IN PARTNERSHIP WITH:
CNRS

Université Rennes 1

Activity Report 2011

Project-Team S4

System Synthesis and Supervision, Scenarios

IN COLLABORATION WITH: Institut de recherche en informatique et systèmes aléatoires (IRISA)

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Embedded and Real Time Systems

Table of contents

Project-Team S4

Keywords: Embedded Systems, Discrete Control Systems, Formal Methods, Timed And Hybrid Systems, Game Theory

S4 *is a joint project of* INRIA, CNRS *and the University of Rennes 1, within* IRISA *(UMR 6074).*

1. Members

Research Scientists

Benoît Caillaud [Team Leader, Junior Researcher, HdR]
Eric Badouel [Junior Researcher, HdR]
Albert Benveniste [Senior Researcher, part-time in S4, HdR]
Philippe Darondeau [Senior Researcher, HdR]

Faculty Member

Sophie Pinchinat [Professor, HdR]

PhD Students

Bastien Maubert
Benoît Delahaye [Research Technologist, funded by the CESAR European project until 30 June 2011]

Post-Doctoral Fellows

Timothy Bourke [Post-Doctoral Fellow, until 31 october 2011, started 16 November 2009, funded by the SYNCHRONICS large-scale initiative action of INRIA]
Uli Farhenberg

Administrative Assistants

Laurence Dinh [TR, part-time in S4, until 14 November 2011]
Angélique Jarnoux [TR, part-time in S4, from 15 November 2011]

2. Overall Objectives

2.1. Introduction

The objective of the project is the realization by algorithmic methods of reactive and distributed systems from partial and heterogeneous specifications. Methods, algorithms and tools are developed to synthesize reactive software from one or several incomplete descriptions of the system's expected behavior, regarding functionality (synchronization, conflicts, communication), control (safety, reachability, liveness), deployment architecture (mapping, partitioning, segregation), or even quantitative performances (response time, communication cost, throughput).

These techniques are better understood on fundamental models, such as automata, Petri nets, event structures and their timed extensions. The results obtained on these basic models are then adapted to those realistic but complex models commonly used to design embedded and telecommunication systems.

The behavioral views of the *Unified Modeling Language* (UML) (sequence diagrams and statecharts), the *High-Level Message Sequence Charts* (HMSC) and the synchronous reactive language Signal are the heart of the software prototypes being developed and the core of the technology transfer strategy of the project.

The scientific objectives of the project can be characterized by the following elements:

A focus on a precise type of applications: The design of real-time embedded software to be deployed over dedicated distributed architectures. Engineers in this field face two important challenges. The first one is related to system specification. Behavioral descriptions should be adaptable and composable. Specifications are expressed as requirements on the system to be designed. These requirements fall into four categories: (i) functional (synchronization, conflict, communication), (ii) control (safety, reachability, liveness), (iii) architectural (mapping, segregation) and (iv) quantitative (response time, communication cost, throughput, etc). The second challenge is the deployment of the design on a distributed architecture. Domain-specific software environments, known as *middleware* or *real-time operating systems* or *communication layers*, are now part of the usual software design process in industry. They provide a specialized and platform-independent distributed environment to higher-level software components. Deployment of software components and services should be done in a safe and efficient manner.

A specific methodology: The development of methods and tools which assist engineers since the very first design steps of reactive distributive software. The main difficulty is the adequacy of the proposed methods with standard design methods based on components and model engineering, which most often rely on heterogeneous formalisms and require correct-by-construction component assembly.

A set of scientific and technological foundations: Those models and methods which encompass (i) the distributed nature of the systems being considered, (ii) true concurrency, and (iii) real-time.

The contribution of the S4 Project-Team consists of algorithms and tools producing distributed reactive software from partial heterogeneous specifications of the system to be synthesized (functionality, control, architecture, quantitative performances). This means that several heterogeneous specifications (for instance, sequence diagrams and state machines) can be combined, analyzed (are the specifications consistent?) and mapped to lower-level specifications (for instance, communicating automata, or Petri nets).

The scientific approach of Team S4 begins with a rigorous modeling of problems and the development of sound theoretical foundations. This not only allows to prove the correctness (functionality and control) of the proposed transformations or analysis; but this can also guarantee the optimality of the quantitative performances of the systems produced with our methods (communication cost, response time).

Synthesis and verification methods are best studied within fundamental models, such as automata, Petri nets, event structures, synchronous transition systems. Then, results can be adapted to more realistic but complex formalisms, such as the UML. The research work of Team S4 is divided in four main tracks:

Petri net synthesis: This track follows up the main research theme of the former Team PARAGRAPHE at INRIA Rennes on the synthesis of Petri net models using the theory of regions.

