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

CONVECS is a host team for the computer science master entitled "*Mathématiques, Informatique, spécialité : Systèmes et Logiciels*", common to Grenoble INP and University Joseph Fourier.

In 2014, we carried out the following teaching activities:

- G. Salaün is co-responsible of the ISI (*Ingéniérie des Systèmes d'Information*) department of ENSIMAG since September 1, 2011.
- F. Lang and W. Serwe gave a course on "*Spécification et vérification de systèmes concurrents et temps-réel*" to the third year computer science engineering students of ENSIMAG (18 hours).
- G. Salaün gave a course on "*Programmation orientée-objet*" to the second year computer science engineering students of ENSIMAG (18 hours).
- G. Salaün gave a course on "*Théorie des langages*" to the first year computer science engineering students of ENSIMAG (18 hours).
- R. Oliveira served as a teaching assistant in a course on "*Bases de Données*", given by Aurélien Faravelon to the first year computer science engineering students of IUT2 (25 hours).
- R. Oliveira served as a teaching assistant in a course on "*Algorithmique et programmation*", given by Vanda Luengo to the first year computer science engineering students of IUT (40 hours).

9.2.2. Juries

- G. Salaün was external reviewer for José Antonio Mateo Cortés's PhD thesis, entitled "*Verification and Validation of Web Service Compositions using Formal Methods*", defended at Universidad de Castilla La Mancha, Spain, on July 1st, 2014.
- G. Salaün was external reviewer for Diana Allam's PhD thesis, entitled "*Enforcing Loose Coupling and Substitution Principles in an Object-Oriented Framework for Web Services*", defended at Ecole des Mines, Nantes, France, on July 10, 2014.
- R. Mateescu was external reviewer for Ala Eddine Ben Salem's PhD thesis, entitled "*Improving the Model Checking of Stutter-Invariant LTL Properties*", defended at Université Pierre et Marie Curie, Paris, France, on September 25, 2014.
- G. Salaün was external reviewer for Lakhdar Akroun's PhD thesis, entitled "*Décidabilité et complexité de la relation de simulation des services Web orientés données*", defended at Université Blaise Pascal, Clermont-Ferrand, France, on December 8, 2014.
- G. Salaün was external reviewer for Dimitris Vekris's PhD thesis, entitled "*Vérification de spécifications EB3 à l'aide de techniques de model checking*", defended at Université Paris-Est Créteil, France, on December 10, 2014.

9.3. Popularization

H. Garavel participates to the committee in charge of organizing the Aerospace Valley series of industrial conferences on formal methods. The third conference [16], devoted to theorem proving, held on February 4 in Toulouse and retransmitted by video-conference in Grenoble, attracted over 100 participants from industry and academia.

The fourth conference [17], devoted to model checking, held on October 16 in Toulouse and retransmitted by video-conference in Grenoble and Saclay, attracted over 120 participants from industry and academia. H. Garavel gave a talk entitled "*Présentation de l'outil CADP*". A. Kriouile gave a talk entitled "*Application de CADP à la vérification de matériel*". R. Mateescu gave a talk entitled "*Introduction au model checking*".

9.4. Miscellaneous Activities

H. Evrard is a member of the council of the MSTII doctoral school.

H. Evrard and G. Salaün are members of the council of the LIG laboratory.

H. Garavel is a member of the LIG commission in charge of preparing candidates selected for recruitment interviews at CNRS.

H. Garavel is a member of the operational committee of the EMSOC cluster ("Embedded System on Chip") within the Minalogic "*pôle de compétitivité mondial*".

H. Garavel was a reviewer for various ongoing ANR (*Agence Nationale de la Recherche*) pre-propositions and projects evaluated in 2014.

F. Lang is a member of the "*commission du développement technologique*", which is in charge of selecting R&D projects for Inria Grenoble - Rhône-Alpes.

