INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Pop Art

Programming languages, Operating systems, Parallelism, and Aspects for Real-Time

Grenoble - Rhône-Alpes

Theme : Embedded and Real Time Systems

Table of contents

1. Team

Research Scientists

Alain Girault [Team Leader, DR INRIA, HdR]
Pascal Fradet [CR INRIA, HdR]
Gregor Gössler [CR INRIA]
Bertrand Jeannet [CR INRIA]

Faculty Member

Gwenaël Delaval [Associate professor, Université Joseph Fourier]

External Collaborator

Emil Dumitrescu [Associate Professor, INSA Lyon]

PhD Students

Mouaiad Alras [Région Rhône-Alpes grant, ISLE cluster, until 06/2010]
Marnes Hoff [INRIA grant, AUTOCHEM project]
Henri-Charles Blondeel [INRIA, since 10/2010]
Lies Lakhdar-Chaouch [INRIA, OPENTLM project]
Peter Schrammel [INRIA, SYNCHRONICS project]
Gideon Smeding [Digiteo grant]

Post-Doctoral Fellows

Petro Poplavko [INRIA and PILSI/CRI, since 10/2010]
Jean-Baptiste Raclet [INRIA, COMBEST project, until 04/2010]
Pascal Sotin [INRIA, ASOPT project]
Javier Cámara-Moreno [INRIA, VEDECY project]
Avinash Malik [Région Rhône-Alpes grant]

Administrative Assistant

Diane Courtiol [Secretary INRIA]

2. Overall Objectives

2.1. Overall Objectives

We work on the problem of the safe design of real-time control systems. This area is related to control theory as well as computer science. Application domains are typically safety-critical systems, as in transportation (avionics, railways), production, medical, or energy production systems. Both methods and formal models for the construction of correct systems, as well as their implementation in computer assisted design tools, targeted to specialists of the applications, are needed. We contribute to propose solutions all along the design flow, from the specification to the implementation: we develop techniques for the specification and automated generation of safe real-time executives for control systems, as well as static analysis techniques to check additional properties on the generated systems. Our research themes concern:

- implementations of synchronous reactive programs, generated automatically by compilation, particularly from the point of view of automatic distribution (in relation with the HEPTAGON and LUCID SYNCHRONE languages[1]) and fault tolerance (in relation with the SYNDEX environment[2]);
- high-level design and programming methods, with support for automated code generation, including: the automated generation of correct controllers using discrete control synthesis (in relation with the SIGALI synthesis tool[3]); compositionality for the verification, and construction of correct systems; reactive programming, aspect-oriented programming;

[1] http://www.di.ens.fr/~pouzet/lucid-synchrone
[2] http://www-rocq.inria.fr/syndex
[3] http://www.irisa.fr/vertecs/Logiciels/sigali.html

- static analysis and abstract interpretation techniques, which are applied both to low-level synchronous models/programs and to more general imperative programs; this includes the verification of general safety properties and the absence of runtime errors.

Our applications are in embedded systems, typically in the robotics, automotive, and telecommunications domains with a special emphasis on dependability issues (*e.g.*, fault tolerance, availability). International and industrial relations feature:

- an IST European FP7 network of excellence: ARTISTDESIGN [4], about embedded real-time systems;
- an FP7 European STREP project: COMBEST [5] on component-based design;
- an ARTEMISIA European project: CESAR [6] on cost-efficient methods and processes for safety relevant embedded systems;
- three ANR French projects: ASOPT (on static analysis), AUTOCHEM (on chemical programming), and VEDECY (on cyber-physical systems);
- a MINALOGIC Pôle de Compétitivité project: OPENTLM, dedicated to the design flow for next generation SoC and SystemC;
- an INRIA large scale action: SYNCHRONICS on a language platform for embedded system design;
- an INRIA associated team with the University of Auckland (New Zealand), called AFMES [7] on advanced formal methods for embedded systems.

3. Scientific Foundations

3.1. Embedded systems and their safe design

3.1.1. Safe Design of Embedded Real-time Control Systems

The context of our work is the area of embedded real-time control systems, at the intersection between control theory and computer science. Our contribution consists of methods and tools for their safe design. The systems we consider are intrinsically safety-critical because of the interaction between the embedded, computerized controller, and a physical process having its own dynamics. What is important is to analyze and design the safe behavior of the whole system, which introduces an inherent complexity. This is even more crucial in the case of systems whose malfunction can have catastrophic consequences, for example in transport systems (avionics, trains), production, medical, or energy production systems.

Therefore, there is a need for methods and tools for the design of safe systems. The definition of adequate mathematical models of the behavior of the systems allows the definition of formal calculi. They in turn form a basis for the construction of algorithms for the analysis, but also for the transformation of specifications towards an implementation. They can then be implemented in software environments made available to the users. A necessary complement is the setting-up of software engineering, programming, modeling, and validation methodologies. The motivation of these problems is at the origin of significant research activity, internationally and, in particular, in the European IST network of excellence ARTISTDESIGN (Advanced Real-Time Systems).

3.1.2. Models, Methods and Techniques

The state of the art upon which we base our contributions is twofold.

[4] http://www.artist-embedded.org
[5] http://www.combest.eu/home
[6] http://www.cesarproject.eu
[7] http://pop-art.inrialpes.fr/~girault/Projets/Afmes

From the point of view of discrete control, there is a set of theoretical results and tools, in particular in the synchronous approach, often founded on finite or infinite labeled transition systems [31], [39]. During the past years, methodologies for the formal verification [79], [42], control synthesis [81] and compilation, as well as extensions to timed and hybrid systems [76], [33] have been developed. Asynchronous models consider the interleaving of events or messages, and are often applied in the field of telecommunications, in particular for the study of protocols. A well-known formalism for reactive systems is STATECHARTS [69], which can be encoded in a synchronous model [34].

From the point of view of verification, we use the methods and tools of symbolic model-checking and of abstract interpretation. From symbolic model-checking, we reuse BDD techniques [37] for manipulating Boolean functions and sets, and their MTBDD extension for more general functions. Abstract interpretation [44] is used to formalize complex static analysis, in particular when one wants to analyze the possible values of variables and pointers of a program. Abstract interpretation is a theory of approximate solving of fix-point equations applied to program analysis. Most program analysis problems, among which reachability analysis, come down to solving a fix-point equation on the state space of the program. The exact computation of such an equation is generally not possible for undecidability (or complexity) reasons. The fundamental principles of abstract interpretation are: (i) to substitute to the state-space of the program a simpler domain and to transpose the equation accordingly (static approximation); and (ii) to use extrapolation (widening) to force the convergence of the iterative computation of the fix-point in a finite number of steps (dynamic approximation). Examples of static analysis based on abstract interpretation are linear relation analysis [45] and shape analysis [41].

The synchronous approach[8] [67], [68] to reactive systems design gave birth to complete programming environments, with languages like ARGOS, LUSTRE [9], ESTEREL [10], SIGNAL/ POLYCHRONY [11], LUCID SYNCHRONE [12], SYNDEX [13], or Mode Automata. This approach is characterized by the fact that it considers periodically sampled systems whose global steps can, by synchronous composition, encompass a set of events (known as simultaneous) on the resulting transition. Generally speaking, formal methods are often used for analysis and verification; they are much less often integrated into the compilation or generation of executives (in the sense of executables of tasks combined with the host real-time operating system). They are notoriously difficult to use by end-users, who are usually experts in the application domain, not in formal techniques. This is why encapsulating formal techniques into an automated framework can dramatically improve their diffusion, acceptance, and hence impact. Our work is precisely oriented towards this direction.

3.2. Issues in Design Automation for Complex Systems

3.2.1. Hard Problems

The design of safe real-time control systems is difficult due to various issues, among them their complexity in terms of the number of interacting components, their parallelism, the difference of the considered time scales (continuous or discrete), and the distance between the various theoretical concepts and results that allow the study of different aspects of their behaviors, and the design of controllers.

A currently very active research direction focuses on the models and techniques that allow the automatic use of formal methods. In the field of verification, this concerns in particular the technique of model checking. The verification takes place after the design phase, and requires, in case of problematic diagnostics, expensive backtracks on the specification. We want to provide a more constructive use of formal models, employing them to derive correct executives by formal computation and synthesis, integrated in a compilation process. We therefore use models throughout the design flow from specification to implementation, in particular by automatic generation of embeddable executives.

[8] http://www.synalp.org
[9] http://www-verimag.imag.fr/SYNCHRONE
[10] http://www.inria.fr/domaines-epi/algorithmique-programmation-logiciels-et-architectures/systemes-embarques-et-temps-reel/aoste
[11] http://www.irisa.fr/espresso/Polychrony
[12] http://www.di.ens.fr/~pouzet/lucid-synchrone/
[13] http://www-rocq.inria.fr/syndex

3.2.2. Applicative Needs

Applicative needs initially come from the fields of safety-critical systems (avionics, energy) and complex systems (telecommunications), embedded in an environment with which they strongly interact (comprising aspects of computer science and control theory). Fields with less criticality, or which support variable degrees of quality of service, such as in the multi-media domain, can also take advantage of methodologies that improve the quality and reliability of software, and reduce the costs of test and correction in the design.

Industrial acceptance, the dissemination, and the deployment of the formal techniques inevitably depend on the usability of such techniques by specialists in the application domain — and not in formal techniques themselves — and also on the integration in the whole design process, which concerns very different problems and techniques. Application domains where the actors are ready to employ specialists in formal methods or advanced control theory are still uncommon. Even then, design methods based on the systematic application of these theoretical results are not ripe. In fields like industrial control, where the use of PLC (Programmable Logic Controller [28]) is dominant, this question can be decisive.

Essential elements in this direction are the proposal of realistic formal models, validated by experiments, of the usual entities in control theory, and functionalities (*i.e.*, algorithms) that correspond indeed to services useful for the designer. Take for example the compilation and optimization taking into account the platforms of execution, possible failures, or the interactions between the defined automatic control and its implementation. A notable example for the existence of an industrial need is the activity of the ATHYS company (now belonging to DASSAULT SYSTEMES) concerning the development of a specialized programming environment, CELLCONTROL, which integrates synchronous tools for compilation and verification, tailored to the application domain. In these areas, there are functionalities that commercial tools do not have yet, and to which our results contribute.

3.2.3. Our Approach

We are proposing effective trade-offs between, on the one hand, expressiveness and formal power, and on the other hand, usability and automation. We focus on the area of specification and construction of correct real-time executives for discrete and continuous control, while keeping an interest in tackling major open problems, relating to the deployment of formal techniques in computer science, especially at the border with control theory. Regarding the applications, we propose new automated functionalities, to be provided to the users in integrated design and programming environments.

3.3. Main Research Directions

The objective of the POP ART team is the **safe design of real-time control systems**. This area is related to control theory as well as computer science. Application domains are typically safety-critical systems, as in transportation (avionics, railways), production, medical, or energy production systems. Both methods and formal models for the construction of correct systems are needed. Such methods must be implemented in computer-assisted design tools, targeted at specialists of the application domains.

Our contribution is to propose solutions covering the entire design flow, from the specification to the implementation. We develop techniques for the specification and automated generation of safe real-time executives for control systems, as well as static analysis techniques to check additional properties on the generated systems.

The integration of formal methods in an automated process of generation/compilation is founded on the formal modeling of the considered mechanisms. This modeling is the base for the automation, which operates on models well-suited for their efficient exploitation, by analysis and synthesis techniques that are difficult to use by end-users.

The creation of easily usable models aims at giving the user the role rather of a pilot than of a mechanics *i.e.*, to offer her/him pre-defined functionalities which respond to concrete demands, for example in the generation of fault tolerant or distributed executives, by the intermediary use of dedicated environments and languages.

The proposal of validated models with respect to their faithful representation of the application domain is done through case studies in collaboration with our partners, where the typical multidisciplinarity of questions across control theory and computer science is exploited.

The overall consistency of our approach comes from the fact that the main research directions address, under different aspects, the specification and generation of safe real-time control executives based on *formal models*.

We explore this field by linking, on the one hand, the techniques we use, with on the other hand, the functionalities we want to offer. We are interested in questions related to:

Component-Based Design. We investigate two main directions: (i) compositional analysis and design techniques; (ii) adapter synthesis and converter verification.

Programming for embedded systems. Programming for embedded real-time systems is considered within POP ART along three axes: (i) synchronous programming languages, (ii) aspect-oriented programming, (iii) static analysis (type systems, abstract interpretation, ...).

