IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2017

Project-Team PARKAS

Parallélisme de Kahn Synchrone

IN COLLABORATION WITH: Département d'Informatique de l'Ecole Normale Supérieure

RESEARCH CENTER
Paris

THEME
Embedded and Real-time Systems

Table of contents

1. Personnel **1**
2. Overall Objectives **2**
3. Research Program **2**
3.1. Programming Languages for Cyber-Physical Systems 2
3.2. Efficient Compilation for Parallel and Distributed Computing 3
3.3. Validation and Proof of Compilers 4
3.3.1. Lustre: 4
3.3.2. C/C++: 4
3.3.3. Static Analysis of x10 5
3.3.4. Toward a Polynomial Model 5
4. Highlights of the Year **5**
5. New Software and Platforms **5**
5.1. Cmmtest 5
5.2. GCC 6
5.3. Heptagon 6
5.4. isl 6
5.5. Lem 6
5.6. Lucid Synchrone 7
5.7. Lucy-n 7
5.8. Ott 7
5.9. PPCG 8
5.10. ReactiveML 8
5.11. SundialsML 8
5.12. Zelus 9
6. New Results **9**
6.1. Compiler Optimisations for Multicore Architectures 9
6.2. Julia Subtyping Reconstructed 9
6.3. Comparing Designs for Gradual Types 9
6.4. Symbolic Simulation for a timed-automaton subset of Zélus 10
6.5. Verified compilation of Lustre 10
6.6. Zélus: Synchronous Languages + Ordinary Differential Equations 11
6.7. Compiling synchronous languages for multi-processor implementations 12
7. Bilateral Contracts and Grants with Industry **12**
8. Partnerships and Cooperations **12**
8.1. National Initiatives 12
8.1.1. ANR 12
8.1.2. Investissements d'avenir 12
8.1.3. Others 12
8.2. European Initiatives 13
8.2.1.1. Eurolab-4-HPC 13
8.2.1.2. TETRACOM 13
8.2.1.3. EMC2 13
8.3. International Initiatives 14
8.3.1. Inria Associate Teams Not Involved in an Inria International Labs 14
8.3.2. Participation in Other International Programs 15
8.4. International Research Visitors 15
8.4.1. Visits of International Scientists 15
8.4.2. Visits to International Teams 15
9. Dissemination **15**

9.1. Promoting Scientific Activities 15
9.1.1. Scientific Events Organisation 15
9.1.2. Scientific Events Selection 15
9.1.3. Journal 15
9.1.3.1. Member of the Editorial Boards 15
9.1.3.2. Reviewer - Reviewing Activities 15
9.1.4. Invited Talks 16
9.1.5. Leadership within the Scientific Community 16
9.1.6. Scientific Expertise 16
9.2. Teaching - Supervision - Juries 16
9.2.1. Teaching 16
9.2.2. Supervision 16
9.2.3. Juries 17
10. Bibliography **17**

Project-Team PARKAS

Creation of the Team: 2011 April 01, updated into Project-Team: 2012 January 01

Keywords:

Computer Science and Digital Science:

A1.1.1. - Multicore, Manycore
A1.1.3. - Memory models
A2.1.1. - Semantics of programming languages
A2.1.3. - Functional programming
A2.1.6. - Concurrent programming
A2.1.8. - Synchronous languages
A2.2.2. - Memory models
A2.2.3. - Run-time systems
A2.2.4. - Parallel architectures
A2.2.5. - GPGPU, FPGA, etc.
A2.2.6. - Adaptive compilation
A2.3. - Embedded and cyber-physical systems
A2.3.1. - Embedded systems
A2.3.2. - Cyber-physical systems
A2.3.3. - Real-time systems
A2.4.3. - Proofs
A3.1.3. - Distributed data
A3.1.8. - Big data (production, storage, transfer)
A6.2.1. - Numerical analysis of PDE and ODE
A6.2.7. - High performance computing

Other Research Topics and Application Domains:

B5.2.1. - Road vehicles
B5.2.2. - Railway
B5.2.3. - Aviation
B6.4. - Internet of things
B6.6. - Embedded systems
B9.2.1. - Music, sound
B9.4.1. - Computer science
B9.4.2. - Mathematics

1. Personnel

Research Scientists

Timothy Bourke [Inria, Starting Research Position]
Albert Cohen [Inria, Senior Researcher, HDR]
Francesco Zappa Nardelli [Inria, Senior Researcher, HDR]

Faculty Member

Marc Pouzet [Team leader, Univ Pierre et Marie Curie, Professor]

Post-Doctoral Fellow

Guillaume Iooss [Ecole Normale Supérieure Paris]

PhD Students

Guillaume Baudart [Ecole Normale Supérieure Paris, until Mar 2017]
Ulysse Beaugnon [Ecole Normale Supérieure Paris]
Lelio Brun [Ecole Normale Supérieure Paris]
Robin Morisset [Inria, until Feb 2017]
Chandan Reddy Gopal [Inria]
Jie Zhao [National Digital Switching System Engineering and Technological Research Center, Zhengzhou, China]

Technical staff

Michael Kruse [Inria]
Oleksandr Zinenko [Inria]

Administrative Assistant

Christine Anocq [Inria]

External Collaborator

Paul Feautrier [Univ de Lyon]

2. Overall Objectives

2.1. Overall Objectives

The research in PARKAS focuses on the design, semantics, and compilation of programming languages which allow going from parallel deterministic specifications to target embedded code executing on sequential or multi-core architectures. We are driven by the ideal of a mathematical and executable language used both to program and simulate a wide variety of systems, including real-time embedded controllers in interaction with a physical environment (e.g., fly-by-wire, engine control), computationally intensive applications (e.g., video), and compilers that produce provably correct and efficient code.

The team bases its research on the foundational work of Gilles Kahn on the semantics of deterministic parallelism, the theory and practice of synchronous languages and typed functional languages, synchronous circuits, modern (polyhedral) compilation, and formal models to prove the correctness of low level code running on weak-memory processors.

To realize our research program, we develop languages (LUCID SYNCHRONE, REACTIVEML, LUCY-N, ZELUS), compilers (PPCG), contributions to open-source projects (isl, LLVM, gcc), tools to study language semantics (Ott) and to test optimization compilers in the presence of threads (cmmtest), and formalizations in Interactive Theorem Provers of language semantics (VÃ©lus, *n*-synchrony, quasi-synchrony). These software projects constitute essential "laboratories": they ground our scientific contributions, guide and validate our research through experimentation, and are an important vehicle for mutually beneficial and long standing collaborations with industry.

