RESEARCH CENTRE

Inria Center at the University of Lille

IN PARTNERSHIP WITH:

Université de Lille, CNRS

2022

ACTIVITY REPORT

Project-Team

SYCOMORES

Symbolic analysis and Component-based design for Modular Real-Time Embedded Systems

IN COLLABORATION WITH: Centre de Recherche en Informatique, Signal et Automatique de Lille

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Embedded and Real-time Systems

Inria

Contents

Project-Team SYCOMORES

Creation of the Project-Team: 2021 October 01

Keywords

Computer sciences and digital sciences

A2.1.9. – Synchronous languages

A2.3.1. – Embedded systems

A2.3.3. – Real-time systems

A2.4.1. – Analysis

A2.4.3. – Proofs

A2.6.1. – Operating systems

A7.2. – Logic in Computer Science

Other research topics and application domains

B6.6. – Embedded systems

1 Team members, visitors, external collaborators

Research Scientists

- Patrick Baillot [CNRS, Senior Researcher]
- Raphaël Monat [Inria, Researcher, from Sep 2022]
- Vlad Rusu [Inria, Researcher, HDR]

Faculty Members

- Giuseppe Lipari [Team leader, UNIV LILLE, Professor]
- Clément Ballabriga [UNIV LILLE, Associate Professor]
- Julien Forget [UNIV LILLE, Associate Professor]

Post-Doctoral Fellow

- Leandro Moreira Gomes [UNIV LILLE, from Jun 2022]

PhD Students

- Fabien Bouquillon [UNIV LILLE, until Oct 2022]
- Nordine Feddal [Inria, from Nov 2022]
- Frédéric Fort [UNIV LILLE, until Oct 2022]
- Sandro Grebant [UNIV LILLE]
- Ikram Senoussaoui [UNIV LILLE, ATER]

Administrative Assistant

- Nathalie Bonte [Inria]

2 Overall objectives

The SYCOMORES project-team aims at developing a **framework for the design and the analysis of embedded real-time systems** based on **symbolic analysis of parametric components**.

To reduce the complexity, we will use a modular approach to the design, development and analysis of ERTS. In particular, we will decline *modularity* along several directions (Figure 1):

- a *component-based design and development* methodology; a ERTS system is decomposed in generic, configurable and reusable components that can be combined and instantiated in the final system;
- *parametric specification* of inputs, configurations and properties; a component is characterised by its interface that will be specified using parameters whose concrete values are unknown at design time;
- *symbolic analysis* of components; by using symbolic techniques, the properties of a component will be proved correct at design time even when parameters have unknown values.

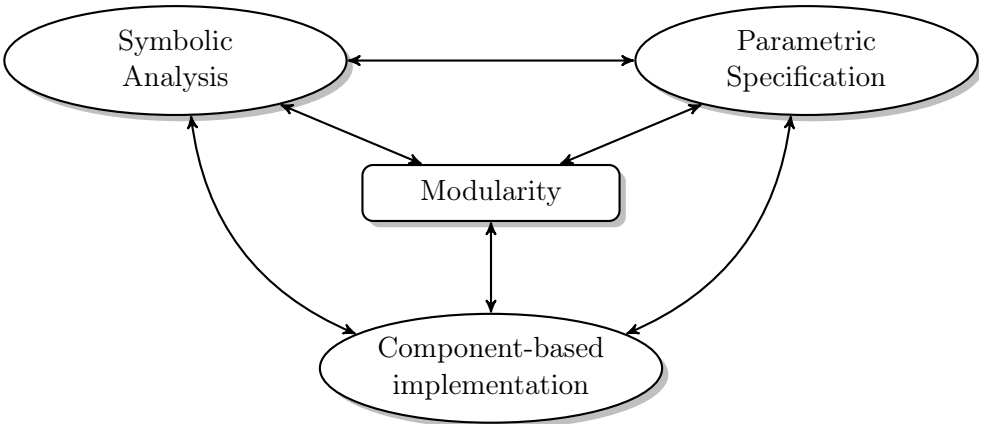

Figure 1: Ontology of terms in SYCOMORES.

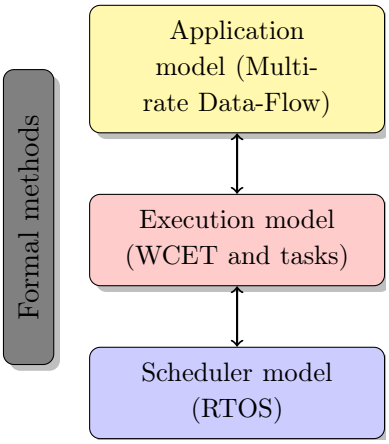

Figure 2: Abstraction Layers

Finally, we will propose *composition operators* to integrate a set of components into a larger component, or into a complete system. The operators will allow to (partially or totally) instantiate parameters and connect component interfaces guaranteeing their compatibility and preserving their properties.

To this end, we will investigate several challenging problems at three different levels of abstraction, as shown in Figure 2. At the application level, we will focus on *multi-rate data-flow* programming languages like SCADE [21], SIMULINK [34], or in our case PRELUDE [35], as they are widely used in safety-critical domains like avionics and automotive. We want to tackle the notions of component, parametric interfaces and contracts in the ERTS context. We will investigate **modular code generation** (compilation) of a synchronous component (as opposed to a synchronous program) with real-time constraints. In the long term, we would like to **prove the semantic preservation of properties** of the program during compilation by using interactive proof systems.

The compiler will produce the component implementation as a set of concurrent real-time tasks with their scheduling parameters. At this level, we will work on **parametric and modular worst-case execution time (WCET) analysis** of tasks' code, and **symbolic schedulability analysis** of the components' tasks. We will also work on analysis of the correctness of system integration, and **safe dynamic replacement/upgrade** of components.

At the lowest level, we will work on the scheduler implementation in the Operating System. We will propose a **hierarchical scheduling architecture**, in order to provide temporal isolation to real-time components. We will formally prove the scheduler properties by using **modular proof techniques** in the Coq proof assistat [40], and we will provide a **correct-by construction implementation of the scheduler**

by using code generation techniques from a Coq program specification.

As a cross-cutting activity, since safety will be a prime concern in our project, we will rely on *formal methods* throughout the project: synchronous programming languages, abstract interpretation, symbolic analyses and proof techniques. Our research objectives will be tightly coupled to ensure the safety of the final framework.

For instance, our tools and techniques will share proof facts (e.g. timing analysis is correct only if schedulability analysis and WCET analysis are both correct), models (e.g. the schedulability analysis, programming language and WCET analysis must share the same task model), and results (e.g. the WCET and schedulability results will impact the program design).

