Activity Report 2018

Project-Team TEA

Time, Events and Architectures

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Embedded and Real-time Systems

Table of contents

Project-Team TEA

Creation of the Team: 2014 January 01, updated into Project-Team: 2015 January 01

Keywords:

Computer Science and Digital Science:

A1.2.5. - Internet of things
A1.2.7. - Cyber-physical systems
A1.5.2. - Communicating systems
A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.6. - Concurrent programming
A2.1.9. - Synchronous languages
A2.1.10. - Domain-specific languages
A2.2.1. - Static analysis
A2.2.4. - Parallel architectures
A2.3. - Embedded and cyber-physical systems
A2.3.1. - Embedded systems
A2.3.2. - Cyber-physical systems
A2.3.3. - Real-time systems
A2.4. - Formal method for verification, reliability, certification
A2.4.1. - Analysis
A2.4.2. - Model-checking
A2.4.3. - Proofs
A2.5. - Software engineering
A2.5.1. - Software Architecture & Design
A2.5.2. - Component-based Design
A4.4. - Security of equipment and software
A4.5. - Formal methods for security
A7.2. - Logic in Computer Science
A7.2.3. - Interactive Theorem Proving
A7.3. - Calculability and computability
A8.1. - Discrete mathematics, combinatorics
A8.3. - Geometry, Topology

Other Research Topics and Application Domains:

B5.1. - Factory of the future
B6.1.1. - Software engineering
B6.4. - Internet of things
B6.6. - Embedded systems

1. Team, Visitors, External Collaborators

Research Scientists

Thierry Gautier [Inria, Researcher]
Jean-Pierre Talpin [Inria, Team leader, Senior Researcher, HDR]

Faculty Members
Rajesh Kumar Gupta [University of California at San Diego, Inria International Chair]
Shuvra Bhattacharyya [University of Maryland, INSA-Inria International Chair, from Deccember 2018]

PhD Students
Lucas Franceschino [Inria, from October 2018]
Simon Lunel [Mitsubishi Electric]
Jean Joseph Marty [Inria]
Liangcong Zhang [China Scholarship Council]

Technical staff
Loïc Besnard [CNRS]
Hai Nam Tran [Inria, until August 2018]

Administrative Assistant
Armelle Mozziconacci [CNRS]

Visiting Scientists
Alexandre Honorat [IETR]
Deian Stefan [UC San Diego]
Shravan Narayan [UC San Diego]
Liangtai Wang [Chinese Academy of Science]

2. Overall Objectives

2.1. Introduction

An embedded architecture is an artifact of heterogeneous constituents and at the crossing of several design viewpoints: software, embedded in hardware, interfaced with the physical world. Time takes different forms when observed from each of these viewpoints: continuous or discrete, event-based or time-triggered. Unfortunately, modeling and programming formalisms that represent software, hardware and physics significantly alter this perception of time. Moreover, time reasoning in system design is usually isolated to a specific design problem: simulation, profiling, performance, scheduling, parallelization, simulation. The aim of project-team TEA is to define conceptually unified frameworks for reasoning on composition and integration in cyber-physical system design, and to put this reasoning to practice by revisiting analysis and synthesis issues in real-time system design with soundness and compositionality gained from formalization.

2.2. Context

In the construction of complex systems, information technology (IT) has become a central force of revolutionary changes, driven by the exponential increase of computational power. In the field of telecommunication, IT provides the necessary basis for systems of networked distributed applications. In the field of control engineering, IT provides the necessary basis for embedded control applications. The combination of telecommunication and embedded systems into networked embedded systems opens up a new range of systems, capable of providing more intelligent functionality thank to information and communication (ICT). Networked embedded systems have revolutionized several application domains: energy networks, industrial automation and transport systems.

20th-century science and technology brought us effective methods and tools for designing both computational and physical systems. But the design of cyber-physical systems (CPS) is much more than the union of those two fields. Traditionally, information scientists only have a hazy notion of requirements imposed by the physical environment of computers. Similarly, mechanical, civil, and chemical engineers view computers strictly as devices executing algorithms. To the extent we have designed CPS, we have done so in an ad hoc, on-off manner that is not repeatable. A new science of CPS design will allow us to create new machines with complex dynamics and high reliability, to apply its principles to new industries and applications in a reliable and economically efficient way. Progress requires nothing less than the construction of a new science and technology foundation for CPS that is simultaneously physical and computational.

2.3. Motivations

Beyond the buzzword, a CPS is an ubiquitous object of our everyday life. CPSs have evolved from individual independent units (e.g an ABS brake) to more and more integrated networks of units, which may be aggregated into larger components or sub-systems. For example, a transportation monitoring network aggregates monitored stations and trains through a large scale distributed system with relatively high latency. Each individual train is being controlled by a train control network, each car in the train has its own real-time bus to control embedded devices. More and more, CPSs are mixing real-time low latency technology with higher latency distributed computing technology.

In the past 15 years, CPS development has moved towards Model Driven Engineering (MDE). With MDE methodology, first all requirements are gathered together with use cases, then a model of the system is built (sometimes several models) that satisfy the requirements. There are several modeling formalisms that have appeared in the past ten years with more or less success. The most successful are the *executable* models [1] [2] [3], i.e., models that can be simulated, exercised, tested and validated. This approach can be used for both software and hardware.

A common feature found in CPSs is the ever presence of concurrency and parallelism in models. Large systems are increasingly mixing both types of concurrency. They are structured hierarchically and comprise multiple synchronous devices connected by buses or networks that communicate asynchronously. This led to the advent of so-called GALS (Globally Asynchronous, Locally Synchronous) models, or PALS (Physically Asynchronous, Logically Synchronous) systems, where reactive synchronous objects are communicating asynchronously. Still, these infrastructures, together with their programming models, share some fundamental concerns: parallelism and concurrency synchronization, determinism and functional correctness, scheduling optimality and calculation time predictability.

Additionally, CPSs monitor and control real-world processes, the dynamics of which are usually governed by physical laws. These laws are expressed by physicists as mathematical equations and formulas. Discrete CPS models cannot ignore these dynamics, but whereas the equations express the continuous behavior usually using real numbers (irrational) variables, the models usually have to work with discrete time and approximate floating point variables.

2.4. Challenges

A cyber-physical, or reactive, or embedded system is the integration of heterogeneous components originating from several design viewpoints: reactive software, some of which is embedded in hardware, interfaced with the physical environment through mechanical parts. Time takes different forms when observed from each of these viewpoints: it is discrete and event-based in software, discrete and time-triggered in hardware, continuous in mechanics or physics. Design of CPS often benefits from concepts of multiform and logical time(s) for their natural description. High-level formalisms used to model software, hardware and physics additionally alter this perception of time quite significantly.