Dependable embedded systems. Here we address the following research axes: (i) static multiprocessor scheduling for fault-tolerance, (ii) multi-criteria scheduling for reliability, (iii) automatic program transformations, (iv) formal methods for fault-tolerant real-time systems.

3.3.1. Component-Based Design

Component-based construction techniques are crucial to overcome the complexity of embedded systems design. However, two major obstacles need to be addressed: the heterogeneous nature of the models, and the lack of results to guarantee correction of the composed system.

The heterogeneity of embedded systems comes from the need to integrate components using different models of computation, communication, and execution, on different levels of abstraction and different time scales. The BIP component framework [5] has been designed, in cooperation with VERIMAG, to support this heterogeneous nature of embedded systems.

Our work focuses on the underlying analysis and construction algorithms, in particular compositional techniques and approaches ensuring correctness by construction (adapter synthesis, strategy mapping). This work is motivated by the strong need for formal, heterogeneous component frameworks in embedded systems design.

3.3.2. Programming for Embedded Systems

Programming for embedded real-time systems is considered along three directions: (i) synchronous programming languages to implement real-time systems; (ii) aspect-oriented programming to specify non-functional properties separately from the base program; (iii) abstract interpretation to ensure safety properties of programs at compile time. We advocate the need for well defined programming languages to design embedded real-time systems with correct-by-construction guarantees, such as bounded time and bounded memory execution. Our original contribution resides in programming languages inheriting features from both synchronous languages and functional languages. In collaboration with Marc Pouzet (ENS Uml, PARKAS team), we have designed the programming language HEPTAGON, the key features of which are: data-flow formal synchronous semantics, strong typing, modular compilation. In particular, we are working on type systems for the clock calculus and the spatial modular distribution.

The goal of Aspect-Oriented Programming (AOP) is to isolate aspects (such as security, synchronization, or error handling) that cross-cut the program basic functionality and whose implementation usually yields tangled code. In AOP, such aspects are specified separately and integrated into the program by an automatic transformation process called weaving. Although this new paradigm has great practical potential, it still lacks formalization and undisciplined uses make reasoning on programs very difficult. Our work on AOP addresses these issues by studying foundational issues of AOP (semantics, analysis, verification) and by considering domain-specific aspects (availability or fault tolerance aspects) as formal properties.

Finally, the aim of the verification activity in POP ART is to check (safety) properties on programs, with emphasis on the analysis of the values of data variables (numerical variables, memory heap), mainly in the context of embedded and control-command systems, which exibit concurrency features. The applications are not only the proof of functional properties on programs, but also test selection and generation, program transformation, controller synthesis, and fault-tolerance. Our approach is based on abstract interpretation, which consists in inferring properties of the program by solving semantic equations on abstract domains. Much effort is spent on implementing developed techniques in tools for experimentation and diffusion.

3.3.3. Dependable Embedded Systems

Embedded systems must often satisfy safety critical constraints. We address this issue by providing methods and algorithms to design embedded real-time systems with guarantees on their fault-tolerance and/or reliability level.

A research direction concerns static multiprocessor scheduling of an application specification on a distributed target architecture. We increase the fault-tolerance level of the system by replicating the computations and the communications, and we schedule the redundant computations according to the faults to be tolerated. We also optimize the schedule *w.r.t.* several criteria, including the schedule length, the reliability, and the power consumption.

A second research direction concerns the fault-tolerance management, by reconfigurating the system (for instance by migrating the tasks that were running on a processor upon the failure of this processor) following objectives of fault-tolerance, consistent execution, functionality fulfillment, boundedness and optimality of response time. We base such formal methods on discrete controller synthesis.

A third research direction concerns AOP to weave fault-tolerance aspects in programs and electronic circuits (seen as synthesizable HDL programs) as mentioned in the previous section.

4. Application Domains

4.1. Industrial Applications

Our applications are the embedded system area, typically: robotics, automotive, telecommunications, systems on chip (SoC). In some areas, safety is critical, and motivates the investment in formal methods and techniques for design. But even in less critical contexts, like telecommunications and multimedia, these techniques can be beneficial in improving the efficiency and quality of designs, as well as the design, production, and test costs themselves.

Industrial acceptance of formal techniques, as well as their deployment, goes necessarily through their usability by specialists of the application domain, rather than of the formal techniques themselves. Hence our orientation towards the proposal of domain-specific (but generic) realistic models, validated through experience (*e.g.*, control tasks systems), based on formal techniques with a high degree of automation (*e.g.*, synchronous models), and tailored for concrete functionalities (*e.g.*, code generation).

4.2. Industrial Design Tools

The commercially available design tools (such as UML with real-time extensions, MATLAB/ SIMULINK/ dSPACE [14]) and execution platforms (OS such as VXWORKS, QNX, real-time versions of LINUX ...) propose a collection of functionalities without accompanying it by design or verification methods. Some of them, founded on models of reactive systems, come close to tools with a formal basis, such as for example STATEMATE by iLOGIX.

[14] http://www.dspaceinc.com

Regarding the synchronous approach, commercial tools are available: SCADE (based on LUSTRE), ESTEREL STUDIO [15], SILDEX (based on SIGNAL), specialized environments like CELLCONTROL for industrial automatism (by the INRIA spin-off ATHYS– now part of DASSAULT SYSTEMES). One can note that behind the variety of actors, there is a real consistency of the synchronous technology, which makes sure that the results of our work related to the synchronous approach are not restricted to some language due to compatibility issues.

4.3. Current Industrial Cooperations

Regarding applications and case studies with industrial end-users of our techniques, we cooperate with STMicroelectronics on two topics: (i) compositional analysis and abstract interpretation for the TLM-based System-on-Chip design flow, and (ii) dynamic models of computation for streaming applications.

5. Software

5.1. NBac

Participant: Bertrand Jeannet.

NBAC (Numerical and Boolean Automaton Checker)[16] is a verification/slicing tool for reactive systems containing combination of Boolean and numerical variables, and continuously interacting with an external environment. NBAC can also handle the same class of hybrid systems as the HyTech tool. It aims at handling efficiently systems combining a non-trivial numerical behaviour with a complex logical (Boolean) behaviour.

NBAC is connected to two input languages: the synchronous dataflow language LUSTRE, and a symbolic automaton-based language, AUTOC/AUTO, where a system is defined by a set of symbolic hybrid automata communicating via valued channels. It can perform reachability analysis, co-reachability analysis, and combination of the above analyses. The result of an analysis is either a verdict to a verification problem, or a set of states together with a necessary condition to stay in this set during an execution. NBAC is founded on the theory of abstract interpretation: sets of states are approximated by abstract values belonging to an abstract domain, on which fix-point computations are performed.

It has been used for verification and debugging of LUSTRE programs [72] [51]. It is connected to the LUSTRE toolset[17]. It has also been used for controller synthesis of infinite-state systems. The fact that the analyses are approximated results simply in the obtention of a possibly non-optimal controller. In the context of conformance testing of reactive systems, it is used by the test generator STG [18] [43] [74] for selecting test cases.

5.2. Prometheus

Participant: Gregor Gössler.

The BIP component model (Behavior, Interaction model, Priority) [65][5] has been designed to support the construction of heterogeneous embedded systems involving different models of computation, communication, and execution, on different levels of abstraction. By separating the notions of behavior, interaction model, and execution model, it enables both heterogeneous modeling, and separation of concerns.

The verification and design tool Prometheus [64] implements the BIP component framework. Prometheus is regularly updated to implement new developments in the framework and the analysis algorithms. It has allowed us to carry out several complex case studies from the system-on-chip and bioinformatics domains.

[15] http://www.esterel-technologies.com
[16] http://pop-art.inrialpes.fr/people/bjeannet/nbac/
[17] http://www-verimag.imag.fr/Lustre-V6.html
[18] http://www.irisa.fr/prive/ployette/stg-doc/stg-web.html

5.3. Implementations of Synchronous Programs

Participant: Alain Girault.

5.3.1. Fault Tolerance

We have been cooperating for several years with the INRIA team AOSTE (INRIA Sophia-Antipolis and Rocquencourt) on the topic of fault tolerance and reliability of safety critical embedded systems. In particular, we have implemented several new heuristics for fault tolerance and reliability within their software SYNDEX [19]. This has taken place within the framework of the European project EAST-EEA in which we participated together with AOSTE. Our first scheduling heuristic produces static multiprocessor schedules tolerant to a specified number of processor and communication link failures [54]. The basic principles upon which we rely to make the schedules fault tolerant is, on the one hand, the active replication of the operations [56], and on the other hand, the active replication of communications for point-to-point communication links, or their passive replication coupled with data fragmentation for multi-point communication media (*i.e.*, buses) [57]. Our second scheduling heuristic is multi-criteria: it produces a static schedule multiprocessor schedule such that the reliability is maximized, the power consumption is minimized, and the execution time is minimized. Our results on fault tolerance are summarized in a web page[20].

5.4. Apron and BDDApron Libraries

Participant: Bertrand Jeannet.

5.4.1. Principles

The APRON library[21] is dedicated to the static analysis of the numerical variables of a program by abstract interpretation [44]. Many abstract domains have been designed and implemented for analysing the possible values of numerical variables during the execution of a program (see Figure 1). However, their API diverge largely (datatypes, signatures, ...), and that does not facilitate their diffusion and experimental comparison w.r.t. efficiency and precision aspects.

The APRON library aims to provide:

- a uniform API for existing numerical abstract domains;
- a higher-level interface to the client tools, by factorizing functionalities that are largely independent of abstract domains.

From an abstract domain implementor point of view, the benefits of the APRON library are:

- the ability to focus on core, low-level functionalities;
- the help of generic services adding higher-level services for free.

For the client static analysis community, the benefits are a unified, higher-level interface, that allows experimenting, comparing, and combining abstract domains.

In 2010, the Taylor1plus domain [52], which is the underlying abstract domain of the tool FLUCTUAT [50] has been integrated in APRON.

The BDDAPRON library[22] aims at a similar goal, by adding finite-types variables and expressions to the concrete semantics of APRON domains. It is built upon the APRON library and provides abstract domains for the combination of finite-type variables (booleans, enumerated types, n-bits integers) and numerical variables (integers, rationals, floating-point numbers). It first allows to manipulate expressions that freely mix, using BDDs and MTBDDs, finite-type and numerical APRON expressions and conditions. It then provides abstract domains that combines BDDs and APRON abstract values for representing invariants holding on both finite-type variables and numerical variables.

[19] http://www-rocq.inria.fr/syndex
[20] http://pop-art.inrialpes.fr/~girault/Projets/FT
[21] http://apron.cri.ensmp.fr/library/
[22] http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/bddapron/index.html

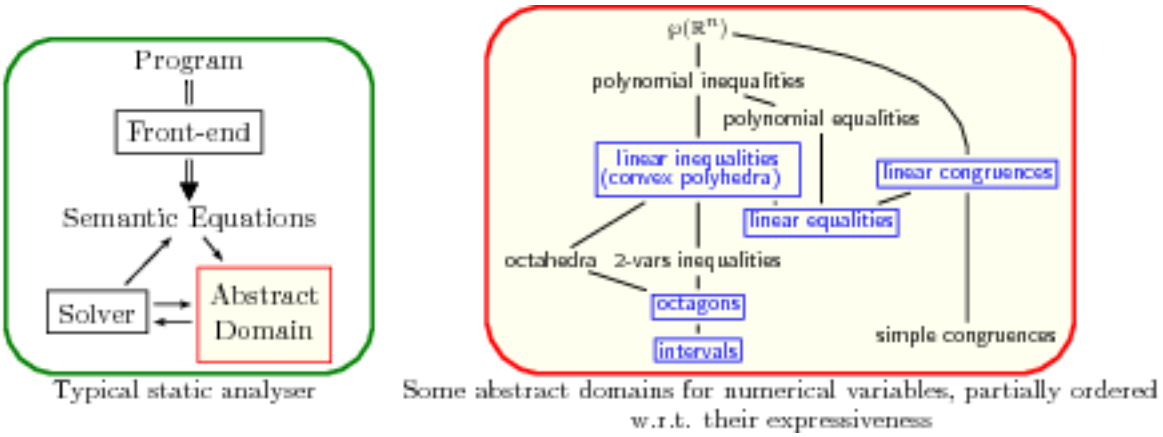

Figure 1. Typical static analyser and examples of abstract domains

5.4.2. Implementation and Distribution

The APRON library (Fig. 2) is written in ANSI C, with an object-oriented and thread-safe design. Both multi-precision and floating-point numbers are supported. A wrapper for the OCAML language is available, and a C++ wrapper is on the way. It is distributed since June 2006 under the LGPL license and available at http://apron.cri.ensmp.fr. Its development has still progressed much since. There are already many external users (ProVal/Démons, LRI Orsay, France — CEA-LIST, Saclay, France — Analysis of Computer Systems Group, New-York University, USA — Sierum software analysis platform, Kansas State University, USA — NEC Labs, Princeton, USA — EADS CCR, Paris, France — IRIT, Toulouse, France) and it is being packaged as a REDHAT and DEBIAN package.