3. Research Program

3.1. Programming Languages for Cyber-Physical Systems

We study the definition of languages for reactive and Cyber-Physical Systems in which distributed control software interacts closely with physical devices. We focus on languages that mix discrete-time and continuous-time; in particular, the combination of synchronous programming constructs with differential equations, relaxed models of synchrony for distributed systems communicating via periodic sampling or through buffers, and the embedding of synchronous features in a general purpose ML language.

The synchronous language SCADE, [1] based on synchronous languages principles, is ideal for programming embedded software and is used routinely in the most critical applications. But embedded design also involves modeling the control software together with its environment made of physical devices that are traditionally defined by differential equations that evolve on a continuous-time basis and approximated with a numerical solver. Furthermore, compilation usually produces single-loop code, but implementations increasingly involve multiple and multi-core processors communicating via buffers and shared-memory.

The major player in embedded design for cyber-physical systems is undoubtedly SIMULINK, [2] with MODELICA[3] a new player. Models created in these tools are used not only for simulation, but also for test-case generation, formal verification, and translation to embedded code. That said, many foundational and practical aspects are not well-treated by existing theory (for instance, hybrid automata), and current tools. In particular, features that mix discrete and continuous time often suffer from inadequacies and bugs. This results in a broken development chain: for the most critical applications, the model of the controller must be reprogrammed into either sequential or synchronous code, and properties verified on the source model have to be reverified on the target code. There is also the question of how much confidence can be placed in the code used for simulation.

We attack these issues through the development of the ZELUS research prototype, industrial collaborations with the SCADE team at ANSYS/Esterel-Technologies, and collaboration with Modelica developers at Dassault-Systèmes and the Modelica association. Our approach is to develop a *conservative extension* of a synchronous language capable of expressing in a single source text a model of the control software and its physical environment, to simulate the whole using off-the-shelf numerical solvers, and to generate target embedded code. Our goal is to increase faithfulness and confidence in both what is actually executed on platforms and what is simulated. The goal of building a language on a strong mathematical basis for hybrid systems is shared with the Ptolemy project at UC Berkeley; our approach is distinguished by building our language on a synchronous semantics, reusing and extending classical synchronous compilation techniques.

Adding continuous time to a synchronous language gives a richer programming model where reactive controllers can be specified in idealized physical time. An example is the so called quasi-periodic architecture studied by Caspi, where independent processors execute periodically and communicate by sampling. We have applied ZELUS to model a class of quasi-periodic protocols and to analyze an abstraction proposed for model-checking such systems.

Communication-by-sampling is suitable for control applications where value timeliness is paramount and lost or duplicate values tolerable, but other applications—for instance, those involving video streams—seek a different trade-off through the use of bounded buffers between processes. We developed the *n*-synchronous model and the programming language LUCY-N to treat this issue.

3.2. Efficient Compilation for Parallel and Distributed Computing

We develop compilation techniques for sequential and multi-core processors, and efficient parallel run-time systems for computationally intensive real-time applications (e.g., video and streaming). We study the generation of parallel code from synchronous programs, compilation techniques based on the polyhedral model, and the exploitation of synchronous Single Static Assignment (SSA) representations in general purpose compilers.

We consider distribution and parallelism as two distinct concepts.

- Distribution refers to the construction of multiple programs which are dedicated to run on specific computing devices. When an application is designed for, or adapted to, an embedded multiprocessor, the distribution task grants fine grained—design- or compilation-time—control over the mapping and interaction between the multiple programs.

[1] http://www.esterel-technologies.com/products/scade-suite
[2] http://www.mathworks.com/products/simulink
[3] https://www.modelica.org

- Parallelism is about generating code capable of efficiently exploiting multiprocessors. Typically this amounts to maing (in)dependence properties, data transfers, atomicity and isolation explicit. Compiling parallelism translates these properties into low-level synchronization and communication primitives and/or onto a runtime system.

We also see a strong relation between the foundations of synchronous languages and the design of compiler intermediate representations for concurrent programs. These representations are essential to the construction of compilers enabling the optimization of parallel programs and the management of massively parallel resources. Polyhedral compilation is one of the most popular research avenues in this area. Indirectly, the design of intermediate representations also triggers exciting research on dedicated runtime systems supporting parallel constructs. We are particularly interested in the implementation of non-blocking dynamic schedulers interacting with decoupled, deterministic communication channels to hide communication latency and optimize local memory usage.

While distribution and parallelism issues arise in all areas of computing, our programming language perspective pushes us to consider four scenarios:

1. designing an embedded system, both hardware and software, and codesign;
2. programming existing embedded hardware with functional and behavioral constraints;
3. programming and compiling for a general-purpose or high-performance, best-effort system;
4. programming large scale distributed, I/O-dominated and data-centric systems.

We work on a multitude of research experiments, algorithms and prototypes related to one or more of these scenarios. Our main efforts focused on extending the code generation algorithms for synchronous languages and on the development of more scalable and widely applicable polyhedral compilation methods.

3.3. Validation and Proof of Compilers

Compilers are complex software and not immune from bugs. We work on validation and proof tools for compilers to relate the semantics of executed code and source programs. We develop techniques to formally prove the correctness of compilation passes for synchronous languages (Lustre), and to validate compilation optimization for C code in the presence of threads.

3.3.1. Lustre:

The formal validation of a compiler for a synchronous language (or more generally for a language based on synchronous block diagrams) promises to reduce the likelihood of compiler-introduced bugs, the cost of testing, and also to ensure that properties verified on the source model hold of the target code. Such a validation would be complementary to existing industrial qualifications which certify the development process and not the functional correctness of a compiler. The scientific interest is in developing models and techniques that both facilitate the verification and allow for convenient reasoning over the semantics of a language and the behavior of programs written in it.

3.3.2. C/C++:

The recently approved C11 and C++11 standards define a concurrency model for the C and C++ languages, which were originally designed without concurrency support. Their intent is to permit most compiler and hardware optimizations, while providing escape mechanisms for writing portable, high-performance, low-level code. Mainstream compilers are being modified to support the new standards. A subtle class of compiler bugs is the so-called concurrency compiler bugs, where compilers generate correct sequential code but break the concurrency memory model of the programming language. Such bugs are observable only when the miscompiled functions interact with concurrent contexts, making them particularly hard to detect. All previous techniques to test compiler correctness miss concurrency compiler bugs.

