RESEARCH CENTRE

Inria Centre at Rennes University

2023
ACTIVITY REPORT

Team

TEA

Time, Events and Architectures

Inria teams are typically groups of researchers working on the definition of a common project, and objectives, with the goal to arrive at the creation of a project-team. Such project-teams may include other partners (universities or research institutions)

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

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Embedded and Real-time Systems

Contents

Team TEA

Creation of the Team: 2023 December 31

Keywords

Computer sciences and digital sciences

- 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 members, visitors, external collaborators

Research Scientist

- Jean-Pierre Talpin [Team leader, INRIA, Senior Researcher, HDR]

Post-Doctoral Fellow

- Benjamin Lion [INRIA]

PhD Students

- Stéphane Kastenbaum [Mitsubishi Electric, until Jan 2023, PhD student]
- Shenghao Yuan [INRIA]

Administrative Assistant

- Armelle Mozziconacci [CNRS]

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. Modeling and programming formalisms that represent software, hardware and physics significantly alter this perception of time. Therefore, 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 functionalities, thanks 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, such as for instance Simulink and Matlab. 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. CPS design is, to date, mostly executed in this ad-hoc manner, without sound, mathematically grounded, integrative methodology. A new science of CPS design will allow to create machines with complex dynamics and high control reliability, and apply to new industries and applications, such as IoT or edge devices, 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 a 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.

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 (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.

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 system 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 reason 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.

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-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 (Polychrony on Polarsys) and in the CCSL standard Clock Constraints in CCSL available from 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 Timed Modeling

To formalize timed semantics for system design, we shall rely on algebraic representations of time as clocks found in previous works and introduce a paradigm of "time system": refinement types that represent timed behaviors. 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 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 timed concurrency.

In particular, the principle of refinement type systems [1], 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 [2]. Being grounded on type and domain theories,

[1] *Abstract Refinement Types.* N. Vazou, P. Rondon, and R. Jhala. European Symposium on Programming. Springer, 2013.

[2] *LTL types FRP.* A. Jeffrey. Programming Languages meets Program Verification.

such type systems 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".

Gaining scalability requires the capacity to modularly decompose systems which can be obtained using Abadi and Lamport's "*Composing Specifications*" and implemented by the notion of assume-guarantee contracts or Dijkstra monads. Verification problems encompassing heterogeneously timed specifications are common and of great variety: checking correctness between abstract (e.g. the synchronous hypothesis) and concrete time models (e.g. real-time architectures) relates to desynchronisation (from synchrony to asynchrony) and scheduling analysis (from synchronous data-flow to hardware). More generally, they can be perceived from heterogeneous timing viewpoints (e.g. mapping a synchronous-time software on a real-time middleware or hardware).

This perspective demands capabilities to use abstraction and refinement mechanisms for time models (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 problems such as these integrating constraints of battery capacity, on-board CPU performance, available memory resources, software schedulability, to logical software correctness and plant controllability.

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 interfaces, connections between ports on component interfaces, and specifies component properties that can be used in analytical reasoning about the model.

In system design, an architectural specification serves several important purposes. First, it breaks down a system model into components of manageable size and complexity, 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, and semantically rich component interfaces facilitate integration by allowing most validation efforts to be conducted modularly. Connections between components, which specify how components interact with each other, help propagate the guaranteed effects of a component to the assumptions of 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.

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 [3] 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 [4]. It consists, first, of approximating task communication patterns (e.g. between Safety-Critical Java threads) using

[3] *A survey of hard real-time scheduling for multiprocessor systems.* R. I. Davis and A. Burns. *ACM Computing Survey* 43(4), 2011.
[4] *Buffer minimization in EDF scheduling of data-flow graphs.* A. Bouakaz and J.-P. Talpin. LCTES, ACM, 2013.

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 [5] [6].

Abstract scheduling, or the use of abstraction and refinement techniques in scheduling borrowed to the theory of abstract interpretation [7] 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.