3 Research program

We detail the objectives by categorising them according to 3 major research axes. Within each axis, we distinguish Short Term (1-3 years) and Medium Term (2-4 years) objectives, and we describe how we will achieve them. We also detail how the objectives are related to one another. Finally, we present our Long Term (4+ years) objectives.

3.1 Axis 1: Design and implementation of RT-components

In this research axis we will focus on the programming of RT-components. We will cover programming aspects at different levels of the development process, from high-level design, to low-level implementation on distributed heterogeneous embedded platforms.

3.1.1 Short term

(A1.S1) Synchronous languages We will continue working on synchronous programming languages for generating correct-by-construction code. In particular, one of the objectives of the ANR projet CORTEVA (which is coordinated by G. Lipari, started in Jan. 2018, and will last for 48 months) is to extend the synchronous language PRELUDE [35] to address systems with extremely variable execution time.

The language is being extended (its semantics and its compilation) with constructs to dynamically change the behaviour of a subset of the system based on rare events, like the occurrence of a large execution time of a task. The proposed extensions will guarantee that the system will respect correctness properties even in the presence of such rare events.

Dependencies: We rely on Parametric WCET to detect the occurence of large execution times (A2.S1).

(A1.S2) Scheduling of complex task models Modern hardware platforms consist of heterogeneous processors with a large degree of parallelism. For example, the NVIDIA Jetson AGX Xavier chipthat is used in modern ADAS systems in the automotive domain, comprises 2 DLAs (Deep Learning Accelerators), 1 integrated GPU (Graphical Processing Unit), and 8 ARM 57 processing cores. Efficiently programming ERTS for such hardware is not easy, as the complexity of the interaction between software and hardware resources makes it difficult to predict performance. Building predictable real-time applications on these platforms requires appropriate programming models. In the recent literature, many graph-based task models have been proposed to exploit the parallelism of such architectures [39, 24]. Such models are neither based on components, nor are they parametric. It is our intention to investigate the possibility to apply component-based techniques to such complex task models.

3.1.2 Medium term

(A1.M1) Parametric scheduling analysis of Real-Time components. Analyzing the schedulability of a system under variations of the task parameters is a complex problem: the complexity of the analysis grows exponentially with the size of the system, and it explodes for medium-to-large sized systems.

The problem can be decomposed by analysing the schedulability of smaller real-time components under variation of some parameters. The idea is to follow a Resource Reservation strategy, where components are assigned and guaranteed a fraction of the system resources. In this way, the temporal behaviour of one component is *isolated* from the interference of other components.

First, we will introduce the notion of *parametric interface* for a component modelled as a multi-rate data-flow program. The interface will specify the temporal properties and constraints on the input/outputs of a component: periodicity and deadline constraints, dimension of the buffer for communicating with other components, and communication protocols.

To analyse the schedulability of a components under the hypothesis of variation of its parameters (the ones specified in the interface as well as the WCET of the tasks), we will extend the *sensitivity analysis* techniques, first proposed by Bini et al. [26, 25] to more complex task models and to hierarchical scheduling systems. The objective is to find the range of the parameters for which the component is schedulable.

(A1.M2) Predictable communication costs. ERTS are increasingly being built using multicore hardware. This transition is however still significantly delayed by the difficulty to provide *safe* and *tight* timing guarantees. Indeed, even though multicore architectures enable the simultaneous parallel execution of programs on different cores, these programs are not completely independent. They have to be synchronized at some point to exchange data and they also share some common resources, such as peripherals, communication bus and (parts of the) memory. Synchronizations and shared resource contentions cause hard-to-predict interferences and timing overheads, which significantly increase the complexity of timing analyzes (WCET and scheduling).

One solution for mitigating these overheads, and making them more predictable, consists in relying on multi-phase task models that decouple communication phases from computation phases [37, 29]. This greatly simplifies WCET analysis of computation phases, and also makes it possible to schedule communication phases so as to reduce their interference.

Memory architectures based on local addressable memories (scratchpads), instead of caches, are also being proposed [30, 36], to avoid the hard-to-predict timing effects of cache consistency mechanisms (which are used to speed-up access to the main shared memory).

We plan to integrate these two approaches in our project, since predictability is a major concern. The PRELUDE compiler will be extended to provide multi-phase task code generation, to be executed on scratchpad-based memory architectures. We will also develop the corresponding schedulability analysis method. Our objective is to devise our development framework such that the programmer abstracts from the implementation of communication mechanisms related on the target OS and hardware. This will enable the programmer to seamlessly transition between different architectures (unicore/multicore, cache-based/scratchpad-based).

3.2 Axis 2: Static Analysis of RT components

In this research axis, we will design static program analysis techniques for RT components at the binary level. While these techniques will mainly focus on WCET analysis, some results will also be reused to analyze security properties instead.

3.2.1 Short Term

(A2.S1) Symbolic WCET analysis. We will work towards improving static analysis techniques for Worst-Case Execution Time Analysis. In particular, we will extend the Parametric WCET method [23] with more powerful symbolic capabilities. Currently, Parametric WCET can represent the WCET of a function as a formula in a number of parameters (e.g. input data). It does however suffer from two important limitations. First, it cannot represent relations between distinct parameters. We will extend the Parametric WCET by abstract interpretation of binary code to detect linear relations between distinct parameters in the code, by using techniques similar to [22]. Second, it struggles to represent program properties that relate instructions of the program that are not close to one another, such as for instance infeasible execution paths [38]. We are currently extending this work to enable the representation of properties related to such *global execution contexts*.

3.2.2 Medium Term

(A2.M1) Modular WCET analysis Traditional WCET analysis is performed on a whole program. This approach is limited by two factors: first, performance concerns (analysis time tends to grow faster than the analyzed program size), and second, the analysis requires access to the complete program (including libraries, etc.). A modular and composable WCET analysis approach would reduce the analysis time, and enable the integration of third-party library or system calls in the analysis process.

To this end, we will extend our polyhedra-based abstract interpretation [22] to support inter-procedural analysis, based on function *summaries* [28], describing relations between the inputs and outputs of the function. This will enable us to compute WCET-related functional properties, such as e.g. loop bounds, in a composable way.

This work on composable analysis will lead to a full composable WCET analysis, by integrating it in our symbolic WCET computation framework [23].

Dependencies: To achieve this objective, we will need the advanced parameter handling techniques of S2.

(A2.M2) Security analysis of binary code Program analysis of ERTS has historically focused mainly on *safety*, i.e. ensuring the absence of system failures. However, in recent years ERTS have become increasingly more connected to other computer systems through communication networks. This makes ERTS more vulnerable to external attacks, as illustrated by various reports of hacks of highly computerized modern cars. This results in an increased need for analyses that focus on ensuring the *security* of ERTS, i.e. ensuring there is no vulnerability to external malevolent computer attacks.