E. Léo and W. Serwe are members of the "*comité de centre*" of Inria Grenoble - Rhône-Alpes.

R. Mateescu is the correspondent of the "*Département des Partenariats Européens*" for Inria Grenoble - Rhône-Alpes.

G. Salaün is a member of the scientific council of Grenoble INP (*Conseil scientifique de l'institut*).

G. Salaün is a member of the MSTIC council of Grenoble-Alpes University (*Conseil de pôle MSTIC*).

[16] http://www.inria.fr/centre/grenoble/agenda/forum-methodes-formelles2
[17] http://projects.laas.fr/IFSE/FMF/J4/index.html

W. Serwe is "*chargé de mission*" for the scientific axis *Formal Methods, Models, and Languages* of the LIG laboratory.

F. Lang is (together with Laurent Lefèvre from the AVALON Inria project-team) correspondent in charge of the Inria Activity Report at Inria Grenoble - Rhône-Alpes.

10. Bibliography

Major publications by the team in recent years

[1] H. GARAVEL, F. LANG, R. MATEESCU, W. SERWE. *CADP 2011: A Toolbox for the Construction and Analysis of Distributed Processes*, in "International Journal on Software Tools for Technology Transfer", 2013, vol. 15, n° 2, pp. 89-107 [*DOI :* 10.1007/s10009-012-0244-z], http://hal.inria.fr/hal-00715056

[2] F. LANG, R. MATEESCU. *Partial Model Checking using Networks of Labelled Transition Systems and Boolean Equation Systems*, in "Logical Methods in Computer Science", October 2013, vol. 9, n° 4, pp. 1–32, http://hal.inria.fr/hal-00872181/en

[3] R. MATEESCU, P. POIZAT, G. SALAÜN. *Adaptation of Service Protocols using Process Algebra and On-the-Fly Reduction Techniques*, in "IEEE Transactions on Software Engineering", 2012 [*DOI :* 10.1109/TSE.2011.62], http://hal.inria.fr/hal-00717252

[4] R. MATEESCU, W. SERWE. *Model Checking and Performance Evaluation with CADP Illustrated on Shared-Memory Mutual Exclusion Protocols*, in "Science of Computer Programming", February 2012 [*DOI :* 10.1016/j.scico.2012.01.003], http://hal.inria.fr/hal-00671321

[5] R. MATEESCU, A. WIJS. *Sequential and Distributed On-the-Fly Computation of Weak Tau-Confluence*, in "Science of Computer Programming", 2012, vol. 77, n° 10–11, pp. 1075–1094, http://hal.inria.fr/hal-00676451/en

[6] G. SALAÜN, T. BULTAN, N. ROOHI. *Realizability of Choreographies using Process Algebra Encodings*, in "IEEE Transactions on Services Computing", August 2012, vol. 5, n° 3, pp. 290-304, http://hal.inria.fr/hal-00726448

Publications of the year

Articles in International Peer-Reviewed Journals

[7] E. LANTREIBECQ, W. SERWE. *Formal Analysis of a Hardware Dynamic Task Dispatcher with CADP*, in "Science of Computer Programming", 2014, vol. 80, pp. 130-149 [*DOI :* 10.1016/j.scico.2013.01.003], https://hal.inria.fr/hal-00782069

[8] R. MATEESCU, A. WIJS. *Property-Dependent Reductions Adequate with Divergence-Sensitive Branching Bisimilarity*, in "Science of Computer Programming", April 2014 [*DOI :* 10.1016/j.scico.2014.04.004], https://hal.inria.fr/hal-01016922

International Conferences with Proceedings

[9] C. CANAL, G. SALAÜN. *Adaptation of Asynchronously Communicating Software*, in "12th International Conference on Service Oriented Computing (ICSOC 2014)", Paris, France, November 2014, https://hal.inria.fr/hal-01053682

