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 Scientist

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

External Collaborator

Emil Dumitrescu [INSA Lyon]

PhD Student

Mouaiad Alras [Region Rhône-Alpes grant, ISLE cluster]
Gérald Vaisman [INRIA grant, DCN, until 09/2009]
Marnes Hoff [INRIA grant]
Lies Lakhdar-Chaouch [INRIA grant, OPENTLM]
Bhasker Vanamali [INRIA grant, OPENTLM]
Peter Schrammel [INRIA grant, since 07/2009]
Gideon Smeding [INRIA grant, since 12/2009]

Post-Doctoral Fellow

Xavier Briand [INRIA, until 03/2009]
Jean-Baptiste Raclet [COMBEST]
Na Xu [COMBEST, until 11/2009]
Pascal Sotin [INRIA, since 10/2009]

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 distribution (in relation with the LUSTRE [1] and ESTEREL [2] languages) and fault tolerance (in relation with the SYNDEX [3] environment);
- 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 Mode Automata and SIGNAL [4] languages, and with the SIGALI synthesis tool); compositionality for the verification, and construction of correct systems; reactive programming, aspect-oriented programming.

[1] http://www-verimag.imag.fr/SYNCHRONE
[2] http://www.inria.fr/recherche/equipes/aoste.en.html
[3] http://www-rocq.inria.fr/syndex
[4] http://www.irisa.fr/espresso

- 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 [5], about embedded real-time systems;
- an FP7 European STREP project: COMBEST [6] on component-based design;
- an ARTEMISIA European project: CESAR [7] 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 [8] on a language platform for embedded system design.

3. Scientific Foundations

3.1. Embedded systems and their safe design

3.1.1. The 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.

[5] http://www.artist-embedded.org
[6] http://www.combest.eu/home
[7] http://www.cesarproject.eu
[8] http://www.inria.org/recherche/equipes/synchronics.en.html

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 [29], [34]. During the past years, methodologies for the formal verification [64], [36], control synthesis [66] and compilation, as well as extensions to timed and hybrid systems [62], [30] 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 [55], which can be encoded in a synchronous model [31].

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 [32] for manipulating Boolean functions and sets, and their MTBDD extension for more general functions. Abstract interpretation [38] 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 others 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 [39] and shape analysis [35].

The synchronous approach [9] [53], [54] to reactive systems design gave birth to complete programming environments, with languages like ARGOS, LUSTRE [10], ESTEREL [11], SIGNAL/ POLYCHRONY [12], LUCID SYNCHRONE [13], SYNDEX [14], 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 in 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 specialists in the application domain, not in formal techniques. This is why encapsulating formal techniques in 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.

[9] http://www.synalp.org
[10] http://www-verimag.imag.fr/SYNCHRONE
[11] http://www.inria.fr/recherche/equipes/aoste.en.html
[12] http://www.irisa.fr/espresso/Polychrony
[13] http://www.lri.fr/~pouzet/lucid-synchrone/
[14] 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 (telecommunication), 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 (University of Orsay – Paris Sud, LRI lab), we have designed the programming language HEPTAGON, the key features of which are: data-flow formal synchronous semantics, strong typing with type inference, and polymorphism. 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 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 [15]) 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.

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

Regarding the synchronous approach, commercial tools are available: SCADE (based on LUSTRE), ESTEREL STUDIO [16], SILDEX (based on SIGNAL), specialized environments like CELLCONTROL for industrial automatism (by the INRIA spin-off ATHYS). 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 compositional analysis and abstract interpretation for the TLM-based System-on-Chip design flow.

5. Software

5.1. NBac

Participant: Bertrand Jeannet.

NBAC (Numerical and Boolean Automaton Checker)[17] 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 [57] [44]. It is connected to the LUSTRE toolset[18]. 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 [19] [37] [58] for selecting test cases.

5.2. Prometheus

Participant: Gregor Gössler.

The BIP component model (Behavior, Interaction model, Priority) [51][5] has been designed to support the construction of heterogeneous reactive 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 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.

[16] http://www.esterel-technologies.com
[17] http://pop-art.inrialpes.fr/people/bjeannet/nbac/
[18] http://www-verimag.imag.fr/SYNCHRONE/index.php?page=tools
[19] http://www.irisa.fr/prive/ployette/stg-doc/stg-web.html

5.3. Implementations of synchronous programs

Participant: Alain Girault.

5.3.1. Code distribution

OCREP distributes automatically synchronous programs according to specifications given by the user. Concretely, starting from a centralized source synchronous program obtained either with the LUSTRE or the ESTEREL compiler, from a number of desired computing locations, and an indication of where each input and output of the source program must be computed, OCREP produces several programs, one for each location, each one computing only its assigned variables and outputs, and communicating harmoniously. Their combined behavior is equivalent to the behavior of the centralized source program.