[1] *Matlab/Simulink*, https://fr.mathworks.com/products/simulink.html
[2] *Ptolemy*, http://ptolemy.eecs.berkeley.edu
[3] *SysML*, http://www.uml-sysml.org

In model-based system design, time is usually abstracted to serve the purpose of one of many design tasks: verification, simulation, profiling, performance analysis, scheduling analysis, parallelization, distribution, or virtual prototyping. For example in non-real-time commodity software, timing abstraction such as number of instructions and algorithmic complexity is sufficient: software will run the same on different machines, except slower or faster. Alternatively, in cyber-physical systems, multiple recurring instances of meaningful events may create as many dedicated logical clocks, on which to ground modeling and design practices.

Time abstraction increases efficiency in event-driven simulation or execution (i.e SystemC simulation models try to abstract time, from cycle-accurate to approximate-time, and to loosely-time), while attempting to retain functionality, but without any actual guarantee of valid accuracy (responsibility is left to the model designer). Functional determinism (a.k.a. conflict-freeness in Petri Nets, monotonicity in Kahn PNs, confluence in Milner's CCS, latency-insensitivity and elasticity in circuit design) allows for reducing to some amount the problem to that of many schedules of a single self-timed behavior, and time in many systems studies is partitioned into models of computation and communication (MoCCs). Multiple, multiform time(s) raises the question of combination, abstraction or refinement between distinct time bases. The question of combining continuous time with discrete logical time calls for proper discretization in simulation and implementation. While timed reasoning takes multiple forms, there is no unified foundation to reasoning about multi-form time in system design.

The objective of project-team TEA is henceforth to define formal models for timed quantitative reasoning, composition, and integration in embedded system design. Formal time models and calculi should allow us to revisit common domain problems in real-time system design, such as time predictability and determinism, memory resources predictability, real-time scheduling, mixed-criticality and power management; yet from the perspective gained from inter-domain timed and quantitative abstraction or refinement relations. A regained focus on fundamentals will allow to deliver better tooled methodologies for virtual prototyping and integration of embedded architectures.

3. Research Program

3.1. Previous Works

The challenges of team TEA support the claim that sound Cyber-Physical System design (including embedded, reactive, and concurrent systems altogether) should consider multi-form time models as a central aspect. In this aim, architectural specifications found in software engineering are a natural focal point to start from. Architecture descriptions organize a system model into manageable components, establish clear interfaces between them, collect domain-specific constraints and properties to help correct integration of components during system design. The definition of a formal design methodology to support heterogeneous or multi-form models of time in architecture descriptions demands the elaboration of sound mathematical foundations and the development of formal calculi and methods to instrument them. This constitutes the research program of team TEA.

System design based on the "synchronous paradigm" has focused the attention of many academic and industrial actors on abstracting non-functional implementation details from system design. This elegant design abstraction focuses on the logic of interaction in reactive programs rather than their timed behavior, allowing to secure functional correctness while remaining an intuitive programming model for embedded systems. Yet, it corresponds to embedded technologies of single cores and synchronous buses from the 90s, and may hardly cover the semantic diversity of distribution, parallelism, heterogeneity, of cyber-physical systems found in 21st century Internet-connected, true-time^{TM}-synchronized clouds, of tomorrow's grids.

By contrast with a synchronous hypothesis, yet from the same era, the polychronous MoCC is inherently capable of describing multi-clock abstractions of GALS systems. Polychrony is implemented in the data-flow specification language Signal, available in the Eclipse project POP [4] and in the CCSL standard [5] available from

[4] *Polychrony on Polarsys*, https://www.polarsys.org/projects/polarsys.pop
[5] *Clock Constraints in UML/MARTE CCSL*. C. André, F. Mallet. RR-6540. Inria, 2008. http://hal.inria.fr/inria-00280941

the TimeSquare project. Both provide tooled infrastructures to refine high-level specifications into real-time streaming applications or locally synchronous and globally asynchronous systems, through a series of model analysis, verification, and synthesis services. These tool-supported refinement and transformation techniques can assist the system engineer from the earliest design stages of requirement specification to the latest stages of synthesis, scheduling and deployment. These characteristics make polychrony much closer to the required semantic for compositional, refinement-based, architecture-driven, system design.

While polychrony was a step ahead of the traditional synchronous hypothesis, CCSL is a leap forward from synchrony and polychrony. The essence of CCSL is “multi-form time” toward addressing all of the domain-specific physical, electronic and logical aspects of cyber-physical system design.

3.2. Modeling Times

To make a sense and eventually formalize the semantics of time in system design, we should most certainly rely on algebraic representations of time found in previous works and introduce the paradigm of "time systems" (type systems to represent time) in a way reminiscent to CCSL. Just as a type system abstracts data carried along operations in a program, a time system abstracts the causal interaction of that program module or hardware element with its environment, its pre and post conditions, its assumptions and guarantees, either logical or numerical, discrete or continuous. Some fundamental concepts of the time systems we envision are present in the clock calculi found in data-flow synchronous languages like Signal or Lustre, yet bound to a particular model of concurrency, hence time.

In particular, the principle of refinement type systems [6], is to associate information (data-types) inferred from programs and models with properties pertaining, for instance, to the algebraic domain on their value, or any algebraic property related to its computation: effect, memory usage, pre-post condition, value-range, cost, speed, time, temporal logic [7]. Being grounded on type and domain theories, a time system should naturally be equipped with program analysis techniques based on type inference (for data-type inference) or abstract interpretation (for program properties inference) to help establish formal relations between heterogeneous component “types”. Just as a time calculus may formally abstract timed concurrent behaviors of system components, timed relations (abstraction and refinement) represent interaction among components.

Scalability and compositionality requires the use of assume-guarantee reasoning to represent them, and to facilitate composition by behavioral sub-typing, in the spirit of the (static) contract-based formalism proposed by Passerone et al. [8]. Verification problems encompassing heterogeneously timed specifications are common and of great variety: checking correctness between abstract and concrete time models relates to desynchronisation (from synchrony to asynchrony) and scheduling analysis (from synchrony to hardware). More generally, they can be perceived from heterogeneous timing viewpoints (e.g. mapping a synchronous-time software on a real-time middle-ware or hardware).

This perspective demands capabilities not only to inject time models one into the other (by abstract interpretation, using refinement calculi), to compare time abstractions one another (using simulation, refinement, bi-simulation, equivalence relations) but also to prove more specific properties (synchronization, determinism, endochrony). All this formalization effort will allow to effectively perform the tooled validation of common cross-domain properties (e.g. cost v.s. power v.s. performance v.s. software mapping) and tackle equally common yet tough case studies such as these linking battery capacity, to on-board CPU performance, to static software schedulability, to logical software correctness and plant controllability (by the choice of the correct sampling period across system components).

3.3. Modeling Architectures

To address the formalization of such cross-domain case studies, modeling the architecture formally plays an essential role. An architectural model represents components in a distributed system as boxes with well-defined

[6] *Abstract Refinement Types*. N. Vazou, P. Rondon, and R. Jhala. European Symposium on Programming. Springer, 2013.
[7] *LTL types FRP*. A. Jeffrey. Programming Languages meets Program Verification.
[8] *A contract-based formalism for the specification of heterogeneous systems*. L. Benvenistu, et al. FDL, 2008

interfaces, connections between ports on component interfaces, and specifies component properties that can be used in analytical reasoning about the model. Several architectural modeling languages for embedded systems have emerged in recent years, including the SAE AADL [9], SysML [10], UML MARTE [11].