3.3.3. Static Analysis of x10

x10 is an explicit parallel programming language, originally developped by IBM Research. Parallelism is expressed by the `async / finish` construct (a disymetric variant of `fork / join`), and synchronization uses *clocks*, a sophisticated version of barriers. Programs in this language can be analysed at compile time provided their control statements obey the restrictions of of the polyhedral model. The analysis focuses on the extraction of the *happens before* relation of the subject program, and can be used for the detection of races and deadlocks. A first version of this analysis, which did not take clocks into account, was published in 2013. Its extension to clocked programs is a complex problem, which requires the use of a proof assistant, Coq. Work in collaboration with Alain Ketterlin and Eric Violard (Inria Camus) and Tomofumi Yuki (Inria Cairn).

3.3.4. Toward a Polynomial Model

The polyhedral model is a powerful tool for program analysis and verification, autoparallelization, and optimization. However, it can only be applied to a very restricted class of programs : counted loops, affine conditionals and arrays with affine subscripts. The key mathematical result at the bottom of this model is Farkas lemma, which characterizes all affine function non negative on a polyhedron. Recent mathematical results on the *Positiv Stellen Satz* enable a similar characterization for polynomials positive on a semi-algebraic set. Polynomials may be native to the subject code, but also appears as soon as counting is necessary, for instance when a multidimensional array is linearized or when messages are transmitted through a one dimensional channel. Applying the above theorems allows the detection of polynomial dependences and the construction of polynomial schedules, hence the detection of deadlocks. Code generation from a polynomial schedule is the subject of present work. These methods are applied to the language openStream. Work in collaboration with Albert Cohen and Alain Darte (Xilinx).

4. Highlights of the Year

4.1. Highlights of the Year

4.1.1. Awards

- Francesco Zappa Nardelli received the *Most Influential ICFP Paper Award* for 2007 paper “Ott: Effective Tool Support for the Working Semanticist” (http://www.sigplan.org/Awards/ICFP/).

5. New Software and Platforms

5.1. Cmmtest

FUNCTIONAL DESCRIPTION: Cmmtest is a tool for hunting concurrency compiler bugs. The Cmmtest tool performs random testing of C and C++ compilers against the C11/C++11 memory model. A test case is any well-defined, sequential C program, for each test case, cmmtest:

compiles the program using the compiler and compiler optimisations that are being tested,

runs the compiled program in an instrumented execution environment that logs all memory accesses to global variables and synchronisations,

compares the recorded trace with a reference trace for the same program, checking if the recorded trace can be obtained from the reference trace by valid eliminations, reorderings and introductions.

Cmmtest identified several mistaken write introductions and other unexpected behaviours in the latest release of the gcc compiler. These have been promptly fixed by the gcc developers.

- Participants: Anirudh Kumar, Francesco Zappa Nardelli, Pankaj More, Pankaj Pawan, Pankaj Prateek Kewalramani and Robin Morisset
- Contact: Francesco Zappa Nardelli
- URL: http://www.di.ens.fr/~zappa/projects/cmmtest/

5.2. GCC

KEYWORDS: Compilation - Polyhedral compilation
FUNCTIONAL DESCRIPTION: The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgcj,...). GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100

- Participants: Albert Cohen, Feng Li, Nhat Minh Le, Riyadh Baghdadi and Tobias Grosser
- Contact: Albert Cohen
- URL: http://gcc.gnu.org/

5.3. Heptagon

KEYWORDS: Compilers - Synchronous Language - Controller synthesis
FUNCTIONAL DESCRIPTION: Heptagon is an experimental language for the implementation of embedded real-time reactive systems. It is developed inside the Synchronics large-scale initiative, in collaboration with Inria Rhones-Alpes. It is essentially a subset of Lucid Synchrone, without type inference, type polymorphism and higher-order. It is thus a Lustre-like language extended with hierchical automata in a form very close to SCADE 6. The intention for making this new language and compiler is to develop new aggressive optimization techniques for sequential C code and compilation methods for generating parallel code for different platforms. This explains much of the simplifications we have made in order to ease the development of compilation techniques.

The current version of the compiler includes the following features: - Inclusion of discrete controller synthesis within the compilation: the language is equipped with a behavioral contract mechanisms, where assumptions can be described, as well as an "enforce" property part. The semantics of this latter is that the property should be enforced by controlling the behaviour of the node equipped with the contract. This property will be enforced by an automatically built controller, which will act on free controllable variables given by the programmer. This extension has been named BZR in previous works. - Expression and compilation of array values with modular memory optimization. The language allows the expression and operations on arrays (access, modification, iterators). With the use of location annotations, the programmer can avoid unnecessary array copies.

- Participants: Adrien Guatto, Brice Gelineau, Cédric Pasteur, Eric Rutten, Gwenaël Delaval, Léonard Gérard and Marc Pouzet
- Partners: UGA - ENS Paris - Inria - LIG
- Contact: Gwenaël Delaval
- URL: http://heptagon.gforge.inria.fr

5.4. isl

FUNCTIONAL DESCRIPTION: isl is a library for manipulating sets and relations of integer points bounded by linear constraints. Supported operations on sets include intersection, union, set difference, emptiness check, convex hull, (integer) affine hull, integer projection, transitive closure (and over-approximation), computing the lexicographic minimum using parametric integer programming. It includes an ILP solver based on generalized basis reduction, and a new polyhedral code generator. isl also supports affine transformations for polyhedral compilation, and increasingly abstract representations to model source and intermediate code in a polyhedral framework.

- Participants: Albert Cohen, Sven Verdoolaege and Tobias Grosser
- Contact: Sven Verdoolaege
- URL: http://freshmeat.net/projects/isl

5.5. Lem

lightweight executable mathematics

FUNCTIONAL DESCRIPTION: Lem is a lightweight tool for writing, managing, and publishing large scale semantic definitions. It is also intended as an intermediate language for generating definitions from domain-specific tools, and for porting definitions between interactive theorem proving systems (such as Coq, HOL4, and Isabelle). As such it is a complementary tool to Ott. Lem resembles a pure subset of Objective Caml, supporting typical functional programming constructs, including top-level parametric polymorphism, datatypes, records, higher-order functions, and pattern matching. It also supports common logical mechanisms including list and set comprehensions, universal and existential quantifiers, and inductively defined relations. From this, Lem generates OCaml, HOL4, Coq, and Isabelle code.

- Participants: Francesco Zappa Nardelli, Peter Sewell and Scott Owens
- Contact: Francesco Zappa Nardelli
- URL: http://www.cl.cam.ac.uk/~pes20/lem/

5.6. Lucid Synchrone

FUNCTIONAL DESCRIPTION: Lucid Synchrone is a language for the implementation of reactive systems. It is based on the synchronous model of time as provided by Lustre combined with features from ML languages. It provides powerful extensions such as type and clock inference, type-based causality and initialization analysis and allows to arbitrarily mix data-flow systems and hierarchical automata or flows and valued signals.

