RESEARCH CENTRE

Inria Centre at the University of Lille

IN PARTNERSHIP WITH:

Université de Lille, CNRS

2023

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, HDR]
- Raphaël Monat [INRIA, Researcher]
- Vlad Rusu [INRIA, Researcher, HDR]

Faculty Members

- Giuseppe Lipari [Team leader, UNIV LILLE, Professor, HDR]
- Clément Ballabriga [UNIV LILLE, Associate Professor, until Sep 2023, Currently on leave]
- Julien Forget [UNIV LILLE, Associate Professor, HDR]

Post-Doctoral Fellow

- Leandro Moreira Gomes [UNIV LILLE, from Sep 2023, ATER]

PhD Students

- Chiara Daini [INRIA, from Jul 2023]
- Nordine Feddal [INRIA]
- Sandro Grebant [UNIV LILLE, until Nov 2023]
- Victor Sannier [UNIV LILLE, from Sep 2023]
- Ikram Senoussaoui [UNIV LILLE, ATER]

Interns and Apprentices

- Pierre Goutagny [ENS DE LYON, Intern, from Oct 2023]

Administrative Assistant

- Nathalie Bonte [INRIA]

External Collaborator

- Aymeric Fromherz [Inria Paris]

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;

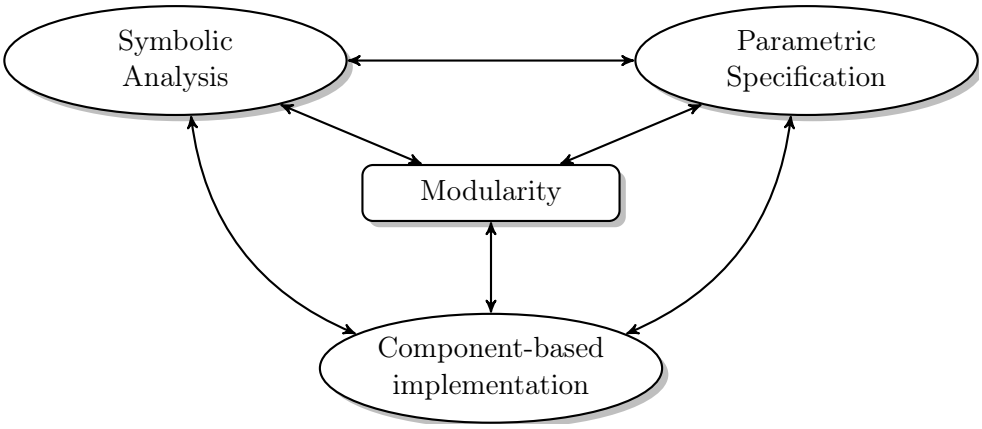

Figure 1: Ontology of terms in SYCOMORES.

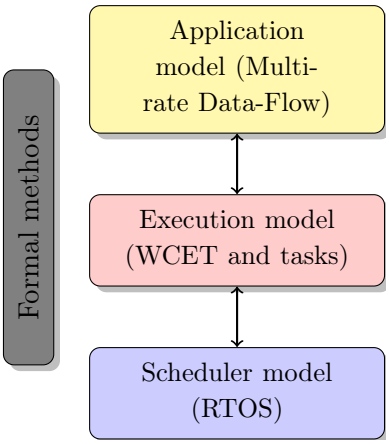

Figure 2: Abstraction Layers.

- *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.

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 [15], SIMULINK [28], or in our case PRELUDE [29], 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 [34], 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 [29] 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 [33, 18]. 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. [20, 19] 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 [31, 23]. 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 [24, 30], 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 [17] 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 [16]. 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 [32]. 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 [16] to support inter-procedural analysis, based on function *summaries* [22], 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 [17].

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 [16] 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 [35] 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 [14, 26], 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 [27] 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 [25] (from C code to binary code) and the LUSTRE verified compiler [21]. 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 Social and environmental responsibility

Raphaël Monat takes part in the gender-equality commission of the lab, and attended some meetings of the sustainable development group.

