IN PARTNERSHIP WITH:
CNRS

Institut polytechnique de Grenoble

Université Joseph Fourier (Grenoble)

Activity Report 2012

Project-Team POP ART

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

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

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Embedded and Real Time Systems

Table of contents

1. Members **1**
2. Overall Objectives **1**
3. Scientific Foundations **2**
3.1. Embedded systems and their safe design 2
3.1.1. Safe Design of Embedded Real-time Control Systems 2
3.1.2. Models, Methods and Techniques 2
3.2. Issues in Design Automation for Complex Systems 3
3.2.1. Hard Problems 3
3.2.2. Applicative Needs 4
3.2.3. Our Approach 4
3.3. Main Research Directions 4
3.3.1. Component-Based Design 5
3.3.2. Programming for Embedded Systems 5
3.3.3. Dependable Embedded Systems 5
4. Application Domains **6**
4.1. Industrial Applications 6
4.2. Industrial Design Tools 6
4.3. Current Industrial Cooperations 6
5. Software **6**
5.1. NBac 6
5.2. ReaVer 7
5.3. Implementations of Synchronous Programs 7
5.4. Apron and BddApron Libraries 8
5.4.1. Principles 8
5.4.2. Implementation and Distribution 8
5.5. Prototypes 9
5.5.1. Logical Causality 9
5.5.2. Cosyma 10
5.5.3. Automatic Controller Generation 10
5.5.4. Rapture 10
5.5.5. The Interproc family of static analyzers 10
5.5.6. Heptagon/BZR 11
6. New Results **11**
6.1. Dependable Distributed Real-time Embedded Systems 11
6.2. Controller Synthesis for the Safe Design of Embedded Systems 12
6.2.1. Synthesis of switching controllers using approximately bisimilar multiscale abstractions 12
6.2.2. Modular discrete controller synthesis 13
6.3. Automatic Distribution of Synchronous Programs 13
6.3.1. Modular distribution 13
6.3.2. Distribution of synchronous programs under real-time constraints 14
6.4. New Programming Languages for Embedded Systems 14
6.5. Static Analysis and Abstract Interpretation 15
6.5.1. Translating data-flow languages for hybrid systems simulation to hybrid automata for hybrid systems verification 15
6.5.2. Abstract Acceleration of general linear loops 15
6.5.3. Logico-Numerical Max-Strategy Iteration 16
6.6. Component-Based Construction 16
6.6.1. Incremental converter synthesis 16
6.6.2. Analysis of logical causality 17

6.6.3. A Theory of fault recovery for component-based models 17
6.7. Aspect-Oriented Programming 17
6.7.1. Aspects preserving properties 17
6.7.2. Fault tolerance aspects 18
7. Bilateral Contracts and Grants with Industry **18**
8. Partnerships and Cooperations **18**
8.1. National Initiatives 18
8.1.1. Inria Large Scale Actions 18
8.1.2. ANR 19
8.1.2.1. ANR Asopt: Analyse Statique et OPTimisation 19
8.1.2.2. ANR Vedecy: Verification and Design of Cyber-physical Systems 19
8.2. International Initiatives 19
8.3. International Research Visitors 20
8.3.1. Visits of International Scientists 20
8.3.2. Visits to International Teams 20
9. Dissemination **20**
9.1. Scientific Animation 20
9.2. Teaching - Supervision - Juries 20
9.2.1. Supervision 20
9.2.2. Juries 21
10. Bibliography **21**

Project-Team POP ART

Keywords: Aspect Oriented Programming, Embedded Systems, Fault Tolerance, Scheduling, Verification

Creation of the Project-Team: July 01, 2003 .

1. Members

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

Faculty Member
Gwenaël Delaval [Associate professor, Université Joseph Fourier]

External Collaborators
Emil Dumitrescu [Associate Professor, INSA Lyon]
Xavier Nicollin [Associate Professor, Grenoble INP]

PhD Students
Vagelis Bebelis [CIFRE STMicroelectronics]
Peter Schrammel [INRIA, SYNCHRONICS project, until 10/2012]
Gideon Smeding [DIGITEO grant]
Dmitry Burlyaev [MNR grant, since 10/2012]

Post-Doctoral Fellows
Roopak Sinha [INRIA, CESAR project, until 02/2012]
Wei-Tsun Sun [INRIA grant, since 12/2012]
Sebti Mouelhi [VEDECY project, until 09/2012]

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 (discrete) control theory as well as computer science. Application domains are typically safety-critical systems, as in transportation (avionics, railways, automotive), production, medical, or energy production systems. These application domains require both formal methods and models for the construction of correct systems. Such design methods must be implemented in computer assisted design tools, targeted at specialists of the applications. We contribute to this research domain by offering solutions all along the design flow, from the specification to the implementation: we develop techniques for the specification, the programmation and the 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 and fault tolerance;

- high-level design and programming methods, with support for automated code generation, including: the automated generation of correct controllers using discrete control synthesis; compositionality for the verification and construction of correct systems; component-based and contract-based design methods; and aspect-oriented programming;
- static analysis and abstract interpretation techniques, which are applied both to low-level synchronous models/programs and to more general imperative or concurrent 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 [1], on embedded real-time systems;
- an FP7 European STREP project: COMBEST [2] on component-based design;
- an ARTEMISIA European project: CESAR [3] on cost-efficient methods and processes for safety relevant embedded systems;
- three ANR French projects: ASOPT (on static analysis), CTRL-GREEN (on control of autonomic systems) and VEDECY (on cyber-physical systems);
- an INRIA large scale action: SYNCHRONICS [4] on a language platform for embedded system design;
- an INRIA associated team with the University of Auckland (New Zealand): AFMES [5] 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. Such systems are known under various names, notably *cyberphysical systems* and *embedded control systems*. What is important is to design and to analyze 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, railways, automotive), production, medical, or energy production systems (nuclear).

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.

[1] http://www.artist-embedded.org
[2] http://www.combest.eu/home
[3] http://www.cesarproject.eu
[4] http://www.inria.fr/en/research/research-fields/large-scale-initiatives
[5] 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 [70], [41], control synthesis [72] and compilation, as well as extensions to timed and hybrid systems [69], [32] 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.