[10] F. DURÁN, G. SALAÜN. *Robust Reconfiguration of Cloud Applications*, in "The 17th International ACM Sigsoft Symposium on Component-Based Software Engineering (CBSE 2014)", Lille, France, June 2014, https://hal.inria.fr/hal-01016401

[11] X. ETCHEVERS, G. SALAÜN, F. BOYER, T. COUPAYE, N. DE PALMA. *Reliable Self-Deployment of Cloud Applications*, in "SAC 2014 - 29th ACM Symposium on Applied Computing", Gyeongju, South Korea, March 2014, https://hal.inria.fr/hal-00934042

[12] H. EVRARD, F. LANG. *Automatic Distributed Code Generation from Formal Models of Asynchronous Concurrent Processes*, in "PDP - 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing", Turku, Finland, March 2015, https://hal.inria.fr/hal-01086522

[13] A. GRAF-BRILL, H. HERMANNS, H. GARAVEL. *A Model-Based Certification Framework for the EnergyBus Standard*, in "Proceedings of the 34th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems - FORTE'14", Berlin, Germany, E. ABRAHAM, C. PALAMIDESSI (editors), Lecture Notes in Computer Science, Springer, June 2014, vol. 8461, pp. 84-99 [*DOI :* 10.1007/978-3-662-43613-4_6], https://hal.inria.fr/hal-01098360

[14] F. JEBALI, F. LANG, R. MATEESCU. *GRL: A Specification Language for Globally Asynchronous Locally Synchronous Systems*, in "Formal Methods and Software Engineering", Luxembourg, November 2014, vol. 8829, pp. 219 - 234 [*DOI :* 10.1007/978-3-319-11737-9_15], https://hal.inria.fr/hal-01082348

[15] A. KRIOUILE, W. SERWE. *Using a Formal Model to Improve Verification of a Cache-Coherent System-on-Chip*, in "21th International Conference on Tools and Algorithms for the Construction and Analysis of Systems", London, UK, France, April 2015, https://hal.inria.fr/hal-01104747

[16] R. MATEESCU, G. SALAÜN, L. YE. *Quantifying the Parallelism in BPMN Processes using Model Checking*, in "The 17th International ACM Sigsoft Symposium on Component-Based Software Engineering (CBSE 2014)", Lille, France, June 2014, https://hal.inria.fr/hal-01016412

[17] R. OLIVEIRA, S. DUPUY-CHESSA, C. GAËLLE. *Formal verification of UI using the power of a recent tool suite*, in "EICS 2014 : Proceedings of the 2014 ACM SIGCHI symposium on Engineering Interactive Computing Systems", Florence, Italy, June 2014, pp. 235-240 [*DOI :* 10.1145/2607023.2610280], https://hal.inria.fr/hal-01110183

[18] M. OUEDERNI, G. SALAÜN, J. CÁMARA, E. PIMENTEL. *Comparator: A Tool for Quantifying Behavioural Compatibility*, in "FASE 2014 - 17th International Conference on Fundamental Approaches to Software Engineering", Grenoble, France, April 2014, https://hal.inria.fr/hal-00934057

[19] G. SALAÜN, L. YE. *Debugging Process Algebra Specifications*, in "VMCAI 2015", Mumbai, India, Springer, January 2015, vol. 8931, 18 p. [*DOI :* 10.1007/978-3-662-46081-8_14], https://hal.inria.fr/hal-01087505

[20] Z. ZHANG, W. SERWE, J. WU, T. YONEDA, H. ZHENG, C. MYERS. *Formal Analysis of a Fault-Tolerant Routing Algorithm for a Network-on-Chip*, in "FMICS - 19th International Conference on Formal Methods for Industrial Critical Systems", Florence, Italy, F. LANG, F. FLAMMINI (editors), Lecture Notes in Computer Science, Springer, September 2014, vol. 8718, pp. 48-62 [*DOI :* 10.1007/978-3-319-10702-8_4], https://hal.inria.fr/hal-01064829