Currently, OCREP is distributed in the form of executable on the web[20]. It consists in 15000 lines of C++ code. In 2002, a contract for industrial transfer was drawn up with France Télécom R&D in order to integrate OCREP into their compiler SAXO-RT for ESTEREL programs.

5.3.2. Fault tolerance

We have been cooperating for several years with the INRIA team AOSTE (INRIA Sophia-Antipolis and Rocquencourt) on the subject of fault tolerance. In particular, we have implemented several new heuristics for fault tolerance and reliability within their software SYNDEX [21]. This has taken place within the framework of the European project EAST-EEA in which we participated together with AOSTE. In this context, we have developed several new scheduling heuristics that produce static multiprocessor schedules tolerant to a specified number of processor and communication link failures [45]. The basic principles upon which we rely to make the schedules fault tolerant is, on the one hand, the active replication of the operations [46], 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) [47]. Our results on fault-tolerance are summarized in a web page[22].

5.4. Apron and BDDApron libraries

Participant: Bertrand Jeannet.

5.4.1. Principles

The APRON library[23] is dedicated to the static analysis of the numerical variables of a program by abstract interpretation [38]. 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.

[20] http://pop-art.inrialpes.fr/people/girault/Ocrep/
[21] http://www-rocq.inria.fr/syndex
[22] http://pop-art.inrialpes.fr/~girault/Projets/FT
[23] http://apron.cri.ensmp.fr/library/

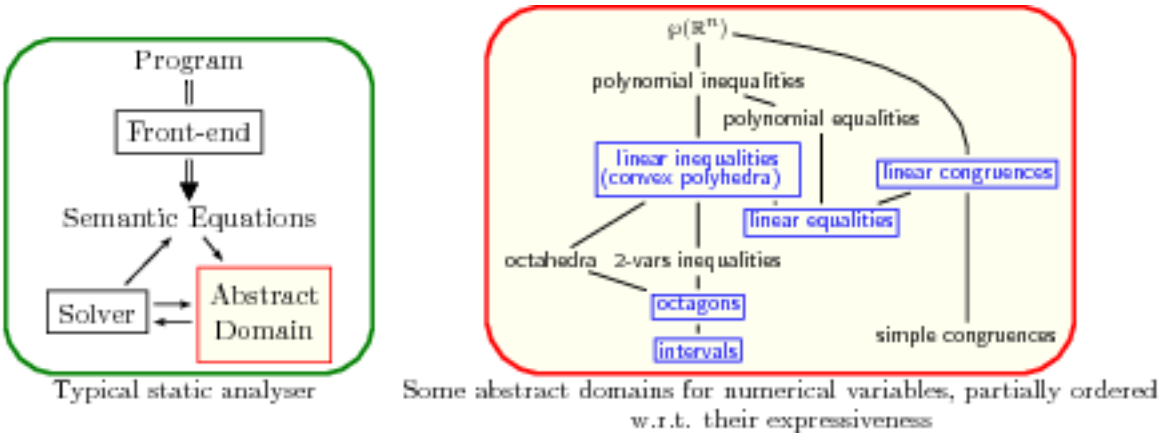

Figure 1. Typical static analyser and examples of abstract domains

The BDDAPRON library[24] 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-points). 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.

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, Saclauy, 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 [25] and Mode Automata [26].

[24] http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/bddapron/index.html
[25] http://www.irisa.fr/vertecs/Logiciels/sigali.html
[26] http://www-verimag.imag.fr

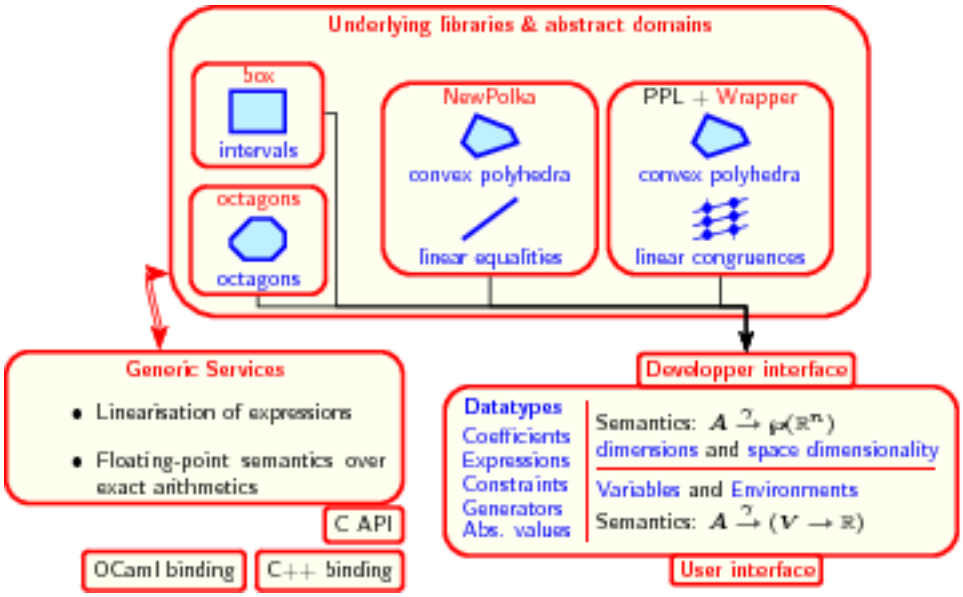

Figure 2. Organisation of the APRON *library*

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.