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 considers actions and event timings 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 (API) 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 a logical interface expressed using contracts or refinement 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 based on concepts of contracts for reasoning on networked devices and integrate them as cyber-physical systems [8]. An invited paper [9] outlines our progress with respect to 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 constraints 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 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, a developer could build a verified IoT application by ensuring that a well-typed program cannot violate the physical constraints of its architecture and environment.

4 Application domains

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 focuses on different CPS abstractions that interact in subtle ways: logic of PLC programs, real-time electro-mechanical 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

[5] *ADFG for the synthesis of hard real-time applications.* A. Bouakaz, J.-P. Talpin, J. Vitek. ACSD, IEEE, June 2012.
[6] *Design of SCJ Level 1 Applications Using Affine Abstract Clocks.* A. Bouakaz and J.-P. Talpin. SCOPES, ACM, 2013.
[7] *La vérification de programmes par interprétation abstraite.* P. Cousot. Séminaire au Collège de France, 2008.
[8] Refinement types for system design. Jean-Pierre Talpin. FDL’18 keynote.
[9] Steps toward verified programming of embedded computing systems. Jean-Pierre Talpin, Jean-Joseph Marty, Deian Stefan, Shravan Nagarayan, Rajesh Gupta, DATE’18.

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

This last year of project-team TEA was the occasion to publish three major articles presenting the latest results of its participants on: the polychronous model of computation and its relation with Kahn networks [9], the applications of affine abstraction for scheduling synchronous data-flow graphs [10], and a semantics model of cyber-physical systems in the unified theory of programming [11]. It is the place and occasion to wholeheardedly thank all the coauthors of these articles for the trepident adventure that TEA has indeed been and wish all the best on their carrier in science, to the youngest, and a joyful retirement to Paul, Loïc and Thierry :)

6 New software, platforms, open data

6.1 New software

6.1.1 CertFC

Name: End-to-end Mechanized Proof of an eBPF Virtual Machine for Micro-controllers

Keywords: Virtualization, Network Function Virtualization, Proof, Isolation, Code generation

Functional Description: CertrBPF includes a verified C verifier and interpreter. The verifier performs static checks to verify that the rBPF instructions are syntactically correct. If the verification succeeds, the program is run by the interpreter. The static verification and the dynamic checks ensure software fault isolation of the running rBPF propgram. Namely, we have the guarantee that:

The interpreter never crashes. More precisely, the proved C program is free of undefined behaviours such as division by zero or invalid memory accesses. All the memory accesses are performed in designated memory regions which are an argument of the interpreter.

The development of CertrBPF follows a refinement methodology with three main layers:

The proof model: an executable Coq specification of the rBPF virtual machine The synthesis model: a refined and optimised executable Coq program that is close in style to a C program. Eventually, we have the synthesis model (named dx model) which is compliant with the dx C extraction tool. The implementation model: the extracted C implementation in the form of CompCert Clight AST.

URL: https://github.com/future-proof-iot/CertFC

Authors: Shenghao Yuan, Jean-Pierre Talpin, Frederic Besson, Samuel Hym, Koen Zandberg, Emmanuel Baccelli

Contact: Jean-Pierre Talpin

6.1.2 ADFG

Name: 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., relations 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 scheduler for the actors.

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

URL: `https://gitlab.inria.fr/ADFG/ADFG`

Authors: Thierry Gautier, Jean-Pierre Talpin, Adnan Bouakaz, Alexandre Honorat, Loïc Besnard, Hai Nam Tran

Contact: Jean-Pierre Talpin

7 New results

7.1 End-to-end verification of operating system services

Participants: Shenghao Yuan, Jean-Pierre Talpin.