In system design, an architectural specification serves several important purposes. First, it breaks down a system model into manageable components to establish clear interfaces between components. In this way, complexity becomes manageable by hiding details that are not relevant at a given level of abstraction. Clear, formally defined, component interfaces allow us to avoid integration problems at the implementation phase. Connections between components, which specify how components affect each other, help propagate the effects of a change in one component to the linked components.

Most importantly, an architectural model is a repository to share knowledge about the system being designed. This knowledge can be represented as requirements, design artifacts, component implementations, held together by a structural backbone. Such a repository enables automatic generation of analytical models for different aspects of the system, such as timing, reliability, security, performance, energy, etc. Since all the models are generated from the same source, the consistency of assumptions w.r.t. guarantees, of abstractions w.r.t. refinements, used for different analyses becomes easier, and can be properly ensured in a design methodology based on formal verification and synthesis methods.

Related works in this aim, and closer in spirit to our approach (to focus on modeling time) are domain-specific languages such as Prelude [12] to model the real-time characteristics of embedded software architectures. Conversely, standard architecture description languages could be based on algebraic modeling tools, such as interface theories with the ECDAR tool [13].

In project TEA, it takes form by the normalization of the AADL standard's formal semantics and the proposal of a time specification annex in the form of related standards, such as CCSL, to model concurrency time and physical properties, and PSL, to model timed traces.

3.4. Scheduling Theory

Based on sound formalization of time and CPS architectures, real-time scheduling theory provides tools for predicting the timing behavior of a CPS which consists of many interacting software and hardware components. Expressing parallelism among software components is a crucial aspect of the design process of a CPS. It allows for efficient partition and exploitation of available resources.

The literature about real-time scheduling [14] provides very mature schedulability tests regarding many scheduling strategies, preemptive or non-preemptive scheduling, uniprocessor or multiprocessor scheduling, etc. Scheduling of data-flow graphs has also been extensively studied in the past decades.

A milestone in this prospect is the development of abstract affine scheduling techniques [15]. It consists, first, of approximating task communication patterns (e.g. between Safety-Critical Java threads) using cyclo-static data-flow graphs and affine functions. Then, it uses state of the art ILP techniques to find optimal schedules and to concretize them as real-time schedules in the program implementations [16] [17].

Abstract scheduling, or the use of abstraction and refinement techniques in scheduling borrowed to the theory of abstract interpretation [18] is a promising development toward tooled methodologies to orchestrate thousands of heterogeneous hardware/software blocks on modern CPS architectures (just consider modern cars or aircrafts). It is an issue that simply defies the state of the art and known bounds of complexity theory in the field, and consequently requires a particular focus.

[9] *Architecture Analysis and Design Language*, AS-5506. SAE, 2004. http://standards.sae.org/as5506b
[10] *System modeling Language*. OMG, 2007. http://www.omg.org/spec/SysML
[11] *UML Profile for MARTE*. OMG, 2009. http://www.omg.org/spec/MARTE
[12] *The Prelude language*. LIFL and ONERA, 2012. http://www.lifl.fr/~forget/prelude.html
[13] *PyECDAR, timed games for timed specifications*. Inria, 2013. https://project.inria.fr/pyecdar
[14] *A survey of hard real-time scheduling for multiprocessor systems*. R. I. Davis and A. Burns. *ACM Computing Survey* 43(4), 2011.
[15] *Buffer minimization in EDF scheduling of data-flow graphs*. A. Bouakaz and J.-P. Talpin. LCTES, ACM, 2013.
[16] *ADFG for the synthesis of hard real-time applications*. A. Bouakaz, J.-P. Talpin, J. Vitek. ACSD, IEEE, June 2012.
[17] *Design of SCJ Level 1 Applications Using Affine Abstract Clocks*. A. Bouakaz and J.-P. Talpin. SCOPES, ACM, 2013.
[18] *La vérification de programmes par interprétation abstraite*. P. Cousot. Séminaire au Collège de France, 2008.

To develop the underlying theory of this promising research topic, we first need to deepen the theoretical foundation to establish links between scheduling analysis and abstract interpretation. A theory of time systems would offer the ideal framework to pursue this development. It amounts to representing scheduling constraints, inferred from programs, as types or contract properties. It allows to formalize the target time model of the scheduler (the architecture, its middle-ware, its real-time system) and defines the basic concepts to verify assumptions made in one with promises offered by the other: contract verification or, in this case, synthesis.

3.5. Verified programming for system design

The IoT is a network of devices that sense, actuate and change our immediate environment. Against this fundamental role of sensing and actuation, design of edge devices often treats action and event timing to be primarily software implementation issues: programming models for IoT abstract even the most rudimentary information regarding timing, sensing and the effects of actuation. As a result, applications programming interfaces for IoT allow wiring systems fast without any meaningful assertions about correctness, reliability or resilience.

We make the case that the "API glue" must give way to the logical closure of interface types. Interfaces can be governed by a calculus – a refinement type calculus – to enable reasoning on time, sensing and actuation, in a way that provides both deep specification refinement, for mechanized verification of requirements, and multi-layered abstraction, to support compositionality and scalability, from one end of the system to the other.

Our project seeks to elevate the “function as type” paradigm to that of “system as type”: to define a refinement type calculus for reasoning on networked devices and integrate them as cyber-physical systems [19]. Our companion paper [20] outlines our progress in this aim and plans towards building a verified programming environment for networked IoT devices: we propose a type-driven approach to verifying and building safe and secure IoT applications.

Accounting for such constrains in a more principled fashion demands reasoning about the composition of all the software and hardware components of the application. Our proposed framework takes a step in this direction by (1) using refinement types to make make physical constraints explicit and (2) imposing an event-driven programming discipline to simplify the reasoning of system-wide properties to that of an event queue. In taking this approach, our framework makes it possible for developers to build verified IoT application by making it a type error for code to violate physical constraints.

4. Application Domains

4.1. Automotive and Avionics

From our continuous collaboration with major academic and industrial partners through projects TOPCASED, OPENEMBEDD, SPACIFY, CESAR, OPEES, P and CORAIL, our experience has primarily focused on the aerospace domain. The topics of time and architecture of team TEA extend to both avionics and automotive. Yet, the research focuses on time in team TEA is central in any aspect of, cyber-physical, embedded system design in factory automation, automotive, music synthesis, signal processing, software radio, circuit and system on a chip design; many application domains which, should more collaborators join the team, would definitely be worth investigating.

Multi-scale, multi-aspect time modeling, analysis and software synthesis will greatly contribute to architecture modeling in these domains, with applications to optimized (distributed, parallel, multi-core) code generation for avionics (project Corail with Thales avionics, section 8) as well as modeling standards, real-time simulation and virtual integration in automotive (project with Toyota ITC, section 8).