Heterogeneous systems: This track contributes to the extension of the well-established synchronous paradigm to distributed systems. The aim is to provide a unified framework in which both synchronous systems, and particular asynchronous systems (so-called weakly-synchronous systems) can be expressed, combined, analyzed and transformed.

Reactive components: The design of reusable components calls for rich specification formalisms, with which the interactions of a component with its environment combines expectations with guarantees on its environment.We are investigating questions related to reactive component refinement and composition. We are also investigating the issues of coherence of views and modularity in complex specifications.

Discrete event system synthesis and supervisory control: Many synthesis and supervisory control problems can be expressed with full generality in the *quantified mu-calculus*, including the existence of optimal solutions to such problems. Algorithms computing winning strategies in parity games (associated with formulas in this logic) provide effective methods for solving such control problems. This framework offers means of classifying control problems, according to their decidability or undecidability, but also according to their algorithmic complexity.

2.2. Highlights

The three main achievements of the S4 team during 2011 are:

- The non-standard semantics of hybrid systems (Section 6.2) and its application to the compilation of hybrid data-flow synchronous languages.
- The modal interface theory (Section 6.3) and its implementation in the Mica toolbox (Section 5.1).
- Several results on the separability and the distributability of Petri nets (Section 6.1).

3. Scientific Foundations

3.1. Scientific Foundations

The research work of the team is built on top of solid foundations, mainly, algebraic, combinatorial or logical theories of transition systems. These theories cover several sorts of systems which have been studied during the last thirty years: sequential, concurrent, synchronous or asynchronous. They aim at modeling the behavior of finite or infinite systems (usually by abstracting computations on data), with a particular focus on the control flow which rules state changes in these systems. Systems can be autonomous or reactive, that is, embedded in an environment with which the system interacts, both receiving an input flow, and emitting an output flow of events and data. System specifications can be explicit (for instance, when the system is specified by an automaton, extensively defined by a set of states and a set of transitions), or implicit (symbolic transition rules, usually parameterized by state or control variables; partially-synchronized products of finite transition systems; Petri nets; systems of equations constraining the transitions of synchronous reactive systems, according to their input flows; etc.). Specifications can be non-ambiguous, meaning that they fully define at most one system (this holds in the previous cases), or they can be ambiguous, in which case more than one system is conforming to the specification (for instance, when the system is described by logical formulas in the modal mu-calculus, or when the system is described by a set of scenario diagrams, such as *Sequence Diagrams* or *Message Sequence Charts*).

Systems can be described in two ways: either the state structure is described, or only the behavior is described. Both descriptions are often possible (this is the case for formal languages, automata, products of automata, or Petri nets), and moving from one representation to the other is achieved by folding/unfolding operations.

Another taxonomy criteria is the concurrency these models can encompass. Automata usually describe sequential systems. Concurrency in synchronous systems is usually not considered. In contrast, Petri nets or partially-synchronized products of automata are concurrent. When these models are transformed, concurrency can be either preserved, reflected or even, infused. An interesting case is whenever the target architecture requires distributing events among several processes. There, communication-efficient implementations require that concurrency is preserved as far as possible and that, at the same time, causality relations are also preserved. These notions of causality and independence are best studied in models such as concurrent automata, Petri nets or Mazurkiewicz trace languages.

Here are our sources of inspiration regarding formal mathematical tools:

1. Jan van Leeuwen (ed.), *Handbook of Theoretical Computer Science - Volume B: Formal Models and Semantics*, Elsevier, 1990.
2. Jörg Desel, Wolfgang Reisig and Grzegorz Rozenberg (eds.), *Lectures on Concurrency and Petri nets*, Lecture Notes in Computer Science, Vol. 3098, Springer, 2004.
3. Volker Diekert and Grzegorz Rozenberg (eds.), *The Book of Traces*, World Scientific, 1995.
4. André Arnold and Damian Niwinski, *Rudiments of Mu-Calculus*, North-Holland, 2001.
5. Gérard Berry, *Synchronous languages for hardware and software reactive systems - Hardware Description Languages and their Applications*, Chapman and Hall, 1997.

Our research exploits decidability or undecidability results on these models (for instance, inclusion of regular languages, bisimilarity on automata, reachability on Petri nets, validity of a formula in the mu-calculus, etc.) and also, representation theorems which provide effective translations from one model to another. For instance, Zielonka's theorem yields an algorithm which maps regular trace languages to partially-synchronized products of finite automata. Another example is the theory of regions, which provides methods for mapping finite or infinite automata, languages, or even *High-Level Message Sequence Charts* to Petri nets. A further example concerns the mu-calculus, in which algorithms computing winning strategies for parity games can be used to synthesize supervisory control of discrete event systems.