Our work on abstract interpretation of binary code [22] enables to detect linear relations between the data-locations accessed by a binary program (registers, memory addresses and their contents). While this work was initially targeted for WCET analysis, we plan to apply the developped techniques to the security domain as well. In particular, we will analyze security properties specific to the binary structure (e.g. unauthorized accesses to specific memory sections) and study the discovery of potential security threats in closed-source software (to detect malwares).

3.3 Axis 3: Proof of RT Components

The formal proof activity in the project will focus on compositional techniques for proving RT components of operating systems. Our plan is to focus mainly on RT schedulers.

3.3.1 Short term

(A3.S1) A proof methodology for a standard scheduling policy We shall start with the standard scheduling policy EDF (Earliest Deadline First) and develop a proof methodology for it, which we shall later adapt to more advanced policies. The methodology incorporates a formal notion *refinement* as a means to master complexity and to smoothly descend from abstract definitions down to executable schedulers.

First, we are planning to model EDF at an abstract level in Coq. We shall formally prove at this level the *schedulability* property: under adequate hypotheses any given set of periodic hard real-time tasks can be scheduled, such that each task completes before its deadline. Since in the short term we shall only deal with strictly periodic, hard real-time tasks (reading data from sensors, performing some computation and sending the results to actuators), one only needs to consider the schedulability on finite executions, whose length equals the hyper-period: the least common multiplier of the task's periods. As a result, we expect that *induction*, well supported by Coq, will be the appropriate proof technique for this stage of the project.

Then, we shall refine the abstract EDF scheduling policy into a scheduling *algorithm* written in Coq's input language *Gallina*, a purely functional language, and shall prove again by induction that the algorithm preserves the already established properties of the policy.

Next, our plan is to further refine these *functional* algorithms into *imperative* Gallina programs, in order to get a step closer to executable code[1]. Imperative programs in purely functional languages such

[1] Going directly from fonctional to executable code is not an option, since that would require a complex compilation process with a high risk of losing the schedulability properties already proved on the functional code.

as Gallina are traditionally written using *monads* e.g., *state* monads, which enable variable assignments in functional code. For doing this we shall benefit from the experience of our colleagues in the 2XS team, our closest collaborators within the CRIStAL laboratory. They have been developing a minimalistic OS kernel called Pip [41] in imperative Gallina using the state-monad technology and have directly proved properties of the kernel in Coq. Unlike them, we do not prove properties directly on the monadic code, but shall prove a refinement step (from functional to imperative) ensuring that schedulability holds on the imperative scheduler.

The final step is translating the imperative Gallina scheduler to executable code. For this we shall use a translator also developped by the 2XS team, which essentially maps word-for-word imperartive Gallina programs to C programs with appropriate primitives that, after compilation of the C code, turn our schedulers into executable programs within Pip.

3.3.2 Medium term

In the medium term we are planning to make progress on two directions: the RT components being proved and the proof techniques.

(A3.M1) More advanced schedulers and other RT components Once the validation of the proof/refinement methodology on the EDF example is complete, we are planning to progressively generalise it to other schedulers. The next likely candidate is the *Grub* scheduler, developped by Giuseppe Lipari [20, 32], which generalises EDF and makes it possible to mix periodic tasks (hard real-time) with sporadic tasks (soft real-time). The algorithm guarantees that the deadlines of the hard real-time tasks are met, while for the soft real-time ones a certain quality of service is guaranteed.

We are also planning to formally verify existing resource reservation hierarchical schedulers [33] by extending the proof/refinement approach with compositional features that exploit the component-based nature of the considered applications.

Depending on the availability of human resources, we would also like to formally verify other RT components, such as interruption multiplexers, memory managers, or synchronisation mechanisms.

(A3.M2) More advanced proof techniques We expect that different verification techniques will be necessary to prove these more advanced schedulers. The EDF scheduler can be proved correct by considering its behaviour over a limited period of time, which as explained above can be dealt with using induction. However, Grub also has to schedule sporadic tasks that, by definition, do not repeat themselves periodically. Without a periodic repetition the behaviour cannot be studied over a finite interval, infinite executions have to be considered. Hence, we are planning to use coinduction, the natural technique for defining and reasoning about infinite objects. As stated in the Preliminaries we already have experience with coinduction, especially, with improving the way it is dealt with in Coq to make it better suited for use in nontrivial applications. We are planning to continue doing this both for corecursive program definitions (e.g., reactive systems, including schedulers, are such programs) and for coinductive reasoning techniques.

3.4 Long term

In the long term, we will integrate the elements studied during the medium term phase, in order to provide a seamless framework that goes from high-level design to final implementation. Translations will be automated and proved correct. These objectives are transverse by nature, so all members of the project-team will participate. Since these objectives focus on integration of our previous results, they are related to all of our short and medium term objectives.

(L1) Integrated framework During the medium term phase, we will develop bricks that contribute to the design and analysis of ERTS. In the long term phase, we will integrate these bricks into a complete framework. This will raise several research topics.

First, we will have to evaluate the impact of scheduling on WCET analysis. Though this topic has already been studied before, our symbolic approach to both problems will raise new opportunities and challenges.

Second, we will evaluate the scalability of the approach with respect to realistic and complex real-time programs. In particular, the advent of powerful heterogeneous hardware platform permits to exploit their large scale parallelism. It is then important to check the suitability and the expressiveness of our framework with respect to these new powerful platforms.

(L2) Proving semantics preservation In our framework, an ERTS is first specified with a high-level data-flow semantics (in PRELUDE). Then, it undergoes translations to produce the final program (in C) embedded on the hardware platform. One of our long term objectives is to prove that the complete translation process, from PRELUDEto C, is semantics-preserving.

There exists previous work and techniques for the formal verification of compilers such as COMPCERT [31] (from C code to binary code) and the LUSTRE verified compiler [27]. However, these works focus on the preservation of the *functional semantics* (computing the correct values). A major novelty of our work will be to focus on the preservation of the *temporal semantics* (computing values at the correct time) as well. As far as we know, proving the preservation of temporal semantics was previously explored in theoretical models (e.g. timed automata, or Petri nets), but not in programming languages and compilers. It is an ambitious challenge, because it connects compilation with scheduling and WCET analysis.

(L3) Corecursion-preserving compilation and coinductive verification techniques An alternative view of reactive programs (such as PRELUDE programs) is that of corecursive transformers from infinite flows of inputs to infinite flows of outputs. We envisage an enhanced compilation chain that, in addition to what is planned in L2, enables the tracing of corecursion down to the executable code. Since corecursive calls typically compile to low-level instructions such as loops or jumps, such a tracing mechanism would enable recognising the corecursive calls at the low-level. This, in turn, would enable the formal reasoning about low-level corecursive programs, using coinductive techniques that we shall develop in A3. We note that schedulers can also be expressed as corecursive programs, hence the verification boils down to compositionally verifying compositions of corecursive programs. It is, again, an ambitious challenge at every step, since corercursive programs are notoriously difficult to define, compose, and verify. This envisaged works depends on progress in essentially all the objectives enumerated above.