[19] Refinement types for system design. Jean-Pierre Talpin. FDL’18 keynote.

[20] Steps toward verified programming of embedded computing systems. Jean-Pierre Talpin, Jean-Joseph Marty, Deian Stefan, Shravan Nagarayan, Rajesh Gupta, DATE’18.

Together with the importance of open-source software, one of these projects, the FUI Project P (section 8), demonstrated that a centralized model for system design could not just be a domain-specific programming language, such as discrete Simulink data-flows or a synchronous language. Synchronous languages implement a fixed model of time using logical clocks that are abstraction of time as sensed by software. They correspond to a fixed viewpoint in system design, and in a fixed hardware location in the system, which is not adequate to our purpose and must be extended.

In project P, we first tried to define a centralized model for importing discrete-continuous models onto a simplified implementation of SIMULINK: P models. Certified code generators would then be developed from that format. Because this does not encompass all aspects being translated to P, the P meta-model is now being extended to architecture description concepts (of the AADL) in order to become better suited for the purpose of system design. Another example is the development of System modeler on top of SCADE, which uses the more model-engineering flavored formalism SysML to try to unambiguously represent architectures around SCADE modules.

An abstract specification formalism, capable of representing time, timing relations, with which heterogeneous models can be abstracted, from which programs can be synthesized, naturally appears better suited for the purpose of virtual prototyping. RT-Builder, based on Signal like Polychrony and developed by TNI, was industrially proven and deployed for that purpose at Peugeot. It served to develop the virtual platform simulating all on-board electronics of PSA cars. This ‘hardware in the loop” simulator was used to test equipments supplied by other manufacturers with respect to virtual cars. In the advent of the related automotive standard, RT-Builder then became AUTOSAR-Builder.

4.2. Factory Automation

In collaboration with Mitsubishi R&D, we explore another application domain where time and domain heterogeneity are prime concerns: factory automation. In factory automation alone, a system is conventionally built from generic computing modules: PLCs (Programmable Logic Controllers), connected to the environment with actuators and detectors, and linked to a distributed network. Each individual, physically distributed, PLC module must be timely programmed to perform individually coherent actions and fulfill the global physical, chemical, safety, power efficiency, performance and latency requirements of the whole production chain. Factory chains are subject to global and heterogeneous (physical, electronic, functional) requirements whose enforcement must be orchestrated for all individual components.

Model-based analysis in factory automation emerges from different scientific domains and focus on different CPS abstractions that interact in subtle ways: logic of PLC programs, real-time electromechanical processing, physical and chemical environments. This yields domain communication problems that render individual domain analysis useless. For instance, if one domain analysis (e.g. software) modifies a system model in a way that violates assumptions made by another domain (e.g. chemistry) then the detection of its violation may well be impossible to explain to either the software or chemistry experts. As a consequence, cross-domain analysis issues are discovered very late during system integration and lead to costly fixes. This is particularly prevalent in multi-tier industries, such as avionic, automotive, factories, where systems are prominently integrated from independently-developed parts.

5. Highlights of the Year

5.1. Highlights of the Year

Inria created a new International Chair in collaboration with Insa-Rennes and appointed American computer engineer Shuvra Bhattacharyya, Professor at the University of Maryland, to the part-time position. Shuvra will hold the International Chair for a period of five years, fostering our joint collaboration with the CNRS-Insa laboratory IETR.

TEA becomes the first Inria group to host two International Chairs: last year, Rajesh Gupta, Director of UC San Diego Data Science Institute, was appointed Inria International Chair with project-team TEA.

Jean-Pierre Talpin spent the first semester to prepare an ERC advanced grant. He also gave a keynote presentation at the FDL'18 conference on "refinement types for system design".

Thierry Gautier and Albert Benveniste gave an invited seminar at the Collège de France in Gérard Berry's 2017-2018 lecture course [15].

6. New Software and Platforms

6.1. ADFG

Affine data-flow graphs schedule synthesizer
KEYWORDS: Code generation - Scheduling - Static program analysis
FUNCTIONAL DESCRIPTION: ADFG is a synthesis tool of real-time system scheduling parameters: ADFG computes task periods and buffer sizes of systems resulting in a trade-off between throughput maximization and buffer size minimization. ADFG synthesizes systems modeled by ultimately cyclo-static dataflow (UCSDF) graphs, an extension of the standard CSDF model.

Knowing the WCET (Worst Case Execute Time) of the actors and their exchanges on the channels, ADFG tries to synthezise the scheduler of the application. ADFG offers several scheduling policies and can detect unschedulable systems. It ensures that the real scheduling does not cause overflows or underflows and tries to maximize the throughput (the processors utilization) while minimizing the storage space needed between the actors (i.e. the buffer sizes).

Abstract affine scheduling is first applied on the dataflow graph, that consists only of periodic actors, to compute timeless scheduling constraints (e.g. relation between the speeds of two actors) and buffering parameters. Then, symbolic schedulability policies analysis (i.e., synthesis of timing and scheduling parameters of actors) is applied to produce the scheduller for the actors.

ADFG, initially defined to synthesize real-time schedulers for SCJ/L1 applications, may be used for scheduling analysis of AADL programs.

- Authors: Thierry Gautier, Jean-Pierre Talpin, Adnan Bouakaz, Alexandre Honorat and Loïc Besnard
- Contact: Loïc Besnard

6.2. POLYCHRONY

KEYWORDS: Code generation - AADL - Proof - Optimization - Multi-clock - GALS - Architecture - Cosimulation - Real time - Synchronous Language
FUNCTIONAL DESCRIPTION: Polychrony is an Open Source development environment for critical/embedded systems. It is based on Signal, a real-time polychronous data-flow language. It provides a unified model-driven environment to perform design exploration by using top-down and bottom-up design methodologies formally supported by design model transformations from specification to implementation and from synchrony to asynchrony. It can be included in heterogeneous design systems with various input formalisms and output languages. The Polychrony tool-set provides a formal framework to: validate a design at different levels, by the way of formal verification and/or simulation, refine descriptions in a top-down approach, abstract properties needed for black-box composition, compose heterogeneous components (bottom-up with COTS), generate executable code for various architectures. The Polychrony tool-set contains three main components and an experimental interface to GNU Compiler Collection (GCC):

* The Signal toolbox, a batch compiler for the Signal language, and a structured API that provides a set of program transformations. Itcan be installed without other components and is distributed under GPL V2 license.

* The Signal GUI, a Graphical User Interface to the Signal toolbox (editor + interactive access to compiling functionalities). It can be used either as a specific tool or as a graphical view under Eclipse. It has been transformed and restructured, in order to get a more up-to-date interface allowing multi-window manipulation of programs. It is distributed under GPL V2 license.

* The POP Eclipse platform, a front-end to the Signal toolbox in the Eclipse environment. It is distributed under EPL license.