RAPTURE [56] [40] 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 [27]: 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 very recent techniques [48].

[27] http://bjeannet.gforge.inria.fr/fixpoint

INTERPROC [28]: 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[29]. It has been cited in at least 3 published research papers in 2008.

CONCURINTERPROC extends Interproc with concurrency, for the analysis of multithreaded programs interacting via shared global variables. It is also deployed through a web-interface[30].

6. New Results

6.1. Dependable distributed real-time embedded systems

Participants: Pascal Fradet, Alain Girault [contact person], 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 [9], [11] 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. This work is conducted in collaboration with Hamoudi Kalla (University of Batna, Algeria).

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*. 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 [65]. 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

[28] http://bjeannet.gforge.inria.fr/interproc
[29] http://pop-art.inrialpes.fr/interproc/interprocweb.cgi
[30] http://pop-art.inrialpes.fr/interproc/concurinterprocweb.cgi

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 [10]:

- The **automatization**, 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 are extending 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 are adapting 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.

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 6.5.3.

6.2. Automatic distribution of synchronous programs

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

6.2.1. Modular distribution

Synchronous programming languages describe functionally centralized systems, where every value, input, output, or function are 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 Gwenaël Delaval (SARDES team from Grenoble) and Marc Pouzet (Orsay University and PROVAL 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 [33]). 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 [41], [42], 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.

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 [52]. Concretely, starting from a high-level model specified in the de-facto standard Simulink, we first generate Lustre code along with some necessary structured "glue code", and then we generate embedded real-time code for the Xenomai RTOS[31]. Thanks to Lustre's clean mathematical semantics, we are able to guarantee the faithfulness of the generated multi-tasked real-time code [14]. This is the topic of the PhD of Mouaiad Alras, co-advised by Alain Girault and Pascal Raymond (CNRS, Verimag).

6.3. Static analysis and abstract interpretation

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

6.3.1. 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.

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 [17]. A journal version is in preparation.

We are also working on the definition of a synchronous hybrid language for the design, simulation, and verification of discrete-continuous hybrid systems. This is the topic of the PhD of Peter Schrammel, co-advised by A. Girault and B. Jeannet, and funded by SYNCHRONICS.

6.3.2. 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, accepted in october 2008 has been revised this year and should be published in 2010 [13]. This work has been done in collaboration with T. Reps (Univ. of Madison-Wisconsin), M. Sagiv (Univ. of Tel-Aviv) and A. Loginov (GrammaTech).

6.3.3. 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 [20].

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

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.

We are also working on modular analyzes of concurrent programs with abstract interpretation techniques. This is the topic of the PhD of Lies Lakhdar-Chaouch, co-advised by A. Girault and B. Jeannet, and funded by OPENTLM.

6.3.4. Distributed controller synthesis using static analysis of FIFO channels

As explained in previous section, controller synthesis aims at modifying an existing specification/system in order to make it satisfy a property. We study this problem in the particular case of distributed systems modeled as a set of sequential machines communicating via unbounded FIFO channels, for which we want to ensure safety properties.

The static analysis of stacks and FIFO queues was the topic of the PhD of Tristan Le Gall, defended in June 2008. We proposed in [6] a new abstract domain for languages on infinite alphabets, which acts as a functor taking an abstract domain for a concrete alphabet, and lifts it to an abstract domain for words on this alphabet.

We studied this year the application of this technique to the controller synthesis of a set of sequential machines communicating via unbounded FIFO channels, for which we consider simple state-avoidance properties. It is well-known that there exists no optimal (most permissive) controller in such a context, so our ambition is to propose a technique for computing "permissive-enough" controller. Our approach is based on the computation of global controller, which is then projected on local sites so as to obtain a controller per site in the controlled distributed system. We exploit the abstract domain mentioned above in the fixpoint computations involved in the computation of a correct controller, and we take into account the following partial observation constraints: the global controller cannot observe the contents of the FIFO channels to take its decisions (they model communication links), and the projected local controller has a knowledge only on their local state.

This work is conducted in collaboration with H. Marchand and T. Le Gall (VERTECS team from INRIA Rennes).

6.3.5. Tools developpement

Several man/month efforts have been devoted to the developpement of libraries and tools (see 5.5.3). This year has been more particularly devoted to the BDDAPRON library, which has been publicy released as a deliverable of the ASOPT project (§8.2.2), and the FIXPOINT library.[32] We have published a tool paper (6 pages) on the now mature APRON library [21]. We also plan to submit in 2010 research and tool paper(s) on these libraries.