National Conferences with Proceedings

[21] F. JEBALI, M. TKA MNAD, C. DELEUZE, F. LANG, R. MATEESCU, I. PARISSIS. *Modélisation et validation formelle de systèmes globalement asynchrones et localement synchrones*, in "Approches Formelles dans l'Assistance au Développement de Logiciels", Paris, France, June 2014, pp. 97–102, http://hal.univ-grenoble-alpes.fr/hal-01007674

Books or Proceedings Editing

[22] M. R. MOUSAVI, G. SALAÜN (editors). *Preface: Special Section on Foundations of Coordination Languages and Software Architectures (Selected Papers from FOCLASA'10)*, Science of Computer Programming, Elsevier, February 2014, vol. 80, 2 p. [*DOI :* 10.1016/J.SCICO.2012.03.003], https://hal.inria.fr/hal-00919799

[23] C. PASAREANU, G. SALAÜN (editors). *Special Issue on Formal Aspects of Component Software (Selected Papers from FACS'12)*, Elsevier, October 2014, 3 p. , https://hal.inria.fr/hal-01016471

[24] G. SALAÜN, B. SCHÄTZ (editors). *Preface: Special Section on Formal Methods for Industrial Critical Systems (Selected Papers from FMICS'11)*, Science of Computer Programming, Elsevier, February 2014, vol. 80, 2 p. [*DOI :* 10.1016/J.SCICO.2013.01.008], https://hal.inria.fr/hal-00919803

Research Reports

[25] F. JEBALI, F. LANG, R. MATEESCU. *GRL: A Specification Language for Globally Asynchronous Locally Synchronous Systems (Syntax and Formal Semantics)*, April 2014, n° RR-8527, https://hal.inria.fr/hal-00983711

[26] M. OUEDERNI, U. FAHRENBERG, A. LEGAY, G. SALAÜN. *Flooding-Based Algorithm for Behavioural Compatibility Measuring*, Inria Rennes, 2014, https://hal.inria.fr/hal-01088157

[27] G. SALAÜN, L. YE. *Stability of Asynchronously Communicating Systems*, July 2014, n° RR-8561, https://hal.inria.fr/hal-01020777

Other Publications

[28] F. JEBALI, F. LANG, E. LÉO, R. MATEESCU. *Formal Modeling and Verification of GALS Systems Using GRL and CADP*, November 2014, https://hal.inria.fr/hal-01082950

References in notes

[29] *AMBA AXI and ACE Protocol Specification*, ARM IHI 0022D (ID102711), ARM, October 22 2011

[30] F. AARTS. *Tomte: Bridging the Gap Between Active Learning and Real-World Systems*, Radboud University Nijmegen, The Netherlands, October 2014

[31] F. AARTS, H. KUPPENS, J. TRETMANS, F. W. VAANDRAGER, S. VERWER. *Improving Active Mealy Machine Learning for Protocol Conformance Testing*, in "Machine Learning", July 2014, vol. 96, n° 1–2, pp. 189–224

[32] R. ABID, G. SALAÜN, F. BONGIOVANNI, N. DE PALMA. *Verification of a Dynamic Management Protocol for Cloud Applications*, in "11th International Symposium, ATVA 2013", Hanoi, Viet Nam, Dang Van Hung and Mizuhito Ogawa, 2013, vol. 8172, pp. 178-192 [*DOI :* 10.1007/978-3-319-02444-8_14], http://hal.inria.fr/hal-00863262

[33] A. BELINFANTE. *JTorX: A Tool for On-Line Model-Driven Test Derivation and Execution*, in "Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems TACAS'2010 (Paphos, Cyprus)", Lecture Notes in Computer Science, Springer Verlag, March 2010, vol. 6015, pp. 266–270

[34] G. CHALUPAR, S. PEHERSTORFER, E. POLL, J. DE RUITER. *Automated Reverse Engineering using Lego*, in "Proceedings of the 8th USENIX Workshop on Offensive Technologies WOOT'14 (San Diego, CA, USA", S. BRATUS, F. F. X. LINDNER (editors), USENIX Association, August 2014