(L4) Dynamic update of components ERTS may evolve over time, in particular, one component may be upgraded while the system is operational. In the traditional development process of critical software, once the system has been developed, tested and *certified*, it is not possible to change it anymore, with the only exception of correcting a critical bug. Dynamic updates are not possible since they would require to re-certify the whole system.

In the long term, we would like to apply our component based development process to the problem of guaranteeing the correctness of dynamic updates. This means that the system must be able to evolve over time, during operation, without jeopardizing the guarantee on existing properties.

Given the large complexity of the formal methods we will use for off-line analysis, it is unthinkable to apply the same type of analysis on-line. One possibility is to separate the analysis into two distinct parts: an off-line part which may be complex and does most of the work; and an on-line part which is much simpler but relies on the off-line computation. Alternatively, it is possible to off-load part of the analysis to the cloud, where there is abundance of computational resources.

4 Application domains

The long term research we propose to pursue in this project will advance the current development practice for embedded real-time critical systems.

First, it will impact the design and development of critical software for domains like avionics, automotive, train, etc. It is well known that developing safety-critical software is a long and costly process, where each error could endanger human life. It has been estimated that the cost to certify 30K lines of DAL A code is around 2M $ if the code has been developed by experienced programmers, and it jumps to 8M$ if the code has been produced by non-experienced ones.

The avionic industry is slowly adopting formal methods to reduce the cost by reducing (or, in certain cases, eliminating altogether) testing and improve confidence in the methodology. Our integrated framework (L1) will greatly reduce the development cost of safety-critical software because it will automatise many of the steps in the design and development methodology. For example, the use of semantic preserving transformations (L2, L3) will enhance the confidence in the correctness of the generated code, reducing the need for extensive testing, thus further reducing cost.

Other critical domains, like automotive, have not yet fully adopted safety-critical standard methodologies like in the avionic domain, mainly for cost reasons. Our framework will lower the cost of developing safety critical software, thus improving the safety of critical software in a wider range of domains.

Finally, thanks to the research in (L4), it will be possible to update a software component on-line without performing a complete analysis of the system, *while the system is operational*. An example of futuristic application will be the possibility to update one software subsystem of an autonomous vehicle while the vehicle is running, without compromising its functionality.

5 Highlights of the year

In September 2022 Raphaël Monat has joined the team as *Chargé de Recherche* (Researcher). His research interests include abstract interpretation, language and compilers, formal methods. He will reinforce the *Axis 2: Static Analysis of RT components*.

Two students defended their PhD in October 2022:

- Fréderic Fort defended his thesis *Programmation de systèmes temps réel adaptatifs* [17]; He was supervised by Giuseppe Lipari and Julien Forget.
- Fabien Bouquillon defended his thesis *Améliorer la fiabilité des architectures multicœurs hétérogènes pour les systèmes de transport intelligents* [16]. He was supervised by Giuseppe Lipari and Smail Niar.

Two new PhD students joined the team in October 2022: Andrei Florea under the direction of Vlad Rusu and Julien Forget; and Nordine Feddal under the direction of Giuseppe Lipari.

Leandro Moreira Gomes, a new post-doctoral fellow, joined our team in June 2022, to work with Patrick Baillot on the application of dynamic logic to differential privacy. His stay is funded by an I-SITE project of the University of Lille.

On the 10th of June 2022 Prof. Sanjoy Baruah of the University of Washington at Saint Louis (USA) visited our team to present his recent work on real-time scheduling. We discussed the possibility of future research collaborations.

On September 23rd 2022 Prof. Luis Soares Barbosa of the University of Minho (Portugal) visited our team and presented his work on dynamic logic. We exchanged ideas and discussed possible future collaborations.

In Fall 2022, Raphaël Monat participated to the Software Verification Competition (SV-Comp) by submitting the Mopsa static analyzer he is co-developing. Mopsa earned a **bronze medal** in the *SoftwareSystems* category. There were 19 competing tools in this category and the gold medal winner has been competing for 7 years. This is the first French participation to this competition.

6 New software and platforms

6.1 New software

6.1.1 otawa

Keywords: Static analysis, WCET

Functional Description: Otawa is an open source static analysis tool developed and maintained by the TRACES team at the University of Toulouse and co-developed by the team SYCOMORES at Inria lille Nord-Europe. Specifically, SYCOMORES has developed two plugins for OTAWA: Polymalys, for polyhedra-based analysis and loop bound estimation, and WSymb, a Control Flow Tree extractor and symbolic WCET evaluator.

URL: https://www.tracesgroup.net/otawa/

Contact: Clément Ballabriga

Partner: Université de Toulouse

6.1.2 prelude

Keywords: Synchorous language, Real time

Functional Description: Prelude is a synchronous data-flow language with real-time constraints, designed by Julien Forget. Prelude programs are compiled into muli-thread C code. The language and its compiler are developed in collaboration with ONERA Toulouse. Prelude is currently being extended to support adaptive applications.

URL: https://www.cristal.univ-lille.fr/~forget/prelude.html

Contact: Julien Forget

6.1.3 ptask

Keywords: Real time, Library

Functional Description: PTASK is a library for programming real-time systems in Linux. It serves as the target RTOS API in the SYCOMORES project. The PTASK library is authored by Giuseppe Lipari. It has been initially developed to support teaching real-time systems at the Scuola Sant'Anna. It has later been extended with additional capabilities and it is being used as target for design tools such as CPAL19 from RTaW20 and by other companies.

URL: https://github.com/glipari/ptask

Contact: Giuseppe Lipari

6.1.4 Catala

Keywords: Domain specific, Programming language, Law

Functional Description: Catala is a domain-specific programming language designed for deriving correct-by-construction implementations from legislative texts. Its specificity is that it allows direct translation from the text of the law using a literate programming style, that aims to foster interdisciplinary dialogue between lawyers and software developers. By enjoying a formal specification and a proof-oriented design, Catala also opens the way for formal verification of programs implementing legislative specifications.

Release Contributions: Changelog:

- Performance improvements - Better error message formatting - New Markdown syntax - Better lexer factorization - ...