6.4. Component-Based Construction

Participants: Alain Girault, Gregor Gössler [contact person], Jean-Baptiste Raclet, Gideon Smeding, Na Xu.

6.4.1. 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

[32] http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/bddapron/index.html, http://pop-art.inrialpes.fr/~bjeannet/bjeannet-forge/fixpoint/index.html

also satisfies some desired specification. A converter is responsible for bridging different kinds of mismatches such as *control*, *data*, 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.

We have formulated a generalization of this convertibility verification problem, by using a new refinement relation called *Specification Enforcing Refinement* (SER) between a protocol composition and a desired specification. The existence of such a refinement is shown to be a necessary and sufficient condition for the existence of suitable a converter. We have also proposed an approach to automatically synthesize a converter if a SER refinement relation exists. These results have been published in [24].

We are currently working on an improvement of this framework that supports incremental converter synthesis.

6.4.2. Compositional strategy mapping

In the context of our work on compositionality and reconfigurability, we are studying the issue of implementing a component system on a lower-level platform. With the increasing complexity of embedded systems, coupled with the need for faster time-to-market and high confidence in the reliability of the product, design methods that ensure correctness by construction are, when available, the solution of choice. When dealing with reactive systems, which interact with their environment, the behavior of the system to be designed has to be considered in terms of *strategies*: can some desired behavior be enforced in spite of the — potentially non cooperative — environment?

Computing a strategy satisfying some property is expensive, and although modular and compositional controller synthesis have been studied for some decades, this remains a hard problem. In particular, progress properties are notoriously more difficult to tackle compositionally than safety properties.

We are interested in a design flow supporting the refinement of strategies, rather than in controller synthesis performed on some given level of abstraction. We consider a platform-based design process consisting of successive mapping steps [59]. The goal of each step is to constructively map a strategy constructed so far onto a lower-level platform. The mapping is performed component-wise, using an abstraction of the environment of each component. We have developed compositionality results ensuring that the refinement carries over to the global strategy [18].

The result of strategy mapping is a strategy for each target component, such that the composition of the strategies over-approximates the source strategy. We are currently investigating an improvement of this technique in order to effectively derive an implementation on the target platform, rather than an over-approximation.

6.4.3. Contract-based design

Contracts have first been introduced as a type system for classes [63]: 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* [67], allowing to reason about the temporal behavior of environments with which a component can be composed.

6.4.3.1. Modal Assume/Guarantee Contracts

We define contracts as pairs (A, G) of *modal automata* [60]: one describes an assumption on the usage of the component made by its environment; the other one corresponds to a guarantee offered by the component as long as the assumption is satisfied [19]. Modal automata extend automata with a modality that indicates for each transition whether it *may* or *must* be implemented. Modal contracts leverage the well-established theory of modal automata. They finitely represent an infinite number of implementations. In contrast to a premature choice of implementing or not a given transition, which would prematurely narrow the design space and

rule out possible implementations, modal contracts allow to preserve a larger solution space along the design process.

In contrast to a *specification* defining how a component *must* behave, contracts can be seen as implications, providing a guarantee depending on an assumption on the context. Accordingly, different semantics of contract composition are conceivable, with the two special cases of *conjunction of implications* yielding a *lazy* composition, and *implication of a conjunction* for an *eager* composition. The latter approach is adopted by [61], where the assumption of the composed contract is defined as the weakest assumption ensuring the conjunction of both guarantees. In the present work we choose the former approach: a component satisfying the composition of two contracts must satisfy each guarantee if and only if the corresponding assumption holds. This notion of composition is consistent with the component paradigm mentioned above, enabling the component to offer different guarantees depending on the context.

Our theory of modal assume/guarantee contracts relies on *weak implication*. This operation computes a modal specification whose implementations are also implementations of a given assume/guarantee contract $C = (A, G)$. This modal specification is thus called the *implicit form* of C. Based on weak implication, we have introduced three composition operations between modal contracts, responding to different requirements in the design flow and satisfying different properties. This is the first work formalizing and allowing to effectively combine these contract composition operations.

Component composition. A contract may be used to describe the guarantees a component is able to give, depending on its environment. We call this a *component contract*. For component contracts over disjoint components we define a "best effort" composition operation that is parametrized by an *interaction model* inspired by the BIP framework [5]. The composition ensures each guarantee depending on the satisfaction of its assumption, provided that the guarantee is feasible under the specified interaction model. We show that this composition is the *strongest* contract satisfying the property of independent implementability: the component composition of two contracts is satisfied by the composition of any pair of implementations of both contracts, which allows to reason about contracts in a bottom-up manner.