[35] D. CHAMPELOVIER, X. CLERC, H. GARAVEL, Y. GUERTE, C. MCKINTY, V. POWAZNY, F. LANG, W. SERWE, G. SMEDING. *Reference Manual of the LOTOS NT to LOTOS Translator (Version 5.7)*, November 2012, Inria/VASY, 153 pages

[36] F. CHÉRIAUX, D. GALARA, M. VIEL. *Interfaces for Nuclear Power Plant Overview*, in "Proceedings of the 8th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies NPIC & HMIT 2012 (San Diego, California, USA)", American Nuclear Society, July 2012

[37] E. M. CLARKE, E. A. EMERSON, A. P. SISTLA. *Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications*, in "ACM Transactions on Programming Languages and Systems", April 1986, vol. 8, n^{o} 2, pp. 244–263

[38] P. CROUZEN, F. LANG. *Smart Reduction*, in "Proceedings of Fundamental Approaches to Software Engineering FASE'2011 (Saarbrücken, Germany)", D. GIANNAKOPOULOU, F. OREJAS (editors), Lecture Notes in Computer Science, Springer Verlag, March 2011, vol. 6603, pp. 111–126

[39] R. DE NICOLA, F. W. VAANDRAGER. *Action versus State Based Logics for Transition Systems*, Lecture Notes in Computer Science, Springer Verlag, 1990, vol. 469, pp. 407–419

[40] A. FANTECHI, S. GNESI, G. RISTORI. *Model Checking for Action-Based Logics*, in "Formal Methods in System Design", 1994, vol. 4, pp. 187–203

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

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

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

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

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

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

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

[48] A. GRAF-BRILL. *Model-based Testing Approaches for the EnergyBus*, Department of Computer Science, Faculty of Natural Sciences and Technology I, Saarland University, October 2013

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

[50] C. HELMSTETTER. *TLM. open: a SystemC/TLM Frontend for the CADP Verification Toolbox*, in "Leibniz Transactions on Embedded Systems", April 2014, vol. 1, n^{o} 1, pp. 02:1-02:18

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

[52] L. HENRIO, O. KULANKHINA, D. LIU, E. MADELAINE. *Verifying the Correct Composition of Distributed Components: Formalisation and Tool*, in "Proceedings of the 13th International Workshop on the Foundations of Coordination Languages and Software Architectures, FOCLASA'2014 (Rome, Italy)", J. CÁMARA, J. PROENÇA (editors), September 2014

[53] C. JARD, T. JÉRON. *TGV: Theory, Principles and Algorithms — A Tool for the Automatic Synthesis of Conformance Test Cases for Non-Deterministic Reactive Systems*, in "Springer International Journal on Software Tools for Technology Transfer (STTT)", August 2005, vol. 7, n^{o} 4, pp. 297–315

[54] M. KWIATKOWSKA, G. NORMAN, D. PARKER. *PRISM: Probabilistic Symbolic Model Checker*, in "Computer Performance Evaluation: Modelling Techniques and Tools", T. FIELD, P. HARRISON, J. BRADLEY, U. HARDER (editors), Lecture Notes in Computer Science, Springer Verlag, April 2002, vol. 2324, pp. 200–204

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

[56] R. MATEESCU, D. THIVOLLE. *A Model Checking Language for Concurrent Value-Passing Systems*, in "Proceedings of the 15th International Symposium on Formal Methods FM'08 (Turku, Finland)", J. CUELLAR, T. MAIBAUM, K. SERE (editors), Lecture Notes in Computer Science, Springer Verlag, May 2008, vol. 5014, pp. 148–164