URL: https://catala-lang.org/en

Publications: hal-03128248, hal-03159939, hal-02936606

Contact: Denis Merigoux

Partner: Université Panthéon-Sorbonne

6.1.5 dates-calc

Keywords: Law, Programming language

Functional Description: A date calculation library with a well-defined semantics

URL: https://github.com/CatalaLang/dates-calc

Contact: Raphael Monat

6.1.6 Mopsa

Keywords: Formal methods, Static analysis, Abstraction

Functional Description: Mopsa is an open-source static analysis platform relying on abstract interpretation. It provides a novel way to combine abstract domains, in order to offer extensibility and cooperation between them, which is especially beneficial when relational numerical domains are used. It is able to analyze C, Python, and programs mixing these two languages. Mopsa was originally developed at LIP6, Sorbonne Université following an ERC Consolidator Grant award to Antoine Miné. It is now partially developed at Inria.

URL: https://gitlab.com/mopsa/mopsa-analyzer/

Contact: Antoine Miné

Partner: Sorbonne Université

7 New results

7.1 A Formal Correctness Proof for an EDF Scheduler Implementation

The scheduler is a critical piece of software in real-time systems. A failure in the scheduler can have serious consequences; therefore, it is important to provide strong correctness guarantees for it. In [11] together witth colleagues in the 2XS team (CRIStAL, CNRS & Univ. Lille) we propose a formal proof methodology that we apply to an Earliest Deadline First (EDF) scheduler. It consists first in proving the correctness of the election function algorithm and then lifting this proof up to the implementation through refinements. The proofs are formalized in the Coq proof assistant, ensuring that they are free of human errors and that all cases are considered. Our methodology is general enough to be applied to other schedulers or other types of system code. To the best of our knowledge, this is the first time that an implementation of EDF applicable to arbitrary sequences of jobs has been proven correct.

Participants: Vlad Rusu.

7.2 Relational abstract interpretation of arrays in assembly code

In [4], we proposed a static analysis technique for assembly code, based on abstract interpretation, to discover properties on arrays. Considering assembly code rather than source code has important advantages: we do not require to make assumptions on the compiler behaviour, and we can handle closed-source programs. The main disadvantage however, is that information about source code variables and their types, in particular about arrays, is unavailable. Instead, the binary code reasons about data-locations (registers and memory addresses) and their sizes in bytes. Without any knowledge of the source code, our analysis infers which sets of memory addresses correspond to arrays, and establishes properties on these addresses and their contents. The underlying abstract domain is relational, meaning that we can infer relations between variables of the domain. As a consequence, we can infer properties on arrays whose start address and size are defined with respect to variables of the domain, and thus can be unknown statically. Currently, no other tool operating at the assembly or binary level can infer such properties.

Participants: Clément Ballabriga, Julien Forget.

7.3 Synchronous semantics of multi-mode multi-periodic systems

This work has been carried out in the scope of the thesis of Fréderic Fort [17], who successfully defended is thesis on October 2022. He was supervised by Giuseppe Lipari and Julien Forget. In [8], we addressed the problem of designing and programming a real-time system with multiple modes of execution, where each mode executes a different set of periodic tasks. The main difficulty is that the period of Mode Change Requests (MCR) and the period of tasks are not all the same. Thus, not all tasks perceive MCRs in the same way. When programming such a system with traditional languages without mechanisms dedicated to mode changes (e.g. C), it is difficult to ensure that the system is sound and deterministic. We proposed an extension to synchronous data-flow languages to support mode changes. The semantics of the resulting language is defined formally, which prevents ambiguous programs. The language is flexible enough to support different types of mode changes. The compiler of the language includes a static analysis that rejects programs whose semantics is ill-defined. The extension consists in transposing Synchronous State Machines to the Prelude language.

Participants: Frédéric Fort, Julien Forget.

7.4 Mopsa at the Software Verification Competition

Mopsa is a multilanguage static analysis platform relying on abstract interpretation. It is able to analyze C, Python, and programs mixing these two languages

We participated for the first time to the Software Verification Competition (SV-Comp). Mopsa earned a **bronze medal** in the *SoftwareSystems* category. This is the first French participation to this event. The *SoftwareSystems* category aims at "representing verification tasks from real software systems". There were 19 competing tools in this category and the gold medal winner has been competing for 7 years.

Participants: Raphaël Monat.

7.5 Static Analysis of Functional Programs Handling Recursive Algebraic Data Types

In [13], we describe a static value analyzer by abstract interpretation for a typed first-order functional language, which is a safe and automatic approach to guarantee the absence of errors at runtime. Based on relational abstract domains and realizing summaries of recursive fields of algebraic types, this approach allows to analyze recursive functions manipulating recursive algebraic types and to infer in an abstract domain their input-output relationship. An implementation is underway on the MOPSA multilanguage analysis platform and successfully analyzes short programs of a few lines. This work thus paves the way towards accurate and relational value analysis based on abstract interpretation for functional languages of higher order than ML.

Participants: Raphaël Monat.

7.6 Cache Analysis for Real-Time, Fault-Tolerant Systems

This work has been carried out in the scope of the thesis of Fabien Bouquillon, who successfully defended is thesis on October 2022. He was supervised by Giuseppe Lipari and Smail Niar (Polytechnique University of Hauts des France). His thesis consists of two new important contributions.

In [9], Fabien has presented two contributions for reducing the pessimism in the analysis of real-time systems that use set-associative cache memories.

Cache Related Preemption Delay (CRPD) analysis is a methodology for bounding the cost of cache reloads due to preemptions. Given the complexity of the problem, existing methods make simplifying assumptions to speed up the analysis, but they also introduce large amounts of pessimism. In the paper, we present a new way to compute a bound on the number of preemptions on a task under EDF scheduling. Second, we propose a novel algorithm that trades off speed for precision. We show improvements in the schedulability ratio on classical benchmarks in comparison with the state-of-the-art.

In [6], Fabien has proposed a new original methodology to evaluate and reduce the vulnerability of hard real-time applications to soft errors in IL1 cache memories.

With the progress of the technology, the presence of transient faults (e.g. bit-flipping errors) in cache memories becomes a challenge, especially in embedded real-time systems. These are mission critical systems that are often subject to both fault-tolerant and real-time constraints. To reduce the impact of transient faults, hardware protection mechanisms are usually proposed. However, these mechanisms introduce too much pessimism in the computation of the worst-case execution time of a task, decreasing the overall system performance.

We use static analysis tools to analyze a binary program and compute the overall vulnerability of its instructions. Then, we propose to reduce this vulnerability by invalidating some cache blocks at specific instants during the execution, thus forcing vulnerable instruction blocks to be reloaded from higher layers of memory. Since adding invalidation points will likely increase the WCETs of the tasks, we perform a static analysis to guarantee that the application deadlines are respected. Finally, we analyze how our methodology can be combined with hardware protection mechanisms as ECC memories, and we evaluate the performance on a set of benchmarks.

Participants: Giuseppe Lipari, Fabien Bouquillon.