Aspect composition. On the other hand, a contract may specify a *requirement* as a guarantee that must be ensured under some hypothesis. We call this a requirement contract, or *aspect*. Aspects are usually implemented top-down. For a set of aspects on the same component or sub-system, a crucial question is whether they are consistent, and how to compute a common implementation, also called their *shared refinement* [43]. We define a composition operation based on modal conjunction, as the *weakest* contract refining both arguments. This is motivated by the fact that different aspects express different requirements whose conjunction is to be satisfied. The conjunction of contracts is shown to be sound and, under some conditions, complete.

Therefore, contracts are an elegant way to combine bottom-up construction of a system from simpler components, and top-down design by successive refinement of abstract components.

Priority composition. In practice, aspects are not equally important. For instance, an aspect "safety" may be chosen to override an aspect "quality of service". Therefore we define the operation of *priority* which composes aspects in a hierarchical order, such that in case of inconsistency, an aspects of higher priority overrides a lower-priority contract.

6.4.3.2. Probabilistic contracts for reliability of components

We are studying probabilistic contracts as a means to reason about reliability in a component-based framework. A probabilistic contract is an interactive Markov chain specification (IMCS) where each state is either an interaction state or a probabilistic state. From interaction states, only interactions (in the sense of the BIP component framework) are enabled. From probabilistic states, only probabilistic transitions are enabled. Each of the latter is labeled with an interval of allowed probabilities. We have defined operations such as refinement and composition allowing to reason about probabilistic contracts. This work is still in progress.

6.4.4. A modal interface theory

Nowadays, OEM perform system design and integration by importing/reusing entire subsystems provided by equipment suppliers. It is crucial that the subsystems are designed according to some rules; which highlights the importance of providing good notions of component interfaces. According to our understanding of industrial needs, the following list of requirements applies to the notion of interface in the context of embedded systems: (i) Interfaces act as legal bindings and therefore must clearly identify roles and responsibilities. (ii) Interfaces must support the operation of conjunction; the two reasons for this are, one the one hand, that current practice of requirements capture leads to large data bases of requirements to be seen in a conjunctive way, and, on the other hand, that large systems possess multiple viewpoints (functional, timing, safety, etc) that concern the same sub-systems or component and apply conjunctively. (iii) Interface-based design must allow for flexible architecture choices. The mathematical requirements for interfaces are then:

1. Interfaces must explicit roles (component versus environment, at least)
2. They must be equipped with a parallel composition and a conjunction; in addition, it is useful to have the adjoint of parallel composition to address incomplete designs, the latter is called quotient or residuation.
3. They must support the following fundamental relations, involving interfaces and their associated implementations (or models): satisfaction, consistency, compatibility, and refinement.
4. Last but not least, interfaces must support local alphabets of actions.

We have reviewed candidate classes for interface theories failing to address all these requirements. Then we have advocated using modal interfaces, a unification of interface automata by de Alfaro and Henzinger and modal specifications by Larsen. (see [23] and [22]). This work is conducted in collaboration with E. Badouel, A. Benveniste, B. Caillaud and A. Legay (S4 team from IRISA/INRIA-Rennes) and R. Passerone (University of Trento).

6.4.5. Timed aspects of interface theories

As we have seen, modal interfaces are classic, convenient, and expressive mathematical objects to represent interfaces of component-based systems. On the other hand, time is a crucial aspect of systems for practical applications, e.g. in the area of embedded systems. And yet, only few results exist on the design of timed component-based systems. In [16], we have proposed a timed extension of modal specifications, defined their notions of refinement and consistency, and established their decidability. In [15], we have considered the subclass of modal event-clock automata, where clock resets are easy to handle. We then have developed an entire theory with conjunction, product, and quotient, that promotes efficient incremental design techniques and that enables to reason in a compositional way about timed system. This work is conducted in collaboration with N. Bertrand (VERTECS team from IRISA/INRIA-Rennes) and S. Pinchinat (S4 team from IRISA/INRIA-Rennes).

6.5. Aspect-oriented programming

Participants: 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 or fault tolerance aspects) as formal properties.

6.5.1. 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 [7], [27]. Our categories of aspects comprise, among others, observers, aborters, and confiners. Observers do not modify the base program's state and control-flow (*e.g.*, persistence, profiling, and debugging aspects). Aborters are observers which may also abort executions (*e.g.*, security aspects). Confiners only ensure that the executions remain in the set of reachable states of the base program (*e.g.*, optimization or fault-tolerance 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 [7], [27]. These languages preserve the corresponding classes of properties by construction. The aspect languages share the same expressive pointcut language and are designed *w.r.t.* a common imperative base language.

This work was the central topic of Simplice Djoko Djoko's PhD thesis, which has been defended in June 2009. It is conducted in collaboration with Rémi Douence from the ASCOLA INRIA team at École des Mines de Nantes.

6.5.2. Resource management and aspects of availability

We have studied the use of aspect-oriented programming for resource management with the aim of enforcing availability properties [8]. Our technique allows us to keep resource management and availability issues separate from the rest of the system.