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 use BDD techniques [37] for manipulating Boolean functions and sets, and their MTBDD extension for more general functions. Abstract interpretation [43] 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 analyses based on abstract interpretation are linear relation analysis [44] and shape analysis [40].

The synchronous approach [6] [60], [61] to reactive systems design gave birth to complete programming environments, with languages like ARGOS, LUSTRE [7], ESTEREL [8], SIGNAL/ POLYCHRONY [9], LUCID SYNCHRONE [10], SYNDEX [11], 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.

[6] http://www.synalp.org
[7] http://www-verimag.imag.fr/SYNCHRONE
[8] http://www.inria.fr/equipes/aoste
[9] http://www.irisa.fr/espresso/Polychrony
[10] http://www.di.ens.fr/~pouzet/lucid-synchrone/
[11] http://www-rocq.inria.fr/syndex

3.2.2. Applicative Needs

Applicative needs initially come from the fields of safety-critical systems (*e.g.*, avionics, nuclear) 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 [29]) 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, the possible failures, or the interactions between the defined automatic control and its implementation. 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 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.

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.

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, at 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. We contribute to the compiler of the HEPTAGON language (whose main inventor is Marc Pouzet, ENS Paris, PARKAS team), 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 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 that 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 first 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. A first step in this direction has been the design of automatic transformation method for fault tolerance, which implement a limited (but nonetheless interesting) form of AOP.

4. Application Domains

4.1. Industrial Applications

Our applications are in the embedded system area, typically: transportation, energy production, robotics, 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 the quality of designs, as well as the cost of the programmation and the validation processes.

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 [12]) and execution platforms (OS such as VXWORKS, QNX, real-time versions of LINUX ...) starts now to provide besides their core functionalities 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.

Regarding the synchronous approach, commercial tools are available: SCADE [13] (based on LUSTRE), CONTROLBUILD and RT-BUILDER (based on SIGNAL) from GEENSYS [14] (part of DASSAULT SYSTEMES), specialized environments like CELLCONTROL for industrial automatism (by the INRIA spin-off ATHYS– now part of DASSAULT SYSTEMES). One can observe 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 dynamic data-flow models of computation for streaming applications, dedicated to high definition video applications for their new STHORM manycore chip.

5. Software

5.1. NBac

Participant: Bertrand Jeannet.

[12] http://www.dspaceinc.com
[13] http://www.esterel-technologies.com
[14] http://www.geensoft.com

NBAC (Numerical and Boolean Automaton Checker) [15] 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 [63]. 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.

It has been used for verifying and debugging LUSTRE programs [65] [52] [36]. It is connected to the LUSTRE toolset [16]. 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 has been used by the test generator STG [17] [42] [66] for selecting test cases.

It has recently been superseded by REAVER (see Section 5.2).

5.2. ReaVer

Participant: Peter Schrammel.

REAVER (REActive VERifier [18]) is a tool framework for the safety verification of discrete and hybrid systems specified by logico-numerical data-flow languages, like LUSTRE, LUCID SYNCHRONE or ZELUS. It provides time-unbounded analysis based on abstract interpretation techniques. In many aspects it is the successor of NBAC (see Section 5.1).

It features partitioning techniques and several logico-numerical analysis methods based on Kleene iteration with widening and descending iterations, abstract acceleration, max-strategy iteration, and relational abstractions; logico-numerical product and power domains (based on the APRON and BddApron domain libraries) with convex polyhedra, octagons, intervals, and template polyhedra; and frontends for the hybrid NBAC format, LUSTRE via lus2nbac, and ZELUS/LUCID SYNCHRONE. Compared to NBAC, it is connected to higher-level, more recent synchronous and hybrid languages, and provides much more options regarding analysis techniques.

It has been used for several experimental comparisons published in papers and it integrates all the methods developped by Peter Schrammel in its PhD.

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]. Our first scheduling heuristic produces static multiprocessor schedules tolerant to a specified number of processor and communication link failures [55]. 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

[15] http://pop-art.inrialpes.fr/people/bjeannet/nbac/
[16] http://www-verimag.imag.fr/The-Lustre-Toolbox.html
[17] http://www.irisa.fr/prive/ployette/stg-doc/stg-web.html
[18] http://members.ktvam.at/schrammel/research/reaver
[19] http://www-rocq.inria.fr/syndex

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 [3] [33][17], [11]. 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 [43]. 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 this does not ease their diffusion and experimental comparison *w.r.t.* efficiency and precision aspects.

The APRON library provides:

- 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 designer 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, which allows experimenting, comparing, and combining abstract domains.

In 2011, the Taylor1plus domain [53], which is the underlying abstract domain of the tool FLUCTUAT [51] has been improved. Glue code has also been added to enable the connection of an abstract domain implemented in OCaml to the APRON infrastructure written in C (this requires callbacks from C to OCaml that are safe *w.r.t.* garbage collection). This will enable the integration in APRON of the MaxPlus polyhedra library written by X. Allamigeon [30] in the context of the ANR ASOPT project.

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, bitvectors) and numerical variables (integers, rationals, floating-point numbers). It first allows the manipulation of 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.

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 has been 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 is currently packaged as a REDHAT and DEBIAN package.

[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

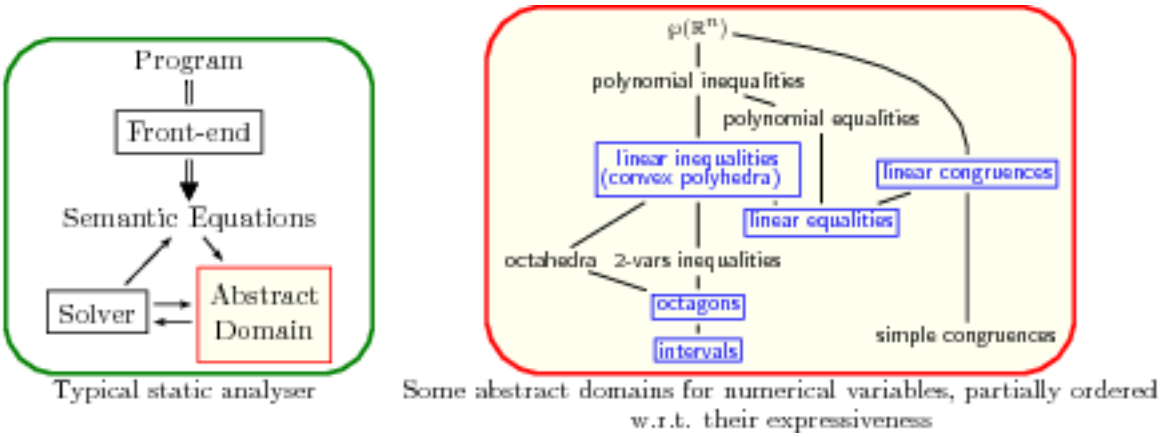

Figure 1. Typical static analyser and examples of abstract domains

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.