RELEASE FUNCTIONAL DESCRIPTION: The language is still used for teaching and in our research but we do not develop it anymore. Nonetheless, we have integrated several features from Lucid Synchrone in new research prototypes described below. The Heptagon language and compiler are a direct descendent of it. The new language Zélus for hybrid systems modeling borrows many features originaly introduced in Lucid Synchrone.

- Contact: Marc Pouzet
- URL: http://www.di.ens.fr/~pouzet/lucid-synchrone/

5.7. Lucy-n

Lucy-n: an n-synchronous data-flow programming language
FUNCTIONAL DESCRIPTION: Lucy-n is a language to program in the n-synchronous model. The language is similar to Lustre with a buffer construct. The Lucy-n compiler ensures that programs can be executed in bounded memory and automatically computes buffer sizes. Hence this language allows to program Kahn networks, the compiler being able to statically compute bounds for all FIFOs in the program.

- Participants: Adrien Guatto, Albert Cohen, Louis Mandel and Marc Pouzet
- Contact: Albert Cohen
- URL: https://www.lri.fr/~mandel/lucy-n/

5.8. Ott

FUNCTIONAL DESCRIPTION: Ott is a tool for writing definitions of programming languages and calculi. It takes as input a definition of a language syntax and semantics, in a concise and readable ASCII notation that is close to what one would write in informal mathematics. It generates output:

a LaTeX source file that defines commands to build a typeset version of the definition,

a Coq version of the definition,

an Isabelle version of the definition, and

a HOL version of the definition.

Additionally, it can be run as a filter, taking a LaTeX/Coq/Isabelle/HOL source file with embedded (symbolic) terms of the defined language, parsing them and replacing them by typeset terms.

The main goal of the Ott tool is to support work on large programming language definitions, where the scale makes it hard to keep a definition internally consistent, and to keep a tight correspondence between a definition and implementations. We also wish to ease rapid prototyping work with smaller calculi, and to make it easier to exchange definitions and definition fragments between groups. The theorem-prover backends should enable a smooth transition between use of informal and formal mathematics.

- Participants: Francesco Zappa Nardelli, Peter Sewell and Scott Owens
- Contact: Francesco Zappa Nardelli
- URL: http://www.cl.cam.ac.uk/~pes20/ott/

5.9. PPCG

FUNCTIONAL DESCRIPTION: PPCG is our source-to-source research tool for automatic parallelization in the polyhedral model. It serves as a test bed for many compilation algorithms and heuristics published by our group, and is currently the best automatic parallelizer for CUDA and OpenCL (on the Polybench suite).

- Participants: Albert Cohen, Riyadh Baghdadi, Sven Verdoolaege and Tobias Grosser
- Contact: Sven Verdoolaege
- URL: http://freshmeat.net/projects/ppcg

5.10. ReactiveML

FUNCTIONAL DESCRIPTION: ReactiveML is a programming language dedicated to the implementation of interactive systems as found in graphical user interfaces, video games or simulation problems. ReactiveML is based on the synchronous reactive model due to Boussinot, embedded in an ML language (OCaml).

The Synchronous reactive model provides synchronous parallel composition and dynamic features like the dynamic creation of processes. In ReactiveML, the reactive model is integrated at the language level (not as a library) which leads to a safer and a more natural programming paradigm.

- Participants: Cédric Pasteur, Guillaume Baudart and Louis Mandel
- Contact: Guillaume Baudart

5.11. SundialsML

Sundials/ML
KEYWORDS: Simulation - Mathematics - Numerical simulations
SCIENTIFIC DESCRIPTION: Sundials/ML is a comprehensive OCaml interface to the Sundials suite of numerical solvers (CVODE, CVODES, IDA, IDAS, KINSOL). Its structure mostly follows that of the Sundials library, both for ease of reading the existing documentation and for adapting existing source code, but several changes have been made for programming convenience and to increase safety, namely:

solver sessions are mostly configured via algebraic data types rather than multiple function calls,

errors are signalled by exceptions not return codes (also from user-supplied callback routines),

user data is shared between callback routines via closures (partial applications of functions),

vectors are checked for compatibility (using a combination of static and dynamic checks), and

explicit free commands are not necessary since OCaml is a garbage-collected language.
FUNCTIONAL DESCRIPTION: Sundials/ML is a comprehensive OCaml interface to the Sundials suite of numerical solvers (CVODE, CVODES, IDA, IDAS, KINSOL, ARKODE).

- Participants: Jun Inoue, Marc Pouzet and Timothy Bourke
- Partner: UPMC
- Contact: Marc Pouzet
- URL: http://inria-parkas.github.io/sundialsml/

5.12. Zelus

SCIENTIFIC DESCRIPTION: The Zélus implementation has two main parts: a compiler that transforms Zélus programs into OCaml programs and a runtime library that orchestrates compiled programs and numeric solvers. The runtime can use the Sundials numeric solver, or custom implementations of well-known algorithms for numerically approximating continuous dynamics.
FUNCTIONAL DESCRIPTION: Zélus is a new programming language for hybrid system modeling. It is based on a synchronous language but extends it with Ordinary Differential Equations (ODEs) to model continuous-time behaviors. It allows for combining arbitrarily data-flow equations, hierarchical automata and ODEs. The language keeps all the fundamental features of synchronous languages: the compiler statically ensure the absence of deadlocks and critical races, it is able to generate statically scheduled code running in bounded time and space and a type-system is used to distinguish discrete and logical-time signals from continuous-time ones. The ability to combines those features with ODEs made the language usable both for programming discrete controllers and their physical environment.

- Participants: Marc Pouzet and Timothy Bourke
- Contact: Marc Pouzet

6. New Results

6.1. Compiler Optimisations for Multicore Architectures

Participants: Robin Morisset, Francesco Zappa Nardelli.

Robin has completed his research work on sound optimisations for modern multicore architectures. This covered optimisations that can be expressed inside the semantics of the C11/C++11 programming language, as well as optimisations that can be expressed only at the harware level. In particular we have shown how partial redundancy elimination (PRE) can be instantiated to perform *provably correct* fence elimination for multi-threaded programs running on top of the x86, ARM and IBM Power relaxed memory models. We have implemented our algorithm in the x86, ARM and Power backends of the `LLVM` compiler infrastructure. The optimisation does not induce an observable overhead at compile-time and can result in up-to 10% speedup on some benchmarks.

This work has been published in CC 2017 [10]. The implementation of the optimisations will be submitted for inclusion in the LLVM compiler suite.

Robin Morisset completed this line of research and defended his PhD Thesis in April 2017.

6.2. Julia Subtyping Reconstructed

Participant: Francesco Zappa Nardelli.