We have proposed a domain-specific aspect language aimed at preventing denial of service caused by resource management (*e.g.*, starvation, deadlocks, etc.). 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 automaton representing a program specifies a superset of all possible (timed) execution traces whereas the automaton representing an aspect specifies a set of desired/allowed (timed) execution traces. Weaving can be seen as a *product* of two timed automata (*i.e.*, the intersection of execution traces) which restricts the execution of the base program to the behaviors allowed by the aspect. 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.

6.5.3. Fault tolerance aspects for real-time software

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 [2]. Failure detection is implemented using heartbeating, and failure masking using checkpointing and roll-back. These techniques are described and implemented by automatic program transformations of the tasks' source programs.

We are currently working on an aspect language allowing users to specify and tune a wider range of fault tolerance techniques. For example, the user may want to use checkpointing, code or data replication at different places of the same program. For checkpointing, the user may also want to specify the subset of variables which must be saved. We are currently working on the definition of an aspect language allowing the designer to

specify such choices. This work is conducted in collaboration with T. Ayav from Izmir University who spent a one-month visit in POP ART last June.

6.6. Other results

6.6.1. Chemical programming

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

The chemical reaction metaphor describes computation in terms of a chemical solution in which molecules (representing data) interact freely according to reaction rules (representing the program). Chemical programs can be formalized as associative-commutative rewritings (reactions) of multisets (chemical solutions). This model of computation is well-suited to the specification of complex computing infrastructures. In particular, the orderless interactions between elements that occur in large parallel or open systems as well as autonomicity (e.g. self-healing, self-protection, self-optimization, etc.) are naturally expressed as reaction rules. We have described classical coordination mechanisms and parallel programming models (Linda, Petri Nets, Kahn Networks) in the same chemical setting [25]. All these examples put forward the simplicity and expressivity of the chemical paradigm.

A drawback of chemical languages is that their very high-level nature usually leads to very inefficient programs. We are currently looking at approaches to refine chemical programs to more efficient lower-level programs. The idea is to specify separately the data structures, the selection of elements and the scheduling of rules using domain specific languages. The goal is to use these additional components to automatically refine chemical programs into C-like programs. The overall approach is related to aspect-oriented programming where the chemical program represents the base functionality and the other components can be seen as implementation aspects.

This line of research is followed by Marnes Hoff in his PhD thesis. It takes place within the AUTOCHEM project (see Section 8.2.1).

6.6.2. Component-based modeling and reachability analysis of genetic networks

Participant: Gregor Gössler.

Genetic regulatory networks usually encompass a multitude of complex, interacting feedback loops. Being able to model and analyze their behavior is crucial for understanding the interactions between the proteins, and their functions. Genetic regulatory networks have been modeled as discrete transition systems by many approaches, benefiting from a large number of formal verification algorithms available for the analysis of discrete transition systems. However, most of these approaches face the problem of state space explosion, as even models of modest size (from a biological point of view) usually lead to large transition systems, due to a combinatorial blow-up of the number of states. This problem has been addressed with the component-based approach of [50] — based on the mathematically well-founded formalism of qualitative simulation [68] — where the discrete abstraction is constructed and analyzed modularly, allowing to deal with complex, high-dimensional systems.

We have further improved this technique by allowing for a more precise, conservative abstraction, and provided both correctness and completeness results [12].

We are currently working, in cooperation with H. de Jong (IBIS team from Grenoble) and G. Batt (CONTRAINTES team from Rocquencourt), on parametric models of genetic networks that reflect the lack of knowledge about the position of focal points with respect to the thresholds, and the ordering of thresholds. The goal is to determine automatically feasible parameter values corresponding to an observed or desired behavior.

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 [49]. We intend to develop methods for abstraction, and interprocedural and compositional analysis of SystemC models. Interesting results have been obtained regarding (see Section 6.3.3). 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 6.6.1) to program complex computing infrastructures such as grids and real-time deeply-embedded systems. The consortium includes IRISA (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.

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 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 (IRISA), A. Cohen, L. Mandel, and M. Pouzet (INRIA-Saclay), 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-Rocquencourt is working with us on fault tolerant heuristics for their software SYNDEX.
- VERTECS at IRISA/INRIA-Rennes 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, IRISA/INRIA-Rennes) and with R. Douence (ASCOLA, Ecole des Mines de Nantes).
- A. Girault cooperates with the MOAIS and GRAAL projects (CR Grenoble – Rhône-Alpes) on multi-criteria scheduling. A. Girault cooperates also with the VERIMAG lab on model-based design and a compilation tool chain from SIMULINK to distributed platforms, and with the DEMON team of LRI (Orsay) on the distribution of higher-order synchronous data-flow programs.
- G. Gössler cooperates with D. Le Métayer (LICIT action, CR Rhône-Alpes), H. de Jong (IBIS project, CR Rhône-Alpes), and G. Batt (CONTRAINTES project, CR Rocquencourt).
- B. Jeannet cooperates with T. Le Gall (VERTECS, IRISA/INRIA-Rennes) on the analysis of communicating systems, and with C. Constant, T. Jéron and F. Ployette (VERTECS, IRISA/INRIA-Rennes) on test generation.
- J.-B. Raclet cooperates with E. Badouel, A. Benveniste, B. Caillaud and A. Legay (S4 team from IRISA/INRIA-Rennes) on interface theories, and with N. Bertrand (VERTECS team from IRISA/INRIA-Rennes) and S. Pinchinat (S4 team from IRISA/INRIA-Rennes) on timed modal specifications.