6 Highlights of the year

Julien Forget defended his Habilitation à Diriger les Recherches on "Programming and analysis of critical real-time systems", June 12, 2023.

6.1 Awards

In Fall 2023, Raphaël Monat participated to the Software Verification Competition (SV-Comp) by submitting the Mopsa static analyzer he is co-developing. Mopsa earned a gold medal in the SoftwareSystems category. There were 22 competing tools in this category.

6.2 PhD graduation

Two PhD students of the team graduated in 2023.

Sandro Grebant defended his thesis titled "Efficient tree-based symbolic WCET computation" on the 7th of November 2023 at the University of Lille. Sandro was co-supervised by Julien Forget and Giuseppe Lipari

Ikram Senoussaoui defended her thesis titled "Processor and memory co-scheduling of embedded real-time applications on multicore platforms" on the 14th of Decembre 2023 at the University of Lille. She was co-supervised by Giuseppe Lipari, Benahoua Kamel from the University of Oran 1, and Houssam-Eddine Zahaf from the University of Nantes.

7 New software, platforms, open data

7.1 New software

7.1.1 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: Clement Ballabriga

Partner: Université de Toulouse

7.1.2 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

7.1.3 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

7.1.4 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.

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

Publications: hal-03712130, hal-03781578, hal-03128248, hal-03159939, hal-02936606, hal-03869335

Contact: Denis Merigoux

Participants: Denis Merigoux, Louis Gesbert, Aymeric Fromherz, Alain Delaet–Tixeuil, Raphael Monat

Partner: Université Panthéon-Sorbonne

7.1.5 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

7.1.6 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

7.1.7 Haddock

Keywords: Partial function, (Co)Recursive Function, Coq

Functional Description: A Coq library for defining and reasoning about partial (co)recursive functions.

URL: https://github.com/vladmgrusu/haddock

Contact: Vlad Rusu

Partners: Université de Lille, CNRS, Université de Bucarest

8 New results

8.1 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

In Fall 2023, Raphaël Monat participated to the Software Verification Competition (SV-Comp) by submitting the Mopsa static analyzer he is co-developing. Mopsa earned a gold medal in the SoftwareSystems category; this category aims at "representing verification tasks from real software systems". There were 22 competing tools.

This is our second participation, our first participation is described in [5].

Participants: Raphaël Monat.

8.2 Contention-free scheduling of AECR dag 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.

We aim at avoiding contention for a set of tasks on a hardware multicore architecture, where each core has its private scratchpad memory and all cores share access to the main memory. By avoiding contention, we make multicore programming predictable again.

We propose a new task model, AECR-dag (Acquisition, Execution, Communication Restitution – Directed Acyclice Graph), where a periodic or sporadic task is models by a set of subtask nodes and communication between then, organised as a DAG. The model captures the parallelism of the task and it makes explicit the communication between the subtasks.

Then, we propose a methodology based on Integer Linear Programming, for allocating the subtasks on the cores using partitioned preemptive EDF, so to reduce the cost of communication between subtasks allocated to different cores. Then, we propose a method based on Evolutionary Programming, that allocates the intermediated deadlines to the subtasks so to respect the precedence constraints and the end-to-end deadline.

This work has been presented in the last chapter of the thesis of Ikram Senoussaoui. It has also been submitted to the Journal of Systems Architectures where it is now under major revision.

Participants: Giuseppe Lipari, Ikram Senoussaoui.

8.3 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 [10] 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.

8.4 Formal Definitions and Proofs for Partial (Co)Recursive Functions

Participants: Vlad Rusu.

Partial functions are a key concept in programming. Without partiality a programming language has limited expressiveness - it is not Turing-complete, hence, some programs cannot be written. In *functional* programming languages, partiality mostly originates from the non-termination of recursive functions. *Corecursive* functions are another source of partiality: here, the issue is not non-termination, but the inability to produce arbitrary large, finite approximations of a theoretically infinite output.