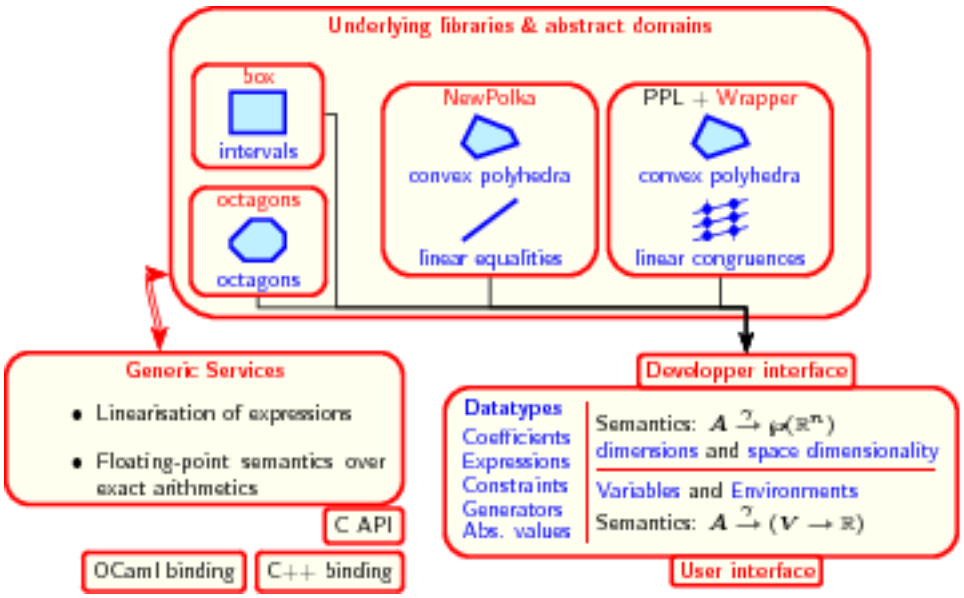

Figure 2. Organisation of the APRON *library*

5.5. Prototypes

5.5.1. Logical Causality

Participant: Gregor Goessler [contact person].

We have developed LOCA, a new prototype tool written in Scala that implements the analysis of logical causality described in 6.6.2. LOCA currently supports causality analysis in BIP. The core analysis engine is implemented as an abstract class, such that support for other models of computation (MOC) can be added by instantiating the class with the basic operations of the MOC.

5.5.2. Cosyma

Participants: Gregor Goessler [contact person], Sebti Mouelhi.

We have developed COSYMA, a tool for automatic controller synthesis for incrementally stable switched systems based on multi-scale discrete abstractions (see 6.2.1). The tool accepts a description of a switched system represented by a set of differential equations and the sampling parameters used to define an approximation of the state-space on which discrete abstractions are computed. The tool generates a controller — if it exists — for the system that enforces a given safety or time-bounded reachability specification.

5.5.3. 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]. It is the result of a collaboration with Eric Rutten from the SARDES team.

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.

5.5.4. Rapture

Participant: Bertrand Jeannet.

RAPTURE [25] [64] [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.

5.5.5. The Interproc family of static analyzers

Participant: Bertrand Jeannet [contact person].

[23] http://www.irisa.fr/vertecs/Logiciels/sigali.html
[24] http://www-verimag.imag.fr
[25] http://pop-art.inrialpes.fr/people/bjeannet/rapture/rapture.html

These analyzers and libraries are of general use for people working in the static analysis and abstract interpretation community, and serve as an experimental platform for the ANR project ASOPT (see §8.1.2.1).

- 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 [59].
- 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 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].

5.5.6. 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. It is developped within the SYNCHRONICS INRIA large-scale action.

HEPTAGON has been used to built BZR [32], 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 [33]. 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: Gwenaël Delaval, Pascal Fradet, Alain Girault [contact person], Emil Dumitrescu.

6.1.1. Tradeoff exploration between reliability, power consumption, and execution time

For autonomous critical real-time embedded systems (*e.g.*, satellite), guaranteeing a very high level of reliability is as important as keeping the power consumption as low as possible. We have designed an off-line ready list scheduling heuristics which, from a given software application graph and a given multiprocessor architecture (homogeneous and fully connected), 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*

[26] http://http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/fixpoint
[27] http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/interproc
[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://bzr.inria.fr
[33] http://www.irisa.fr/vertecs/Logiciels/sigali.html

voltage and frequency scaling to lower the power consumption [17], [11]. By running TSH on a single problem instance, we are able to provide the Pareto front for this instance in 3D, therefore exposing the user to several tradeoffs between the power consumption, the reliability and the execution time. The new contribution for 2012 has been the formulation of a new multi-criteria cost function for our ready list scheduling heuristics, such that we are able to prove rigorously that the static schedules we generate meet both the reliability constraint and the power consumption constraint.

Thanks to extensive simulation results, we have shown how TSH behaves in practice. Firstly, we have compared TSH versus an optimal Mixed Linear Integer Program on small instances; the experimental results show that TSH behaves very well compared to the the ILP. Secondly, we have compared TSH versus the ECS heuristic (Energy-Conscious Scheduling [68]); the experimental results show that TSH performs systematically better than ECS.

This is a joint work with Ismail Assayad (U. Casablanca, Morocco) and Hamoudi Kalla (U. Batna, Algeria), who both visit the team regularly.

6.2. Controller Synthesis for the Safe Design of Embedded Systems

Participants: Gwenaël Delaval [contact person], Gregor Goessler, Sebti Mouelhi.