Our research aims at providing effective representation theorems, with a particular emphasis on algorithms and tools which, given an instance of one model, synthesize an instance of another model. In particular we have contributed a theory, several algorithms and a tool for synthesizing Petri nets from finite or infinite automata, regular languages, or languages of *High-Level Message Sequence Charts*. This also applies to our work on supervisory control of discrete event systems. In this framework, the problem is to compute a system (the controller) such that its partially-synchronized product with a given system (the plant) satisfies a given behavioral property (control objective, such as a regular language or satisfaction of a mu-calculus formula).

Software engineers often face problems similar to *service adaptation* or *component interfacing*, which in turn, often reduce to particular instances of system synthesis or supervisory control problems.

4. Application Domains

4.1. Panorama

Results obtained in Team S4 apply to the design of real-time systems consisting of a distributed hardware architecture, and software to be deployed over that architecture. A particular emphasis is put on *embedded* systems (automotive, avionics, production systems, *etc.*), and also, to a lesser extent, *telecommunication* and *production* systems.

Our work on contract-based modular reasoning has found applications in embedded system design, by supporting and controlling concurrent design activities in aeronautics.

Our work on heterogeneous reactive systems facilitates the mapping of pure synchronous designs onto a distributed architecture where communication is done by non-instantaneous message passing. These architectures can be usual *asynchronous* distributed systems or, more interestingly, *loosely time-triggered architectures* (LTTA), such as those found on board of recent Airbus aircrafts. In the latter, communication is done by periodically reading or writing (according to local inaccurate real-time clocks) distributed shared variables, without any means of synchronizing these operations. The consequence is that values may be lost or duplicated, and software designed for such specific architectures must resist losses or duplications of messages. In the context of the IST European network of excellence ARTIST we have developed a theoretical and methodological framework in which the correct mapping of synchronous designs to such particular distributed architectures can be best understood, at a high level of abstraction.

5. Software

5.1. Mica: A Modal Interface Compositional Analysis Toolbox

Participant: Benoît Caillaud.

http://www.irisa.fr/s4/tools/mica/

Mica is an Ocaml library developed by Benoît Caillaud implementing the Modal Interface algebra published in [18]. The purpose of Modal Interfaces is to provide a formal support to contract based design methods in the field of system engineering. Modal Interfaces enable compositional reasoning methods on I/O reactive systems.

In Mica, systems and interfaces are represented by extension. However, a careful design of the state and event heap enables the definition, composition and analysis of reasonably large systems and interfaces. The heap stores states and events in a hash table and ensures structural equality (there is no duplication). Therefore complex data-structures for states and events induce a very low overhead, as checking equality is done in constant time.

Thanks to the Inter module and the mica interactive environment, users can define complex systems and interfaces using Ocaml syntax. It is even possible to define parameterized components as Ocaml functions.

Mica is available as an open-source distribution, under the CeCILL-C Free Software License Agreement (http://www.cecill.info/licences/Licence_CeCILL-C_V1-en.html).

5.2. Synet: A General Petri-Net Sythesis Toolbox

Participant: Benoît Caillaud.

http://www.irisa.fr/s4/tools/synet/

Synet is a software tool for the synthesis of bounded and unbounded Petri-nets, based on the theory of regions [31]. It can synthesize Petri-nets from automata or regular expression and can be configured by command-line options to synthesize nets modulo graph isomorphism or language equality. Petri nets computed by Synet can be displayed using the GraphViz 2D graph layout software, or saved to a file for further transformation and analysis.

The tool actually implements two linear-algebraic synthesis methods: A first method uses the simplex algorithm and the second one is based on the computation of extremal rays of polyhedral cones, using Chernikova's algorithm [34]. Both methods imply that the input graphs are given by extension. Nevertheless, Synet yields good performances on many practical use-cases and is the only tool supporting unbounded net synthesis.

The main application of Synet is the synthesis of communicating distributed protocols and controllers [30]. Synthesis is constrained to produce so-called distributables nets [33], a class of nets that can be turned into networks of communicating automata by automated methods. This allows to divide the synthesis problem in two steps: Given the specification of a protocol as a finite automaton, (i) synthesize (if exists) a distributable net, and then (ii) derive a network of communicating automata from the distributable net. While the second step is automatic and straightforward, the first step is in essence a computer assisted design task, where the distributed Petri-net synthesis algorithm helps the designer to refine the protocol specification into a graph isomorphic to the marking graph of a distributable net.

6. New Results

6.1. Petri Nets and their Synthesis

Participant: Philippe Darondeau.

6.1.1. Petri Net Reachability Graphs: Decidability Status of FO Properties

In [24], we investigate the decidability and complexity status of model-checking problems on unlabelled reachability graphs of Petri nets by considering first-order, modal and pattern-based languages without labels on transitions or atomic propositions on markings. We consider several parameters to separate decidable problems from undecidable ones. Not only are we able to provide precise borders and a systematic analysis, but we also demonstrate the robustness of our proof techniques.