- Participants: Loïc Besnard, Paul Le Guernic and Thierry Gautier
- Partners: CNRS - Inria
- Contact: Loïc Besnard
- URL: https://www.polarsys.org/projects/polarsys.pop

6.3. Polychrony AADL2SIGNAL

KEYWORDS: Real-time application - Polychrone - Synchronous model - Polarsys - Polychrony - Signal - AADL - Eclipse - Meta model
FUNCTIONAL DESCRIPTION: This polychronous MoC has been used previously as semantic model for systems described in the core AADL standard. The core AADL is extended with annexes, such as the Behavior Annex, which allows to specify more precisely architectural behaviors. The translation from AADL specifications into the polychronous model should take into account these behavior specifications, which are based on description of automata.

For that purpose, the AADL state transition systems are translated as Signal automata (a slight extension of the Signal language has been defined to support the model of polychronous automata).

Once the AADL model of a system transformed into a Signal program, one can analyze the program using the Polychrony framework in order to check if timing, scheduling and logical requirements over the whole system are met.

We have implemented the translation and experimented it using a concrete case study, which is the AADL modeling of an Adaptive Cruise Control (ACC) system, a highly safety-critical system embedded in recent cars.

- Participants: Huafeng Yu, Loïc Besnard, Paul Le Guernic, Thierry Gautier and Yue Ma
- Partner: CNRS
- Contact: Loïc Besnard
- URL: http://www.inria.fr/equipes/tea

6.4. POP

Polychrony on Polarsys
KEYWORDS: Synchronous model - Model-driven engineering
FUNCTIONAL DESCRIPTION: The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. a major achievement of the ESPRESSO (and now TEA) project-team. The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. It was finalised in the frame of project OPEES, as a case study: by passing the POLARSYS qualification kit as a computer aided simulation and verification tool. This qualification was implemented by CS Toulouse in conformance with relevant generic (platform independent) qualification documents. Polychrony is now distributed by the Eclipse project POP on the platform of the POLARSYS industrial working group. Team TEA aims at continuing its dissemination to academic partners, as to its principles and features, and industrial partners, as to the services it can offer.

Project POP is composed of the Polychrony tool set, under GPL license, and its Eclipse framework, under EPL license. SSME (Syntactic Signal-Meta under Eclipse), is the meta-model of the Signal language implemented with Eclipse/Ecore. It describes all syntactic elements specified in Signal Reference Manual [21]: all Signal

operators (e.g. arithmetic, clock synchronization), model (e.g. process frame, module), and construction (e.g. iteration, type declaration). The meta-model primarily aims at making the language and services of the Polychrony environment available to inter-operation and composition with other components (e.g. AADL, Simulink, GeneAuto, P) within an Eclipse-based development tool-chain. Polychrony now comprises the capability to directly import and export Ecore models instead of textual Signal programs, in order to facilitate interaction between components within such a tool-chain. The download site for project POP has opened in 2015 at https://www.polarsys.org/projects/polarsys.pop. It should be noted that the Eclipse Foundation does not host code under GPL license. So, the Signal toolbox useful to compile Signal code from Eclipse is hosted on our web server.

- Participants: Jean-Pierre Talpin, Loïc Besnard, Paul Le Guernic and Thierry Gautier
- Contact: Loïc Besnard
- URL: https://www.polarsys.org/projects/polarsys.pop

6.5. Sigali

FUNCTIONAL DESCRIPTION: Sigali is a model-checking tool that operates on ILTS (Implicit Labeled Transition Systems, an equational representation of an automaton), an intermediate model for discrete event systems. It offers functionalities for verification of reactive systems and discrete controller synthesis. The techniques used consist in manipulating the system of equations instead of the set of solutions, which avoids the enumeration of the state space. Each set of states is uniquely characterized by a predicate and the operations on sets can be equivalently performed on the associated predicates. Therefore, a wide spectrum of properties, such as liveness, invariance, reachability and attractivity, can be checked. Algorithms for the computation of predicates on states are also available. Sigali is connected with the Polychrony environment (Tea project-team) as well as the Matou environment (VERIMAG), thus allowing the modeling of reactive systems by means of Signal Specification or Mode Automata and the visualization of the synthesized controller by an interactive simulation of the controlled system.

- Contact: Hervé Marchand

7. New Results

7.1. ADFG: Affine data-flow graphs scheduler synthesis

Participants: Loïc Besnard, Thierry Gautier, Alexandre Honorat, Jean-Pierre Talpin, Hai Nam Tran.

We consider with ADFG (Affine DataFlow Graph) the synthesis of scheduling parameters for real-time systems modeled as synchronous data flow (SDF), cyclo-static dataflow (CSDF), and ultimately cyclo-static dataflow (UCSDF) graphs. This synthesis aims for a trade-off between throughput maximization and total buffer size minimization. The synthesizer inputs are a graph which describes tasks by their Worst Case Execution Time (WCET), and directed buffers connecting tasks by their data production and consumption rates; the number of processors in the target system and the real-time scheduling synthesis algorithm to be used. The outputs are synthesized scheduling parameters such as tasks periods, offsets, processor bindings, priorities, buffer initial markings and buffer sizes. In this section, we present new results on two aspects: (1) the improvement of ADFG's usability and tool interoperability, (2) the integration of new scheduling analysis and scheduler synthesis algorithms.

21

SIGNAL V4-Inria version: Reference Manual. Besnard, L., Gautier, T. and Le Guernic, P. http://www.irisa.fr/espresso/Polychrony, 2010

ADFG was originally the implementation of Adnan Bouakaz's work [22]. However, the tool had not been packaged yet to be easily installed and used. Moreover, code refactoring led to improve the theory and to add new features. Firstly, more accurate bounds and Integer Linear Programming (ILP) formulations have been used. Besides, dataflow graphs do not need to be weakly connected for EDF policy on multiprocessor systems. The new implementation also avoids to use a fixed parameter for some multiprocessor partitioning algorithms, now an optional strategy enables to compute it. Finally, implementation has been adapted to standard technologies to be more easily installed and used. As the synthesizer evolved a lot, new evaluations have been made. Moreover, many scheduled examples have been simulated with Cheddar [23], which provides relevant metrics to analyze the scheduling efficiency.

Actor models and scheduling algorithms in ADFG are extended to investigate the contention-aware scheduling problem on multi/many-core architectures. The problem we tackled is that the scheduler synthesis for these platforms must account for the non-negligible delay due to shared memory accesses. We exploited the deterministic communications exposed in SDF graphs to account for the contention and further optimize the synthesized schedule. Two solutions are proposed and implemented in ADFG: contention-aware and contention-free scheduling synthesis. In other words, we either take into account the contention and synthesize a contention-aware schedule or find a one that results in no contention.