7.7 Contention-free scheduling of PREM tasks on partitioned multicore platforms

This work is based on the thesis of Ikram Senoussaoui. She is supervised by Giuseppe Lipari, Kamel Benahoua (University of Oran 1) and Houssam Zahaf (University of Nantes).

Commercial-off-the-shelf (COTS) platforms feature several cores that share and contend for memory resources. In real-time system applications, it is of paramount importance to correctly estimate tight upper bounds to the delays due to memory contention. However, without proper support from the hardware (e.g. a real-time bus scheduler), it is difficult to estimate such upper bounds. [15] aims at avoiding contention for a set of tasks modeled using the Predictable Execution Model (PREM), i.e. each task execution is divided into a memory phase and a computation phase, on a hardware multicore architecture where each core has its private scratchpad memory and all cores share the main memory.

We consider non-preemptive scheduling for memory phases, whereas computation phases are scheduled using partitioned preemptive EDF. In this work, we propose three novel approaches to avoid contention in memory phases: (i) a task-level time-triggered approach, (ii) job-level time-triggered approach, and (iii) on-line scheduling approach. We compare the proposed approaches against the state of the art using a set of synthetic experiments in terms of schedulability and analysis time. Furthermore, we implemented the different approaches on an Infineon AURIX TC397 multicore microcontroller and validated the proposed approaches using a set of tasks extracted from well-known benchmarks from the literature.

An extension of this work has been presented at the EDIS conference [14], where tasks' allocation is imposed by other functional constraints.

Participants: Giuseppe Lipari, Ikram Senoussaoui.

7.8 Real-Time scheduling on FPGAs

The research described in this section has been carried out in collaboration with the RETIS Lab of the Scuola Superiore Sant'Anna in the context of the PhD thesis of Marco Pagani, co-directed by Giuseppe Lipari and Giorgio Buttazzo. The research concerns the real-time scheduling of tasks on dynamically reconfigurable FPGAs.

Dynamic partial reconfiguration allows virtualizing the FPGA resources by sharing them among multiple hardware accelerators over time. Although very promising, FPGA-based hardware acceleration also introduces new challenges, such as managing and scheduling multiple concurrent acceleration and reconfiguration requests.

In this research, the FRED-Linux library has been proposed to address these challenges while preserving the predictability required by real-time systems. Fred-Linux allows developing rich applications while leveraging predictable FPGA-based hardware acceleration for performing heavy computations. This research has been published in [7].

Participants: Giuseppe Lipari.

7.9 Scheduling for Networks on Chips

This work is an offspring of the thesis of Chawki Benchehida, who defended on October 2021, and was supervised by Giuseppe Lipari, Houssam Zahaf (Univ. Nantes), Kamel Benahoua (Univ. Oran 1).

Networks-on-Chips (NoC) provide a viable solutions to bus-contention problems in classical Multi/Many core architectures. However, NoC complex design requires particular attention to support the execution of real-time workloads. In fact, it is necessary to take into account task-to-core allocation and inter-task communication, so that all timing constraints are respected. The problem is more complex when considering task-to-main-memory communication, as the main memory is off-chip and usually connected to the network edges, within the 2D-Mesh topology, which generates a particular additional pattern of traffic.

In [5], we tackle these problems by considering the allocation of tasks and inter-task communications, and memory-to-task communications (modeled using Directed Acyclic Graphs DAGs) at the same time, rather than separating them, as it has been addressed in the literature of real-time systems. This problem is highly combinatorial, therefore our approach transforms it at each step, to a simpler problem until reaching the classical single-core scheduling problem. The goal is to find a trade-off between the problem combinatorial explosion and the loss of generality when simplifying the problem. We study the effectiveness of the proposed approaches using a large set of synthetic experiments.

7.10 Bunched Fuzz: Sensitivity for Vector Metrics

Program sensitivity measures the distance between the outputs of a program when run on two related inputs. This notion, which plays a key role in areas such as differential privacy and optimization, has been the focus of several program analysis techniques introduced in recent years. Among the most successful ones, we can highlight type systems inspired by linear logic, as pioneered by Reed and Pierce in the Fuzz functional programming language. In Fuzz, each type is equipped with its own notion of distance, and sensitivity analysis boils down to type checking. In particular, Fuzz features two product types, corresponding to two different notions of distance: the *tensor product* combines the distances of each component by *adding* them, while the *with product* takes their *maximum*.

In [12] with colleagues from Boston University we proposed an extension of Fuzz where product types are generalized to arbitrary L^p *distances*, metrics that are often used in privacy and optimization. The original Fuzz products, tensor and with, correspond to the special cases L^1 and L^∞. To support the handling of such products, we extend the Fuzz type system with *bunches*—as in the logic of bunched implications—where the distances of different groups of variables can be combined using different L^p distances. We show that our extension can be used to reason naturally about quantitative properties of probabilistic programs.

Participants: Patrick Baillot.

7.11 Defining corecursive functions in Coq using approximations

In [10] together with a colleague from the 2XS teams (CRIStAL, CNRS & Univ. Lille) we present two methods for defining corecursive functions that go beyond what is accepted by the builtin corecursion mechanisms of the Coq proof assistant. This gain in expressiveness is obtained by using a combination of axioms from Coq's standard library that, to our best knowledge, do not introduce inconsistencies but enable reasoning in standard mathematics. Both methods view corecursive functions as limits of sequences of approximations, and both are based on a property of productiveness that, intuitively, requires that for each input, an arbitrarily close approximation of the corresponding output is eventually obtained. The first method uses Coq's builtin corecursive mechanisms in a non-standard way, while the second method uses none of the mechanisms but redefines them. Both methods are implemented in Coq and are illustrated with examples.

Participants: Vlad Rusu.

7.12 Sufficiently complete partial orders: towards corecursion without corecursion in Coq

Coinduction is an important concept in functional programming. To formally prove properties of corecursive functions one can try to define them in a proof assistant such as Coq. But there are strong limitations on the corecursive functions that can be defined. In particular, one cannot freely mix corecursion with recursion. In this paper we introduce the notion of Sufficiently Complete Partial Order (SCPO) that enables us to transform a program's coinductive types by means of "sufficiently completing" inductive types, thereby escaping Coq's builtin limitations. We also show how corecursive functions can be defined on SCPOs without actually using corecursion, by adapting recent work [10] where we show how this can be done by means of the unique solution of a fixpoint equation. This work [19] was presented at the WPTE 2022 workshop Haïfa, Israel, August 2022, part of the FLOC federated conference. An extended version of this work be found in [18].

Participants: Vlad Rusu.

8 Dissemination

8.1 Promoting scientific activities

8.1.1 Scientific events

Chair of conference program committees Vlad Rusu was chair of the FROM 2022 conference (Working Formal Methods Symposium).

Patrick Baillot was co-chair of the LCC 2022 workshop (Logic and Computational Complexity).