Partial functions have been formally studied in the branch of theoretical computer science called *domain theory*. In this paper we propose to step up the level of formality by using the Coq proof assistant. The main difficulty is that Coq requires all functions to be total, since partiality would break the soundness of its underlying logic. In [12] (currently submitted to a journal) we propose practical solutions for this issue, as well as other issues that appear when one attempts to define and reason about partial (co)recursive functions in a total functional language such as that of the Coq proof assistant.

Our solutions have been implemented in an open-source Coq library 7.1.7. We present several examples of partial recursive and corecursive functions defined with our approach. For the functions that are actually total, a coinductive proof technique enables users to prove the totality.

8.5 While Loops in Coq

Participants: Vlad Rusu.

While loops are present in virtually all imperative programming languages. They are important both for practical reasons (performing a number of iterations not known in advance) and theoretical reasons (achieving Turing completeness). In [9] we propose an approach for incorporating while loops in an imperative language shallowly embedded in the Coq proof assistant. The main difficulty is that proving the termination of while loops is nontrivial, or impossible in the case of nontermination, whereas Coq only accepts programs endowed with termination proofs. Our solution is based on a new, general method for defining possibly non-terminating recursive functions in Coq. We illustrate the approach by proving termination and partial correctness of a program on linked lists. The development is available here.

8.6 WCET analysis with procedure arguments as parameters

In [8] we present a methodology for obtaining the WCET of a program procedure as a function of the its arguments. The methodology extends a previous method for symbolic WCET to the case of conditional statements (if-then-else). Then, a new technique based on abstract interpretation has been developed to 1) detect the arguments in the function code, 2) compute a symbolic expression of the WCET based on the arguments. Finally, a compiler generates efficient C-code for evaluating the formula at run-time, thus enabling the use for this technique for adaptive scheduling. The resulting methodology has been tested on several benchmarks from the literature, proving its efficiency and tightness.

This work is part of the thesis of Sandro Grebant. He was supervised by Julien Forget and Giuseppe Lipari, and he defended succesfully in November 2023.

Participants: Sandro Grebant, Julien Forget, Clément Ballabriga, Giuseppe Lipari.

8.7 Static Analysis of Functional Programs Handling Recursive Algebraic Data Types

In [11], 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.

9 Partnerships and cooperations

9.1 International research visitors

Horatiu Cheval

Status: PhD student

Institution of origin: University of Bucharest

Country: Romania

Dates: 4-10 Sept. 2023

Context of the visit: During Horatiu's visit we initiated the collaboration that resulted in the (currently, submitted) paper [12].

Mobility program/type of mobility: funded by the EuroProofNet COST action

Participants: Vlad Rusu.

9.2 European initiatives

9.2.1 Other european programs/initiatives

COST Action EuroProofNet: all relevant information is here. This action funded the visit of Horatiu Cheval, PhD student at Univ. Bucharest, Romania.

9.3 National initiatives

Raphaël Monat is co-PI of an Inria Exploratory Action called AVoCat, aiming at exploring Automatic Verification of Catala programs. The project is shared with Aymeric Fromherz at Inria Paris.

Participants: Raphaël Monat.

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

Chair of conference program committees Julien Forget has been co-chair of the ECRTS'23 Artifact Evaluation committee.

Member of the conference program committees Patrick Baillot has been member of the PC of the conference FoSSaCS 2023 and since 2023 of the steering committee of the conference FSCD.

Julien Forget has been PC member for ECRTS'23 and DATE'23.

Raphaël Monat has been a PC member of SOAP'23, SV-Comp 2024 and JFLA 2024.

Vlad Rusu had been PC member for FROM 2023 and Tacas 2024.

10.1.2 Journal

Member of the editorial boards Giuseppe Lipari is member of the editorial board of Real-Time System Journal.

10.1.3 Invited talks

Vlad Rusu gave an invited talk at the tenth congress of Romanian mathematicians (July 2023, Pitesti, Romania).

Giuseppe Lipari gave an invited talk "Parametric WCET" at the University of Nantes during his visit on September 29-29.

10.1.4 Scientific expertise

From october to december 2023, Giuseppe Lipari has served as member of the HCERES commitee for the evaluation of the LIP6 laboratory of the Sorbonne University in Paris.