ADFG is extended to apply a transformation known as partial expansion graphs (PEG). This transformation can be applied as a pre-processing stage to improve the exploitation of data parallelism in SDF graphs on parallel platforms. In contrast to the classical approaches of transforming SDF graphs into equivalent homogeneous forms, which could lead to an exponential increase in the number of actors and excessive communication overhead, PEG-based approaches allow the designer to control the degree to which each actor is expanded. A PEG algorithm that employs cyclo-static data flow techniques is developed in ADFG. Compared to exist PEG-based approach, our solution requires neither buffer managers nor split-join actors to coordinate data production and consumption rates. This allows us to reduce the number of added actors and communication overhead in the expanded graphs.

7.2. Hardware synthesis in Polychrony

Participants: Loïc Besnard, Hafiz Muhamad Amjad.

In the context of the Convex associate-project with the Chinese Academy of Science, we have this year developped code generators (VHDL, Verilog) for modeling hardware in Signal language [24]. The first scheme of the translation had been proposed by Mohammed Belhadj PhD Thesis. Independent on the HDL used, VHDL or Verilog, the translation of Signal to a HDL is quite simple, considering only the functional (executable) Signal programs and a behavioral translation. Indeed, behavioral translation is quite similar to a sequential code generator. In this case, the Signal compiler generates the clock of each SIGNAL signal and orders the execution of the equations. This control structure can be easily translated in the HDL. The generated code may contain conditionals, loops and signal assignements in a HDL process.

7.3. Modular verification of cyber-physical systems using contract theory

Participants: Jean-Pierre Talpin, Benoit Boyer, David Mentre, Simon Lunel.

The primary goal of our project, in collaboration with Mitsubishi Electronics Research Centre Europe (MERCE), is to ensure correctness-by-design in realistic cyber-physical systems, i.e., systems that mix software and hardware in a physical environment, e.g., Mitsubishi factory automation lines or water-plant factory. To achieve that, we develop a verification methodology based on decomposition into components enhanced with contract reasoning.

[22] Real-Time Scheduling of Dataflow Graphs. A. Bouakaz. Ph.D. Thesis, University of Rennes 1, 2013.
[23] The Cheddar project: a GPL real-time scheduling analyzer: http://beru.univ-brest.fr/~singhoff/cheddar/
[24] Verilog Code Generation Scheme from Signal Language. Hafiz Muhammad, Amjad and Jianwei, Niu and Kai, Hu and Naveed, Akram and Loïc, Besnard. International Bhurban Conference on Applied Sciences and Technology (IBCAST). IEEE, 2019

The work of A. Platzer on Differential Dynamic Logic ($d\mathcal{L}$) held our attention [25]. This formalism is built upon the Dynamic Logic of V. Pratt and augmented with the possibility of expressing Ordinary Differential Equations (ODEs). Combined with the ability of Dynamic Logic to specify and verify hybrid programs, $d\mathcal{L}$ is a particularly fit model cyber-physical systems. The proof system associated with the logic is implemented into the theorem prover KeYmaera X. Aimed toward automation, it is a promising tool to spread formal methods into industry.

We have defined a syntactic parallel composition operator in $d\mathcal{L}$ which enjoys associativity and commutativity[6]. Commutativity provides compositionality: the possibility to compose and prove components and modules in every possible order. Associativity is mandatory to modularly design a system; it allows to construct step-by-step a system by adding new components. We have proved a theorem to automatically build contracts from the composition of components. We have exemplified our results with a cruise-controller example.

This contribution to $d\mathcal{L}$ defines a component-based approach to modularly model and prove cyber-physical systems. We have developed a working prototype in the interactive theorem prover KeYmaera X to show the feasibility of an implementation of our approach. To validate our methodology, we have case studied the example of a water-recycling plant, which rose several challenges.

The timing aspects in cyber-physical systems are a key aspect. A monitor regulating a plant, for example the water-level in a tank, must execute sufficiently often to ensure the correct behavior, for example that the water-level does not overflow. We have adapted our approach to automatically handle the compliance of execution time of monitor with the controllability of plant. We retain commutativity and associativity, thus the modularity of our approach. More importantly, we are still able to automatically build contracts from the composition of components.

We have also adapted our component-based approach to handle modes, a frequent construct in cyber-physical systems. It is also frequent to have to model causal composition between two components, for example that the sensor must execute before the monitor using the data of the sensor. Once again, we have adapted our component-based approach to take into account such design choice. It is important to emphasize that all these adaptations remain within our framework and are thus compatible.

To conclude, we have presented a methodology to tackle complexity of modeling and verification of cyber-physical systems by breaking a systems into smaller parts, the components. We have showed that it is easily adaptable to take into account new challenges. A future work would be to blend our component-based approach with refinement reasoning.

7.4. Verified information flow of embedded programs

Participants: Jean-Joseph Marty, Jean-Pierre Talpin, Shravan Narayan, Deian Stefan, Rajesh Gupta.

This PhD project is about applying refinement types theory to verified programming of applications and modules of library operting systems, such as unikernels, for embedded devices (the Internet of Things (IoT). We focus on developping a model of information flow control approach using labelled input-outputs (LIO).

We are collaborating with the ProgSys group at UC San Diego in the frame of Inria associate-team Composite, which develops the LIO framework. The LIO framework allows to avoid the "label creep" problem and supports the modeling of concurrency.

Currentlymost of the properties implemented in LIO rely on Haskell properties which is not friendly for embedded devices (IoT), as Haskell requires a huge run-time compared to low resources micro-controllers with less than 32KB of memory.

Instead, we actively use the new Microsoft's verified programming language F*. This programming language is a proof assistant like language that allows us to formalize, verify (using SMT solver and tactics) and extract to clean C (without system dependency) . We succeeded in making proved programs on Arduino compatible micro-controller. Our aim is to develop a version of LIO that could be verified and then extracted to C for targeting operating systems or IoT.

[25] *Differential Dynamic Logic for Hybrid Systems*, André Platzer, http://symbolaris.com/logic/dL.html

At present, F* is a mix of three domain specific languages: Meta* for proof automation, Low* for system level code including memory safety and F* that glues everything. We successfully implemented a simple Low*-only LIO library allowing to use labeled values. We are now working on a formalized version that will ensure that an F* program is safe w.r.t. information flow, before code generation.

In parallel we continue to work with the ProgSys team on a second project: code-named Gluco*. The goal of this project is to strengthen the F* programming knowledge and to make a example of a safety-critical application where F* can be used [26].

7.5. Modeling cyber-physical systems: from Signal to Signal+

Participants: Thierry Gautier, Albert Benveniste.