The BDDAPRON library is written in OCAML, using polymorphism features of OCAML to make it generic. It is also thread-safe. It provides two different implementations of the same domain, each one presenting pros and cons depending on the application. It is currently used by the CONCURINTERPROC interprocedural and concurrent program analyzer.

5.5. Prototypes

5.5.1. Automatic Controller Generation

Participants: Emil Dumitrescu, Alain Girault [contact person].

We have developed a software tool chain to allow the specification of models, the controller synthesis, and the execution or simulation of the results. It is based on existing synchronous tools, and thus consists primarily in the use and integration of SIGALI [23] and Mode Automata[24].

Useful component templates and relevant properties can be materialized, on one hand by libraries of task models, and, on the other hand, by properties and synthesis objectives.

5.5.2. Rapture

Participant: Bertrand Jeannet.

[23] http://www.irisa.fr/vertecs/Logiciels/sigali.html
[24] http://www-verimag.imag.fr

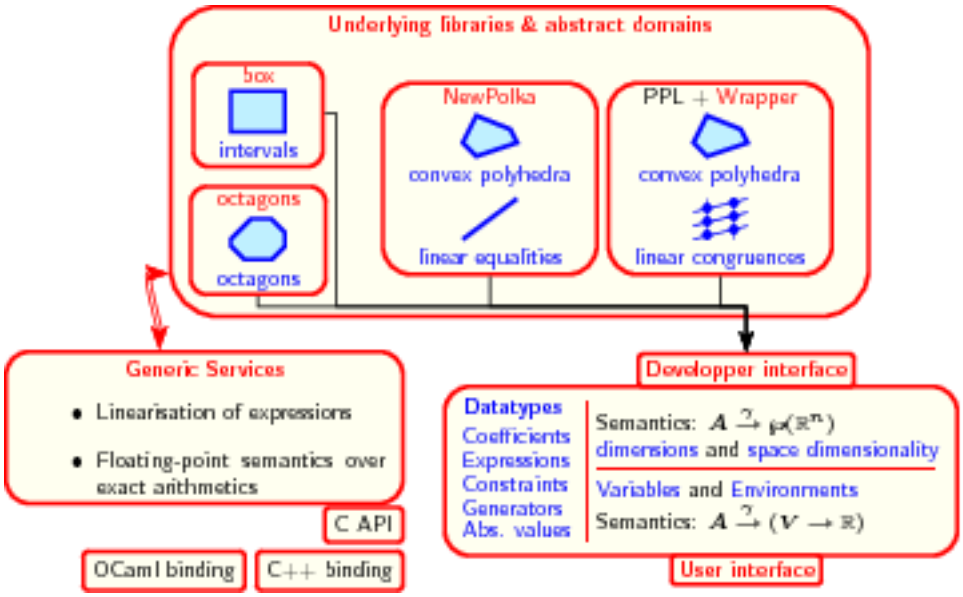

Figure 2. Organisation of the APRON *library*

RAPTURE[25] [71] [46] is a verification tool that was developed jointly by BRICS (Denmark) and INRIA in years 2000–2002. The tool is designed to verify reachability properties on Markov Decision Processes (MDP), also known as Probabilistic Transition Systems. This model can be viewed both as an extension to classical (finite-state) transition systems extended with probability distributions on successor states, or as an extension of Markov Chains with non-determinism. We have developed a simple automata language that allows the designer to describe a set of processes communicating over a set of channels *à la* CSP. Processes can also manipulate local and global variables of finite type. Probabilistic reachability properties are specified by defining two sets of initial and final states together with a probability bound. The originality of the tool is to provide two reduction techniques that limit the state space explosion problem: automatic abstraction and refinement algorithms, and the so-called essential states reduction.

5.5.3. Abstract Interpretation Tools and Libraries

Participant: Bertrand Jeannet.

We also develop and maintain smaller libraries of general use for people working in the static analysis and abstract interpretation community.

FIXPOINT [26]: a generic fix-point engine written in OCAML. It allows the user to solve systems of fix-point equations on a lattice, using a parameterized strategy for the iteration order and the application of widening. It also implements recent techniques for improving the precision of analysis by alternating post-fixpoint computation with widening and descending iterations in a sound way [62].

INTERPROC [27]: a simple interprocedural static analyzer that infers properties on the numerical variables of programs in a toy language. It is aimed at demonstrating the use of the previous library and the above-described APRON library, and more generally at disseminating the knowledge in abstract interpretation. It is also deployed through a web-interface[28]. It is used as the experimental platform

[25] http://pop-art.inrialpes.fr/people/bjeannet/rapture/rapture.html
[26] http://http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/fixpoint
[27] http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/interproc

of the ASOPT ANR project.

CONCURINTERPROC extends Interproc with concurrency, for the analysis of multithreaded programs interacting via shared global variables. It is also deployed through a web-interface[29].

PINTERPROC extends Interproc with pointers to local variables. It is also deployed through a web-interface[30].

5.5.4. Heptagon/BZR

Participant: Gwenaël Delaval.

HEPTAGON is a dataflow synchronous language, inspired from LUCID SYNCHRONE[31]. Its compiler is meant to be simple and modular, allowing this language to be a good support for the prototyping of compilation methods of synchronous languages.

HEPTAGON has been used to built BZR, which is an extension of the former with contracts constructs. These contracts allow to express dynamic temporal properties on the inputs and outputs of HEPTAGON node. These properties are then enforced, within the compilation of a BZR program, by discrete controller synthesis, using the SIGALI tool[32]. The synthesized controller is itself generated in HEPTAGON, allowing its analysis and compilation towards different target languages (C, JAVA, VHDL).

6. New Results

6.1. Dependable Distributed Real-time Embedded Systems

Participants: Javier Cámara-Moreno, Gwenaël Delaval, Pascal Fradet, Alain Girault [contact person], Gregor Gössler, Bertrand Jeannet, Emil Dumitrescu.

6.1.1. Static Multiprocessor Scheduling with Tradeoff Between Performance and Reliability

We have extended our work on bicriteria (length, reliability) scheduling [55], [59] in two directions. The first direction takes into account the power consumption as a third criterion to be minimized. We have designed a scheduling heuristics called TSH that, given a software application graph and a multiprocessor architecture, produces a static multiprocessor schedule that optimizes three criteria: its *length* (crucial for real-time systems), its *reliability* (crucial for dependable systems), and its *power consumption* (crucial for autonomous systems). Our tricriteria scheduling heuristics, TSH, uses the *active replication* of the operations and the data-dependencies to increase the reliability, and uses *dynamic voltage scaling* to lower the power consumption. By setting a bound on the minimal reliability, a bound on the maximal power consumption, and making theses two bounds vary, we are able to produce with TSH a Pareto surface of the best compromises found in the 3D space (length, reliability, power consumption). TSH is implemented within the SYNDEX tool. This work is conducted in collaboration with Hamoudi Kalla (University of Batna, Algeria) and Ismail Assayad (University of Casablanca, Morocco).

[28] http://pop-art.inrialpes.fr/interproc/interprocweb.cgi
[29] http://pop-art.inrialpes.fr/interproc/concurinterprocweb.cgi
[30] http://pop-art.inrialpes.fr/interproc/pinterprocweb.cgi
[31] http://www.di.ens.fr/~pouzet/lucid-synchrone
[32] http://www.irisa.fr/vertecs/Logiciels/sigali.html

The second direction studies the mapping of chains of tasks on multi-processor platforms. We have proposed *mapping by interval techniques*, where the chain of tasks is divided in a sequence of intervals, each interval being executed on a different processor in a pipe-lined manner, and each processor executing no more than one interval. Because of this pipe-lined execution, we have two antagonistic criteria, the input-output latency and the period. Then, to increase the reliability, we replicate the intervals by mapping them to several processors. We have proved that, for homogeneous platforms, computing a mapping that optimizes the reliability only is *polynomial*, but that optimizing both the reliability and the period is *NP-complete*, as well as optimizing both the reliability and the latency. For heterogeneous platforms, we have proved that optimizing the reliability only is *NP-complete*, and hence all the multi-criteria mapping problems that include the reliability in their criteria are also *NP-complete* [16]. Finally, we have proposed heuristics to find solutions in the NP-complete cases. This work is done in collaboration with Anne Benoit, Fanny Dufossé, and Yves Robert (ENS Lyon and GRAAL team).

Unlike most work found in the literature, all our contributions are truly bicriteria, in the sense that the user can gain several orders of magnitude on the reliability of his schedule thanks to the active replication of tasks onto processors. In contrast, most of the other algorithms do not replicate the tasks, and hence have a very limited impact on the reliability.

6.1.2. Automating the Addition of Fault Tolerance with Discrete Controller Synthesis

We have defined a new framework for the *automatic* design of fault tolerant embedded systems, based on discrete controller synthesis (DCS), a formal approach based on the same state-space exploration algorithms as model-checking [80]. Its interest lies in the ability to obtain automatically systems satisfying by construction formal properties specified *a priori*. Our aim is to demonstrate the feasibility of this approach for fault tolerance. We start with a fault intolerant program, modeled as the synchronous parallel composition of finite labeled transition systems. We specify formally a fault hypothesis, state fault tolerance requirements and use DCS to obtain automatically a program having the same behavior as the initial fault intolerant one in the absence of faults, and satisfying the fault tolerance requirements under the fault hypothesis. Our original contribution resides in the demonstration that DCS can be elegantly used to design fault tolerant systems, with guarantees on key properties of the obtained system, such as the fault tolerance level, the satisfaction of quantitative constraints, and so on. We have shown with numerous examples taken from case studies that our method can address different kinds of failures (crash, value, or Byzantine) affecting different kinds of hardware components (processors, communication links, actuators, or sensors). Besides, we have shown that our method also offers an optimality criterion very useful to synthesize fault tolerant systems compliant to the constraints of embedded systems, like power consumption. In summary, our framework for fault tolerance has the following advantages [58]:

- The **automation**, because DCS produces automatically a fault tolerant system from an initial fault intolerant one.
- The **separation of concerns**, because the fault intolerant system can be designed independently from the fault tolerance requirements.
- The **flexibility**, because, once the system is entirely modeled, it is easy to try several fault hypotheses, several environment models, several fault tolerance goals, several degraded modes, and so on.
- The **safety**, because, in case of positive result obtained by DCS, the specified fault tolerance properties are guaranteed by construction on the controlled system.
- The **optimality** when optimal synthesis is used, modulo the potential numerical equalities (hence a non strict optimality).

In collaboration with Emil Dumitrescu (INSA Lyon), Hervé Marchand (VERTECS team from Rennes), and Eric Rutten (SARDES team from Grenoble), we have extended this work in the direction of optimal synthesis considering weights cumulating along bounded-length paths, and its application to the control of sequences of reconfigurations. We have adapted our models in order to take into account the additive costs of, *e.g.*, execution time or power consumption, and adapting synthesis algorithms in order to support the association of costs with

transitions, and the handling of these new weight functions in the optimal synthesis. We therefore combine, on the one hand, guarantees on the safety of the execution by tolerating faults, and on the other hand, guarantees on the worst cumulated consumption of the resulting dynamically reconfiguring fault tolerant system [19].

In collaboration with Tolga Ayav (University of Izmir, Turkey), we are also working on an AOP approach for fault tolerance. This is described in details in Section 6.6.3.