Member of conference program committees

- Patrick Baillot has been in the program committee of FoSSaCS 2023 (PC);
- Julien Forget has been in the program committee of DATE 2022 (PC), RTNS 2022 (PC);

- Giuseppe Lipari has been in the program committee of RTNS 2022 (PC), ISPDC 2022 (PC), Microservices 2022 (PC), EDiS 2022 (PC);
- Raphaël Monat has been in the program committee of SAS 2022 (PC), SV-COMP 2023 (PC), as well as the artefact evaluation and external review committees of OOPSLA 2023 (AEC+ERC), ECOOP 2023 (AEC+ERC).

8.1.2 Journal

Member of the editorial boards

- Giuseppe Lipari is member of the editorial board of the Journal of System Architecture (Online ISSN: 1873-6165), and of the Real-Time Systems Journal (Online ISSN: 1573-1383).

8.1.3 Scientific expertise

Raphaël Monat is a member of the "Source code and software" subgroup from the French Committee for Open Science.

8.2 Supervision - Juries

8.2.1 Supervision

Leandro Moreira Gomes (postdoc) is supervised by Patrick Baillot. Fabien Bouquillon (PhD student until Oct. 2022) was supervised by Giuseppe Lipari. Frédéric Fort (PhD student until Oct. 2022) was supervised by Giuseppe Lipari and Julien Forget. Sandro Grebant (PhD student) is also supervised by Giuseppe Lipari and Julien Forget. Nordine Feddal (PhD student) and Ikram Senoussaoui (ATER) are supervised by Giuseppe Lipari. Andrei Florea (PhD student) is supervised by Julien Forget and Vlad Rusu.

8.2.2 Juries

- Giuseppe Lipari has been president of the PhD thesis committee of Guillame Fieni, who defended on Decembre 15th 2022 at INRIA Lille.

9 Scientific production

9.1 Major publications

[1] C. Ballabriga, J. Forget and J. Ruiz. 'Relational abstract interpretation of arrays in assembly code'. In: *Formal Methods in System Design* (2nd Oct. 2022). DOI: 10.1007/s10703-022-00399-3. URL: https://hal.inria.fr/hal-03794951.

[2] F. Bouquillon, S. Niar and G. Lipari. 'Reducing the fault vulnerability of hard real-time systems'. In: *Journal of Systems Architecture* 133 (Dec. 2022), p. 102758. DOI: 10.1016/j.sysarc.2022.102758. URL: https://hal.archives-ouvertes.fr/hal-03842393.

[3] F. Vanhems, V. Rusu, D. Nowak and G. Grimaud. 'A Formal Correctness Proof for an EDF Scheduler Implementation'. In: *Proc. 28th IEEE Real-Time and Embedded Technology and Applications Symposium*. RTAS 2022: 28th IEEE Real-Time and Embedded Technology and Applications Symposium. Milan, Italy, 4th May 2022. DOI: 10.1109/RTAS54340.2022.00030. URL: https://hal.inria.fr/hal-03671598.

9.2 Publications of the year

International journals

[4] C. Ballabriga, J. Forget and J. Ruiz. 'Relational abstract interpretation of arrays in assembly code'. In: *Formal Methods in System Design* (2nd Oct. 2022). DOI: 10.1007/s10703-022-00399-3. URL: https://hal.inria.fr/hal-03794951.

[5] C. Benchehida, M. Kamel Benhaoua, H. Zahaf and G. Lipari. 'Memory-processor co-scheduling for real-time tasks on network-on-chip manycore architectures'. In: *International Journal of High Performance Systems Architecture (IJHPSA)* 11.1 (31st Jan. 2022), pp. 1–11. DOI: 10.1504/IJHPSA.2022.121877. URL: https://hal-cnrs.archives-ouvertes.fr/hal-03595577.

[6] F. Bouquillon, S. Niar and G. Lipari. 'Reducing the fault vulnerability of hard real-time systems'. In: *Journal of Systems Architecture* 133 (Dec. 2022), p. 102758. DOI: 10.1016/j.sysarc.2022.102758. URL: https://hal.science/hal-03842393.

[7] M. Pagani, A. Biondi, M. Marinoni, L. Molinari, G. Lipari and G. Buttazzo. 'A Linux-Based Support for Developing Real-Time Applications on Heterogeneous Platforms with Dynamic FPGA Reconfiguration'. In: *Future Generation Computer Systems* 129 (2nd Apr. 2022), pp. 125–140. DOI: 10.1016/j.future.2021.11.007. URL: https://hal.science/hal-03512476.

International peer-reviewed conferences

[8] F. Fort and J. Forget. 'Synchronous semantics of multi-mode multi-periodic systems'. In: SAC '22: The 37th ACM/SIGAPP Symposium on Applied Computing. Virtual Event, France: ACM, 25th Apr. 2022, pp. 1248–1257. DOI: 10.1145/3477314.3507271. URL: https://hal.science/hal-03817684.

[9] G. Lipari, F. Bouquillon and S. Niar. 'Improving CRPD Analysis for EDF Scheduling: Trading Speed for Precision'. In: The 37th ACM/SIGAPP Symposium On Applied Computing. Brno, Czech Republic, 18th Jan. 2022. DOI: 10.1145/3477314.3507027. URL: https://hal.science/hal-03531143.

[10] V. Rusu and D. Nowak. 'Defining Corecursive Functions in Coq Using Approximations'. In: *Proceedings ECOOP 20222, 8-10 Jube 2022, Berlin, Germany*. ECOOP. Berlin, Germany, 8th June 2022. DOI: 10.4230/LIPIcs.ECOOP.2022.12. URL: https://hal.inria.fr/hal-03671876.

[11] F. Vanhems, V. Rusu, D. Nowak and G. Grimaud. 'A Formal Correctness Proof for an EDF Scheduler Implementation'. In: *Proc. 28th IEEE Real-Time and Embedded Technology and Applications Symposium*. RTAS 2022: 28th IEEE Real-Time and Embedded Technology and Applications Symposium. Milan, Italy, 4th May 2022. DOI: 10.1109/RTAS54340.2022.00030. URL: https://hal.inria.fr/hal-03671598.

[12] J. Wunder, A. A. D. Amorim, P. Baillot and M. Gaboardi. 'Bunched Fuzz: Sensitivity for Vector Metrics'. In: European Symposium on Programming (ESOP). Proceedings of ESOP 2023 (European Symposium on Programming). Paris, France: Springer, 24th Apr. 2023. DOI: 10.48550/arXiv.2202.01901. URL: https://hal.science/hal-03870966.