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), M. Pouzet (LRI, University of Paris VI), P. Roop, Z. Salcic, A. Malik, and S. Andalam (University of Auckland, New Zealand), and H. Kalla (University of Batna, Algeria).
- P. Fradet and A. Girault collaborate with T. Ayav (University of Izmir, Turkey).

- G. Gössler cooperates with A. Girard (LJK, Grenoble), T. Dang, J. Sifakis, and S. Bensalem (VERIMAG).
- 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[33]. 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[34]. 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.

8.4. International actions

8.4.1. CMCU Tunisia

We have a cooperation in the framework of CMCU (*Comité Mixte pour la Coopération Universitaire*), on the topic of analysis and verification of the safety of safety-critical systems, with ENSI (*Ecole Nationale des Sciences de l'Informatique*) at La Manouba in Tunisia. The other French partner is GIPSA (team of H. Alla).

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

9. Dissemination

9.1. Scientific community

- P. Fradet served in the program committees of FOAL'09 (*International Workshop on Foundations of Aspect-Oriented Languages*), RV'09 (*International Workshop on Runtime Verification*) and AOSD'10 (*International Conference on Aspect-Oriented Software Development*).
- A. Girault served in the programme committee of the DATE'09 Conference[35]. He was publicity chair for the LCTES'09 Conference on Languages, Compilers and Tools for Embedded Systems[36], and web chair for the ESWEEK 2009 Joint Conference on Embedded Systems[37]. Finally, he was assessor for the PhD of Julien Forget (University of Toulouse).

9.2. Teaching

9.2.1. Advising

PhDs:

- Simplice Djoko Djoko, co-advised by P. Fradet (with R. Douence, ASCOLA, Ecole des Mines de Nantes), since 10/2005, PhD in computer science, University of Nantes, defended on June 26th, 2009.
- Marnes Hoff, co-advised by P. Fradet (with J.-L. Giavitto, Université d'Evry), since 04/2008, PhD in computer science, Grenoble INP.
- Mouaiad Alras, co-advised by Alain Girault (with P. Raymond, VERIMAG Grenoble), 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.
- 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/INRIA) since 12/2009, PhD in computer science, UJF, Grenoble.

10. Bibliography

Major publications by the team in recent years

[1] K. ALTISEN, G. GÖSSLER, J. SIFAKIS. *Scheduler Modeling Based on the Controller Synthesis Paradigm*, in "Journal of Real-Time Systems, special issue on "control-theoretical approaches to real-time computing"", vol. 23, n^{o} 1/2, 7-9 2002, p. 55–84.

[2] T. AYAV, P. FRADET, A. GIRAULT. *Implementing fault-tolerance in real-time programs by automatic program transformations*, in "ACM Transactions on Embedded Computing Systems (TECS)", vol. 7, n^{o} 4, july 2008, p. 1-43.

[3] 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.

[35] http://www.date-conference.com
[36] http://www.cse.psu.edu/lctes09
[37] http://www.esweek.org

[4] A. GIRAULT, H. KALLA. *A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate*, in "TDSC", vol. 6, n^{o} 4, December 2009, p. 241-254, http://hal.inria.fr/inria-00177117.

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

[6] 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, vol. 4634, August 2007, http://pop-art.inrialpes.fr/people/bjeannet/publications/sas07.ps.

Year Publications

Doctoral Dissertations and Habilitation Theses

[7] S. DJOKO DJOKO. *Programmation par aspects et préservation de propriétés*, Université de Nantes, June 2009, Ph. D. Thesis.

Articles in International Peer-Reviewed Journal

[8] P. FRADET, S. HONG TUAN HA. *Aspects of Availability - Enforcing timed properties to prevent denial of service*, in "Science of Computer Programming", 2010, In Press.

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

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

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

[12] G. GÖSSLER. *Component-based Modeling and Reachability Analysis of Genetic Networks*, in "ACM/IEEE TCBB", 2010.

[13] 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.

International Peer-Reviewed Conference/Proceedings

[14] 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.