Julia is a programming language recently designed at MIT to support the needs of the scientific community. Julia occupies a unique position in the design landscape, it is a dynamic language with no type system, yet it has a surprisingly rich set of types and type annotations used to specify multimethod dispatch. The types that can be expressed in function signatures include parametric union types, covariant tuple types, parametric user-defined types with single inheritance, invariant type application, and finally types and values can be reified to appear in signatures. With Vitek started a research project to study the design and the pragmatic use of the Julia language. At first we focused on the Julia subtyping algorithm. We studied the empirical evidence that users appeal to all the features provided by Julia and we report on a formalisation and implementation of the subtyping algorithm. The work on subtyping is under submission to an international conference. This line of research will be pursued in the next year, studying method dispatch and type inference.

6.3. Comparing Designs for Gradual Types

Participant: Francesco Zappa Nardelli.

The enduring popularity of dynamically typed languages has given rise to a cottage industry of static type systems, often called gradual type systems, that let developers annotate legacy code piecemeal. Type soundness for a program which mixes typed and untyped code does not ensure the absence of errors at runtime, rather it means that some errors will caught at type checking time, while other will be caught as the program executes. After a decade of research it is clear that the combination of mutable state, self references and subtyping presents interesting challenges to designers of gradual type systems. We have reviewed the state of the art in gradual typing for objects, and introduced a class-based object calculus with a static type system, dynamic method dispatch, transparent wrappers and dynamic class generation that we use to model key features of several gradual type systems by translation to it, and discuss the implications of the respective designs. We have submitted this work to an international conference.

6.4. Symbolic Simulation for a timed-automaton subset of Zélus

Participants: Guillaume Baudart, Timothy Bourke, Marc Pouzet.

Synchronous languages like Lustre are ideal for programming an important class of embedded controllers. Their discrete model of time and deterministic semantics facilitate the precise expression of reactive behaviors. That said, many systems are naturally modeled using physical timing constraints that almost inevitably involve some 'timing nondeterminism' due to tolerances in requirements or uncertainties in implementations. Conversely, such constraints are readily modeled using Timed Automata, and simulated symbolically in Uppaal, but large-scale discrete-time behaviors are more cumbersome to express in such tools.

In this work, we combined existing techniques and data structures for Timed Safety Automata with typing and compilation techniques for synchronous languages to develop a novel programming language where discrete reactive logic can be mixed with nondeterministic continuous-time features. In particular, we developed an extension of Lustre and a specialization of Zélus for modeling real-time reactive systems, proposed a symbolic simulation scheme based on 'sweeping', and showed how to implement it via source-to-source compilation. A type system, based on that of Zélus, ensures the correct composition of discrete-time and continuous-time elements.

Our proposal has been implemented using the Zélus compiler and a small library of operations on Difference-Bound Matrices (DBMs). Unlike the work around Uppaal, we do not address verification or treat industrial case studies. A future direction could be to verify programs in our 'extended version of Lustre' by either generating C code and using the highly-tuned Uppaal DBM library, or combining symbolic techniques for Lustre programs with those for Timed Automata.

This work was presented at FDL 2017 [5]. A prototype implementation is available online.

This work is also described with extended examples in Baudart's PhD thesis [1] which was defended in March of 2017.

6.5. Verified compilation of Lustre

Participants: Timothy Bourke, Lélio Brun, Marc Pouzet.

Synchronous dataflow languages and their compilers are increasingly used to develop safety-critical applications, like fly-by-wire controllers in aircraft and monitoring software for power plants. A striking example is the SCADE Suite tool of ANSYS/Esterel Technologies which is DO-178B/C qualified for the aerospace and defense industries. This tool allows engineers to develop and validate systems at the level of abstract block diagrams that are automatically compiled into executable code.

Formal modelling and verification in an interactive theorem prover can potentially complement the industrial certification of such tools to give very precise definitions of language features and increased confidence in their correct compilation; ideally, right down to the binary code that actually executes.

This year we continued work on our verified Lustre compiler. We developed a set of benchmarks and evaluated the Worst Case Execution time of code generated by our compiler with that of code generated by the academic Heptagon and Lustre v6 compilers. This work also required numerous improvements to the parser and elaborator. We also tested the compiler on an industrial example in the context of the ASSUME project. We completed the end-to-end theorem showing that the dataflow semantics of input programs is preserved by the assembly language semantics generated by our compiler combined with the CompCert compiler. This work was presented in June at PLDI [8].

In the latter half to the year we worked on extending the compiler to accept nodes with clocked arguments, treating non-normalized Lustre, and adding a modular reset to the language.

To accept clocked arguments, we extended the semantic model, developed a richer encoding of the clock system, added a new invariant to forbid non-trivial sub-clocked expressions, and adapted the correctness proof. An unexpected complication was the need to pass undefined variables in function call arguments: this required changes to our intermediate Obc language and introduces minor technical difficulties in the translation to Clight which requires that variables be defined. This work is now almost complete.

To treat non-normalized Lustre, we introduced new syntactic and semantic definitions, updated the parser, and complete reworked the elaboration and type-checking passes. We developed many small Lustre programs to confirm our understanding of the language and test the updated front-end; this also revealed several bugs in other academic Lustre compilers. This work is now complete. The next step is to implement the normalization pass to connect the new front-end to the existing compilation passes.

The work on modular resets continues as part of L. Brun's PhD thesis. This year we developed a novel semantic model for modular resets and started considering how to generate provably correct code.

In collaboration with Pierre-Évariste Dagand (CNRS), Lionel Reig (Collège de France), and Xavier Leroy (Inria, GALLIUM team).

6.6. Zélus: Synchronous Languages + Ordinary Differential Equations

Participants: Timothy Bourke, Marc Pouzet.

Zélus is a synchronous language extended with Ordinary Differential Equations (ODEs) to model systems with complex interactions between discrete-time and continuous-time dynamics. It shares the basic principles of Lustre with features from Lucid Synchrone (type inference, hierarchical automata, and signals). The compiler is written in OCaml and is structured as a series of source-to-source and traceable transformations that ultimately yield statically scheduled sequential code. Continuous components are simulated using off-the-shelf numerical solvers (here Sundials CVODE) and, for the moment, two built-in solvers (ode23 and ode45).

Zélus is used to experiment with new techniques for building hybrid modelers like Simulink/Stateflow and Modelica on top of a synchronous language. The language exploits novel techniques for defining the semantics of hybrid modelers, it provides dedicated type systems to ensure the absence of discontinuities during integration and the generation of sequential code. In particular, all discrete computations must be aligned to zero-crossing events; programs with causality loops and uninitialized values are statically rejected.