6.1.2. Separability in Persistent Petri Nets

We prove in [14] that the separability of plain, bounded, reversible and persistent Petri nets, a class of nets that extends the well-known live and bounded marked graphs. We establish first a weak form of separability, already known to hold for marked graphs, in which every firing sequence is simulated by a firing sequence of k parallel instances identical firing counts. We establish on top of this a strong form of separability, in which every firing sequence of is simulated by identical firing sequences.

6.1.3. Petri Net Distributability

A Petri net is distributed if, given an allocation of transitions to (geographical) locations, no two transitions at different locations share a common input place. A system is distributable if there is some distributed Petri net implementing it. We address in [23] the question of which systems can be distributed, while respecting a given allocation. The paper states the problem formally and discusses several examples illuminating — to the best of the authors' knowledge — the current status of this work.

6.2. Heterogeneous Systems

Participants: Eric Badouel, Albert Benveniste, Timothy Bourke, Benoît Caillaud.

6.2.1. Hybrid Modeling

Hybrid modeling tools like Simulink have evolved from simulation platforms into development platforms on which testing, verification and code generation are also performed. It is critical to ensure that the results of simulation, compilation and verification are consistent. Synchronous languages have addressed these issues but only for discrete systems. Reprising earlier work [32], we present in [21] a hybrid modeler built from a synchronous language and an of-the-shelf numerical solver. The main novelty is a language with hierarchical automata that can be arbitrarily mixed with data-flow and ordinary differential equations (ODEs). A type system statically ensures that discrete state changes are aligned with zero-crossing events and that the function passed to the numerical solver has no side-effects during integration. Well-typed programs are compiled by source-to-source translation into synchronous code which is then translated into sequential code using an existing synchronous language compiler.

Starting from a minimal, yet full-featured, Lustre-like synchronous language, we present in [22] a conservative extension where data-flow equations can be mixed with ordinary differential equations (ODEs) with possible reset. A type system is proposed to statically distinguish discrete computations from continuous ones and to ensure that signals are used in their proper domains. We propose a semantics based on non-standard analysis which gives a synchronous interpretation to the whole language, clarifies the discrete/continuous interaction and the treatment of zero-crossings, and also allows the correctness of the type system to be established. The extended data-flow language is realized through a source-to-source transformation into a synchronous subset, which can then be compiled using existing tools into routines that are both efficient and bounded in their use of memory. These routines are orchestrated with a single off-the-shelf numerical solver using a simple but precise algorithm which treats causally-related cascades of zero-crossings. We have validated the viability of the approach through experiments with the Sundials library.

6.2.2. Distributed Structured Documents

Evaluation of attributes w.r.t. an attribute grammar can be obtained by inductively computing a function expressing the dependencies of the synthesized attributes on inherited attributes. This higher-order functional approach to attribute grammars leads to a straightforward implementation using a higher-order lazy functional language like Haskell. The resulting evaluation functions are, however, not easily amenable to optimization rules. We present in [12] an alternative first-order functional interpretation of attribute grammars where the input tree is replaced with an extended cyclic tree each node of which is aware of its context viewed as an additional child tree. By the way, we demonstrate that these cyclic representations of zippers (trees with their context) are natural generalizations of doubly-linked lists to trees over an arbitrary signature.

6.3. Component-Based Design

Participants: Eric Badouel, Albert Benveniste, Benoît Caillaud, Benoît Delahaye, Sophie Pinchinat.

6.3.1. The Modal Interface Theory

In [18], we present the *modal interface* theory, a unification of *interface automata* and *modal specifications*, two radically dissimilar models for interface theories. Interface automata is a game-based model, which allows the designer to express assumptions on the environment and which uses an optimistic view of composition: *two components can be composed if there is an environment where they can work together*. Modal specifications are a language theoretic account of a fragment of the modal mu-calculus logic with a rich composition algebra which meets certain methodological requirements but which does not allow the environment and the component to be distinguished. The present paper contributes a more thorough unification of the two theories by correcting a first attempt in this direction by Larsen et al., drawing a complete picture of the modal interface algebra, and pushing the comparison between interface automata, modal automata and modal interfaces even further.

6.3.2. A Stochastic Interface Theory

Notions of specification, implementation, satisfaction, and refinement, together with operators supporting stepwise design, constitute a specification theory. In [16], we construct such a theory for Markov Chains (MCs) employing a new abstraction of a Constraint MC. Constraint MCs permit rich constraints on probability distributions and thus generalize prior abstractions such as Interval MCs. Linear (polynomial) constraints suffice for closure under conjunction (respectively parallel composition). This is the first specification theory for MCs with such closure properties. We discuss its relation to simpler operators for known languages such as probabilistic process algebra. Despite the generality, all operators and relations are computable.