RIOT is a micro-kernel dedicated to IoT applications that adopts eBPF (extended Berkeley Packet Filters) to implement so-called femto-containers. As micro-controllers rarely feature hardware memory protection, the isolation of eBPF virtual machines (VM) is critical to ensure system integrity against potentially malicious programs. Low-power operating system runtimes used on IoT microcontrollers typically provide rudimentary APIs, basic connectivity and, sometimes, a (secure) firmware update mechanism. In contrast, on less constrained hardware, networked software has entered the age of serverless, microservices and agility. With a view to bridge this gap, we design Femto-Containers, a new middleware runtime which can be embedded on heterogeneous low-power IoT devices. Femto-Containers enable the secure deployment, execution and isolation of small virtual software functions on low-power IoT devices, over the network. We implement FemtoContainers, and provide integration in RIOT, a popular open source IoT operating system. We then evaluate the performance of our implementation, which was formally verified for fault-isolation, guaranteeing that RIOT is shielded from logic loaded and executed in a Femto-Container. Our experiments on various popular microcontroller architectures (Arm Cortex-M, ESP32 and RISC-V) show that Femto-Containers offer an attractive trade-off in terms of memory footprint overhead, energy consumption, and security. Additionally, we show how to directly derive, within the Coq proof assistant, the verified C implementation of an eBPF virtual machine from a Gallina specification. Leveraging the formal semantics of the CompCert C compiler, we obtain an end-to-end theorem stating that the C code of our VM inherits the safety and security properties of the Gallina specification. Our refinement methodology ensures that the isolation property of the specification holds in the verified C implementation. Preliminary experiments demonstrate satisfying performance [12].

7.2 Semantic foundations for cyber-physical systems using higher-order logic

Participants: Jean-Pierre Talpin.

Model-based design has become the predominant approach to the design of hybrid and cyber-physical systems (CPSs). It advocates the use of mathematically founded models to capture heterogeneous digital and analog behaviours from domain-specific formalisms, allowing all engineering tasks of verification, code synthesis and validation to be performed within a single semantic body. Guaranteeing the consistency among the different views and heterogeneous models of a system at different levels of abstraction however poses significant challenges. To address these issues, Hoare and He's Unifying Theories of Programming (UTP) proposes a calculus to capture domain-specific programming and modelling paradigms into a unified semantic framework. Our goal is to extend UTP to form a semantic foundation for CPS design. Higher-Order UTP (HUTP) is a conservative extension to Hoare and He's theory which supports the specification of discrete, real-time and continuous dynamics, concurrency and communication, and higher-order quantification. Within HUTP, we define a calculus of normal hybrid designs to model, analyse, compose, refine and verify heterogeneous hybrid system models. In addition, we define respective formal semantics for HCSP (Hybrid Communicating Sequential Processes) and Simulink using HUTP. We apply this framework to Matlab/Simulink, a de-facto industrial standard for modelling embedded systems. Reflecting the complexity of cyber-physical system (CPS) design, the semantics of Simulink is complex, mixing discrete and continuous time and events. In this paper, we define a compositional semantics of hierarchical Simulink diagrams using Higher-order Unifying Theories of Programming (HUTP) for CPS design. The HUTP theory satisfies the suitable algebraic properties to serve as a mathematical foundation for expressing the semantics of CPSs, in particular Simulink diagrams. We characterise a class of well-formed Simulink diagrams and prove the determinacy of their HUTP semantics. Moreover, we construct a framework for proving the consistency between Simulink diagrams and their translation to HCSP (Hybrid Communicating Sequential Processes). Finally, we provide a case study to illustrate and justify this translation [11].

7.3 ADFG: Affine data-flow graphs scheduler synthesis

Participants: Thierry Gautier, Jean-Pierre Talpin.

The major drawback of using static schedules to execute dataflow applications is their high inflexibility. In realtime systems, periodic schedules make it easier to assert safety guarantees and to decrease the schedule size, but their characteristics remain hard to compute. This article presents an approach to automatically generate fixed priority schedules from a dataflow specification. To do so, precedence dependencies between actors in the dataflow graphs are abstracted, as well as the task periods, by using affine relations. This abstraction allows us to synthesize schedules efficiently considering two main objectives: the maximization of throughput and the minimization of buffer sizes. Given a dataflow graph to execute in a real-time environment, we transform it into an Affine DataFlow Graph (ADFG) and compute the task priorities, their mapping, the number of delays in the buffers, and the buffer sizes. This article is the first to present an overview of both theoretical and practical aspects of ADFG. On the theoretical side, it presents corrections and improvements on the fixed priority case. On the practical side, benchmark evaluations demonstrate the robustness and maturity of the approach that our scheduling synthesizer implements. Synthesized schedules are evaluated by using scheduling simulation and real-time implementation. Last but not least, the synthesized periods reach the optimal throughput if enough processors are available, and most of the time the periods reach the maximal processor utilization factor in the uni-processor case. Moreover, execution time of the synthesis is about only one second for the main proposed algorithms [10].