6.2.1. 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.*, [73] 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 [71]. The first attempts to compute discrete abstractions for hybrid systems were based on traditional systems behavioral relationships such as simulation or bisimulation, 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 [54].

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 [45] we have proposed a technique for the synthesis of safety controllers for switched systems using multi-scale abstractions that allow us to deal with fast switching while keeping the number of states in the abstraction at a reasonable level. The finest scales of the abstraction are effectively explored only when fast switching is needed, that is when the system approaches the unsafe set.

We have extended the approach of [45] to the synthesis of controllers for time-bounded reachability. Furthermore we have implemented the algorithms for safety and time-bounded reachability in COSYMA, a tool for automatic controller synthesis for incrementally stable switched systems based on multi-scale discrete abstractions. The tool accepts a description of a switched system represented by a set of differential equations and the sampling parameters used to define an approximation of the state-space on which discrete abstractions are computed. The tool generates a controller — if it exists — for the system that enforces a given safety or time-bounded reachability specification.

We are currently exploring, in the SYMBAD project, controller synthesis for switched systems based on a different approach for the construction of multi-scale abstractions. The goal is to further improve the trade-off between cost and precision.

6.2.2. Modular discrete controller synthesis

Discrete controller synthesis (DCS) [71] 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* satisfies 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 [50]. This work is done in collaboration with Hervé Marchand (VERTECS team from Rennes) and Eric 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.

A formal semantics of BZR has been defined, taking into account its underlying nondeterminism related to the presence of controllable variables. A new implementation has been achieved, including an abstraction method based on [47]. We have used BZR for demonstrating the use of Control Theory and Techniques to the administration of computing systems in a closed-loop management [19].

6.3. Automatic Distribution of Synchronous Programs

Participants: Gwenaël Delaval [contact person], Alain Girault, Gregor Goessler, Xavier Nicollin, Gideon Smeding.

6.3.1. 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 Rocquencourt) and Xavier Nicollin (Grenoble INP, VERIMAG lab).

We are working on type systems to formalize, in a 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 be applied to variables with identical 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 be applied to variables located at the same location. Compared to the PhD of Gwenaël Delaval [48], [49], 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.

We currently work on an example of software-defined radio. We have shown on this example how to use a modified clock calculus to describe the localisation of values as clocks, and the architecture as clocks (for the computing resources) and their relations (for communication links).

6.3.2. Distribution of synchronous programs under real-time constraints

With the objective to distribute synchronous data-flow programs (*e.g.*, LUSTRE) over GALS architectures, such that the difference between the original and synchronous systems satisfy given bounds, we have developed a quantitative clock calculus to (1) describe timing properties of the architecture's clock domain, and (2) describe the acceptable difference between the original and distributed programs. The clock calculus is inspired by the network calculus [67], with the difference that clocks are described only with respect to one-another, not with respect to real-time.

As a first result, we have applied our clock calculus to analyze the properties of periodic synchronous data-flow programs executed on a network of processors. Because our clock calculus is relational, it can model and preserve correlated variations of streams. In particular, the common case of a data-flow system that splits a stream for separate treatment, and joins them afterwards, this analysis yields more precise result than comparable methods [24].

We have been able to use the clock calculus as an abstract domain to perform abstract interpretation of synchronous boolean data-flow programs and their distribution on synchronous nodes that communicate asynchronously by sampling shared memory. The analysis discovers the relative clock drift of all clocks of the distributed system as well as bounds on the distance from the original program.

In case the guaranteed maximal distance is too large, we provide methods to synthesize bounds on the relative drift of the architecture's clocks that ensure an acceptable distance. Given the synthesized bounds, we use the known clock drifts and program behavior to synthesize light weight protocols.

6.4. New Programming Languages for Embedded Systems

Participants: Alain Girault [contact person], Pascal Fradet, Vagelis Bebelis, Bertrand Jeannet, Peter Schrammel.

6.4.1. Analysis and scheduling of parametric dataflow models

Recent data-flow programming environments support applications whose behavior is characterized by dynamic variations in resource requirements. The high expressive power of the underlying models (*e.g.*, Kahn Process Networks, the CAL actor language) makes it challenging to ensure predictable behavior. In particular, checking *liveness* (*i.e.*, no part of the system will deadlock) and *boundedness* (*i.e.*, the system can be executed in finite memory) is known to be hard or even undecidable for such models. This situation is troublesome for the design of high-quality embedded systems.

We have introduced the *schedulable parametric data-flow (SPDF)* MoC for dynamic streaming applications [20]. SPDF extends the standard dataflow model by allowing rates to be parametric (*e.g.*, of the form $2xy$, where x and y are parameters, the value of which can change at run-time). SPDF was designed to be statically analyzable while retaining sufficient expressive power. We formulated sufficient and general static criteria for boundedness and liveness. In SPDF, parameters can be changed dynamically even within iterations. The safety of dynamic parameter changes can be checked and their implementation made explicit in the graph. These different analyses are made possible using well-defined static operations on symbolic expressions. The same holds for quasi-static scheduling which is the first step towards code generation for multi-core systems.

We are now focusing on the parallel scheduling of parametric dataflow models. We have proposed a generic and flexible framework to generate parallel ASAP schedules targeted to the new STHORM many-core platform designed by STMicroelectronics. The parametric dataflow graph is associated with generic or user-defined specific constraints aimed at minimizing, timing, buffer sizes, power consumption, or other criteria. The scheduling algorithm executes with minimal overhead and can be adapted to different scheduling policies just by changing some constraints. The safety of both the dataflow graph and constraints can be checked statically and all schedules are guaranteed to be bounded and deadlock free. Our case studies are video decoders for high definition video streaming such as VC-1 or HEVC.

This research is the central topic of Vagelis Bebelis' PhD thesis. It is conducted in collaboration with STMicroelectronics.

6.5. Static Analysis and Abstract Interpretation

Participants: Alain Girault, Bertrand Jeannet [contact person], Peter Schrammel.