10.1.5 Research administration

Patrick Baillot is Délégué Scientifique at CNRS Sciences Informatiques, since September 2022 (40% of his work time).

10.1.6 Hiring committees

Raphaël Monat has been a jury for oral examinations to enter Écoles Normales Supérieures.
Vlad Rusu was a member of the hiring committee of researchers for Inria Lille (CRCN/ISFP).

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

Giuseppe Lipari is responsible for the "IoT and Cybersecurity" specialty of the Master in Computer Science of the University of Lille. The Masters' courses of this specialty are taught in English. The specialty is also part of the Graduate Programme "Information and Knowledge Society" of the University of Lille.

Giuseppe Lipari has taught the following courses at the University of Lille in the academic year 2022-2023:

- *Programmation de Systemes* (Resp., L3, 3 ECTS, 180 students) and *Programmation de Systemes +* (Resp., L3, 3 ECTS, 80 students)
- *Embedded System Design* (Resp., M1, 6 ECTS, 25 students)
- *Operating System 1* (M1, 3 ECTS, 25 students)
- *Operating Systems 2* (Resp., M1, 3 ECTS, 75 students)
- *Advanced Object Oriented Programming* (Resp., M1, 3 ECTS, 75 students)
- *Advanced Operating Systems* (Resp., M2, 3 ECTS, 30 students)
- *Real-Time Systems* (Resp., M2, 3 ECTS, 15 students)

Moreover, Giuseppe Lipari has supervised several internships and memoirs of M2, for a total service of 338.5 hours (eqTD) in 2022-2023.

Julien Forget has taught the following courses at the University of Lille in the academic year 2022-2023:

- *Structured programming* (Resp., Ing3, 48h, 50 students)
- *Operating Systems* (Ing3, 16h, 12 students)
- *Operating Systems* (Resp., Ing3, 16h, 50 students)×2
- *Object-oriented Programming* (Ing4, 10h, 12 students)
- *Theoretical Computing* (Resp., Ing4, 22h, 24 students)
- *Data structures* (Resp., Ing3, 12h, 24 students)

- *Programming project* (Ing3, 22h, 12 students)
- *An Introduction to Research* (Ing4, 6h, 50 students)

Overall, the teaching activities of Julien Forget amount to a total service of 245 hours (eqTD) in 2022-2023.

Vlad Rusu has taught a cours on *Formal Methods for Embedded Systems* (M2) at the "Internet of Things and Cybersecurity" specialty of the Master in Computer Science at the University of Lille (M2, 12 hours)

Vlad Rusu gave a one-day course on introduction to the Coq proof assistant at an autumn school (September 2024, Bucharest, Romania).

Raphaël Monat has taught a course on "Compilation des Logiciels" at the Master in Computer Science of the University of Lille (M1, 3 ECTS, 24 hours).

10.2.2 Supervision

Master2 internship of Victor Sannier, april-july 2023, supervised by Patrick Baillot.

10.2.3 Juries

Julien Forget was examiner for the defense of Basile Pesin's Ph.D. (PSL/ENS) on "Verified compilation of a synchronous dataflow language with state machines".

Julien Forget was examiner for the defense of Rémi Meunier's Ph.D. (U. Toulouse) on "Prédiction du temps d'exécution d'applications dans des architectures multi-cœur".

Giuseppe Lipari was reviewer of the thesis of Gabriele Ara at Scuola Superiore Sant'Anna of Pise, whose defense was succesfully completed on the 16th of June 2023. He was also president of the jury for Clément Boin at the University of Lille on the 18th of December 2023. He was also president of the jury of Lorenzo Canonne at the University of Lille on the 19th of December 2023.

11 Scientific production

11.1 Major publications

[1] S. Grebant, C. Ballabriga, J. Forget and G. Lipari. 'WCET analysis with procedure arguments as parameters'. In: RTNS 2023: The 31st International Conference on Real-Time Networks and Systems. Dortmund, Germany: ACM, 2023, pp. 11–22. DOI: 10.1145/3575757.3593655. URL: https://hal.science/hal-04118213.