6.1.3. Synthesis of Switching Controllers using Approximately Bisimilar Multiscale Abstractions

The use of discrete abstractions for continuous dynamics has become standard in hybrid systems design (see e.g. [85] and the references therein). The main advantage of this approach is that it offers the possibility to leverage controller synthesis techniques developed in the areas of supervisory control of discrete-event systems [80] or algorithmic game theory [32]. The first attempts to compute discrete abstractions for hybrid systems were based on traditional systems behavioral relationships such as simulation or bisimulation [78], initially proposed for discrete systems most notably in the area of formal methods. These notions require inclusion or equivalence of observed behaviors which is often too restrictive when dealing with systems observed over metric spaces. For such systems, a more natural abstraction requirement is to ask for closeness of observed behaviors. This leads to the notions of approximate simulation and bisimulation introduced in [53].

These notions enabled the computation of approximately equivalent discrete abstractions for several classes of dynamical systems, including nonlinear control systems with or without disturbances, and switched systems. These approaches are based on sampling of time and space where the sampling parameters must satisfy some relation in order to obtain abstractions of a prescribed precision. In particular, the smaller the time sampling parameter, the finer the lattice used for approximating the state space; this may result in abstractions with a very large number of states when the sampling period is small. However, there are a number of applications where sampling has to be fast; though this is generally necessary only on a small part of the state-space.

In [17] we have presented a novel class of multiscale discrete abstractions for incrementally stable switched systems that allows us to deal with fast switching while keeping the number of states in the abstraction at a reasonable level. We assume that the controller of the switched system has to decide the control input and the time period during which it will be applied before the controller executes again. In this context, it is natural to consider abstractions where transitions have various durations. For transitions of longer duration, it is sufficient to consider abstract states on a coarse lattice. For transitions of shorter duration, it becomes necessary to use finer lattices. These finer lattices are effectively used only on a restricted area of the state-space where the fast switching occurs.

These abstractions allow us to use multiscale iterative approaches for controller synthesis as follows. An initial controller is synthesized based on the dynamics of the abstraction at the coarsest scale where only transitions of longer duration are enabled. An analysis of this initial controller allows us to identify regions of the state-space where transitions of shorter duration may be useful (e.g. to improve the performance of the controller). Then, the controller is refined by enabling transitions of shorter duration in the identified regions. The last two steps can be repeated until we are satisfied with the obtained controller.

6.1.4. Modular Discrete Controller Synthesis

Discrete controller synthesis (DCS) [80] allows to design programs in a mixed imperative/declarative way. From a program with some freedom degrees left by the programmer (e.g., free controllable variables), and a temporal property to enforce which is not *a priori* verified by the initial program, DCS tools compute off-line automatically a *controller* which will constrain the program (by e.g., giving values to controllable variables) such that, whatever the values of inputs from the environment, the *controlled program* verifies the temporal property.

Our motivation w.r.t. DCS concerns its modular application, improving the scalability of the technique by using contract enforcement and abstraction of components. Moreover, our aim is to integrate DCS into a compilation chain, and thereby improve its usability by programmers, not experts in discrete control. This work has been

implemented into the HEPTAGON/BZR language and compiler [49]. This work is done in collaboration with Hervé Marchand (VERTECS team from Rennes) and Éric Rutten (SARDES team from Grenoble).

The implemented tool allows the generation of the synthesized controller under the form of an HEPTAGON node, which can in turn be analyzed and compiled, together with the HEPTAGON source from which it has been generated. This full integration allows this method to aim different target languages (currently C, JAVA or VHDL), and its integrated use in different contexts.

Several case studies are currently explored. In [18], we show how HEPTAGON/BZR can be used in an Autonomic System context: system administrators have to manage the trade-off between system performances and energy saving goals. Autonomic computing is a promising approach to automate the control of the QoS and the energy consumed by a system. This paper precisely investigates the use of synchronous programming and discrete controller synthesis to automate the generation of a controller that enforces the required coordination between QoS and energy managers. We illustrate our approach by describing the coordination between an admission controller and an energy controller.

6.2. Automatic Distribution of Synchronous Programs

Participants: Mouaiad Alras, Gwenaël Delaval [contact person], Alain Girault.

6.2.1. Modular Distribution

Synchronous programming languages describe functionally centralized systems, where every value, input, output, or function is always directly available for every operation. However, most embedded systems are nowadays composed of several computing resources. The aim of this work is to provide a language-oriented solution to describe *functionally distributed reactive systems*. This research is conducted within the INRIA large scale action SYNCHRONICS and is a joint work with Marc Pouzet (ENS, PARKAS team from Saclay).

We are working on type systems to formalize, in an uniform way, both the clock calculus and the location calculus of a synchronous data-flow programming language (the HEPTAGON language, inspired from LUCID SYNCHRONE [38]). On one hand, the clock calculus infers the clock of each variable in the program and checks the clock consistency: e.g., a time-homogeneous function, like +, should not be applied to variables of different clocks. On the other hand, the location calculus infers the spatial distribution of computations and checks the spatial consistency: e.g., a centralized operator, like +, should not be applied to variables located on different locations. Compared to the recent PhD of Gwenaël Delaval [47], [48], the goal is to achieve *modular* distribution. By modular, we mean that we want to compile each function of the program into a single function capable of running on any computing location. We make use of our uniform type system to express the computing locations as first-class abstract types, exactly like clocks, which allows us to compile a typed variable (typed by both the clock and the location calculi) into `if ... then ... else ...` structures, whose conditions will be valuations of the clock and location variables.

6.2.2. Model-based Development of Fault-tolerant Embedded Systems, Code Generation for Distributed Heterogeneous Platforms

Model-based design (MBD) involves designing a model of a control system, simulating and debugging it with dedicated tools, and finally generating automatically code corresponding to this model. In the domain of embedded systems, it offers the huge advantage of avoiding the time-consuming and error-prone final coding phase. The main issue raised by MBD is the faithfulness of the generated code with respect to the initial model, the latter being defined by the simulation semantics. To bridge the gap between the high-level model and the low-level implementation, we use the synchronous programming language LUSTRE as an intermediary formal model [66]. Concretely, starting from a high-level model specified in the de-facto standard SIMULINK, we proceed in two steps. Firstly, we generate LUSTRE code along with some necessary structured "glue code"; this is based on new "meta-operators" that extend LUSTRE with the non-functional properties extracted from the SIMULINK model (related, e.g., to the activation conditions and the real time). Secondly, from this intermediate format written in LUSTRE with meta-operators, we generate embedded real-time code for the

Xenomai RTOS[33]. Thanks to Lustre's clean mathematical semantics, we are able to guarantee the faithfulness of the generated multi-tasked real-time code [30]. This is the topic of the ongoing PhD of Mouaiad Alras, co-advised by Alain Girault and Pascal Raymond (CNRS, Verimag).

6.3. New Programming Languages for Embedded Systems

Participants: Alain Girault [contact person], Avinash Malik.

6.3.1. The DSystemJ Programming Language for Dynamic Distributed GALS Systems

We have extended the SYSTEMJ programming language [13] with dynamic constructs to better design and implement dynamic distributed *Globally Asynchronous Locally Synchronous* (GALS) systems. The new language is called DSYSTEMJ. We have studied its formal model of computation, its formal syntax and semantics, its compilation, and its implementation. DSYSTEMJ is aimed at dynamic distributed systems, which use socket based communication protocols for communicating between components. The language allows the creation and control at runtime of asynchronous processes called clock-domains, their mobility on a distributed execution platform, as well as the runtime reconfiguration of the system's functionality and topology. Like in SYSTEMJ, all the data computations can be efficiently programmed in JAVA. As DSYSTEMJ is based on a GALS model of computation and has a formal semantics, it offers very safe mechanisms for implementation of distributed systems, as well as potential for their formal verification. The runtime support is implemented in the SYSTEMJ GALS language, which can therefore be considered as a static subset of DSYSTEMJ. A journal article has been submitted. This work has been done in collaboration with Zoran Salcic (University of Auckland).

6.3.2. The PRET-C Programming Language for Time Predictability

We have continued our work on the PRET-C language (Precision Timed C), for predictable and lightweight multi-threading in C. PRET-C supports synchronous concurrency, preemption, and a high-level construct for logical time. In contrast to existing synchronous languages, PRET-C offers C-based shared memory communications between concurrent threads that is guaranteed to be thread safe. Due to the proposed synchronous semantics, the mapping of logical time to physical time can be achieved much more easily than with plain C, thanks to a Worst Case Reaction Time (WCRT) analyzer (not presented here). Associated to the PRET-C programming language, we present a dedicated target architecture, called ARPRET, which combines a hardware accelerator associated to an existing softcore processor. This allows us to improve the throughput while preserving the predictability. With extensive benchmarking, we have demonstrated that ARPRET not only achieves completely predictable execution of PRET-C programs, but also improves the throughput when compared to the pure software execution of PRET-C. We have also shown that the PRET-C software approach is significantly more efficient in comparison to two other light-weight concurrent C variants (namely SC and Protothreads), as well as the well-known ESTEREL synchronous programming language [14], [15]. This work has been done in collaboration with Partha Roop and Sidharta Andalam (University of Auckland).

6.4. Static Analysis and Abstract Interpretation

Participants: Alain Girault, Bertrand Jeannet [contact person], Lies Lakhdar-Chaouch, Peter Schrammel, Pascal Sotin.

6.4.1. Combining Control and Data Abstraction for the Verification of Hybrid Systems

We have studied the verification of hybrid systems built as the composition of a discrete software controller interacting with a physical environment exhibiting a continuous behavior. Our goal is to tackle the problem of the combinatorial explosion of discrete states that may happen when a complex software controller is considered. We propose to extend an existing abstract interpretation technique, namely dynamic partitioning, to hybrid systems. Dynamic partitioning, which shares some common principles with predicate abstraction, allows us to finely tune the tradeoff between precision and efficiency in the analysis.

[33] http://www.xenomai.org

We have extended the NBAC tool (Section 5.1) according to these principle, and showed the efficiency of the approach by a case study that combines a non trivial controller specified in the synchronous dataflow programming language LUSTRE with its physical environment [36][9].

6.4.2. Extending Abstract Acceleration Methods to Data-Flow Programs with Numerical Inputs

Acceleration methods are commonly used for computing precisely the effects of loops in the reachability analysis of counter machine models. Applying these methods on synchronous data-flow programs with Boolean and numerical variables, e.g., LUSTRE programs, firstly requires the enumeration of the Boolean states in order to obtain a control graph with numerical variables only. Secondly, acceleration methods have to deal with the non-determinism introduced by numerical input variables.

We addressed in [23] the latter problem by extending the concept of abstract acceleration of Gonnord et al. [61], [60] to numerical input variables. This extension raises some subtle points. We show how to accelerate loops composed of a translation with resets *and inputs*, provided that the guard of the loop constrains separately state and input variables, and we evaluate the gain in precision that we obtain with this method, compared to the more traditional approach based on the use of widening. A journal version has been submitted to a special issue of Journal of Symbolic Computation, focusing on invariant generation and advanced techniques for reasoning about loops.

We worked more recently on the first point. Our goal is to apply acceleration techniques to data-flow programs without resorting to an exhaustive enumeration of Boolean states. To this end, we are studying (1) methods for *applying abstract acceleration* to general control flow graphs, and (2) heuristics for *controlled partitioning*, i.e., partially unfolding the control structure in order to gain precision on numerical variables during analysis while treating symbolically Boolean states as much as possible.

6.4.3. A Relational Approach to Interprocedural Shape Analysis

This work addresses the verification of properties of imperative programs with recursive procedure calls, heap-allocated storage, and destructive updating of pointer-valued fields, i.e., interprocedural shape analysis. It presents a way to apply some previously known approaches to interprocedural dataflow analysis — which in past work have been applied only to a much less rich setting — so that they can be applied to programs that use heap-allocated storage and perform destructive updating.

Our submission to ACM TOPLAS has been published this year [12]. This work has been done in collaboration with T. Reps (Univ. of Madison-Wisconsin), M. Sagiv (Univ. of Tel Aviv) and A. Loginov (GrammaTech).

6.4.4. Concrete Memory Models for Shape Analysis

The purpose of shape analysis is to infer properties on the runtime structure of the memory heap. Like most static analyses, shape analyses perform approximations. One has thus to distinguish the concrete memory model that a shape analysis tackles, and the abstract memory model/representation used by the analysis to express properties. For instance, in [83] and in [12] the concrete memory model is an unbounded 2-valued logical structure, and the abstract memory representation is a bounded 3-valued logical structure. But other analyses describe concrete (and abstract) memory models with separation-logic formulas [40].