6.5.1. Translating data-flow languages for hybrid systems simulation to hybrid automata for hybrid systems verification

Hybrid systems are used to model embedded computing systems interacting with their physical environment. There is a conceptual mismatch between high-level hybrid system languages like SIMULINK [34], which are used for simulation, and hybrid automata, the most suitable representation for safety verification. Indeed, in simulation languages the interaction between discrete and continuous execution steps is specified using the concept of zero-crossings, whereas hybrid automata exploit the notion of staying conditions.

In the context of the INRIA large scale action SYNCHRONICS (see §8.1.1.1), we studied how to translate the ZELUS hydrid data-flow language [34] developped in this project into logico-numerical hybrid automata by carefully pointing out this issue. We investigated various zero-crossing semantics, proposed a sound translation, and discussed to which extent the original semantics is preserved.

This work is part of the PhD thesis of Peter Schrammel and was presented at the conference HSCC'2012 (Hybrid Systems: Computation and Control) [22], [27].

6.5.2. Abstract Acceleration of general linear loops

We investigated abstract acceleration techniques for computing loop invariants for linear loops with linear assignments in their body and guards defined by the conjunction of linear inequalities.

While standard abstract interpretation considers over approximations over the set of reachable states at any loop iteration, and relies on extrapolation (*a.k.a.* widening) for making the analysis converge, abstract acceleration captures the effect of the loop with a single, non-iterative transfer function applied to the reachable states at the loop head. The concept of abstract acceleration has already been applied to restricted form of linear loops, by us [16] and others [58], and extended to logico-numerical loops [16]; the novelty here is to investigate general linear loops.

The main idea we developped is to over-approximate the set of transformation matrices associated to any number of iterations of the loop body and to apply this "accelerated" transformation to the initial states. This over-approximation is based on the Jordan normal form decomposition that allows deriving closed form symbolic expressions for the entries of the matrix modeling the effect of exactly n iterations of the loop. We then discover linear relationships between the symbolic expressions that hold for any number of iterations, and we obtain a set of matrices described by a polyhedra on its coefficients, which can be applied to a set of vectors also described by a convex polyhedra.

We also developed a technique to take into account the guard of the loop by bounding the number of loop iterations, which relies again on the Jordan normal form decomposition.

These ideas were implemented and evaluated on a series of simple loops, alone or inside outer loops, exhibiting classical behaviors: polynomial, stable and unstable exponential, inward spirals (damped oscillators), ... Our approach enables proofs that are out of the reach of most other techniques, that are either too unprecise (classical abstract interpretation) or limited to a restricted class of loops, *e.g.*, translation with resets in the case of abstract acceleration, or stable loops (in the sense of control theory) for ellipsoid methods.

This work was initiated during a visit to the University of Colorado-Boulder and is under review.

[34] http://www.mathworks.com

6.5.3. Logico-Numerical Max-Strategy Iteration

Strategy iteration methods aim at solving fixed point equations and are an alternative to abstract acceleration for fighting against the loss of precision incurred by extrapolation in classical interpretation. It has been shown that they improve precision in static analysis based on abstract interpretation and template abstract domains, *e.g.*, intervals, octagons or template polyhedra. However, they are limited to numerical programs.

We investigated a method for applying max-strategy iteration to logico-numerical programs, that is, programs with numerical and Boolean variables, without explicitly enumerating the Boolean state space. The method is optimal in the sense that it computes the *least fixed point w.r.t.* the abstract domain.

Our experiments showed that the resulting logico-numerical max-strategy iteration gains one order of magnitude in terms of efficiency in comparison to the purely numerical approach while being almost as precise. Moreover, they are the first experimental results of applying max-strategy iteration to larger programs. This work has been accepted at VMCAI'2013 [23].

6.6. Component-Based Construction

Participants: Alain Girault, Gregor Goessler [contact person], Roopak Sinha, Gideon Smeding.

6.6.1. Incremental converter synthesis

We have proposed and implemented a formal incremental converter-generation algorithm for system-on-chip (SoC) designs. The approach generates a converter, if one exists, to control the interaction between multiple intellectual property (IP) protocols with possible control and data mismatches, and allows pre-converted systems to be re-converted with additional IPs in the future. IP protocols are represented using labeled transition systems (LTS), a simple but elegant abstraction framework which can be extracted from and converted to standard IP description languages such as VHDL. The user can provide control properties, each stated as an LTS with accepting states, to describe desired aspects of the converted system, including fairness and liveness. Furthermore, data specifications can be provided to bound data channels between interacting IPs such that they do not over/under flow. The approach takes into account the uncontrollable environment of a system by allowing users to identify signals exchanged between the SoC and the environment, which the converter can neither suppress nor generate.

Given these inputs, the conversion algorithm first computes the reachable state-space of a maximal non-deterministic converter that ensures (i) the satisfaction of the given data specifications and (ii) the trace equivalence with the given control specifications, using a greatest fix-point computation. It then checks, using the standard algorithm for Büchi games, whether the converter can ensure the satisfaction of the given control specifications (reachability of accepting states) regardless of how the environment behaves. If this is found to be true, deterministic converters can be automatically generated from the maximal non-deterministic converter generated during the first step. The algorithm is proven to be sound and complete, with a polynomial complexity in the state-space sizes of given IP protocols and specifications. It is also shown that it can be used for incremental design of SoCs, where IPs and specifications are added to an SoC in steps. Incremental design allows to constrain the combinatorial explosion of the explored state-space in each step, and also reduces on-chip wire congestion by decentralizing the conversion process.

A Java implementation has been created, and experimental results show that the algorithm can handle complex IP mismatches and specifications in medium to large AMBA based SoC systems. Future work involves creating a library of commonly-encountered specifications in SoC design such as sharing of control signals between interacting IPs using buffers, signal lifespans, and the generation of optimal converters based on quantitative criteria such as minimal power usage.

This work has been done within the AFMES associated team with the Electric and Computer Engineering Department of the University of Auckland.

6.6.2. Analysis of logical causality

The failure of one component may entail a cascade of failures in other components; several components may also fail independently. In such cases, elucidating the exact scenario that led to the failure is a complex and tedious task that requires significant expertise.