[2] D. Nowak and V. Rusu. 'While Loops in Coq'. In: *Electronic Proceedings in Theoretical Computer Science*. 7th Symposium on Working Formal Methods (FROM 2023). Vol. 389. Bucarest, Romania, 22nd Sept. 2023, pp. 96–109. DOI: 10.4204/eptcs.389.8. URL: https://inria.hal.science/hal-04254872.

[3] J. Wunder, A. A. D. Amorim, P. Baillot and M. Gaboardi. 'Bunched Fuzz: Sensitivity for Vector Metrics'. In: ESOP 2023 - European Symposium on Programming. 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.

11.2 Publications of the year

International journals

[4] S. Altmeyer, É. André, S. Dal Zilio, L. Fejoz, M. G. Harbour, S. Graf, J. J. Gutiérrez, R. Henia, D. Le Botlan, G. Lipari, J. Medina, N. Navet, S. Quinton, J. Rivas and Y. Sun. 'From FMTV to WATERS: Lessons Learned from the First Verification Challenge at ECRTS (Artifact)'. In: *Dagstuhl Artifacts Series* 9.1 (3rd July 2023). DOI: 10.4230/DARTS.9.1.4. URL: https://hal.science/hal-04254710.

Invited conferences

[5] R. Monat, A. Ouadjaout and A. Miné. 'Mopsa-C: Modular Domains and Relational Abstract Interpretation for C Programs (Competition Contribution)'. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2023). Vol. 13994. Lecture Notes in Computer Science. Paris, France: Springer, Cham, 20th Apr. 2023, pp. 565–570. DOI: 10.1007/978-3-031-30820-8_37. URL: https://inria.hal.science/hal-04077678.

International peer-reviewed conferences

[6] P. Baillot, U. Dal Lago, C. Kop and D. Vale. 'On Basic Feasible Functionals and the Interpretation Method'. In: *Lecture Notes in Computer Science*. International Conference on Foundations of Software Science and Computation Structures. Luxembourg City, Luxembourg: Springer, 8th Apr. 2024. URL: https://hal.science/hal-04376613.

[7] C. Ballabriga, J. Forget, S. Grebant and G. Lipari. 'New challenges in adaptive real-time systems with parametric WCET'. In: RTSOPS 2023 - 12th International Real-Time Scheduling Open Problems Seminar. Vienne, Austria, 11th July 2023. URL: https://hal.science/hal-04197411.

[8] S. Grebant, C. Ballabriga, J. Forget and G. Lipari. 'WCET analysis with procedure arguments as parameters'. In: RTNS 2023: The 31st International Conference on Real-Time Networks and Systems. Dortmund, Germany: ACM, 2023, pp. 11–22. DOI: 10.1145/3575757.3593655. URL: https://hal.science/hal-04118213.

[9] D. Nowak and V. Rusu. 'While Loops in Coq'. In: *Electronic Proceedings in Theoretical Computer Science*. 7th Symposium on Working Formal Methods (FROM 2023). Vol. 389. Bucarest, Romania, 22nd Sept. 2023, pp. 96–109. DOI: 10.4204/eptcs.389.8. URL: https://inria.hal.science/hal-04254872.

[10] J. Wunder, A. A. D. Amorim, P. Baillot and M. Gaboardi. 'Bunched Fuzz: Sensitivity for Vector Metrics'. In: ESOP 2023 - European Symposium on Programming. 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

[11] 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://inria.hal.science/hal-03936718.

Reports & preprints

[12] H. Cheval, D. Nowak and V. Rusu. *Formal Definitions and Proofs for Partial (Co)Recursive Functions*. 21st Dec. 2023. URL: https://inria.hal.science/hal-04360660.

[13] L. Gomes, P. Baillot and M. Gaboardi. *BiGKAT: an algebraic framework for relational verification of probabilistic programs*. 6th Mar. 2023. URL: https://hal.science/hal-04017128.

11.3 Cited publications

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

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

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