These concrete models do actually abstract some properties, as they do not completely model the physical memory of a computer. For instance, the physical numerical addresses may be ignored, as it is the case for [83] which cannot define the semantics of C pointer arithmetics.

In [25] we propose a classification of various concrete memory models and we try to clarify the equivalences and differences between them. In particular, we discuss to which extend the semantics of the C language can be encoded within these models, as the C-like programming languages are the most expressive ones in term of pointer manipulation.

6.4.5. Relational Interprocedural Analysis of Concurrent Programs

We have studied the extension of the relational approach to interprocedural analysis of sequential programs to concurrent programs, composed of a fixed number of threads [73].

In the relational approach, a sequential program is analyzed by computing summaries of procedures, and by propagating reachability information using these summaries. We propose an extension to concurrent programs, which is technically based on an instrumentation of the standard operational semantics, followed by an abstraction of tuple of call-stacks into sets. This approach allows us to extend relational interprocedural analysis to concurrent programs. We have implemented it for programs with scalar variables, in the CONCURINTERPROC online analyzer (see §5.5.3).

We have experimented several classical synchronisation protocols in order to investigate the precision of our technique, but also to analyze the approximations it performs.

This year a journal version has been submitted to SOSYM (Software and Systems Modeling) and is currently under revision process. The journal version improves on the conference version with better notation and generalization to backward analysis.

We also worked on new techniques for applying the widening extrapolation operator in the context of concurrent programs. This is the topic of the PhD of Lies Lakhdar-Chaouch, co-advised by Bertrand Jeannet and Alain Girault, and funded by OPENTLM. A conference paper is in preparation.

6.4.6. Precise Interprocedural Analysis in the Presence of Pointers to the Stack

In a language with procedure calls and pointers as parameters, an instruction can modify memory locations anywhere in the call-stack. The presence of such side effects breaks most generic interprocedural analysis methods (such as the one described in the sections 6.4.3 and 6.4.5) which assume that only the top of the stack may be modified.

We present a method that addresses this issue, based on the definition of an equivalent local semantics in which writing through pointers has a local effect on the stack. The idea of this local semantics, inspired by [35], is that a procedure works on local copies (called *external locations*) of the locations that it can reach with its pointer parameters. When the procedure returns to its caller, the side-effects performed on such copies will be propagated back the corresponding locations in the caller, which may be themselves local or external w.r.t. their own caller.

Our second contribution in this context is an adequate representation of summary functions that models the effect of a procedure, not only on the values of its scalar and pointer variables, but also on the values contained in pointed memory locations. Our implementation in the interprocedural analyser PINTERPROC (see section 5.5.3) results in a verification tool that infers relational properties on the value of Boolean, numerical and pointer variables.

We submitted this year a paper to the ESOP'2011 conference, which has been accepted [84].

6.4.7. Software Engineering of Abstract Interpretation Tools

The "right" way of writing and structuring compilers is well-known. The situation is a bit less clear for static analysis tools. It seems to us that a static analysis tool is ideally decomposed into three building blocks: (1) a front-end, which parses programs, generates semantic equations, and supervises the analysis process; (2) a fixpoint equation solver, which takes equations and solves them; (3) and an abstract domain, on which equations are interpreted. The expected advantages of such a modular structure is the ability of sharing development efforts between analyzers for different languages, using common solvers and abstract domains. However putting in practice such ideal concepts is not so easy, and some static analyzers merge for instance the blocks (1) and (2).

In [22], we describe how we instantiated these principles with three different static analyzers (addressing resp. imperative sequential programs (INTERPROC), imperative concurrent programs (CONCURINTERPROC), and synchronous dataflow programs (NBAC), a generic fixpoint solver (FIXPOINT), and two different abstract domains (APRON and BDDAPRON), see sections 5.5.3, 5.1, and 5.4. We discuss our experience on the advantages and the limits of this approach compared to related work.

6.5. Component-Based Construction

Participants: Alain Girault, Gregor Gössler [contact person], Gideon Smeding.

6.5.1. Specification Enforcing Refinement for Convertibility Verification

Protocol conversion deals with the automatic synthesis of an additional component or glue logic, often referred to as an *adaptor* or an *interface*, to bridge mismatches between interacting components, often referred to as *protocols*. A formal solution, called convertibility verification, has been recently proposed, which produces such a glue logic, termed as a *converter*, so that the parallel composition of the protocols and the converter also satisfies some desired specification. A converter is responsible for bridging different kinds of mismatches such as *control* and *clock* mismatches. Mismatches are usually removed by the converter (similar to controllers in supervisory control of discrete event systems) by *disabling undesirable paths* in the protocol composition. In [82] we defined a solution to the converter synthesis problem for control mismatches based on a refinement relation called *Specification Enforcing Refinement* (SER) between a protocol composition and a desired specification.

We are currently working on a generalization of this convertibility verification problem in order to take data exchange — and hence, incompatibilities stemming from inconsistent protocol specifications of how data are exchanged — into account.

6.5.2. Contract-based Design

Contracts have first been introduced as a type system for classes [77]: a method guarantees some post-condition under the assumption that its pre-condition is satisfied. In the component-based programming community, contracts are increasingly focus of research as a means to achieve one of the main goals of the component paradigm, namely the deployment and reuse of components in different, *a priori* unknown contexts. As components may interact under various models of communication, the notion of contract has been generalized from pre- and post-conditions in the form of predicates to *behavioral interfaces* such as *interface automata* [86], allowing to reason about the temporal behavior of environments with which a component can be composed.

6.5.2.1. Probabilistic Contracts

Typical embedded and distributed systems often encompass unreliable software or hardware components, as it may be technically or economically impossible to make a system entirely reliable. As a result, system designers have to deal with probabilistic specifications such as "the probability that this component fails at this point of its behavior is less than or equal to 10^{-4}". More generally, uncertainty in the observed behavior is introduced by abstraction of black-box — or simply too complex — behavior of components, the environment, or the execution platform.

In [26], we have introduced a framework for the design of correct systems from probabilistic, interacting components. To model components, we adopt the discrete time Interactive Markov Chain (IMC) semantic model [70], which combines Labeled Transition System (LTS) and Markov Chain. Components communicate through interactions, that is, synchronized action transitions. Interactions are essential in component frameworks, as they allow the modeling of how components cooperate and communicate. We use the BIP framework [5] to model interactions between components.

Since the deploying context of a component is not known at design time, we use probabilistic *contracts* to specify and reason about correct behaviors of a component. Contracts allow us to specify what a component can expect from its context, what it must guarantee, and explicitly limit the responsibilities of both.

The framework we have proposed allows us to model components, their interactions, and uncertainty in their observed behavior. It supports different steps in a design flow: refinement and abstraction, parallel composition, and conjunction (shared refinement). We have proved that these operations satisfy the desired properties of *independent implementability* and *congruence* for parallel composition, and *soundness* for conjunction. Thus,

- refinement is compositional, that is, contracts over different components can be refined and implemented independently;
- the parallel composition of two contracts is satisfied by the parallel composition of any two implementations of the contracts; and

- several contracts C_i over the same component may be used to independently specify different requirements, possibly over different subsets of the component interactions. The conjunction is a common refinement of all C_i. Conjunction of probabilistic specifications is non trivial, as a straightforward approach would introduce spurious behaviors.

6.5.2.2. Causality Analysis in Contract Violation

Establishing liabilities in case of litigation is generally a delicate matter. It becomes even more challenging when IT systems are involved. Generally speaking, a party can be declared liable for a damage if a fault can be attributed to that party and that fault has caused the damage. The two key issues are thus to establish convincing evidence with respect to (1) the occurrence of the fault and (2) the causality relation between the fault and the damage. The first issue concerns the technique used to log the relevant events of the system and to ensure that the logs can be produced (and have some value) in court. The second issue is especially complex when several faults are detected in the logs and the impact of these faults on the occurrence of the failure has to be assessed. In [6] we have focused on this second issue and proposed a formal framework for reasoning about causality. A system based on this framework could be used to provide relevant information to the expert, the judge, or the parties themselves (in case of amicable settlement) to analyze the origin of the failure of an IT system.

The notion of causality has been studied for a long time in computer science, but with very different perspectives and goals. In the distributed systems community, causality (following Lamport's seminal paper [75]) is seen essentially as a temporal property. In our context, the temporal ordering contributes to the analysis, but it is obviously not sufficient to establish the *logical causality* required to rule on a matter of liability: the fact that an event e_1 has occurred before an event e_2 does not imply that e_1 was the cause for e_2 (or that e_2 would not have occurred if e_1 had not occurred).

Our formal model is based on components interacting according to well identified *interaction models* [5]. Each component is associated with an individual *contract* which specifies its expected behavior. The system itself is associated with a *global contract* which is assumed to be implied by the composition of the individual contracts.

We have defined several variants of logical causality. The first variant, *necessary causality*, characterizes cases when the global contract would not have been violated if the local contract had been fulfilled. The second variant, *sufficient causality*, characterizes cases when the global contract would have been violated even if all the other components had fulfilled their contracts. In other words, the violation of its contract by a single component was sufficient to violate the global contract.

We have further shown that our definitions of causality are decidable in the introduced setting. We have also provided conditions for decidability on trace suffixes. Such a possibility is of great practical significance because it makes it possible to analyze traces back to a given point in the past. Indeed, the analysis of liability in real cases can hardly assume that all traces of the past can always be produced and analyzed.

In order to be able to trace the propagation of faults, we have defined *horizontal causality*, which relates prefixes of local traces of components on the same level of hierarchy. Horizontal causality allows to analyze causality among violations of component contracts.

This work opens a number of new directions for further research, in particular, the generalization to different models of communication, and to a setting where the result of causality analysis is not Boolean but a probability.

This research has been conducted as part of the LISE project on liability issues in software engineering [29].

6.6. Aspect-Oriented Programming

Participants: Pascal Fradet [contact person], Alain Girault, Henri-Charles Blondeel.

The goal of Aspect-Oriented Programming (AOP) is to isolate aspects (such as security, synchronization, or error handling) which cross-cut the program basic functionality and whose implementation usually yields tangled code. In AOP, such aspects are specified separately and integrated into the program by an automatic transformation process called *weaving*.

Although this paradigm has great practical potential, it still lacks formalization and undisciplined uses make reasoning on programs very difficult. Our work on AOP addresses these issues by studying foundational issues (semantics, analysis, verification) and by considering domain-specific aspects (availability or fault tolerance aspects) as formal properties.

6.6.1. Aspects Preserving Properties

Aspect Oriented Programming can arbitrarily distort the semantics of programs. In particular, weaving can invalidate crucial safety and liveness properties of the base program.

We have identified categories of aspects that preserve some classes of properties [27]. Our categories of aspects comprise, among others, observers, aborters, and confiners. For example, observers do not modify the base program's state and control-flow (*e.g.*, persistence, profiling, and debugging aspects). These categories are defined formally based on a language independent abstract semantic framework. The classes of properties are defined as subsets of LTL for deterministic programs and CTL* for non-deterministic ones. We have formally proved that, for any program, the weaving of any aspect in a category preserves any property in the related class.

In a second step, we have designed for each aspect category a specialized aspect language which ensures that any aspect written in that language belongs to the corresponding category. These languages preserve the corresponding classes of properties by construction.

This work was conducted in collaboration with Rémi Douence from the ASCOLA INRIA team at École des Mines de Nantes.

6.6.2. Resource Management and Aspects of Availability

We have proposed a domain-specific aspect language aimed at preventing denial of service caused by resource management (*e.g.*, starvation, deadlocks, etc.) [10]. The aspects specify time or frequency limits in the allocation of resources. They can be seen as formal temporal properties on execution traces that specify availability policies. The semantics of base programs and aspects are expressed as *timed automata*. The main advantage of such a formal approach is two-fold:

- aspects are expressed at a higher-level and the semantic impact of weaving is kept under control;
- model checking tools can be used to optimize weaving and verify the enforcement of general availability properties.

6.6.3. Fault Tolerance Aspects

Here, our objective is to design a domain-specific language for specifying fault tolerance aspects as well as efficient techniques based on static analysis, program transformation, and/or instrumentation to weave them into real-time programs.

We have studied the implementation of specific fault tolerance techniques in real-time embedded systems using program transformation [1]. We are now investigating the use of fault-tolerance aspects in hardware description languages (HDL, for instance VERILOG or VHDL). Our goal is to design an aspect language allowing users to specify and tune a wide range of fault tolerance techniques, while ensuring that the woven program remains synthesizable. The objective is to produce fault-tolerant circuits by specifying fault-tolerant strategies separately from the functional specifications.