6.3.3. Contract-Based Compositional Analysis of Stochastic Systems

A contract allows to distinguish hypotheses made on a system (the guarantees) from those made on its environment (the assumptions). In [17], we focus on models of Assume/Guarantee contracts for (stochastic) systems. We consider contracts capable of capturing reliability and availability properties of such systems. We also show that classi- cal notions of Satisfaction and Refinement can be checked by effective methods thanks to a reduction to classical verification problems. Finally, theorems supporting compositional reasoning and enabling the scalable analysis of complex systems are also studied.

6.3.4. Modal event-clock specifications for timed component-based design

On the one hand, modal specifications 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 [13], we propose a timed extension of modal specifications, together with fundamental operations (conjunction, product, and quotient) that enable reasoning in a compositional way about timed system. The specifications are given as modal event-clock automata, where clock resets are easy to handle. We develop an entire theory that promotes ecient incremental design techniques.

6.4. Scheduling and Supervisory Control

Participants: Eric Badouel, Benoît Caillaud, Philippe Darondeau.

6.4.1. Model Identification and Synthesis of Discrete-Event Systems

Book chapter [28] focuses on two important and tightly related problems, namely the identification and synthesis of discrete-event systems. Particular attention is devoted to two main formalisms in this area, i.e., finite state automata and Petri nets. The goal of this chapter is to provide a collection of references in this framework, and discuss the main research areas where such problems have been investigated. Due to the extensive literature, only some of the results are discussed in a certain detail, such as the basic ideas related to the theory of regions and the synthesis of labeled Petri nets, while other results are simply mentioned and the reader is addressed to the specific contributions for more details.

6.4.2. Assembling Sessions

Sessions are a central paradigm inWeb services to implement decentralized transactions with multiple participants. Sessions enable the cooperation of workflows while at the same time avoiding the mixing of workflows from distinct transactions. Languages such as BPEL, ORC, AXML that implement Web Services usually realize sessions by attaching unique identifiers to transactions. The expressive power of these languages makes the properties of the implemented services undecidable. In [25], we propose a new formalism for modelling web services. Our model is session-based, but avoids using session identifiers. The model can be translated to a dialect of Petri nets that allows the verification of important properties of web services.

6.4.3. Towards Distributed Control of Discrete Event Systems

To initiate a discussion on the modeling requirements for distributed control of discrete-event systems, a partially-automated region based methodology is presented in [26]. The methodology is illustrated via a well-known example from distributed computing: the dining philosophers.

6.4.4. Communicating Decentralized Control

Frameworks that incorporate communication into decentralized supervisory control theory address the following problem: find locations in the evolution of the plant behavior where some supervisors send information so that a supervisor that was unable to make the correct control decision prior to receiving external information, is now capable of making the correct control decision. We propose in [19] a solutions to this problem and identify an earliest and a latest placement where such communication results in the synthesis of a correct control solution. In addition to a first and last communication opportunity, there may be a selection of intermediate possibilities where communication would produce the correct control solution. We present a computable procedure to identify a broader range of suitable communication locations.

6.4.5. Residuation of tropical series: rationality issues

In [20], the decidability of existence, rationality of delay controllers and robust delay controllers are investigated for systems with time weights in the tropical and interval semirings. Depending on the (max,+) or (min,+)-rationality of the series specifying the controlled system and the control objective, cases are identified where the controller series defined by residuation is rational, and when it is positive (i.e., when delay control is feasible). When the control objective is specified by a tolerance, i.e. by two bounding rational series, a nice case is identified in which the controller series is of the same rational type as the system specification series.

6.5. Games, Logic and System Synthesis

Participants: Bastien Maubert, Sophie Pinchinat.

6.5.1. Opacity Issues in Games with Imperfect Information

In [27], we study the class of games with opacity condition, which are two-player games with imperfect information in which one of the players only has imperfect information, and where the winning condition relies on the information he has along the play. Those games are relevant for security aspects of computing systems: a play is opaque whenever the player who has imperfect information never "knows" for sure that the current position is one of the distinguished "secret" positions. We study the problems of deciding the existence of a winning strategy for each player, and we call them the opacity-violate problem and the opacity-guarantee problem. Focusing on the player with perfect information is new in the field of games with imperfect-information because when considering classical winning conditions it amounts to solving the underlying perfect-information game. We establish the EXPTIME-completeness of both above-mentioned problems, showing that our winning condition brings a gap of complexity for the player with perfect information, and we exhibit the relevant opacity-verify problem, which noticeably generalizes approaches considered in the literature for opacity analysis in discrete-event systems. In the case of blindfold games, this problem relates to the two initial ones, yielding the determinacy of blindfold games with opacity condition and the PSPACE-completeness of the three problems.