This year we added arrays with iterators and statically expanded higher-order functions to the language. Both extensions required adapting the existing type and causality systems, and extending the compilation algorithms. These extensions allowed us to show that a fairly large set of blocks from the Simulink standard library can be programmed in a precise, purely functional language using stream equations, hierarchical automata, Ordinary Differential Equations (ODEs), and deterministic synchronous parallel composition. Although some blocks cannot be expressed as they mix discrete-time and continuous-time signals in unprincipled ways; they are statically rejected by the type checker. This work was presented at EMSOFT in October [9]

Our work on analyzing causality loops in hybrid systems modelers was published in the NAHS journal [2].

In collaboration with B. Caillaud and A. Benveniste (Inria Rennes); and F. Carcenac, B. Pagano, and C. Pasteur (ANSYS/Esterel Technologies).

6.7. Compiling synchronous languages for multi-processor implementations

Participants: Timothy Bourke, Albert Cohen, Guillaume Iooss, Marc Pouzet.

Working together with industrial partners in the context of the ASSUME project.

We spent a week in Toulouse working at Airbus on their use case and our front-end tools. We can now treat the case and generate code for Lopht (AOSTE team), which, in turn, generates executable code for the Kalray MPPA. We have also advanced significantly on two use cases provided by Safran. The first one is similar to the Airbus use case. The second one is more preliminary, it revealed the need for more general iterators to better express FFT algorithms.

We have made solid progress on a language extension for expressing and manipulating harmonic clocks. In particular, we derive a scheduling problem from the clock constraints in a program and we are working on automatically calculating their initial phases.

We have written an import tool that transforms graphs of dependencies between several Lustre components scheduled with different harmonic periods into a monolithic Lustre program. We are working on a hyper-scheduling transformation that generates a single step function running at the slowest period and that contains multiple instances of the faster tasks with annotations to ensure they execute at the correct time.

In collaboration (this year) with Dumitru Potop-Butucaru and Keryan Didier (Inria, AOSTE team); Jean Souyris and Adrien Gauffriau (Airbus); Philippe Baufreton et Jean-Marie Courtelle (Safran).

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

Polly Labs contract with ARM, 2015-2019, with the participation of Qualcomm, Xilinx and Facebook (human resources, consulting services and and hiring former PARKAS members).

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

ANR/CHIST-ERA DIVIDEND project, 2013-2018.

8.1.2. Investissements d'avenir

Sys2Soft contract (Briques Génériques du Logiciel Embarqué). Partenaire principal: Dassault-Systèmes, etc. Inria contacts are Benoit Caillaud (HYCOMES, Rennes) and Marc Pouzet (PARKAS, Paris).

8.1.3. Others

Marc Pouzet is scientific advisor for the Esterel-Technologies/ANSYS company.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. Eurolab-4-HPC

Title: EuroLab-4-HPC: Foundations of a European Research Center of Excellence in High Performance Computing Systems

Programm: H2020

Duration: September 2015 - September 2017

Coordinator: CHALMERS TEKNISKA HOEGSKOLA AB

Inria contact: Albert Cohen

Europe has built momentum in becoming a leader in large parts of the HPC ecosystem. It has brought together technical and business stakeholders from application developers via system software to exascale systems. Despite such gains, excellence in high performance computing systems is often fragmented and opportunities for synergy missed. To compete internationally, Europe must bring together the best research groups to tackle the longterm challenges for HPC. These typically cut across layers, e.g., performance, energy efficiency and dependability, so excellence in research must target all the layers in the system stack. The EuroLab-4-HPC project's bold overall goal is to build connected and sustainable leadership in high-performance computing systems by bringing together the different and leading performance orientated communities in Europe, working across all layers of the system stack and, at the same time, fuelling new industries in HPC.

8.2.1.2. TETRACOM

Title: Technology Transfer in Computing Systems

Programm: FP7

Duration: September 2013 - August 2016

Coordinator: RHEINISCH-WESTFAELISCHE TECHNISCHE HOCHSCHULE AACHEN

Inria contact: Albert Cohen

The mission of the TETRACOM Coordination Action is to boost European academia-to-industry technology transfer (TT) in all domains of Computing Systems. While many other European and national initiatives focus on training of entrepreneurs and support for start-up companies, the key differentiator of TETRACOM is a novel instrument called Technology Transfer Project (TTP). TTPs help to lower the barrier for researchers to make the first steps towards commercialisation of their research results. TTPs are designed to provide incentives for TT at small to medium scale via partial funding of dedicated, well-defined, and short term academia-industry collaborations that bring concrete R&D results into industrial use. This will be implemented via competitive Expressions-of-Interest (EoI) calls for TTPs, whose coordination, prioritization, evaluation, and management are the major actions of TETRACOM. It is expected to fund up to 50 TTPs. The TTP activities will be complemented by Technology Transfer Infrastructures (TTIs) that provide training, service, and dissemination actions. These are designed to encourage a larger fraction of the R&D community to engage in TTPs, possibly even for the first time. Altogether, TETRACOM is conceived as the major pilot project of its kind in the area of Computing Systems, acting as a TT catalyst for the mutual benefit of academia and industry. The projects primary success metrics are the number and value of coordinated TTPs as well as the amount of newly introduced European TT actors. It is expected to acquire around more than 20 new contractors over the project duration. TETRACOM complements and actually precedes the use of existing financial instruments such as venture capital or business angels based funding.

8.2.1.3. EMC2

Title: Embedded Multi-Core Systems for Mixed Criticality Applications in Dynamic and Changeable Real-Time Environments

Programm: FP7

Duration: April 2014 - March 2017

Coordinator: Infineon Technologies

Inria contact: Albert Cohen

'Embedded systems are the key innovation driver to improve almost all mechatronic products with cheaper and even new functionalities. Furthermore, they strongly support today's information society as inter-system communication enabler. Consequently boundaries of application domains are alleviated and ad-hoc connections and interoperability play an increasing role. At the same time, multi-core and many-core computing platforms are becoming available on the market and provide a breakthrough for system (and application) integration. A major industrial challenge arises facing (cost) efficient integration of different applications with different levels of safety and security on a single computing platform in an open context. The objective of the EMC^2 project (Embedded multi-core systems for mixed criticality applications in dynamic and changeable real-time environments) is to foster these changes through an innovative and sustainable service-oriented architecture approach for mixed criticality applications in dynamic and changeable real-time environments. The EMC2 project focuses on the industrialization of European research outcomes and builds on the results of previous ARTEMIS, European and National projects. It provides the paradigm shift to a new and sustainable system architecture which is suitable to handle open dynamic systems. EMC^2 is part of the European Embedded Systems industry strategy to maintain its leading edge position by providing solutions for: . Dynamic Adaptability in Open Systems . Utilization of expensive system features only as Service-on-Demand in order to reduce the overall system cost. . Handling of mixed criticality applications under real-time conditions . Scalability and utmost flexibility . Full scale deployment and management of integrated tool chains, through the entire lifecycle Approved by ARTEMIS-JU on 12/12/2013 for EoN. Minor mistakes and typos corrected by the Coordinator, finally approved by ARTEMIS-JU on 24/01/2014. Amendment 1 changes approved by ECSEL-JU on 31/03/2015.'