This line of research is followed by Henri-Charles Blondeel in his PhD thesis, co-advised by Alain Girault and Pascal Fradet.

6.7. Other results

6.7.1. Chemical Programming

Participants: Pascal Fradet [contact person], Marnes Hoff.

Chemical programming describes computation in terms of a *chemical solution* in which molecules (representing data) interact freely according to *reaction rules* (representing the program). Solutions are represented by multisets of elements and reactions by rewrite rules which consume and produce new elements according to conditions. This paradigm makes it possible to express programs without artificial sequentiality in a very abstract way. It bridges the gap between specification and implementation languages.

A drawback of chemical languages is that their very high-level nature usually leads to very inefficient programs. We have proposed an approach [20] where the basic functionality is expressed as a chemical program whereas efficiency is achieved separately by:

- structuring the multiset with a data type defining neighborhood relations;
- describing the selection of elements according to their neighborhood;
- specifying the evaluation strategy (*i.e.*, the application of rules and termination).

Using these three implementation aspects (data structure, selection, and strategy), the chemical program can then be refined automatically into an efficient low-level program. The crucial methodological advantage is that logical issues are decoupled from efficiency issues.

This research, that takes place within the AUTOCHEM project (see Section 8.2.1), is the subject matter of Marnes Hoff's PhD thesis.

6.7.2. Efficient Parameter Search for Qualitative Models of Regulatory Networks using Symbolic Model Checking

Participant: Gregor Gössler.

A central problem in the analysis of biological regulatory networks concerns the relation between their structure and dynamics. This problem can be narrowed down to the following two questions: (a) Is a hypothesized structure of the network consistent with the observed behavior? (b) Can a proposed structure generate a desired behavior?

Qualitative models of regulatory networks, such as (synchronous or asynchronous) Boolean models and piecewise-affine differential equation (PADE) models, have been proven useful for addressing the above questions. The models are coarse-grained, in the sense that they do not explicitly specify the biochemical mechanisms. However, they include the logic of gene regulation and allow different expression levels of the genes to be distinguished.

Qualitative models bring specific advantages when studying the relation between structure and dynamics. In order to answer questions (a) and (b), one has to search the parameter space to check if for some parameter values the network is consistent with the data or can attain a desired control objective. In qualitative models, the number of different parametrizations is finite and the number of possible values for each parameter is usually rather low. This makes parameter search easier to handle than in quantitative models, where exhaustive search of the continuous parameter space is in general not feasible. Moreover, qualitative models are concerned with trends rather than with precise quantitative values, which corresponds to the nature of much of the available biological data.

Nevertheless, the parametrization of qualitative models remains a complex problem. For most models of networks of biological interest the state and parameter spaces are too large to exhaustively test all combinations of parameter values. The aim of this work was to address this search problem for PADE models by treating it in the context of formal verification and symbolic model checking.

Our contributions in [8] are twofold. On the methodological side, we have developed a method that makes it possible to efficiently analyze large and possibly incompletely parametrized PADE models. This is achieved by a symbolic encoding of the model structure, constraints on parameter values, and transition rules describing the qualitative dynamics of the system. We can thus take full advantage of symbolic model checkers for testing the consistency of the network structure with dynamic properties expressed in temporal logics. In comparison with related work, our method applies to incompletely instead of fully parametrized models, provides more precise results, and the encoding is efficient without (strongly) simplifying the PADE dynamics.

On the application side, we show that the method performs well on real problems, by means of the IRMA synthetic network and benchmark experimental datasets. More precisely, we are able to find parameter values for which the network satisfies temporal-logic properties describing observed expression profiles. Analysis of these parameter values reveals that biologically relevant constraints have been identified. Moreover, we make suggestions to improve the robustness of the external control of the IRMA behavior by proposing a rewiring of the network.

7. Contracts and Grants with Industry

7.1. Pôle de compétitivité Minalogic: OpenTLM

In the context of the Pôle de Compétitivité Minalogic, we participate in the four-year project OPENTLM on analysis of systems-on-chip modeled at the transaction level in SystemC [63]. We intend to develop methods for abstraction, and interprocedural and compositional analysis of SystemC models. Interesting results have been obtained regarding the precisio of abstract interpretation based analysis of concurrent systems (see Section 6.4.5). One PhD student and one engineer have been hired on this topics, resp. in April and May 2008.

8. Other Grants and Activities

8.1. Regional Actions

8.1.1. Regional Cluster ISLE

We participate in the regional cluster ISLE ("Informatique, Systèmes et Logiciels Embarqués") of the Région Rhône-Alpes, which funds the PhD of Mouaiad Alras (see Section 6.2.2).

8.2. National actions

8.2.1. ANR AutoChem: Chemical Programming

Participants: Pascal Fradet [contact person], Marnes Hoff.

The AUTOCHEM project aims at investigating and exploring the use of chemical languages (see Section 6.7.1) to program complex computing infrastructures such as grids and real-time deeply-embedded systems. The consortium includes INRIA Rennes – Bretagne Atlantique (PARIS team, Rennes), INRIA Grenoble – Rhône-Alpes (POP ART team, Montbonnot), IBISC (CNRS/Université d'Evry) and CEA List (Saclay). The project started at the end of 2007 and will terminate at the end of 2011.

8.2.2. ANR Asopt: Analyse Statique et OPTimisation

Participants: Bertrand Jeannet, Lies Lakhdar-Chaouch, Pascal Sotin, Peter Schrammel.

The ASOPT (Analyse Statique et OPTimisation) project [end of 2008-2011] brings together static analysis (INRIA-POP ART, VERIMAG, CEA LMeASI), optimisation, and control/game theory experts (CEA LMeASI, INRIA-MAXPLUS) around some program verification problems. POP ART is the project coordinator.

Many abstract interpretations attempt to find "good" geometric shapes verifying certain constraints; this not only applies to purely numerical abstractions (for numerical program variables), but also to abstractions of data structures (arrays and more complex shapes). This problem can often be addressed by optimisation techniques, opening the possibility of exploiting advanced techniques from mathematical programming.

The purpose of ASOPT is to develop new abstract domains and new resolution techniques for embedded control programs, and in the longer run, for numerical simulation programs.

8.2.3. ANR Vedecy: Verification and Design of Cyber-physical Systems

Participants: Gregor Gössler [contact person], Bertrand Jeannet.

The VEDECY project aims at pursuing fundamental research towards the development of algorithmic approaches to the verification and design of cyber-physical systems. Cyber-physical systems result from the integration of computations with physical processes: embedded computers control physical processes which in return affect computations through feedback loops. They are ubiquitous in current technology and their impact on lives of citizens is meant to grow in the future (autonomous vehicles, robotic surgery, energy efficient buildings, ...).

Cyber-physical systems applications are often safety critical and therefore reliability is a major requirement. To provide assurance of reliability, model based approaches and formal methods are appealing. Models of cyber-physical systems are heterogeneous by nature: discrete dynamic systems for computations and continuous differential equations for physical processes. The theory of hybrid systems offers a sound modeling framework for cyber-physical systems. The purpose of VEDECY is to develop hybrid systems techniques for the verification and the design of cyber-physical systems.

8.2.4. INRIA Large Scale Action Synchronics: Language Platform for Embedded System Design

Participants: Mouaiad Alras, Alain Girault, Bertrand Jeannet, Peter Schrammel.

The SYNCHRONICS (Language Platform for Embedded System Design) project [beginning of 2008-2011] gathers 9 permanent researchers on the topic of embedded systems design: B. Caillaud (INRIA Rennes – Bretagne Atlantique), A. Cohen, L. Mandel, and M. Pouzet (INRIA-Saclay and ENS Ulm), G. Delaval, A. Girault, and B. Jeannet (INRIA Grenoble – Rhône-Alpes), E. Jahier and P. Raymond (VERIMAG).

SYNCHRONICS capitalizes on recent extensions of data-flow synchronous languages, as well as relaxed forms of synchronous composition or compilation techniques for various platform, to address two main challenges with a language-centered approach: (i) the co-simulation of mixed discrete-continuous specifications, and more generally the co-simulation of programs and properties (either discrete or continuous); (ii) the ability, inside the programming model, to account for the architecture constraints (execution time, memory footprint, energy, power, reliability, etc.).

8.2.5. Collaborations inside Inria

- AOSTE at INRIA Paris – Rocquencourt is working with us on fault tolerant heuristics for their software SYNDEX.
- VERTECS at INRIA Rennes – Bretagne Atlantique is working with us on applications of discrete controller synthesis, and in particular on the tool SIGALI.
- P. Fradet cooperates with J.-P. Banâtre and T. Priol (PARIS, INRIA Rennes – Bretagne Atlantique) and with R. Douence (ASCOLA, Ecole des Mines de Nantes).
- A. Girault cooperates with D. Trystram (MOAIS, INRIA Grenoble – Rhône-Alpes) on scheduling and dependability, with E. Rutten (SARDES, INRIA Grenoble – Rhône-Alpes) and H. Marchand (VERTECS, INRIA Rennes – Bretagne Atlantique) on optimal discrete controller synthesis, with A. Benoit, F. Dufossé and Y. Robert (GRAAL, INRIA Grenoble – Rhône-Alpes) on multi-criteria scheduling, and with P. Raymond (VERIMAG, CNRS) on model-based design and a compilation tool chain from SIMULINK to distributed platforms.
- G. Gössler cooperates with D. Le Métayer (LICIT, INRIA Grenoble – Rhône-Alpes), H. de Jong and M. Page (IBIS, INRIA Grenoble – Rhône-Alpes), G. Batt (CONTRAINTES, INRIA Paris – Rocquencourt), G. Salaün (VASY project, INRIA Grenoble – Rhône-Alpes), and D.N. Xu (GALLIUM, INRIA Paris – Rocquencourt).
- B. Jeannet cooperates with T. Le Gall (VERTECS, INRIA Rennes – Bretagne Atlantique) on the analysis of communicating systems, and with C. Constant, T. Jéron and F. Ployette (VERTECS, INRIA Rennes – Bretagne Atlantique) on test generation.

- G. Delaval cooperates with H. Marchand (VERTECS, INRIA Rennes – Bretagne Atlantique) and É. Rutten (SARDES, INRIA Grenoble – Rhône-Alpes) on modular controller synthesis and its applications.
- G. Delaval, A. Girault and B. Jeannet collaborate with the PARKAS team of ENS Ulm (INRIA Paris – Rocquencourt) on the distribution of higher-order synchronous data-flow programs and on static analysis of hybrid programs.

8.2.6. Cooperations with other laboratories

- P. Fradet cooperates with J.-L. Giavitto (CNRS/Université d'Evry).
- A. Girault cooperates with P. Raymond (VERIMAG), P. Roop, Z. Salcic, and S. Andalam (University of Auckland, New Zealand), H. Kalla (University of Batna, Algeria), and I. Assayad (University of Casablanca, Morocco).
- P. Fradet and A. Girault collaborate with T. Ayav (University of Izmir, Turkey).
- G. Gössler cooperates with A. Girard (LJK, Grenoble), M. Bozga, T. Dang, and J. Sifakis (VERIMAG), J.-B. Raclet (IRIT, Toulouse), and B. Bonakdarpour (U. of Waterloo, Canada).
- A. Girault and G. Gössler collaborate with P. Roop and R. Sinha (University of Auckland, New Zealand).
- B. Jeannet cooperates with N. Halbwachs and M. Péron (VERIMAG) on static analysis and abstract interpretation.
- J.-B. Raclet cooperates with R. Passerone (University of Trento) on interface theories.

8.3. European actions

8.3.1. ArtistDesign European FP7 IST network of excellence

ARTISTDESIGN is a European Network of Excellence on embedded system design, successor of ARTIST II in FP7. The objective for ARTISTDESIGN is to build on existing structures and links forged in ARTIST II, to become a virtual Center of Excellence in Embedded Systems Design. This will be mainly achieved through tight integration between the central players of the European research community. The long-term vision for embedded systems in Europe, established in ARTIST II, will advance the emergence of Embedded Systems as a mature discipline. G. Gössler is the administrator of ARTISTDESIGN for INRIA.