6.5.2. Hardness of preorder checking for basic formalisms

We investigate in [15] the complexity of preorder checking when the specification is a flat finite-state system whereas the implementation is either a non-flat finite-state system or a standard timed automaton. In both cases, we show that simulation checking is Exptime-hard, and for the case of a non-flat implementation, the result holds even if there is no synchronization between the parallel components and their alphabets of actions are pairwise disjoint. Moreover, we show that the considered problems become Pspace-complete when the specification is assumed to be deterministic. Additionally, we establish that comparing a synchronous non-flat system with no hiding and a flat system is Pspace-hard for any relation between trace containment and bisimulation equivalence.

7. Partnerships and Cooperations

7.1. Disc: Distributed Supervisory Control of Large Plants

Participant: Philippe Darondeau.

ICT STREP 224498 Disc (September 2008 to December 2011), http://www.disc-project.eu

Started on 1 September 2008, Disc is a project supported by the ICT program of the European Union.

The aim of the project is to enable the supervisory control of networked embedded systems. These distributed plants are composed by several local agents that take concurrently decisions, based on information that may be local or received from neighbouring agents; they require scalable and self-organising platforms for advanced computing and control. The evolution is guided by the occurrence of asynchronous events, as opposed to other real-time models where the event occurrence is time-triggered.

The partners of the project come from academia (University of Cagliari, CWI - Amsterdam, Ghent University, Technical University of Berlin, University of Zaragoza, INRIA, Czech Academy of Sciences), from industry (Akhela s.r.l., Italy and CyBio AG, Germany), and from a governmental instance (Ministry of the Flemish Government, Belgium).

Philippe Darondeau has worked in this context with Eric Badouel, Anne Bouillard and Jan Komenda (Czech Academy of Sciences, Brno) on the synthesis of robust delay-controllers for timed systems modelled with rational power series. He works also towards applying the synthesis of distributable Petri nets to asynchronous and distributed supervisory control.

7.2. Synchronics: Language Platform for Embedded System Design

Participants: Albert Benveniste, Timothy Bourke, Benoît Caillaud.

Large initiative action funded by INRIA. http://synchronics.inria.fr/

This project, started Jan 1st 2008, is supported by INRIA. It capitalizes on recent extensions of data-flow synchronous languages (mode automata, Lucid Synchrone, Signal, Lustre, ReactiveML, relaxed forms of synchronous composition or compilation techniques for various platforms). We aim to address the main challenges of embedded system design, starting from a single, semantically well founded programming language.

Our contribution in 2011 is detailed in Section 6.2. A detailed account of the work carried out in Synchronics can be found in the slides presented during the mid-term evaluation seminar of the action: http://synchronics.inria.fr/doku.php/mid-term-review

8. Dissemination

8.1. Teaching, Committee Work and Organization of Scientific Events

- Eric Badouel is the secretary of the steering committee of CARI, the African Conference on Research on Computer Science and Applied Mathematics. He takes part in the programme committee and in the organizing committee of CARI 2011. He is a member of the editorial board of the ARIMA Journal. He has taught an advanced course on functional programming and the manipulations of structured documents in the Second year of the Master of Research in Computer Science, Université Cheikh Anta Diop (UCAD) in Dakar, Senegal.
- Albert Benveniste is associated editor at large (AEAL) for the journal *IEEE Trans. on Automatic Control*. He is member of the Strategic Advisory Council of the Institute for Systems Research, Univ. of Maryland, College Park, USA. He belongs to the Scientific Advisory Board of INRIA, where he is in charge of the area of Embedded Systems.[1]
- Benoît Caillaud has been program committee co-chair of the International Conference on Application of Concurrency to System Design (ACSD) [29]. He has also served on the program committee of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011).
- Sophie Pinchinat has organized the GIPSy Workshop on Games, Logics and Security (2nd edition) in Rennes (25-27 October 2011). See details at http://www.irisa.fr/prive/Sophie.Pinchinat/GIPSy/gipsy11.html. Sophie Pinchinat is serving in the editorial board of the Journal on Discrete Event Dynamic Systems, Theory and Applications.

9. Bibliography

Major publications by the team in recent years

[1] E. BADOUEL, M. BEDNARCZYK, A. BORZYSZKOWSKI, B. CAILLAUD, P. DARONDEAU. *Concurrent Secrets*, in "Discrete Event Dynamic Systems", December 2007, vol. 17, n^{o} 4, p. 425-446, http://dx.doi.org/10.1007/s10626-007-0020-5.