8.3. International Initiatives

8.3.1. Inria Associate Teams Not Involved in an Inria International Labs

8.3.1.1. POLYFLOW

Title: Polyhedral Compilation for Data-Flow Programming Languages

International Partner (Institution - Laboratory - Researcher):

IISc Bangalore (India) - Department of Computer Science and Automation (CSA) - Uday Kumar Reddy Bondhugula

Start year: 2016

See also: http://polyflow.gforge.inria.fr

The objective of the associate team is to foster collaborations on fundamental and applied research. It also supports training sessions, exchange of undergraduate and master students, and highlighting opportunities in the partners' research, education and economic environments.

Polyhedral techniques for program transformation are now used in several proprietary and open source compilers. However, most of the research on polyhedral compilation has focused on imperative languages, where computation is specified in terms of computational statements within nested loops and control structures. Graphical data-flow languages, where there is no notion of statements or a schedule specifying their relative execution order, have so far not been studied using a powerful transformation or optimization approach. These languages are extremely popular in the system analysis, modeling and design of embedded reactive control applications. They also underline the construction of domain-specific languages and compiler intermediate representations. The execution semantics of data-flow languages impose a different set of challenges for compilation and optimization. We are studying techniques enabling the extraction of a polyhedral representation from data-flow

programs, to transform them with the goal of generating memory-efficient and high-performance code for modern architectures.

The research conducted in PolyFlow covers both fundamental and applied aspects. The partners also emphasize the development of solid research tools. The associate team will facilitate their dissemination as free software and their exploitation through industrial collaborations.

8.3.2. Participation in Other International Programs

- VerticA (Francesco Zappa Nardelli), 2017-2020, joint project with Northeastern University, USA, financed by the ONR (Office of Naval Research), 1.5M$ (subcontract for 150k$).

8.4. International Research Visitors

8.4.1. Visits of International Scientists

8.4.1.1. Internships

Alex Susu from Polytechnica di Bucarest spent a 3 months internship in the Fall.

8.4.2. Visits to International Teams

8.4.2.1. Sabbatical programme

Francesco Zappa Nardelli, from Feb. 1st, 2017 to July. 29th, 2017 has been on sabbatical leave at Northeastern University, Boston, USA, invited by Prof. Jan Vitek.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

- Albert Cohen was the General Chair of PLDI 2017.

9.1.2. Scientific Events Selection

9.1.2.1. Member of the Conference Program Committees

- Timothy Bourke was a member of the PC of EMSOFT 2017.
- Timothy Bourke was a member of the PC of the Modelica Conference 2017.
- Timothy Bourke was a member of the PC of SCOPES 2017.
- Timothy Bourke was a member of the Student Research Competition panel of PLDI 2017.
- Francesco Zappa Nardelli was a member of the PC of POPL 2017.
- Francesco Zappa Nardelli was a member of the PC of ECOOP 2017.
- Marc Pouzet was a member of the PC of SCOPES 2017, EOOLT 2017, FADL 2017.
- Albert Cohen was a PC member of CGO 2018, Supercomputing 2017, PACT 2017.
- Albert Cohen was the area co-chair for programming models at IPDPS 2018.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

- Albert Cohen is associate editor of the ACM Transactions on Architecture and Code Optimization

9.1.3.2. Reviewer - Reviewing Activities

- Timothy Bourke was a reviewer for the Springer Real-Time Systems Journal.

- Timothy Bourke was a reviewer for IEEE Transactions on Software Engineering.

9.1.4. Invited Talks

- Timothy Bourke was invited to talk about the seL4 project at the Forum Méthodes Formelles: “Méthodes formelles et cyber-sécurité” in Toulouse in January 2017.
- Timothy Bourke was invited to speak on the “Verified Compilation of Lustre” at the University of Birmingham in March 2017.
- Francesco Zappa Nardelli is an invited speaker at the Entropy Workshop, January 2018.
- Francesco Zappa Nardelli was an invited speaker at Dagsthul Seminar 17502 “Testing and Verification of Compilers”, December 2017.
- Francesco Zappa Nardelli was an invited speaker at the ETH Workshop on Software Correctness and Reliability, October 2017.
- Marc Pouzet was an invited speaker of the GT OVSTR Digicosme (CEA - Telecom Paris), in April 2017; the GDR Glace (part of GPL), in June 2017.

9.1.5. Leadership within the Scientific Community

Albert Cohen is a steering committee member of the PLDI, PPoPP and Compiler Construction conferences.

9.1.6. Scientific Expertise

Albert Cohen has been a visiting scientist at Facebook Artificial Intelligence Research.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Master: F. Zappa Nardelli: “A Programmer’s introduction to Computer Architectures and Operating Systems" (M1), 45h, École Polytechnique, France

Master: A. Cohen & F. Zappa Nardelli, “Semantics, languages and algorithms for multicore programming", Lecture, 12h+9h, M2, MPRI: Ecole normale supeérieure and Université Paris Diderot, France

Licence: F. Zappa Nardelli: “Concurrent Programming" (L3), PCs, 32h, E´cole Polytechnique, France

Master : M. Pouzet & T. Bourke: “Synchronous Systems” (M2), Lectures and TDs, MPRI, France

Master: M. Pouzet : “Synchronous reactive Languages” (M2), Lectures. Master Comasic (Polytechnique), France.

Master: T. Bourke participated in reviewing the M1 internships of students at the ENS, France.

Licence : M. Pouzet & T. Bourke: “Operating Systems” (L3), Lectures and TDs, ENS, France.

Licence : T. Bourke, “Digital Systems” (L3), Lectures and TDs, ENS, France

Marc Pouzet is Director of Studies for the CS department, at ENS.