8.3.2. Combest European FP7 IST STREP

COMBEST is a European STREP on formal component-based design of complex embedded systems[34]. Its goal is to develop a design theory for embedded systems, covering heterogeneity, interface specifications, composability, compositionality, and refinement for functional and extra-functional properties.

8.3.3. Cesar European Artemisia project

CESAR is a European ARTEMISIA project on cost-efficient methods and processes for safety relevant embedded systems[35]. It

We are particularly involved in the following sub-programs:

SP1: Task Force Safety 1.5.1 (State of the art survey on safety and diagnosability for cost-efficient safety critical emebedded systems) and 1.5.2 (Identification of requirements for comon cross domain core safety and diagnosability techniques and methods).

SP2: Requirements Engineering, along with two other INRIA teams (S4 and TRISKELL, from INRIA Rennes). We shall work on contracts based design for traceability.

[34] http://www.combest.eu
[35] http://www.cesarproject.eu

8.4. International actions

8.4.1. AFMES associated team

This collaboration involves two teams, the POP ART team and the ACEI team from the University of Auckland, New Zealand (led by Zoran Salcic, professor at the University of Auckland). It is funded by the Direction des Relations Internationales of INRIA and it started in January 2009[36].

We work on some of the most important challenges for the design of embedded systems. Let us recall that embedded systems are characterized by several constraints, such as enormous complexity and heterogeneity, need for determinism or bounded reaction time. Accordingly, design methods for embedded systems should, wherever possible, be automated and guarantee these properties by construction, therefore shifting the burden of checking these constraints from the programmer/system designer to the design method. In order to achieve this, our goal is to improve the existing design methods in several key directions: (i) incremental converter synthesis (see Section 6.5.1); (ii) programming language for adaptive computing – SYSTEMJ and beyond (see Section 6.3.1); (iii) time predictable programming language and execution architectures (see Section 6.3.2). Together, these advanced formal methods will provide foundations for automated design and higher level of safety of the designed embedded systems.

9. Dissemination

9.1. Scientific Community

- Pascal Fradet served in the program committee of AOSD'10 (*International Conference on Aspect-Oriented Software Development*). He is in the external review committee of PLDI'11 (*ACM SIGPLAN conference on Programming Language Design and Implementation*). He was in the selection committee for an assistant professor position at Institut Polytechnique de Bordeaux.
- Alain Girault served in the program committee of the DATE'10[37] and APSIPA–ASC'10[38] conferences. He was tutorial and invited speakers chair for the MEMOCODE'10[39]. He was in the selection committee for an assistant professor position at the University of Toulouse. Finally, he was on the jury of the EuroSys / Roger Needham PhD Award.
- Gregor Gössler served in the program committee of the FOCLASA'10 workshop on foundations of coordination languages and software architectures.

9.2. Teaching

9.2.1. Advising

PhDs:

- Marnes Hoff, co-advised by Pascal Fradet and Jean-Louis Giavitto (Université d'Evry), since 04/2008, PhD in computer science, Grenoble INP.
- Henri-Charles Blondeel, co-advised by P. Fradet and A. Girault, since 10/2010, PhD in computer science, Grenoble INP.
- Mouaiad Alras, co-advised by Alain Girault and Pascal Raymond (VERIMAG/CNRS), since 10/2006, PhD in computer science, UJF, Grenoble.
- Lies Lakhdar-Chaouch, co-advised by Alain Girault and Bertrand Jeannet since 05/2008, PhD in computer science, Grenoble INP.

[36] http://pop-art.inrialpes.fr/~girault/Projets/Afmes
[37] http://www.date-conference.com
[38] http://apsipa2010.i2r.a-star.edu.sg/v3/index.shtml
[39] http://www-memocode2010.imag.fr

- Peter Schrammel, co-advised by Alain Girault and Bertrand Jeannet since 07/2009, PhD in computer science, Grenoble INP.
- Gideon Smeding, co-advised by Gregor Gössler and Joseph Sifakis (VERIMAG/CNRS), since 12/2009, PhD in computer science, UJF, Grenoble.

9.2.2. University Teaching

Gwenaël Delaval is teaching algorithmics and programming at Université Joseph Fourier (96h in 2010–2011).

10. Bibliography

Major publications by the team in recent years

[1] T. AYAV, P. FRADET, A. GIRAULT. *Implementing fault-tolerance in real-time programs by automatic program transformations*, in "ACM Trans. Embedd. Comput. Syst.", July 2008, vol. 7, n° 4, p. 1–43.

[2] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects Preserving Properties*, in "Proc. of the ACM SIGPLAN 2008 Symposium on Partial Evaluation and Program Manipulation (PEPM'08)", ACM, january 2008, p. 135-145.

[3] A. GIRAULT, H. KALLA. *A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate*, in "IEEE Trans. Dependable Secure Comput.", December 2009, vol. 6, n° 4, p. 241–254, Research report INRIA 6319, http://hal.inria.fr/inria-00177117.

[4] A. GIRAULT, É. RUTTEN. *Automating the Addition of Fault Tolerance with Discrete Controller Synthesis*, in "Formal Methods in System Design", October 2009, vol. 35, n° 2, p. 190–225, http://www.springerlink.com/content/w726262156h4822j.

[5] G. GÖSSLER, J. SIFAKIS. *Composition for Component-based Modeling*, in "Science of Computer Programming", 2005, vol. 55, n° 1-3, p. 161-183.

[6] G. GÖSSLER, D. LE MÉTAYER, J.-B. RACLET. *Causality Analysis in Contract Violation*, in "RV", LNCS, Springer-Verlag, 2010, p. 270-284, http://dx.doi.org/10.1007/978-3-642-16612-9_21.

[7] T. LE GALL, B. JEANNET. *Lattice automata: a representation of languages over an infinite alphabet, and some applications to verification*, in "Static Analysis Symposium, SAS'07", LNCS, August 2007, vol. 4634, http://pop-art.inrialpes.fr/people/bjeannet/publications/sas07.ps.

Publications of the year

Articles in International Peer-Reviewed Journal

[8] G. BATT, M. PAGE, I. CANTONE, G. GÖSSLER, P. MONTEIRO, H. DE JONG. *Efficient parameter search for qualitative models of regulatory networks using symbolic model checking*, in "Bioinformatics", 2010, vol. 26, n° 18, http://dx.doi.org/10.1093/bioinformatics/btq387.

[9] X. BRIAND, B. JEANNET. *Combining control and data abstraction in the verification of hybrid systems*, in "Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)", 2010, vol. 29, n° 10 [*DOI :* 10.1109/TCAD.2010.2066010], http://pop-art.inrialpes.fr/people/bjeannet/publications/tcad10.pdf.

[10] P. FRADET, S. HONG TUAN HA. *Aspects of Availability - Enforcing timed properties to prevent denial of service*, in "Science of Computer Programming", July 2010, vol. 75, n^{o} 7, p. 516-542.

[11] G. GÖSSLER. *Component-based Modeling and Reachability Analysis of Genetic Networks*, in "ACM/IEEE TCBB", 2011.

[12] B. JEANNET, A. LOGINOV, T. REPS, M. SAGIV. *A relational approach to interprocedural shape analysis*, in "ACM Trans. On Programming Languages and Systems (TOPLAS)", 2010, vol. 32, n^{o} 2 [*DOI :* 10.1145/1667048.1667050], http://pop-art.inrialpes.fr/people/bjeannet/publications/toplas10.pdf.

[13] A. MALIK, Z. SALCIC, P. ROOP, A. GIRAULT. *SystemJ: A GALS Language for System Level Design*, in "Computer Languages, Systems and Structures", December 2010, vol. 36, n^{o} 4, p. 317–344.

International Peer-Reviewed Conference/Proceedings

[14] S. ANDALAM, P. ROOP, A. GIRAULT. *Deterministic, Predictable and Light-Weight Multithreading Using PRET-C*, in "Design Automation and Test in Europe Conference, DATE'10", Dresden, Germany, April 2010.

[15] S. ANDALAM, P. ROOP, A. GIRAULT. *Predictable Multithreading of Embedded Applications Using PRET-C*, in "International Conference on Formal Methods and Models for Codesign, MEMOCODE'10", Grenoble, France, July 2010.

[16] A. BENOIT, F. DUFOSSÉ, A. GIRAULT, Y. ROBERT. *Reliability and Performance Optimization of Pipelined Real-Time Systems*, in "International Conference on Parallel Processing, ICPP'10", San Diego (CA), USA, September 2010.

[17] J. CÁMARA, A. GIRARD, G. GÖSSLER. *Synthesis of Switching Controllers using Approximately Bisimilar Multiscale Abstractions*, in "HSCC'11", ACM, 2011, accepted.

[18] N. DE PALMA, G. DELAVAL, É. RUTTEN. *QoS and Energy Management Coordination using Discrete Controller Synthesis*, in "1st International Workshop on Green Computing Middleware (GCM'2010)", Bangalore, India, November 2010, http://pop-art.inrialpes.fr/people/delaval/pub/delaval-gcm10.pdf.

[19] E. DUMITRESCU, A. GIRAULT, H. MARCHAND, É. RUTTEN. *Multicriteria Optimal Reconfiguration of Fault-Tolerant Real-Time Tasks*, in "Workshop on Discrete Event Systems, WODES'10", Berlin, Germany, IFAC, September 2010.

[20] P. FRADET, J.-L. GIAVITTO, M. HOFF. *Refinement of Chemical Programs using Strategies*, in "International Workshop on Strategies in Rewriting, Proving, and Programming, IWS'2010", July 2010.

[21] G. GÖSSLER, D. LE MÉTAYER, J.-B. RACLET. *Causality Analysis in Contract Violation*, in "RV", LNCS, Springer-Verlag, 2010, p. 270-284, http://dx.doi.org/10.1007/978-3-642-16612-9_21.

[22] B. JEANNET. *Some Experience on the Software Engineering of Abstract Interpretation Tools*, in "Int. Workshop on Tools for Automatic Program AnalysiS, TAPAS'2010", ENTCS, Elsevier, 2010, vol. 267, n^{o} 2, p. 29–42, http://dx.doi.org/10.1016/j.entcs.2010.09.016.

[23] P. SCHRAMMEL, B. JEANNET. *Extending Abstract Acceleration to Data-Flow Programs with Numerical Inputs*, in "Int. Workshop on Numerical and Symbolic Abstract Domains, NSAD'2010", ENTCS, Elsevier, 2010, vol. 267, n^{o} 1, p. 101–114, http://dx.doi.org/10.1016/j.entcs.2010.09.009.

[24] P. SCHRAMMEL, B. JEANNET. *Extending Abstract Acceleration to Data-Flow Programs with Numerical Inputs*, in "Int. Workshop on Numerical and Symbolic Abstract Domains", ENTCS, Elsevier, 2010, vol. 267, n^{o} 1, p. 101–114.

[25] P. SOTIN, B. JEANNET, X. RIVAL. *Concrete Memory Models for Shape Analysis*, in "Int. Workshop on Numerical and Symbolic Abstract Domains, NSAD'2010", ENTCS, Elsevier, 2010, vol. 267, n^{o} 1, p. 139–150, http://dx.doi.org/10.1016/j.entcs.2010.09.012.

[26] D. XU, G. GÖSSLER, A. GIRAULT. *Probabilistic Contracts for Component-based Design*, in "International Symposium on Automated Technology for Verification and Analysis, ATVA'10", Singapore, A. BOUAJJANI, W.-N. CHIN (editors), LNCS, Springer-Verlag, September 2010, vol. 6252, p. 325–340.

Other Publications

[27] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects Preserving Properties*, September 2010, (submitted to journal publication).

References in notes

[28] *Norme Internationale – Automates programmables : Langages de programmation*, CEI (Commission Électrotechnique Internationale), 1993.

[29] C. ALLEAUNE, V.-L. BENABOU, D. BERAS, C. BIDAN, N. CRAIPEAU, S. FRÉNOT, G. GÖSSLER, R. HARDOUIN, J. LE CLAINCHE, D. LE MÉTAYER, M. MAAREK, E. MAZZA, L. MÉ, M.-L. POTET, S. STEER, V. VIET TRIEM TONG. *Liability in Software Engineering: Overview of the LISE Approach and Illustration on a Case Study*, INRIA, 2009, n^{o} RR-7148, http://hal.inria.fr/inria-00440437/PDF/LISE-HAL-INRIA-RR-7148.pdf.