The notion of causality *(did an event e cause an event e'?)* has been studied in many disciplines, including philosophy, logic, statistics, and law. The definitions of causality studied in these disciplines usually amount to variants of the counterfactual test "e is a cause of e' if both e and e' have occurred, and in a world that is as close as possible to the actual world but where e does not occur, e' does not occur either". Surprisingly, the study of logical causality has so far received little attention in computer science, with the notable exception of [62] and its instantiations. However, this approach relies on a causal model that may not be known, for instance in presence of black-box components.

In [6] we have proposed a formal framework for reasoning about causality, based on black-box components interacting according to well identified *interaction models* [5].

We are currently extending to framework to other models of computation and communication, in particular, to timed automata, and developing a refinement of our original approach that reduces the number of false positives.

6.6.3. A Theory of fault recovery for component-based models

In [35][18] we have introduced a theory of fault recovery for component-based models. A model is specified in terms of a set of atomic components that are incrementally composed and synchronized by a set of glue operators. We have defined what it means for such models to provide a recovery mechanism, so that the model converges to its normal behavior in the presence of faults (*e.g.*, in self-stabilizing systems). We have identified corrector components whose presence in a model is essential to guarantee recovery after the occurrence of faults. We have also formalized component based models that effectively separate recovery from functional concerns. We have shown that any model that provides fault recovery can be transformed into an equivalent model, where functional and recovery tasks are modularized in different components.

6.7. Aspect-Oriented Programming

Participants: Dmitry Burlyaev, Pascal Fradet [contact person], Alain Girault.

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, fault tolerance or refinement aspects) as formal properties.

6.7.1. 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 [13]. 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.

6.7.2. Fault tolerance aspects

In the recent years, 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 digital circuits. To this aim, we consider program transformations for hardware description languages (HDL). 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 HDL program remains synthesizable. The advantage would be to produce fault-tolerant circuits by specifying fault-tolerant strategies separately from the functional specifications.

We have reviewed the different fault tolerant techniques used in integrated circuits: concurrent error detection, error detecting and correcting codes (Hamming, Berger codes, ...), spatial and time redundancy. We have designed a simple hardware description language inspired from LUSTRE and Lucid Synchrone. It is a core functional language manipulating synchronous boolean streams. Faults are represented by bit flips and we take into account all fault models of the form "at most 1 faults within n clock signals". The language semantics as well as the fault model have been formalized in Coq. Many basic (library) properties have been shown on that language.

We are currently expressing different variants of triple modular redundancy (TMR) as program transformations. We are also studying optimizations to prevent the insertion of useless voters in TMR. The next step is to prove that these transformations make the programs fault tolerant *w.r.t.* specific fault models. Further work also includes the study of mixed techniques (*e.g.*, spatial and time redundancy), their high level specification using an AOP-like language and their implementation as transformations.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- With ST Microelectronics: CIFRE contract for the PhD of Vagelis Bebelis. This work is described in Section 6.4.1.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. Inria Large Scale Actions

8.1.1.1. Inria Large Scale Action Synchronics: Language Platform for Embedded System Design

Participants: Gwenaël Delaval, Alain Girault [contact person, co-coordinator], Bertrand Jeannet, Xavier Nicollin, Peter Schrammel.

The SYNCHRONICS (Language Platform for Embedded System Design) project [mid-2008 to mid-2012] 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 Paris), 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.).

8.1.2. ANR

8.1.2.1. ANR Asopt: Analyse Statique et OPTimisation

Participants: Bertrand Jeannet [contact person, coordinator], Peter Schrammel.

The ASOPT (Analyse Statique et OPTimisation) project [january 2009-july 2012] [35] 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.

The main results are 1. improved *numerical abstract domains* (in particular the MaxPLus polyhedra and zonotopes-based abstract domains), and their combination with finite-types domains (using BDDs); 2. new *symbolic domains*, in particular for the accurate analysis of aliased expressions in data-structures and for precise interprocedural analysis in the presence of pointers to the call-stack; 3. improved *equation solving techniques*, with the generalization of the *policy iteration* approach and the widening of its applicability; 4. precise abstractions of full blocks of code, based either on quantifier elimination or on abstract acceleration.

Most of these contributions have been integrated into either the FIXPOINT library or the APRON/BDDAPRON libraries and they can be experimented on-line or off-line with the INTERPROC analyzer (see Section 5.5.5), which was the common experimental platform of the project.

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

Participants: Gregor Goessler [contact person], Bertrand Jeannet, Sebti Mouelhi.

The VEDECY project brings together hybrid systems and formal methods experts. Three partners are involved: Laboratoire Jean Kuntzmann (LJK), Inria POP ART, and VERIMAG.

VEDECY 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. International Initiatives

8.2.1. Inria Associate Teams

8.2.1.1. AFMES

Title: Advanced Formal Methods for Embedded Systems

Inria principal investigator: Alain Girault

International Partner (Institution - Laboratory - Researcher):

[35] http://asopt.inrialpes.fr/index.php

University of Auckland (New Zealand) - Department of Electrical and Computer Engineering

Duration: 2010 - 2012

See also: http://pop-art.inrialpes.fr/~girault/Projets/Afmes/

Embedded systems are characterized by several constraints, such as determinism and bounded reaction time. Accordingly, design methods for embedded systems should, when possible, guarantee these properties by construction. This allows the shifting of the burden of checking these constraints from the programmer to the design method and the associated compilers and code generation tools. In order to achieve this, our goal is to improve the existing design methods in several key directions: (1) Incremental converter synthesis. (2) Programming language for adaptive computing (SystemJ and beyond) [15]. (3) Time predictable programming language and execution architectures [10], [12]. Together, these advanced methods will provide a higher level of safety in the design of embedded systems.

8.3. International Research Visitors

8.3.1. Visits of International Scientists

- Aditya Zutshi, PhD student at the University of Colorado Boulder (USA), visited POP ART from July to August 2012 and worked on the abstract acceleration of general linear loops with inputs.
- Partha Roop, Senior Lecturer at the University of Auckland (New Zealand) visited POP ART in March 2012 to work on the AFMES associated team.
- Eugene Yip, PhD student at the University of Auckland (New Zealand) visited POP ART from October to December 2012 to work on the AFMES associated team.