7.4 The multi-clocked model of cumputation

Participants: Thierry Gautier, Jean-Pierre Talpin.

In 1974, Gilles Kahn defined a seminal semantic model for asynchronous dataflow programming that would then be called as the eponymous Kahn process networks (KPN) and instantiated in as many models of the so-called DPN hierarchy as domain-specific fields of information processing from digital signal processing to hybrid cyber-physical systems. Among these, synchronous programming models have had an important impact in the specific domain of embedded system design. We consider an instance of what seems to be one of the many synchronous models of computation: polychrony, initiated by the dataflow language Signal and its multi-clock (i.e. polychronous) model of computation and, later on, CCSL (clock constraints specification language). We provide an in-depth study of its semantic relationships with respect to the original definition of KPNs and hint toward the idea of polychrony as a methodology to locally synchronize (abstractions of) globally asynchronous processes. In particular, we formally define the property, referred to as "polyendochrony", that allow to consider a given desynchronized network of synchronous Signal processes (a GALS architecture) as an implementation of a corresponding KPN model (an asynchronous network of Khan-deterministic functions). For this class of networks, we formalize the Signal program analysis and transformations that defines synchronous clusters of Signal processes of guaranteed deterministic behavior in an asynchronous network, that is, without synchronizing communications in the entire network. This definition yields a new strategy of multi-threaded code generation that is available in the open-source Polychrony toolset of the Signal language and blurs the limits between asynchronous and polychronous models of computation [9].

8 Bilateral contracts and grants with industry

8.1 Bilateral contracts with industry

Inria – Mitsubishi Electric framework program

Title: Inria – Mitsubishi Electric framework program

INRIA principal investigator: Jean-Pierre Talpin

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

Duration: 2018+

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

8.2 Bilateral grants with industry

Participants: Stéphane Kastenbaum, Jean-Pierre Talpin.

Mitsubishi Electric R&D Europe

Title: A logical framework to verify requirements of hybrid system models

INRIA principal investigator: Jean-Pierre Talpin, Stéphane Kastenbaum

International Partner: Mitsubishi Electric R&D Europe

Duration: 2019 - 2023

Abstract: The goal of this doctoral project is to verify and build cyber-physical systems (CPSs) with a correct-by-construction approach in order to validate system requirements against the two facets of the cyber and physical aspects of such designs. Our approach is based on components augmented with formal contracts that can be composed, abstracted or refined. It fosters on the proof of system-level requirements by composing individual properties proved at component level. While semantically grounded, the tooling of this methodology should be usable by regular engineers (i.e. not proof theory specialists).

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Participation in International Programs

PIFI

Participants: Jean-Pierre Talpin.

Title: Vérification de systèmes hybrides, dynamiques et mobiles

Partner Institution: State Key Laboratory of Computer Science (SKLCS), Chinese Academy of Science (CAS)

Date/Duration: 2024 - 2026

Program: CAS President's International Fellowship Initiative (PIFI), Outstanding International Innovation Team

9.2 International research visitors

9.2.1 Visits of international scientists

Naijun Zhan

Status Research professor

Institution of origin: SKLCS

Country: China

Dates: September, 18-22

Context of the visit: collaboration with SKLCS

Mobility program: funded by the SKLCS, CAS

Xiong Xu

Status Research Assistant

Institution of origin: SKLCS

Country: China

Dates: 1 Juillet - 30 Septembre

Context of the visit: collaboration with SKLCS

Mobility program: funded by the SKLCS, CAS

9.2.2 Visits to international teams

Jean-Pierre Talpin

Visited institution: SKLCS

Country: China

Dates: December 17 - January 28 and October 16 - November 24

Context of the visit: Invited Research Professor with CAS

Mobility program: funded by the SKLCS, CAS

9.3 National initiatives

Participants: Jean-Pierre Talpin, Shenghao Yuan.