[30] M. ALRAS, P. CASPI, A. GIRAULT, P. RAYMOND. *Model-Based Design of Embeded Control Systems by means of a Synchronous Intermediate Model*, in "International Conference on Embedded Systems and Software, ICESS'09", Hangzhou, China, IEEE, Los Alamitos, CA, May 2009, p. 3–10.

[31] A. ARNOLD. *Systèmes de transitions finis et sémantique des processus communicants*, Masson, 1992.

[32] A. ARNOLD, A. VINCENT, I. WALUKIEWICZ. *Games for synthesis of controllers with partial observation*, in "Theoretical Computer Science", 2003, vol. 28, n^{o} 1, p. 7-34.

[33] E. ASARIN, O. BOURNEZ, T. DANG, O. MALER, A. PNUELI. *Effective Synthesis of Switching Controllers for Linear Systems*, in "Proceedings of the IEEE", 2000, vol. 88, n^{o} 7, p. 1011–1025.

[34] J.-R. BEAUVAIS, É. RUTTEN, T. GAUTIER, R. HOUDEBINE, P. LE GUERNIC, Y.-M. TANG. *Modelling Statecharts and Activity Charts as Signal Equations*, in "ACM Transactions on Software Engineering and Methodology", October 2001, vol. 10, n^{o} 4, p. 397–451.

[35] F. BOURDONCLE. *Interprocedural Abstract Interpretation of Block Structured Languages with Nested Procedures, Aliasing and Recursivity*, in "PLILP", 1990, p. 307-323, http://dx.doi.org/10.1007/BFb0024192.

[36] X. BRIAND, B. JEANNET. *Combining control and data abstraction in the verification of hybrid systems*, in "Formal Methods and Models for Codesign, MEMOCODE'2009", IEEE, 2009.

[37] R. BRYANT. *Graph-based algorithms for boolean function manipulation*, in "IEEE Transactions on Computers", 1986, vol. C-35, n^{o} 8, p. 677–692.

[38] P. CASPI, M. POUZET. *Synchronous Kahn Networks*, in "ACM SIGPLAN International Conference on Functional Programming, ICFP'96", Philadelphia (PA), USA, ACM Press, May 1996.

[39] C. CASSANDRAS, S. LAFORTUNE. *Introduction to Discrete Event Systems*, Kluwer, 1999.

[40] B.-Y. E. CHANG, X. RIVAL. *Relational inductive shape analysis*, in "Principles of Programming Languages, POPL'08", ACM, 2008, http://doi.acm.org/10.1145/1328438.1328469.

[41] D. CHASE, M. WEGMAN, F. ZADECK. *Analysis of Pointers and Structures*, in "Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation", ACM Press, 1990, p. 296–310, http://doi.acm.org/10.1145/93542.93585.

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

[43] D. CLARKE, T. JÉRON, V. RUSU, E. ZINOVIEVA. *STG: a Symbolic Test Generation tool*, in "(Tool paper) Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)", LNCS, 2002, vol. 2280.

[44] P. COUSOT, R. COUSOT. *Abstract Interpretation and Application to Logic Programs*, in "Journal of Logic Programming", 1992, vol. 13, n^{o} 2–3, p. 103–179.

[45] P. COUSOT, N. HALBWACHS. *Automatic discovery of linear restraints among variables of a program*, in "5th ACM Symposium on Principles of Programming Languages, POPL'78", Tucson (Arizona), January 1978.

[46] P. D'ARGENIO, B. JEANNET, H. JENSEN, K. LARSEN. *Reduction and Refinement Strategies for Probabilistic Analysis*, in "Process Algebra and Probabilistic Methods - Performance Modelling and Verification, PAPM-PROBMIV'02", Copenhagen (Denmark), LNCS, July 2002, vol. 2399.

[47] G. DELAVAL. *Répartition modulaire de programmes synchrones*, INPG, INRIA Grenoble Rhône-Alpes, projet Pop-Art, July 2008.

[48] G. DELAVAL, A. GIRAULT, M. POUZET. *A Type System for the Automatic Distribution of Higher-order Synchronous Dataflow Programs*, in "International Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES'08", Tucson (AZ), USA, ACM, June 2008, p. 101–110, ftp://ftp.inrialpes.fr/pub/bip/pub/girault/Publications/Lctes08/main.pdf.

[49] G. DELAVAL, H. MARCHAND, É. RUTTEN. *Contracts for Modular Discrete Controller Synthesis*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2010)", Stockholm, Sweden, April 2010, http://pop-art.inrialpes.fr/people/delaval/pub/lctes2010.pdf.

[50] D. DELMAS, E. GOUBAULT, S. PUTOT, J. SOUYRIS, K. TEKKAL, F. VÉDRINE. *Towards an Industrial Use of FLUCTUAT on Safety-Critical Avionics Software*, in "Formal Methods for Industrial Critical Systems, FMICS'09", LNCS, 2009, vol. 5825.

[51] F. GAUCHER, E. JAHIER, B. JEANNET, F. MARANINCHI. *Automatic State Reaching for Debugging Reactive Programs*, in "5th Int. Workshop on Automated and Algorithmic Debugging, AADEBUG'03", September 2003.

[52] K. GHORBAL, E. GOUBAULT, S. PUTOT. *The Zonotope Abstract Domain Taylor1+*, in "Computer Aided Verification, CAV'09", LNCS, 2009, vol. 5643.

[53] A. GIRARD, G. PAPPAS. *Approximation metrics for discrete and continuous systems*, in "IEEE Trans. on Automatic Control", 2007, vol. 52, n^{o} 5, p. 782-798.

[54] A. GIRAULT. *System-Level Design of Fault-Tolerant Embedded Systems*, October 2006, vol. 67.

[55] A. GIRAULT, H. KALLA. *A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate*, in "IEEE Trans. Dependable Secure Comput.", December 2009, vol. 6, n^{o} 4, p. 241–254, Research report INRIA 6319, http://www.computer.org/portal/web/csdl/doi/10.1109/TDSC.2008.50.

[56] A. GIRAULT, H. KALLA, M. SIGHIREANU, Y. SOREL. *An Algorithm for Automatically Obtaining Distributed and Fault-Tolerant Static Schedules*, in "International Conference on Dependable Systems and Networks, DSN'03", San-Francisco (CA), USA, IEEE, June 2003.

[57] A. GIRAULT, H. KALLA, Y. SOREL. *Transient Processor/Bus Fault Tolerance for Embedded Systems*, in "IFIP Working Conference on Distributed and Parallel Embedded Systems, DIPES'06", Braga, Portugal, Springer, October 2006, p. 135–144.

[58] A. GIRAULT, É. RUTTEN. *Automating the Addition of Fault Tolerance with Discrete Controller Synthesis*, in "Formal Methods in System Design", October 2009, vol. 35, n^{o} 2, p. 190–225, http://www.springerlink.com/content/w726262156h4822j.

[59] A. GIRAULT, E. SAULE, D. TRYSTRAM. *Reliability Versus Performance for Critical Applications*, in "J. of Parallel and Distributed Computing", March 2009, vol. 69, n^{o} 3, p. 326–336.

[60] L. GONNORD. *Accélération abstraite pour l'amélioration de la précision en Analyse des Relations Linéaires*, Université Joseph Fourier, Grenoble, October 2007.

[61] L. GONNORD, N. HALBWACHS. *Combining widening and acceleration in linear relation analysis*, in "Static Analysis Symposium (SAS)", Seoul, Korea, Aug 2006, p. 144–160.

[62] D. GOPAN, T. REPS. *Guided Static Analysis*, in "Static Analysis Symposium, SAS'07", LNCS, August 2007, vol. 4634, http://dx.doi.org/10.1007/978-3-540-74061-2_22.

[63] T. GRÖTKER, S. LIAO, G. MARTIN, S. SWAN. *System Design with SystemC*, Kluwer, 2002.

[64] G. GÖSSLER. *Component-based Design of Heterogeneous Reactive Systems in* PROMETHEUS, INRIA, 2006, n^{o} 6057.

[65] G. GÖSSLER, J. SIFAKIS. *Priority Systems*, in "Proc. FMCO'03", F. DE BOER, M. BONSANGUE, S. GRAF, W.-P. DE ROEVER (editors), LNCS, Springer-Verlag, 2004, vol. 3188, p. 314-329.

[66] N. HALBWACHS, P. CASPI, P. RAYMOND, D. PILAUD. *The Synchronous Data-Flow Programming Language Lustre*, in "Proceedings of the IEEE", September 1991, vol. 79, n^{o} 9, p. 1305–1320.

[67] N. HALBWACHS. *Synchronous Programming of Reactive Systems*, Kluwer, 1993.

[68] N. HALBWACHS. *Synchronous Programming of Reactive Systems – a Tutorial and Commented Bibliography*, in "Proc. of the Int. Conf. on Computer-Aided Verification, CAV'98, Vancouver, Canada", Springer-Verlag, 1998, LNCS Vol. 1427.

[69] D. HAREL. *Statecharts: A Visual Formalism for Complex Systems*, in "Science of Computer Programming", 1987, vol. 8, p. 231-274.

[70] H. HERMANNS. *Interactive Markov Chains: The Quest for Quantified Quality*, LNCS, Springer, 2002, vol. 2428.

[71] B. JEANNET, P. D'ARGENIO, K. LARSEN. *RAPTURE: A tool for verifying Markov Decision Processes*, in "Tools Day, International Conference on Concurrency Theory, CONCUR'02", Brno (Czech Republic), August 2002, Technical Report, Faculty of Informatics at Masaryk University Brno.

[72] B. JEANNET. *Dynamic Partitioning in Linear Relation Analysis. Application To The Verification Of Reactive Systems*, in "Formal Methods in System Design", July 2003, vol. 23, n^{o} 1, p. 5–37.

[73] B. JEANNET. *Relational interprocedural verification of concurrent programs*, in "Proc. of Software Engineering and Formal Methods, SEFM'09", IEEE, November 2009, p. 83-92.

[74] B. JEANNET, T. JÉRON, V. RUSU, E. ZINOVIEVA. *Symbolic Test Selection based on Approximate Analysis*, in "11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)", Edinburgh (UK), LNCS, April 2005, vol. 3440.

[75] L. LAMPORT. *Time, Clocks, and the Ordering of Events in a Distributed System*, in "CACM", 1978, vol. 21, n^{o} 7, 558 565.

[76] O. MALER, A. PNUELI, J. SIFAKIS. *On the Synthesis of Discrete Controllers for Timed Systems*, in "Proc. of STACS'95", LNCS, Springer Verlag, 1995, vol. 900.

[77] B. MEYER. *Applying "Design by Contract"*, in "IEEE Computer", 1992, vol. 25, n^{o} 10, p. 40-51.

[78] R. MILNER. *Communication and Concurrency*, Prentice Hall, 1989.

[79] J.-P. QUEILLE, J. SIFAKIS. *Specification and Verification of Concurrent Systems in CESAR*, in "Proc. International Symposium on Programming", LNCS, Springer-Verlag, 1982, vol. 137, p. 337-351.

[80] P. RAMADGE, W. WONHAM. *Supervisory Control of a Class of Discrete Event Processes*, in "SIAM Journal on control and optimization", January 1987, vol. 25, n^{o} 1, p. 206–230.

[81] P. RAMADGE, W. WONHAM. *The Control of Discrete Event Systems*, in "Proceedings of the IEEE", 1989, vol. 77, n^{o} 1.

[82] P. ROOP, A. GIRAULT, R. SINHA, G. GÖSSLER. *Specification Enforcing Refinement for Convertibility Verification*, in "Proc. ACSD'09", S. EDWARDS, R. LORENZ, W. VOGLER (editors), IEEE, 2009, p. 148-157.

[83] M. SAGIV, T. REPS, R. WILHELM. *Parametric shape analysis via 3-valued logic*, in "ACM Transactions on Prog. Languages and Systems", 2002, vol. 24, n^{o} 3.

[84] P. SOTIN, B. JEANNET. *Precise Interprocedural Analysis in the Presence of Pointers to the Stack*, in "European Symposium on Programming, ESOP'11", LNCS, 2011, accepted to the conference.

[85] P. TABUADA. *Verification and Control of Hybrid Systems - A Symbolic Approach*, Springer, 2009.

[86] L. DE ALFARO, T. HENZINGER. *Interface Automata*, in "Proc. 9th Annual Symposium on Foundations of Software Engineering (FSE)", ACM Press, 2001, p. 109-120.