Based, initially, on two small case studies, we started a reflection on the modeling and analysis of cyber-physical systems by extending the model of synchronous languages, and in particular that of the Signal language [15]. The principle considered here is to remain within the traditional framework, in discrete time, of synchronous languages, but to completely generalize the equational style of specifications. In the usual Signal language, clocks are defined in a relational way by constraint systems. In contrast, data are defined using functionally inspired dataflow expressions. In this new Signal+, we propose to generalize the equational style to the data: numerical quantities also become subject to constraints. This typically corresponds to the way of modeling a system that includes physical components: for example, equations respecting balance laws have to be written. A major interest is that this programming style is much more compositional than more traditional Simulink or Lustre, or even Signal-based programming. The question then arises as to whether such a program should be analyzed. There is no notion of syntactic dependence, but the incidence graph can be used to define a matching that uniquely associates an equation with each variable. A scheduling can then be synthesized, provided that the reasoning is valid, which is the case for some classes of numerical algebraic equations, for which a solver can be used. However, we can observe on our case studies that the transition to Signal+ class is a real step forward in terms of difficulty. In discrete time, at each reaction, the free variables of the system must be evaluated using what is known about states and inputs. But in some cases, there will be more variables than usable equations. By reasoning on the fact that we have systems that are invariant in time, it may then be necessary to "shift" equations, just as if we were in continuous time, we could differentiate a constraint (a program is a discrete approximation of a continuous time system). This amounts, in a way, to proposing some modifications, which are considered as legitimate, to the source program.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. Inria – Mitsubishi Electric framework program (2018+)

Title:

Inria principal investigator: Jean-Pierre Talpin, Simon Lunel

International Partner: Mitsubishi Electric R&D Europe (MERCE)

Duration: 2018

Abstract: Following up the fruitfull collaboration of TEA with the formal methods group at MERCE, Inria and Mitsubishi Electric signed a center-wide collaboration agreement, which currently hosts projects with project-teams Sumo and Tea.

8.1.2. Mitsubishi Electric R&D Europe (2015-2018)

Title: Analysis and verification for correct by construction orchestration in automated factories

[26] Towards verified programming of embedded devices. J.-P. Talpin, J.-J. Marty, S. Narayan, D. Stefan, R. Gupta. Design, Automation and Test in Europe (DATE'19). IEEE, to appear 2018.

Inria principal investigator: Jean-Pierre Talpin, Simon Lunel

International Partner: Mitsubishi Electric R&D Europe

Duration: 2015 - 2018

Abstract: The primary goal of our project is to ensure correctness-by-design in cyber-physical systems, i.e., systems that mix software and hardware in a physical environment, e.g., Mitsubishi factory automation lines. We develop a component-based approach in Differential Dynamic Logic allowing to reason about a wide variety of heterogeneous cyber-physical systems. Our work provides tools and methodology to design and prove a system modularly.

9. Partnerships and Cooperations

9.1. International Initiatives

9.1.1. Inria International Labs

Sino-European Laboratory in Computer Science, Automation and Applied Mathematics
Associate Team involved in the International Lab:

9.1.1.1. CONVEX

Title: Compositional Verification of Cyber-Physical Systems

International Partner (Institution - Laboratory - Researcher):

CAS (China) - State Key Laboratory of Computer Science - Naijun Zhan

Start year: 2018

See also: http://www.irisa.fr/prive/talpin/convex

Formal modeling and verification methods have successfully improved software safety and security in vast application domains in transportation, production and energy. However, formal methods are labor-intensive and require highly trained software developers. Challenges facing formal methods stem from rapid evolution of hardware platforms, the increasing amount and cost of software infrastructures, and from the interaction between software, hardware and physics in networked cyber-physical systems.

Automation and expressivity of formal verification tools must be improved not only to scale functional verification to very large software stacks, but also verify non-functional properties from models of hardware (time, energy) and physics (domain). Abstraction, compositionality and refinement are essential properties to provide the necessary scalability to tackle the complexity of system design with methods able to scale heterogeneous, concurrent, networked, timed, discrete and continuous models of cyber-physical systems.

Project CONVEX wants to define a CPS architecture design methodology that takes advantage of existing time and concurrency modeling standards (MARTE, AADL, Ptolemy, Matlab), yet focuses on interfacing heterogeneous and exogenous models using simple, mathematically-defined structures, to achieve the single goal of correctly integrating CPS components.

Inria@SiliconValley
Associate Team involved in the International Lab:

9.1.1.2. Composite

Title: Compositional System Integration

International Partners (Institution - Laboratory - Researcher):

University of California, San Diego (United States) - Microelectronic Embedded Systems Laboratory - Rajesh Gupta

Start year: 2017

See also: http://www.irisa.fr/prive/talpin/composite

Most applications that run somewhere on the internet are not optimized to do so. They execute on general purpose operating systems or on containers (virtual machines) that are built with the most conservative assumptions about their environment. While an application is specific, a large part of the system it runs on is unused, which is both a cost (to store and execute) and a security risk (many entry points).

A unikernel, on the contrary, is a system program object that only contains the necessary the operating system services it needs for execution. A unikernel is build from the composition of a program, developed using high-level programming language, with modules of a library operating system (libOS), to execute directly on an hypervisor. A unikernel can boot in milliseconds to serve a request and shut down, demanding minimal energy and resources, offering stealthiest exposure time and surface to attacks, making them the ideal platforms to deploy on sensor networks, networks of embedded devices, smart grids and clouds.

The goal of COMPOSITE is to develop the mathematical foundations for sound and efficient composition in system programming: analysis, verification and optimization technique for modular and compositional hardware-system-software integration of unikernels. We intend to further this development with the prospect of an end-to-end co-design methodology to synthesize lean and stealth networked embedded devices.

9.2. International Research Visitors

9.2.1. Visits of International Scientists

Deian Stefan, Shravan Narayan (CSD) visited TEA in September for a week. Jonathan Protzenko (MSR Redmond) joined the meeting for a couple of days and gave a seminar at http://68nqrt.irisa.fr. Rajesh Gupta visited TEA for a month in August-September.

Lingtai Wang, ISCAS, visited TEA for a week in November.

9.2.2. Visits to International Teams

Jean-Joseph Marty visited UC San Diego in June for two weeks.

Jean-Pierre Talpin visited ISCAS, BUAA and Nankai in April, June and October, for a total period of two months, thanks to funding provided by the Chinese partners in the context of associate-project Convex.

Simon Lunel visited ISCAS for a week in December.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Organisation

10.1.1.1. Member of the Organizing Committees

Jean-Pierre Talpin served as Finance Chair and Local Organizer of the 16th J16th ACM-IEEE conference on methods and models for system design in Beijing, China (http://memocode.irisa.fr/2018) and its 1st embedded workshop on formal methods in China's industry.

10.1.2. Scientific Events Selection

10.1.2.1. Member of the Conference Program Committees

Jean-Pierre Talpin served the program committee of:

- LCTES'18, 21th. ACM SIGPLAN-SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems
- SAC'19, 34rd. ACM SIGAPP Symposium on Applied Computing
- SCOPES'18, 21th. International Workshop on Software and Compilers for Embedded Systems

10.1.2.2. Reviewer

Thierry Gautier reviewed articles for *IEEE Access* and *Science of Computer Programming*.

10.1.3. Journal

10.1.3.1. Member of the Editorial Boards

Jean-Pierre Talpin is Associate Editor with the ACM Transactions for Embedded Computing Systems (TECS).

10.1.3.2. Reviewer - Reviewing Activities

Thierry Gautier reviewed articles for *IEEE Access* and *Science of Computer Programming*.