8.3.2. Visits to International Teams

- Bertrand Jeannet and Peter Schrammel visited the University of Colorado Boulder (USA) in February 2012 from the 3th to the 21th.
- Alain Girault visited the University of Auckland (New Zealand) to work on the AFMES Associated Team.
- Alain Girault visited the University of California Berkeley (USA) in August 2012 to work on time predictable programming languages and on parametric dataflow models of computation.

9. Dissemination

9.1. Scientific Animation

- Pascal Fradet served in the program committee of JFLA 2013 (*v*ingt quatrièmes Journées Francophones des Langages Applicatifs).
- Gregor Goessler served in the program committees of the international conference DATE 2013.
- Bertrand Jeannet was Program Chair of the TAPAS workshop (Tools for Automatic Program AnalysiS) affiliated to SAS (Static Analysis Symposium), that was held in Deauville in September 2012.
- Alain Girault served in the program committees of the international conferences DAC 2012, MEMOCODE 2012, LCTES 2012, and APSIPA ASC 2012,

9.2. Teaching - Supervision - Juries

9.2.1. Supervision

PhD in progress: Vagelis Bebelis, “Advanced dataflow programming for embedded systems”, Grenoble University, since 12/2011, co-advised by Pascal Fradet and Alain Girault.

PhD in progress: Dmitry Burlyaev, “Specification and synthesis of fault-tolerant circuits”, Grenoble University, since 12/2011, co-advised by Pascal Fradet and Alain Girault.

PhD in progress: Gideon Smeding, Grenoble University, “Distribution of synchronous programs withe respect to real-time constraints”, co-advised by Gregor Goessler and Joseph Sifakis since 12/2009.

PhD : Peter Schrammel, “Logico-numerical verification methods for discrete and hybrid systems”, Grenoble University, defended on 18/10/2012, co-advised by Alain Girault and Bertrand Jeannet

9.2.2. Juries

- Bertrand Jeannet was examiner for the PhD of Romain Testylier (University of Grenoble).
- Alain Girault was referee for the PhD thesis of Aurélien Monot (University of Lorraine), president of the PhD jury of Slim Bouguerra (University of Grenoble), and president of the PhD jury of Christophe Junke (University of Orsay).

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^{o} 4, p. 1–43.

[2] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects preserving properties*, in "Sci. Comput. Program.", 2012, vol. 77, n^{o} 3, p. 393-422.

[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^{o} 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^{o} 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", 3 2005, vol. 55, n^{o} 1–3, p. 161–183.

[6] G. GÖSSLER, D. LE MÉTAYER, J.-B. RACLET. *Causality Analysis in Contract Violation*, in "Runtime Verification", LNCS, Springer-Verlag, 2010, p. 270-284.

[7] B. JEANNET, A. LOGINOV, T. REPS, M. SAGIV. *A Relational Approach to Interprocedural Shape Analysis*, in "ACM Trans. on Programming Languages and Systems", 2010, vol. 32, n^{o} 2, http://doi.acm.org/10.1145/1667048.1667050.

[8] 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", Copenhagen (Denmark), LNCS, August 2007, vol. 4634, http://pop-art.inrialpes.fr/people/bjeannet/publications/sas07.ps.

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

Publications of the year

Articles in International Peer-Reviewed Journals

[10] S. ANDALAM, P. ROOP, A. GIRAULT, C. TRAULSEN. *A Predictable Framework for Safety-Critical Embedded Systems*, in "IEEE Trans. Computers", 2013, To appear.

[11] I. ASSAYAD, A. GIRAULT, H. KALLA. *Tradeoff Exploration between Reliability, Power Consumption, and Execution Time for Embedded Systems*, in "Int. J. Software Tools for Technology Transfer", 2013, To appear.

[12] P. AXER, R. ERNST, H. FALK, A. GIRAULT, D. GRUND, N. GUAN, B. JONSSON, P. MARWEDEL, J. REINEKE, C. ROCHANGE, M. SEBATIAN, R. VON HANXLEDEN, R. WILHELM, W. YI. *Building Timing Predictable Embedded Systems*, in "ACM Trans. Embedd. Comput. Syst.", 2013, To appear.

[13] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects preserving properties*, in "Sci. Comput. Program.", 2012, vol. 77, n^{o} 3, p. 393-422.

[14] G. GOESSLER, D. XU, A. GIRAULT. *Probabilistic contracts for component-based design*, in "Formal Methods in System Design", 2012, vol. 41, n^{o} 2, p. 211-231 [*DOI :* 10.1007/s10703-012-0162-4], http://hal.inria.fr/hal-00747620.

[15] A. MALIK, A. GIRAULT, Z. SALCIC. *Formal Semantics, Compilation and Execution of the GALS Programming Language DSystemJ*, in "IEEE Trans. Parallel and Distributed Systems", July 2012, vol. 23, n^{o} 7, p. 1240–1254.

[16] P. SCHRAMMEL, B. JEANNET. *Applying abstract acceleration to (co-)reachability analysis of reactive programs*, in "Journal of Symbolic Computation", 2012, vol. 47, n^{o} 12, p. 1512-1532 [*DOI :* 10.1016/J.JSC.2011.12.051], http://hal.inria.fr/hal-00753412.

International Conferences with Proceedings

[17] I. ASSAYAD, A. GIRAULT, H. KALLA. *Scheduling of Real-Time Embedded Systems under Reliability and Power Constraints*, in "International Conference on Complex Systems, ICCS'12", Agadir, Morocco, IEEE, November 2012.

[18] B. BONAKDARPOUR, M. BOZGA, G. GOESSLER. *A Theory of Fault Recovery for Component-Based Models*, in "Stabilization, Safety, and Security of Distributed Systems", Toronto, Canada, 2012 [*DOI :* 10.1109/SRDS.2011.39], http://hal.inria.fr/hal-00747622.

[19] F. BOYER, G. DELAVAL, N. DE PALMA, O. GRUBER, E. RUTTEN. *Discrete supervisory control application to computing systems administration*, in "Proc. of the 14th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'012", Bucarest, Romania, May 2012.

[20] P. FRADET, A. GIRAULT, P. POPLAVKO. *SPDF: A Schedulable Parametric Data-Flow MoC*, in "Design Automation and Test in Europe, DATE'12", Dresden, Germany, 2012, http://hal.inria.fr/hal-00744376.