Title: RIOT-fp: Future-proof IoT

Duration: 4 years

Coordinator: Emmanuel Bachelli

Partners: Tribe, Eva, Grace, Prosecco, Tea, Freie Universität Berlin and Fujitsu.

Summary: RIOT-fp is a research project on cyber-security targeting low-end, microcontroller-based IoT devices, on which operating systems such as RIOT run, and the development of a low-power network stack. Taking a global and practical approach, RIOT-fp gathers partners planning to enhance RIOT with an array of security mechanisms. The main challenges tackled by RIOT-fp are: 1/ developing high-speed, high-security, low-memory IoT crypto primitives, 2/ providing guarantees for software execution on low-end IoT devices, and 3/ enabling secure IoT software updates and supply-chain, over the network. Beyond academic outcomes, the output of RIOT-fp is open source code published, maintained and integrated in the open source ecosystem around RIOT. As such, RIOT-fp strives to contribute usable building blocks for an open source IoT solution improving the typical functionality vs. risk tradeoff for end-users. The goal of project-team TEA in RIOT-fp is to build verified operating system libraries for IoT devices using proof-oriented programming techniques in Coq and F*, such as the actual bootloader of RIOT and its femto-containers: virtual machines isolating kernel-level execution of user-supplied applications and services using the eBPF virtual ISA.

10 Dissemination

Participants: Jean-Pierre Talpin.

10.1 Promoting scientific activities

General chair, scientific chair

- Jean-Pierre Talpin chairs the steering committee of the ACM/IEEE Symposium on formal methods and models for system design (MEMOCODE), integrated to ESWEEK (Embedded Systems Week) since 2022.

Member of conference program committees

- Jean-Pierre Talpin participated to the program committee of the Interntional Symposium on Dependable Software Engineering (SETTA).
- Jean-Pierre Talpin participated to the test-of-time Award committee of the ACM/IEEE conference on Embedded Software (EMSOFT).

Reviewer - reviewing activities

- Jean-PIerre Talpin reviewed articles for the journal of logical and algebraic methods in programming, the journal of system architectures, the journal on formal aspects of computing.

10.2 Teaching - Supervision - Juries

- Jean-Pierre Talpin supervised the Ph.D. thesis of Stéphane Kastenbaum, awarded on May 5th.
- Jean-Pierre Talpin supervised the Ph.D. thesis Shenghao Yuan, awarded on December 15th [13].

11 Scientific production

11.1 Major publications

[1] L. Besnard, T. Gautier, P. Le Guernic and J.-P. Talpin. 'Compilation of Polychronous Data Flow Equations'. In: *Synthesis of Embedded Software*. Ed. by S. K. Shukla and J.-P. Talpin. Springer, 2010, pp. 1–40. DOI: 10.1007/978-1-4419-6400-7_1. URL: https://hal.inria.fr/inria-00540493.

[2] T. Gautier, P. Le Guernic, L. Besnard and J.-P. Talpin. 'The Polychronous Model of Computation and Kahn Process Networks'. In: *Science of Computer Programming* (29th Apr. 2023), pp. 1–49. DOI: 10.1016/j.scico.2023.102958. URL: https://inria.hal.science/hal-04216543.

[3] A. Honorat, H. N. Tran, T. Gautier, L. Besnard, S. S. Bhattacharyya and J.-P. Talpin. 'Real-Time Fixed Priority Scheduling Synthesis using Affine DataFlow Graphs: from Theory to Practice'. In: *ACM Transactions on Embedded Computing Systems (TECS)* (18th Aug. 2023), pp. 1–30. DOI: 10.1145/3615586. URL: https://hal.science/hal-04200195.