[2] E. BADOUEL, M. BEDNARCZYK, P. DARONDEAU. *Generalized Automata and their Net Representations*, H. EHRIG, G. JUHÀS, J. PADBERG, G. ROZENBERG (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2128, p. 304–345, http://link.springer.de/link/service/series/0558/bibs/2128/21280304.htm.

[3] E. BADOUEL, B. CAILLAUD, P. DARONDEAU. *Distributing Finite Automata through Petri Net Synthesis*, in "Journal on Formal Aspects of Computing", 2002, vol. 13, p. 447–470, http://dx.doi.org/10.1007/s001650200022.

[4] E. BADOUEL, P. DARONDEAU. *Theory of regions*, in "Lectures on Petri Nets I: Basic Models", Lecture Notes in Computer Science, Springer, 1999, vol. 1491, p. 529–586.

[5] A. BENVENISTE, B. CAILLAUD, LUCA P. CARLONI, P. CASPI, ALBERTO L. SANGIOVANNI-VINCENTELLI. *Composing heterogeneous reactive systems*, in "ACM Trans. Embedded Comput. Syst.", 2008, vol. 7, n^{o} 4, http://doi.acm.org/10.1145/1376804.1376811.

[1] Only facts related to the activities of Team S4 are mentioned. Other roles or duties concern the DistribCom or Sisthem teams, to which A. Benveniste also belongs.

[6] A. BENVENISTE, B. CAILLAUD, P. LE GUERNIC. *Compositionality in dataflow synchronous languages: specification and distributed code generation*, in "Information and Computation", 2000, vol. 163, p. 125-171.

[7] B. CAILLAUD, P. DARONDEAU, L. HÉLOUËT, G. LESVENTES. *HMSCs as specifications... with PN as completions*, F. CASSEZ, C. JARD, B. ROZOY, M. DERMOT (editors), Lecture Notes in Computer Science, Springer, 2001, vol. 2067, p. 125–152, http://www.irisa.fr/s4/download/papers/hmsc2pn_movep2k_lncs.ps.gz.

[8] G. FEUILLADE, S. PINCHINAT. *Modal Specifications for the Control Theory of Discrete-Event Systems*, in "Discrete Event Dynamic Systems", 2007, vol. 17, n^{o} 2, p. 211–232, http://dx.doi.org/10.1007/s10626-006-0008-6.

[9] D. POTOP-BUTUCARU, B. CAILLAUD. *Correct-by-Construction Asynchronous Implementation of Modular Synchronous Specifications*, in "Fundamenta Informaticae", 2007, vol. 78, n^{o} 1, p. 131–159.

[10] S. RIEDWEG, S. PINCHINAT. *Quantified Mu-Calculus for Control Synthesis*, in "MFCS 2003, 28th International Symposium on Mathematical Foundations of Computer Science", Lecture notes in computer science, Springer, aug 2003, vol. 2747, p. 642–651, http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2747&spage=642.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] B. CAILLAUD. *Analysis, Control and Synthesis of Concurrent Systems*, University of Rennes 1, March 2011, Habilitation Thesis.

Articles in International Peer-Reviewed Journal

[12] E. BADOUEL, B. FOTSING, R. TCHOUGONG. *Attribute Grammars as Recursion Schemes over Cyclic Representations of Zippers*, in "Electr. Notes Theor. Comput. Sci.", 2011, vol. 229, n^{o} 5, p. 39-56, http://dx.doi.org/10.1016/j.entcs.2011.02.015.

[13] N. BERTRAND, A. LEGAY, S. PINCHINAT, J.-B. RACLET. *Modal event-clock specifications for timed component-based design*, in "Science of Computer Programming", 2011, to appear, http://dx.doi.org/10.1016/j.scico.2011.01.007.

[14] E. BEST, P. DARONDEAU. *Separability in Persistent Petri Nets*, in "Fundamenta Informaticae", 2011, to appear.

[15] L. BOZZELLI, A. LEGAY, S. PINCHINAT. *Hardness of preorder checking for basic formalisms*, in "Theor. Comput. Sci.", 2011, vol. 412, n^{o} 49, p. 6795-6808, http://dx.doi.org/10.1016/j.tcs.2011.08.037.

[16] B. CAILLAUD, B. DELAHAYE, K. G. LARSEN, A. LEGAY, M. L. PEDERSEN, A. WASOWSKI. *Constraint Markov Chains*, in "Theor. Comput. Sci.", 2011, vol. 412, n^{o} 34, p. 4373-4404, http://dx.doi.org/10.1016/j.tcs.2011.05.010.

[17] B. DELAHAYE, B. CAILLAUD, A. LEGAY. *Probabilistic contracts: a compositional reasoning methodology for the design of systems with stochastic and/or non-deterministic aspects*, in "Formal Methods in System Design", 2011, vol. 38, n° 1, p. 1-32, http://dx.doi.org/10.1007/s10703-010-0107-8.