[21] B. JEANNET, P. SOTIN. *Inferring Effective Types for Static Analysis of C Programs*, in "NSAD - Int. Workshop on Numerical and Symbolic Abstract Domains - 2011", Venise, Italy, D. MASSÉ, L. MAUBORGNE (editors), ENTCS, Elsevier, October 2012, vol. 288, p. 37-47 [*DOI :* 10.1016/J.ENTCS.2012.10.006], http://hal.inria.fr/hal-00763426.

[22] P. SCHRAMMEL, B. JEANNET. *From Hybrid Data-Flow Languages to Hybrid Automata: A Complete Translation*, in "Hybrid Systems: Computation and Control", Beijing, China, T. DANG, I. MITCHELL (editors), ACM, 2012, p. 167-176 [*DOI :* 10.1145/2185632.2185658], http://hal.inria.fr/hal-00749891.

[23] P. SCHRAMMEL, P. SUBOTIC. *Logico-Numerical Max-Strategy Iteration*, in "Verification, Model Checking, and Abstract Interpretation", Rome, Italy, R. GIACOBAZZI, J. BERDINE, I. MASTROENI (editors), LNCS, Springer, 2013, vol. 7737, p. 414-433, http://hal.inria.fr/hal-00756833.

[24] G. SMEDING, G. GOESSLER. *A Correlation Preserving Performance Analysis for Stream Processing Systems*, in "MEMOCODE", Washington DC, United States, July 2012 [*DOI :* 10.1109/MEMCOD.2012.6292295], http://hal.inria.fr/hal-00745819.

Research Reports

[25] G. GÖSSLER, D. XU, A. GIRAULT. *Probabilistic Contracts for Component-based Design*, Inria, July 2012, n^{o} RR-7328, http://hal.inria.fr/hal-00715750.

[26] S. MOUELHI, A. GIRARD, G. GÖSSLER. *CoSyMA: A Tool for Controller Synthesis Using Multi-scale Abstractions*, Inria, October 2012, n^{o} RR-8108, http://hal.inria.fr/hal-00743982.

[27] P. SCHRAMMEL, B. JEANNET. *From Hybrid Data-Flow Languages to Hybrid Automata: A Complete Translation*, Inria, January 2012, n^{o} RR-7859, 37, http://hal.inria.fr/hal-00659698.

[28] P. SCHRAMMEL, B. JEANNET. *Logico-Numerical Abstract Acceleration and Application to the Verification of Data-Flow Programs*, Inria, November 2012, n^{o} RR-7630, 22, http://hal.inria.fr/inria-00596241.

References in notes

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

[30] X. ALLAMIGEON, S. GAUBERT, É. GOUBAULT. *Inferring Min and Max Invariants Using Max-Plus Polyhedra*, in "Static Analysis Symposium, SAS'08", LNCS, 2008, vol. 5079, p. 189–204.

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

[32] 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.

[33] I. ASSAYAD, A. GIRAULT, H. KALLA. *Tradeoff Exploration between Reliability, Power Consumption, and Execution Time*, in "International Conference on Computer Safety, Reliability and Security, SAFECOMP'11", Napoli, Italy, LNCS, Springer-Verlag, September 2011, vol. 6894, p. 437–451.

[34] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *Divide and recycle: types and compilation for a hybrid synchronous language*, in "LCTES", ACM, 2011, p. 61-70.

[35] B. BONAKDARPOUR, M. BOZGA, G. GÖSSLER. *A Theory of Fault Recovery for Component-Based Models*, in "SRDS", 2011, p. 265-270.

[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 Trans. on Computers", 1986, vol. C-35, n° 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] 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.

[41] E. CLARKE, E. EMERSON, A. SISTLA. *Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications*, in "ACM Trans. Programming Languages and Systems", 4 1986, vol. 8, n° 2, p. 244-263, Introduction of Model-checking; impartiality, justice, fairness.

[42] 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.

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

[44] 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.

[45] J. CÁMARA, A. GIRARD, G. GÖSSLER. *Safety Controller Synthesis for Switched Systems Using Multi-Scale Symbolic Models*, in "CDC-ECC", IEEE, 2011, p. 520-525.

[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. *Modular Distribution and Application to Discrete Controller Synthesis*, in "International Workshop on Model-driven High-level Programming of Embedded Systems (SLA++P'08)", Budapest, Hungary, April 2008, http://pop-art.inrialpes.fr/people/delaval/pub/slap08.pdf.

[48] G. DELAVAL. *Répartition modulaire de programmes synchrones*, INPG, Inria Grenoble Rhône-Alpes, July 2008, PhD thesis.

[49] 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.

[50] 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.

[51] D. DELMAS, É. 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.

[52] 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.

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

[54] 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.

[55] A. GIRAULT. *System-Level Design of Fault-Tolerant Embedded Systems*, October 2006, vol. 67, p. 25–26.

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

[59] D. GOPAN, T. REPS. *Guided Static Analysis*, in "Static Analysis Symposium, SAS'07", LNCS, August 2007, vol. 4634, p. 349–365, http://dx.doi.org/10.1007/978-3-540-74061-2_22.

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

[61] 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.

[62] J. HALPERN, J. PEARL. *Causes and Explanations: A Structural-Model Approach. Part I: Causes*, in "British Journal for the Philosophy of Science", 2005, vol. 56, n^{o} 4, p. 843-887.

[63] T. HENZINGER, P. HO, H. WONG-TOÏ. *HyTech: The Next Generation*, in "RTSS'95", 1995.

[64] 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.

[65] 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° 1, p. 5–37.

[66] 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.

[67] J.-Y. LE BOUDEC, P. THIRAN. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, Lecture Notes in Computer Science, Springer, 2001, vol. 2050.

[68] Y. LEE, A. ZOMAYA. *Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling*, in "IEEE/ACM International Symposium on Cluster Computing and the Grid, SCCG'09", 2009.

[69] 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.

[70] 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.

[71] 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° 1, p. 206–230.

[72] P. RAMADGE, W. WONHAM. *The Control of Discrete Event Systems*, in "Proceedings of the IEEE", 1989, vol. 77, n° 1.

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