[15] N. BERTRAND, A. LEGAY, S. PINCHINAT, J.-B. RACLET. *A Compositional Approach on Modal Specifications for Timed Systems*, in "Proc. of the 11th International Conference on Formal Engineering Methods (ICFEM'09)", LNCS, Springer, 2009.

[16] N. BERTRAND, S. PINCHINAT, J.-B. RACLET. *Refinement and Consistency of Timed Modal Specifications*, in "Proc. of the 3rd International Conference on Language and Automata Theory and Applications (LATA'09)", LNCS, Springer, April 2009.

[17] 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.

[18] G. GÖSSLER. *Compositional Strategy Mapping*, in "Proc. FSEN'09", F. ARBAB, M. SIRJANI (editors), LNCS, Springer, 2009.

[19] G. GÖSSLER, J.-B. RACLET. *Modal Contracts for Component-based Design*, in "Proc. of Software Engineering and Formal Methods, SEFM'09", IEEE, November 2009, p. 295-303.

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

[21] B. JEANNET, A. MINÉ. *APRON: A Library of Numerical Abstract Domains for Static Analysis*, in "Computer Aided Verification, CAV'2009", LNCS, vol. 5643, 2009, p. 661-667, http://apron.cri.ensmp.fr/library/.

[22] J.-B. RACLET, E. BADOUEL, A. BENVENISTE, B. CAILLAUD, A. LEGAY, R. PASSERONE. *Modal Interfaces: Unifying Interface Automata and Modal Specification s*, in "Proc. of 9th International Conference on Embedded Software (EMSOFT'09)", ACM, 2009.

[23] J.-B. RACLET, E. BADOUEL, A. BENVENISTE, B. CAILLAUD, R. PASSERONE. *Why are modalities good for Interface Theories?*, in "Proc. of the 9th International Conference on Application of Concurrency to System Design (ACSD'09)", IEEE, 2009, p. 199–127.

[24] 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.

Scientific Books (or Scientific Book chapters)

[25] J.-P. BANÂTRE, P. FRADET, Y. RADENAC. *Classical Coordination Mechanisms in the Chemical Model*, in "From Semantics to Computer Science: Essays in Honor of Gilles Kahn", Cambridge University Press, 2009, p. 29-50.

Books or Proceedings Editing

[26] A. GIRAULT, E. RUTTEN (editors). *International Workshop on Model-driven High-level Programming of Embedded Systems, SLA++P'08*, ENTCS, vol. 238, June 2009.

Research Reports

[27] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects Preserving Properties*, n^{o} 7155, INRIA, december 2009, (submitted to journal publication), Research Report.

References in notes

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

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

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

[31] J.-R. BEAUVAIS, E. 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", vol. 10, n^{o} 4, October 2001, p. 397–451.

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

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

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

[35] 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.

[36] 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", vol. 8, n^{o} 2, 1986, p. 244-263.

[37] 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, vol. 2280, 2002.

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

[39] 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.

[40] 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, vol. 2399, July 2002.

[41] G. DELAVAL. *Répartition Modulaire de Programmes Synchrones*, INPG, INRIA Grenoble Rhône-Alpes, projet Pop-Art, July 2008, Ph. D. Thesis.

[42] 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.

[43] L. DOYEN, T. HENZINGER, B. JOBSTMANN, T. PETROV. *Interface Theories with Component Reuse*, in "Proc. EMSOFT'08", ACM, 2008, p. 79-88.

[44] 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.

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

[46] 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.

[47] 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.

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

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

[50] G. GÖSSLER. *Compositional Reachability Analysis of Genetic Networks*, in "CMSB'06", C. PRIAMI (editor), LNBI, vol. 4210, Springer, 2006, p. 212-226.

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

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

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

[54] 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.

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

[56] 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.

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

[58] 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, vol. 3440, April 2005.

[59] K. KEUTZER, S. MALIK, A. NEWTON, J. RABAEY, A. SANGIOVANNI-VINCENTELLI. *System Level Design: Orthogonalization of Concerns and Platform-Based Design*, in "IEEE Trans. on Computer-Aided Design", vol. 19, n^{o} 12, 2000.

[60] K. LARSEN. *Modal Specifications*, in "Proc. International Workshop on Automatic Verification Methods for Finite State Systems", LNCS, vol. 407, Springer, 1989, p. 232-246.

[61] K. LARSEN, U. NYMAN, A. WASOWSKI. *Interface Input/Output Automata*, in "Proc. FM'06", J. MISRA, T. NIPKOW, E. SEKERINSKI (editors), LNCS, vol. 4085, Springer-Verlag, 2006, p. 92-97.

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

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

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

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

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

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

[68] H. DE JONG, J.-L. GOUZÉ, C. HERNANDEZ, M. PAGE, T. SARI, J. GEISELMANN. *Qualitative Simulation of Genetic Regulatory Networks Using Piecewise-Linear Models*, in "Bulletin of Mathematical Biology", vol. 66, 2004, p. 301-340.