[57] P. MATYASIK. *Alvis Virtual Machine*, in "Proceedings of the 2014 Federated Conference on Computer Science and Information Systems FedCSIS'14 (Warsaw, Poland)", M. GANZHA, L. A. MACIASZEK, M. PAPRZYCKI (editors), IEEE, September 2014, pp. 1639–1645

[58] Y.-J. MOON, A. SILVA, C. KRAUSE, F. ARBAB. *A Compositional Model to Reason about End-to-End QoS in Stochastic Reo Connectors*, in "Science of Computer Programming", 2014, vol. 80, n^{o} A, pp. 3–24

[59] N. OLIVEIRA, A. SILVA, L. S. BARBOSA. *Quantitative Analysis of Reo-based Service Coordination*, in "Proceedings of the 29th Annual ACM Symposium on Applied Computing SAC'2014 (Gyeongju, Republic of Korea)", ACM Computer Society Press, March 2014, pp. 1247–1254

[60] D. ONGARO, J. OUSTERHOUT. *In Search of an Understandable Consensus algorithm*, in "Proceedings of the 2014 USENIX Annual Technical Conference USENIX ATC 14", Philadelphia, PA, USENIX Association, June 2014, pp. 305–319

[61] M. SIJTEMA, A. BELINFANTE, M. STOELINGA, L. MARINELLI. *Experiences with Formal Engineering: Model-Based Specification, Implementation and Testing of a Software Bus at Neopost*, in "Science of Computer Programming", February 2014, vol. 80, n^{o} A, pp. 188–209

[62] J. TRETMANS. *Model Based Testing with Labelled Transition Systems*, in "Formal Methods and Testing, An Outcome of the FORTEST Network, Revised Selected Papers", R. M. HIERONS, J. P. BOWEN, M. HARMAN (editors), Lecture Notes in Computer Science, Springer Verlag, 2008, vol. 4949, pp. 1–38

[63] G. VOSKUILEN, T. N. VIJAYKUMAR. *Fractal++: Closing the Performance Gap between Fractal and Conventional Coherence*, in "Proceedings of the 41st ACM/IEEE International Symposium on Computer Architecture ISCA'2014 (Minneapolis, Minnesota, USA)", S. KECKLER (editor), IEEE, June 2014, pp. 409–420

[64] G. VOSKUILEN, T. N. VIJAYKUMAR. *High-Performance Fractal Coherence*, in "Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems ASPLOS'14 (Salt Lake City, Utah, USA)", R. BALASUBRAMONIAN, A. DAVIS, S. ADVE (editors), ACM, March 2014, pp. 701–714

[65] A. WIJS, L. ENGELEN. *REFINER: Towards Formal Verification of Model Transformations*, in "Proceedings of the NASA Formal Methods 6th International Symposium NFM'14 (Houston, TX, USA)", J. M. BADGER, K. Y. ROZIER (editors), Lecture Notes in Computer Science, Springer Verlag, April 2014, vol. 8430, pp. 258–263

[66] A. WIJS. *Define, Verify, Refine: Correct Composition and Transformation of Concurrent System Semantics*, in "Proceedings of the 10th International Symposium on Formal Aspects of Component Software FACS'2013 (Nanchang, China)", J. L. FIADEIRO, Z. LIU, J. XUE (editors), Lecture Notes in Computer Science, Springer Verlag, October 2014, vol. 8348, pp. 348–368

[67] J. WU, Z. ZHANG, C. MYERS. *A Fault-Tolerant Routing Algorithm for a Network-on-Chip Using a Link Fault Model*, in "Virtual Worldwide Forum for PhD Researchers in Electronic Design Automation", 2011, http://www.async.ece.utah.edu/publications/VW-FEDA2.pdf

[68] G. DE RUVO, A. SANTONE. *An Eclipse-Based Editor to Support LOTOS Newcomers*, in "Proceedings of the 23rd International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises WETICE'2014 (Parma, Italy)", S. REDDY (editor), IEEE Computer Society Press, June 2014, pp. 372–377