National peer-reviewed Conferences

[13] M. Valnet, R. Monat and A. Miné. 'Analyse statique de valeurs par interprétation abstraite de programmes fonctionnels manipulant des types algébriques récursifs'. In: *Journées Francophones des Langages Applicatifs*. JFLA 2023 - 34èmes Journées Francophones des Langages Applicatifs. Praz-sur-Arly, France, 16th Jan. 2023, pp. 211–242. URL: https://hal.inria.fr/hal-03936718.

Conferences without proceedings

[14] I. Senoussaoui, M. K. Benhaoua, H.-E. Zahaf and G. Lipari. 'Toward memory-centric scheduling for PREM task on multicore platforms, when processor assignments are specified'. In: EDiS 2022 - 3rd International Conference on Embedded & Distributed Systems. Oran, France: IEEE, 2nd Nov. 2022, pp. 11–15. DOI: 10.1109/EDiS57230.2022.9996534. URL: https://hal.science/hal-03938665.

[15] I. Senoussaoui, H.-E. Zahaf, G. Lipari and K. Benhaoua. 'Contention-free scheduling of PREM tasks on partitioned multicore platforms'. In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). Stuttgart, Germany, 6th Sept. 2022. URL: https://hal.science/hal-03834177.

Doctoral dissertations and habilitation theses

[16] F. Bouquillon. 'Improving the reliability of heterogeneous multicore architecture for intelligent transportation systems'. Université de Lille, 18th Oct. 2022. URL: https://hal.science/tel-03895529.

[17] F. Fort. 'Programing adaptive real-time systems'. Université de Lille, 4th Oct. 2022. URL: https://hal.science/tel-03948472.

Reports & preprints

[18] V. Rusu and D. Nowak. *Towards Corecursion Without Corecursion in Coq*. 2022. URL: https://hal.inria.fr/hal-03689027.

Other scientific publications

[19] V. Rusu and D. Nowak. *Sufficiently Complete Partial Orders: Towards Corecursion Without Corecursion in Coq*. Aug. 2022. URL: https://hal.inria.fr/hal-03936127.

9.3 Cited publications

[20] L. Abeni, L. Palopoli, C. Scordino and G. Lipari. 'Resource Reservations for General Purpose Applications'. In: *IEEE Trans. Ind. Informatics* 5.1 (2009), pp. 12–21. DOI: 10.1109/TII.2009.2013633. URL: https://doi.org/10.1109/TII.2009.2013633.

[21] ANSYS. *SCADE Suite*. http://www.esterel-technologies.com/products/scade-suite/. 2018.

[22] C. Ballabriga, J. Forget, L. Gonnord, G. Lipari and J. Ruiz. 'Static Analysis Of Binary Code With Memory Indirections Using Polyhedra'. In: *International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'19)*. Lisbon, Portugal, Jan. 2019. URL: https://hal.archives-ouvertes.fr/hal-01939659.

[23] C. Ballabriga, J. Forget and G. Lipari. 'Symbolic WCET Computation'. In: *ACM Transactions on Embedded Computing Systems (TECS)* 17.2 (Dec. 2017), pp. 1–26. DOI: 10.1145/3147413. URL: https://hal.archives-ouvertes.fr/hal-01665076.

[24] S. Baruah. 'Resource-Efficient Execution of Conditional Parallel Real-Time Tasks'. In: *Euro-Par 2018: Parallel Processing*. Cham: Springer International Publishing, 2018, pp. 218–231.

[25] E. Bini and G. Buttazzo. 'The space of EDF deadlines: the exact region and a convex approximation'. In: *Real-Time Systems* 41.1 (2009), pp. 27–51.

[26] E. Bini, M. Di Natale and G. Buttazzo. 'Sensitivity analysis for fixed-priority real-time systems'. In: *Real-Time Systems* 39.1-3 (2008), pp. 5–30.

[27] T. Bourke, L. Brun, P.-É. Dagand, X. Leroy, M. Pouzet and L. Rieg. 'A formally verified compiler for Lustre'. In: *ACM SIGPLAN Notices*. Vol. 52. 6. ACM. 2017, pp. 586–601.

[28] R. Boutonnet and N. Halbwachs. 'Disjunctive relational abstract interpretation for interprocedural program analysis'. In: *International Conference on Verification, Model Checking, and Abstract Interpretation*. Springer. 2019, pp. 136–159.

[29] G. Durrieu, M. Faugere, S. Girbal, D. G. Pérez, C. Pagetti and W. Puffitsch. 'Predictable flight management system implementation on a multicore processor'. In: *Embedded Real Time Software (ERTS'14)*. 2014.

[30] A. Hamann, D. Dasari, S. Kramer, M. Pressler and F. Wurst. 'Communication Centric Design in Complex Automotive Embedded Systems'. In: *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*. Dubrovnik, Croatia, 2017.

[31] X. Leroy. *The CompCert C verified compiler*. http://compcert.inria.fr. 2018.

[32] G. Lipari and S. K. Baruah. 'Greedy reclamation of unused bandwidth in constant-bandwidth servers'. In: *12th Euromicro Conference on Real-Time Systems (ECRTS 2000), 19-21 June 2000, Stockholm, Sweden, Proceedings*. IEEE Computer Society, 2000, pp. 193–200. DOI: 10.1109/EMRTS.2000.854007. URL: https://doi.org/10.1109/EMRTS.2000.854007.

[33] G. Lipari and E. Bini. 'Resource Partitioning among Real-Time Applications'. In: *Proc. 15th Euromicro Conf. Real-Time Systems*. IEEE Computer Society, 2003, pp. 151–158. DOI: 10.1109/EMRTS.2003.1212738.

[34] MathWorks. *Simulink*. https://www.mathworks.com/products/simulink.html. 2018.

[35] C. Pagetti, J. Forget, F. Boniol, M. Cordovilla and D. Lesens. 'Multi-task implementation of multi-periodic synchronous programs'. In: *Discrete Event Dynamic Systems* 21.3 (2011), pp. 307–338.

[36] C. Pagetti, J. Forget, H. Falk, D. Oehlert and A. Luppold. 'Automated generation of time-predictable executables on multi-core'. In: *RTNS 2018*. Oct. 2018.

[37] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo and R. Kegley. 'A predictable execution model for COTS-based embedded systems'. In: *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. 2011.

[38] P. Raymond. 'A general approach for expressing infeasibility in implicit path enumeration technique'. In: *2014 International Conference on Embedded Software (EMSOFT)*. IEEE. 2014, pp. 1–9.

[39] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu and C. D. Gill. 'Parallel Real-Time Scheduling of DAGs'. In: *IEEE Transactions on Parallel and Distributed Systems* 25.12 (Dec. 2014), pp. 3242–3252. DOI: 10.1109/TPDS.2013.2297919.

[40] *The Coq proof assistant reference manual*. http://coq.inria.fr. 2021.

[41] *The Pip protokernel*. http://pip.univ-lille1.fr/. 2018.