[4] S. Kastenbaum, B. Boyer and J.-P. Talpin. 'A Mechanically Verified Theory of Contracts'. In: ICTAC 2021 - 18th International Colloquium on Theoretical Aspects of Computing. Nur-Sultan, Kazakhstan, 20th Aug. 2021, pp. 134–151. DOI: 10.1007/978-3-030-85315-0_9. URL: https://inria.hal.science/hal-03329311.

[5] S. Nakajima, J.-P. Talpin, M. Toyoshima and H. Yu. *Cyber-Physical System Design from an Architecture Analysis Viewpoint*. Communications of NII Shonan Meetings. Springer, Jan. 2017. DOI: 10.1007/978-981-10-4436-6. URL: https://hal.inria.fr/hal-01615144.

[6] X. Xu, B. Zhan, S. Wang, J.-P. Talpin and N. Zhan. 'A denotational semantics of Simulink with higher-order UTP'. In: *Journal of Logical and Algebraic Methods in Programming* 130 (Jan. 2023), p. 100809. DOI: 10.1016/j.jlamp.2022.100809. URL: https://inria.hal.science/hal-03888092.

[7] H. Yu, J. Prashi, J.-P. Talpin, S. K. Shukla and S. Shiraishi. 'Model-Based Integration for Automotive Control Software'. In: *Digital Automation Conference*. ACM. San Francisco, United States, June 2015. URL: https://hal.inria.fr/hal-01148905.

[8] S. Yuan, F. Besson, J.-P. Talpin, S. Hym, K. Zandberg and E. Baccelli. 'End-to-end Mechanized Proof of an eBPF Virtual Machine for Micro-controllers'. In: CAV 2022 - 34th International Conference on Computer Aided Verification. Haifa, Israel, 7th Aug. 2022, pp. 1–23. URL: https://inria.hal.science/hal-03888082.

11.2 Publications of the year

International journals

[9] T. Gautier, P. Le Guernic, L. Besnard and J.-P. Talpin. 'The Polychronous Model of Computation and Kahn Process Networks'. In: *Science of Computer Programming* (29th Apr. 2023), pp. 1–49. DOI: 10.1016/j.scico.2023.102958. URL: https://inria.hal.science/hal-04216543.

[10] A. Honorat, H. N. Tran, T. Gautier, L. Besnard, S. S. Bhattacharyya and J.-P. Talpin. 'Real-Time Fixed Priority Scheduling Synthesis using Affine DataFlow Graphs: from Theory to Practice'. In: *ACM Transactions on Embedded Computing Systems (TECS)* (18th Aug. 2023), pp. 1–30. DOI: 10.1145/3615586. URL: https://hal.science/hal-04200195.

[11] X. Xu, B. Zhan, S. Wang, J.-P. Talpin and N. Zhan. 'A denotational semantics of Simulink with higher-order UTP'. In: *Journal of Logical and Algebraic Methods in Programming* 130 (Jan. 2023), p. 100809. DOI: 10.1016/j.jlamp.2022.100809. URL: https://inria.hal.science/hal-03888092.

International peer-reviewed conferences

[12] S. Yuan, B. Lion, F. Besson and J.-P. Talpin. 'Making an eBPF Virtual Machine Faster on Microcontrollers: Verified Optimization and Proof Simplification'. In: SETTA 2023 - 9th International Symposium Dependable Software Engineering. Theories, Tools, and Applications. Vol. 14464. Lecture Notes in Computer Science. Nanjing (Chine), China: Springer Nature Singapore, 30th Nov. 2023, pp. 385–401. DOI: 10.1007/978-981-99-8664-4_22. URL: https://inria.hal.science/hal-04376380.

Doctoral dissertations and habilitation theses

[13] S. Yuan. 'Verified programming and secure integration of operating system libraries in Coq'. Université de Rennes, 8th Dec. 2023. URL: https://theses.hal.science/tel-04405955.