[18] J.-B. RACLET, E. BADOUEL, A. BENVENISTE, B. CAILLAUD, A. LEGAY, R. PASSERONE. *A Modal Interface Theory for Component-based Design*, in "Fundam. Inform.", 2011, vol. 108, n° 1-2, p. 119-149, http://dx.doi.org/10.3233/FI-2011-416.

[19] L. RICKER, B. CAILLAUD. *Mind the gap: Expanding communication options in decentralized discrete-event control*, in "Automatica", 2011, vol. 47, n° 11, p. 2364-2372, http://dx.doi.org/10.1016/j.automatica.2011.08.040.

International Conferences with Proceedings

[20] E. BADOUEL, A. BOUILLARD, P. DARONDEAU, J. KOMENDA. *Residuation of tropical series: rationality issues*, in "CDC-ECC", IEEE SOCIETY (editor), 2011, to appear.

[21] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *A hybrid synchronous language with hierarchical automata: static typing and translation to synchronous code*, in "Proceedings of the 11th International Conference on Embedded Software, EMSOFT 2011", S. CHAKRABORTY, A. JERRAYA, S. K. BARUAH, S. FISCHMEISTER (editors), ACM, 2011, p. 137-148, http://doi.acm.org/10.1145/2038642.2038664.

[22] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *Divide and recycle: types and compilation for a hybrid synchronous language*, in "LCTES", J. VITEK, B. DE SUTTER (editors), ACM, 2011, p. 61-70, http://doi.acm.org/10.1145/1967677.1967687.

[23] E. BEST, P. DARONDEAU. *Petri Net Distributability*, in "PSI", Lecture Notes in Computer Science, Springer, 2011, vol. 7162, to appear.

[24] P. DARONDEAU, S. DEMRI, R. MEYER, C. MORVAN. *Petri Net Reachability Graphs: Decidability Status of FO Properties*, in "FSTTCS", S. CHAKRABORTY, A. KUMAR (editors), LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, vol. 13, p. 140-151, http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2011.140.

[25] P. DARONDEAU, L. HÉLOUËT, M. MUKUND. *Assembling Sessions*, in "ATVA", T. BULTAN, P.-A. HSIUNG (editors), Lecture Notes in Computer Science, Springer, 2011, vol. 6996, p. 259-274, http://dx.doi.org/10.1007/978-3-642-24372-1_19.

[26] P. DARONDEAU, L. RICKER. *Towards Distributed Control of Discrete Event Systems*, in "Workshop on Applications of Region Theory", J. DESEL, A. YAKOVLEV (editors), CEUR Workshop Proceedings, 2011, vol. 725, p. 63-78, http://ceur-ws.org/Vol-725.

[27] B. MAUBERT, S. PINCHINAT, L. BOZZELLI. *Opacity Issues in Games with Imperfect Information*, in "GandALF", G. D'AGOSTINO, S. LA TORRE (editors), EPTCS, 2011, vol. 54, p. 87-101, http://dx.doi.org/10.4204/EPTCS.54.7.

Scientific Books (or Scientific Book chapters)

[28] M. P. CABASINO, P. DARONDEAU, M. P. FANTI, C. SEATZU. *Model Identification and Synthesis of Discrete-Event Systems*, IEEE-Wiley, 2011, to appear.

Books or Proceedings Editing

[29] B. CAILLAUD, J. CARMONA, K. HIRAISHI (editors). *Eleventh International Conference on Application of Concurrency to System Design*, IEEE, Newcastle Upon Tyne, UK, June 2011, http://dx.doi.org/10.1109/ACSD.2011.1.

References in notes

[30] E. BADOUEL, B. CAILLAUD, P. DARONDEAU. *Distributing Finite Automata through Petri Net Synthesis*, in "Journal on Formal Aspects of Computing", 2002, vol. 13, p. 447–470.

[31] E. BADOUEL, P. DARONDEAU. *Theory of regions*, in "Lectures on Petri Nets I: Basic Models", Lecture Notes in Computer Science, Springer, 1999, vol. 1491, p. 529–586.

[32] A. BENVENISTE, B. CAILLAUD, M. POUZET. *The Fundamentals of Hybrid Systems Modelers*, in "49th IEEE Conference on Decision and Control (CDC 2010)", IEEE Computer Society, 2010.

[33] R. P. HOPKINS. *Distributable nets*, in "Advances in Petri Nets 1991, Papers from the 11th International Conference on Applications and Theory of Petri Nets", G. ROZENBERG (editor), Lecture Notes in Computer Science, Springer, 1991, vol. 524, p. 161–187.

[34] A. SCHRIJVER. *Theory of linear and integer programming*, Wiley, April 1998.