9.2.2. Supervision

- PhD : Guillaume Baudart, 3rd year, supervised by T. Bourke and M. Pouzet. This thesis was defended in March 2017.
- PhD in progress : Ulyssse Beaugnon, 3rd year, supervised by A. Cohen and M. Pouzet.
- PhD in progress : Lélio Brun, 2nd year, supervised by T. Bourke and M. Pouzet.
- PhD : Robin Morisset, 3rd year, supervised by F. Zappa Nardelli. This thesis was defended in April 2017.
- PhD in progress : Chandan Reddy, 3rd year, supervised by A. Cohen.
- PhD in progress : Jie Zhao, 3rd year, supervised by A. Cohen.

9.2.3. Juries

Francesco Zappa Nardelli was jury member of the PhD thesis of Yannick Zakowski, ENS Rennes, Dec 2017.

Francesco Zappa Nardelli will be jury member of the PhD thesis of Francois Ginraud, Grenoble, Jan 2018.

10. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] G. BAUDART. *A synchronous approach to quasi-periodic systems*, PSL Research University, March 2017, https://tel.archives-ouvertes.fr/tel-01507595

Articles in International Peer-Reviewed Journals

[2] A. BENVENISTE, T. BOURKE, B. CAILLAUD, B. PAGANO, M. POUZET. *A Type-based Analysis of Causality Loops in Hybrid Systems Modelers*, in "Nonlinear Analysis: Hybrid Systems", November 2017, vol. 26, pp. 168–189 [*DOI :* 10.1016/J.NAHS.2017.04.004], https://hal.inria.fr/hal-01549183

[3] I. LLOPARD, C. FABRE, A. COHEN. *A From a Formalized Parallel Action Language to its Efficient Code Generation*, in "ACM Transactions on Embedded Computing Systems (TECS)", January 2017 [*DOI :* 10.1145/0000000.0000000], https://hal.inria.fr/hal-01425140

Invited Conferences

[4] J.-L. COLAÇO, B. PAGANO, M. POUZET. *Scade 6: A Formal Language for Embedded Critical Software Development*, in "TASE 2017 - 11th International Symposium on Theoretical Aspects of Software Engineering", Nice, France, September 2017, pp. 1-10, https://hal.inria.fr/hal-01666470

International Conferences with Proceedings

[5] G. BAUDART, T. BOURKE, M. POUZET. *Symbolic Simulation of Dataflow Synchronous Programs with Timers*, in "12th Forum on Specification and Design Languages (FDL 2017)", Vérone, Italy, Electronic Chips & System Design Initiative (ECSI), September 2017, https://hal.inria.fr/hal-01575621

[6] U. BEAUGNON, A. POUILLE, M. POUZET, J. PIENAAR, A. COHEN. *Optimization Space Pruning without Regrets*, in "CC 2017 - 26th International Conference on Compiler Construction", Austin, TX, United States, Proceedings of the International Conference on Compiler Construction, ACM Press, February 2017, pp. 34-44 [*DOI :* 10.1145/3033019.3033023], https://hal.inria.fr/hal-01655602

[7] A. BENVENISTE, B. CAILLAUD, H. ELMQVIST, K. GHORBAL, M. OTTER, M. POUZET. *Structural Analysis of Multi-Mode DAE Systems*, in "Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017", Pittsburgh, PA, United States, April 2017 [*DOI :* 10.1145/3049797.3049806], https://hal.inria.fr/hal-01521918

[8] T. BOURKE, L. BRUN, P.-E. DAGAND, X. LEROY, M. POUZET, L. RIEG. *A Formally Verified Compiler for Lustre*, in "PLDI 2017 - 38th ACM SIGPLAN Conference on Programming Language Design and Implementation", Barcelone, Spain, ACM, June 2017, https://hal.inria.fr/hal-01512286

[9] T. BOURKE, F. CARCENAC, J.-L. COLAÇO, B. PAGANO, C. PASTEUR, M. POUZET. *A Synchronous Look at the Simulink Standard Library*, in "EMSOFT 2017 - 17th International Conference on Embedded Software", Seoul, South Korea, ACM Press, October 2017, 23 p. , https://hal.inria.fr/hal-01575631

[10] R. MORISSET, F. ZAPPA NARDELLI. *Partially Redundant Fence Elimination for x86, ARM and Power processors*, in "International Conference on Compiler Construction (CC)", Austin, United States, February 2017, https://hal.inria.fr/hal-01423612

[11] J. ZHAO, A. COHEN. *A general compilation algorithm to parallelize and optimize counted loops with dynamic data-dependent bounds*, in "IMPACT 2017 - 7th International Workshop on Polyhedral Compilation Techniques", Stockholm, Sweden, January 2017, pp. 1-10, https://hal.inria.fr/hal-01657608

[12] R. VON HANXLEDEN, T. BOURKE, A. GIRAULT. *Real-Time Ticks for Synchronous Programming*, in "FDL 2017 - 12th Forum on Specification and Design Languages", Vérone, Italy, Electronic Chips & System Design Initiative (ECSI), September 2017, https://hal.inria.fr/hal-01575629

National Conferences with Proceedings

[13] T. BOURKE, P.-E. DAGAND, M. POUZET, L. RIEG. *Vérification de la génération modulaire du code impératif pour Lustre*, in "JFLA 2017 - Vingt-huitième Journées Francophones des Langages Applicatifs", Gourette, France, January 2017, https://hal.inria.fr/hal-01403830

Conferences without Proceedings

[14] A. SUSUNGI, A. COHEN, C. TADONKI. *More Data Locality for Static Control Programs on NUMA Architectures*, in "IMPACT 2017 - 7th International Workshop on Polyhedral Compilation Techniques IMPACT 2017", Stockholm, Sweden, January 2017, 11 p. , https://hal-mines-paristech.archives-ouvertes.fr/hal-01529354

Research Reports

[15] K. DIDIER, A. COHEN, A. GAUFFRIAU, A. GRAILLAT, D. POTOP-BUTUCARU. *Sheep in wolf's clothing: Implementation models for data-flow multi-threaded software*, Inria Paris, April 2017, n^{o} RR-9057, 31 p. , https://hal.inria.fr/hal-01509314

[16] O. ZINENKO, S. VERDOOLAEGE, C. REDDY, J. SHIRAKO, T. GROSSER, V. SARKAR, A. COHEN. *Unified Polyhedral Modeling of Temporal and Spatial Locality*, Inria Paris, November 2017, n^{o} RR-9110, 41 p. , https://hal.inria.fr/hal-01628798

Other Publications

[17] A. SUSUNGI, N. A. RINK, J. CASTRILLÓN, I. HUISMANN, A. COHEN, C. TADONKI, J. STILLER, J. FRÖHLICH. *Towards Compositional and Generative Tensor Optimizations*, October 2017, ACM SIGPLAN conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH), Poster, https://hal-mines-paristech.archives-ouvertes.fr/hal-01666818