10.1.4. Invited Talks

Jean-Pierre Talpin gave a keynote entitled "Refinement types for system design" at the Symposium on Dependable Software Engineering (SETTA'17) in Changsha, October 25.

Thierry Gautier and Albert Benveniste gave a seminar on "The Signal synchronous language: the principles beyond the language and how to exploit and extend them", in March 2018 at the Collège de France, in Gérard Berry's 2017-2018 lecture course (https://www.college-de-france.fr/site/gerard-berry/seminar-2018-03-07-17h30.htm).

10.1.5. Expertise

Jean-Pierre Talpin served as vice-president of CES 25 (Computer Science Evaluation Committee) of ANR.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Jean-Pierre Talpin gave a seminar at Beihang Science Park on June 11th , at ISCAS on May 2nd and on June 14th, and at Beida (PKU) on December 7th. Jean-Pierre Talpin gave an introductory course on model-checking at Nankai University, May 7-8th.

10.2.2. Supervision

Jean-Pierre Talpin co-supervises the PhD Theses of Simon Lunel, Liangcong Zhang, Jean-Joseph Marty and Lucas Franceschino

10.2.3. Juries

Jean-Pierre Talpin served as reviewer at the HDR Thesis defense of Katell Morin, entitled "Assertions and hardware design", which took place at TIMA, Grenoble, on November 15.

Jean-Pierre Talpin served as reviewer at the PhD Thesis defense of Guillaume Plassan, entitled "Conclusive formal verification of clock domain crossing properties", which took place at TIMA, Grenoble, on March 28.

11. Bibliography

Major publications by the team in recent years

[1] L. BESNARD, T. GAUTIER, P. LE GUERNIC, C. GUY, J.-P. TALPIN, B. LARSON, E. BORDE. *Formal Semantics of Behavior Specifications in the Architecture Analysis and Design Language Standard*, in "Cyber-Physical System Design from an Architecture Analysis Viewpoint", Cyber-Physical System Design from an Architecture Analysis Viewpoint, Springer, January 2017 [*DOI :* 10.1007/978-981-10-4436-6_3], https://hal.inria.fr/hal-01615143

[2] L. BESNARD, T. GAUTIER, P. LE GUERNIC, J.-P. TALPIN. *Compilation of Polychronous Data Flow Equations*, in "Synthesis of Embedded Software", S. K. SHUKLA, J.-P. TALPIN (editors), Springer, 2010, pp. 1-40 [*DOI :* 10.1007/978-1-4419-6400-7_1], http://hal.inria.fr/inria-00540493

[3] A. BOUAKAZ, J.-P. TALPIN. *Design of Safety-Critical Java Level 1 Applications Using Affine Abstract Clocks*, in "International Workshop on Software and Compilers for Embedded Systems", St. Goar, Germany, June 2013, pp. 58-67 [*DOI :* 10.1145/2463596.2463600], https://hal.inria.fr/hal-00916487

[4] A. GAMATIÉ, T. GAUTIER, P. LE GUERNIC. *Synchronous design of avionic applications based on model refinements*, in "Journal of Embedded Computing (IOS Press)", 2006, vol. 2, n^{o} 3-4, pp. 273-289, http://hal.archives-ouvertes.fr/hal-00541523

[5] A. HONORAT, H. N. TRAN, L. BESNARD, T. GAUTIER, J.-P. TALPIN, A. BOUAKAZ. *ADFG: a scheduling synthesis tool for dataflow graphs in real-time systems*, in "International Conference on Real-Time Networks and Systems", Grenoble, France, October 2017, pp. 1-10 [*DOI :* 10.1145/3139258.3139267], https://hal.inria.fr/hal-01615142

[6] S. LUNEL, B. BOYER, J.-P. TALPIN. *Compositional proofs in differential dynamic logic dL*, in "17th International Conference on Application of Concurrency to System Design", Zaragoza, Spain, June 2017, https://hal.inria.fr/hal-01615140

[7] S. NAKAJIMA, J.-P. TALPIN, M. TOYOSHIMA, H. YU. *Cyber-Physical System Design from an Architecture Analysis Viewpoint*, Communications of NII Shonan Meetings, Springer, January 2017 [*DOI :* 10.1007/978-981-10-4436-6], https://hal.inria.fr/hal-01615144

[8] D. POTOP-BUTUCARU, Y. SOREL, R. DE SIMONE, J.-P. TALPIN. *From Concurrent Multi-clock Programs to Deterministic Asynchronous Implementations*, in "Fundamenta Informaticae", January 2011, vol. 108, n^{o} 1-2, pp. 91–118, http://dl.acm.org/citation.cfm?id=2362088.2362094

[9] O. SANKUR, J.-P. TALPIN. *An Abstraction Technique For Parameterized Model Checking of Leader Election Protocols: Application to FTSP*, in "23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)", Uppsala, Sweden, Lecture Notes in Computer Science, April 2017, vol. 10206, https://hal.archives-ouvertes.fr/hal-01431472

[10] H. YU, J. PRASHI, J.-P. TALPIN, S. K. SHUKLA, S. SHIRAISHI. *Model-Based Integration for Automotive Control Software*, in "Digital Automation Conference", San Francisco, United States, ACM, June 2015, https://hal.inria.fr/hal-01148905

Publications of the year

Articles in International Peer-Reviewed Journals

[11] P. DERLER, K. SCHNEIDER, J.-P. TALPIN. *Guest Editorial: Special Issue of ACM TECS on the ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE 2017)*, in "ACM Transactions on Embedded Computing Systems (TECS)", November 2018, pp. 1-2, https://hal.inria.fr/hal-01926923

[12] T. GAUTIER, C. GUY, A. HONORAT, P. LE GUERNIC, J.-P. TALPIN, L. BESNARD. *Polychronous Automata and their Use for Formal Validation of AADL Models*, in "Frontiers of Computer Science", 2018 [*DOI :* 10.1007/s11704-017-6134-5], https://hal.inria.fr/hal-01411257

Invited Conferences

[13] J.-P. TALPIN. *Refinement types for system design (abstract)*, in "Forum on specification and Design Languages", Munich, Germany, September 2018, https://hal.inria.fr/hal-01926978

International Conferences with Proceedings

[14] H. N. TRAN, S. S. BHATTACHARYYA, J.-P. TALPIN, T. GAUTIER. *Toward Efficient Many-core Scheduling of Partial Expansion Graphs*, in "SCOPES 2018 - 21st International Workshop on Software and Compilers for Embedded Systems", Saint Goar, Germany, May 2018, vol. 4, pp. 1-4 [*DOI :* 10.1145/3207719.3207734], https://hal.inria.fr/hal-01926955

Other Publications

[15] A. BENVENISTE, T. GAUTIER. *The Signal synchronous language: the principles beyond the language and how to exploit and extend them*, March 2018, pp. 1-68, Lecture, https://hal.archives-ouvertes.fr/hal-01929567