IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2018

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

Project-Team PARKAS

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

The PARKAS team is located at the École normale supérieure

Keywords:

Computer Science and Digital Science:

A1.1.1. - Multicore, Manycore
A1.1.3. - Memory models
A2.1.1. - Semantics of programming languages
A2.1.4. - Functional programming
A2.1.6. - Concurrent programming
A2.1.9. - Synchronous languages
A2.2.2. - Memory models
A2.2.4. - Parallel architectures
A2.2.5. - Run-time systems
A2.2.6. - GPGPU, FPGA...
A2.2.7. - 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
B9.5.1. - Computer science
B9.5.2. - Mathematics

1. Team, Visitors, External Collaborators

Research Scientists

Timothy Bourke [Inria, Researcher]
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
Ulysse Beaugnon [Ecole Normale Supérieure Paris]
Lelio Brun [Ecole Normale Supérieure Paris]
Chandan Reddy Gopal [Inria]
Jie Zhao [Ecole d'ingénieurs]

Technical staff
Theophile Bastian [Inria, from Oct 2018 until Nov 2018]
Andi Drebes [Inria, from Oct 2018]
Michael Kruse [Inria, until Jan 2018]
Adilla Susungi [Inria, from Nov 2018]
Nicolas Tollenaere [Inria]
Oleksandr Zinenko [Inria, until Sep 2018]

Interns
Ismail Lahkim Bennani [Inria, from Apr 2018 until Aug 2018]
Remi Oudin [Inria, from Nov 2018]

Administrative Assistants
Christine Anocq [Inria]
Nelly Maloisel [Inria, from Dec 2018]

External Collaborator
Paul Feautrier [Univ de Lyon]

2. Overall Objectives

2.1. Overall Objectives

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 (Vélus, *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

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

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.
- 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

Guillaume Baudart was awarded the **ACM SIGBED Paul Caspi Memorial Dissertation Award** for his thesis “A Synchronous Approach to Quasi-Periodic Systems” [27] prepared in the PARKAS Team under the supervision of Marc Pouzet and Timothy Bourke and defended in 2017.

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).

RELEASE FUNCTIONAL DESCRIPTION: Adds support for v3.1.x of the Sundials Suite of numerical solvers.

Notably this release adds support for the new generic matrix and linear solver interfaces. The OCaml interface changes but the library is backward compatible with Sundials 2.7.0.

OCaml 4.02.3 or greater is now required and optionally OCamlMPI 1.03.

* New Sundials.Matrix and Sundials.LinearSolver modules. * Better treatment of integer type used for matrix indexing. * Refactor Dls and Sls modules into Sundials.Matrix. * Add confidence intervals to performance graph. * Miscellaneous improvements to configure script. * Potential incompatibility: changes to some label names: com_fn -> comm, ite_type -> iter. * Untangle the ARKODE mass-solver interface from the Jacobian interface.

- 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. 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 modeling 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 new semantic model for the modular reset feature provided by the Scade language and required for the compilation of hierarchical state machines. This work was presented at the SCOPES workshop in Germany in May [17]. Work continues on connecting this semantic model to the intermediate compilation target.

We completed work on generalizing the compiler to treat clocked arguments. This involved changes to our intermediate Obc language and the addition of a pass to add some (necessary) variable initializations in an efficient way. This work was accepted for presentation at the Journées Francophones des Langages Applicatifs in 2019.

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. This has been published in [15]. We are pursuing this line of research studying of the algorithm advances of Julia can be integrated into other programming languages.

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. This has been published in [18].

6.4. Fast and reliable unwinding via DWARF tables

Participants: Theophile Bastian, Francesco Zappa Nardelli.

DWARF is a widely-used debugging data format. DWARF is obviously relied upon by debuggers, but it plays an unexpected role in the runtime of high-level programming languages and in the implementation of program analysis tools. The debug information itself can be pervaded by subtle bugs, making the whole infrastructure unreliable. In this project we are investigating techniques and tools to perform validation and synthesis of the DWARF stack unwinding tables, to speedup DWARF-based unwinding, as well as exploring adventurous projects that can be built on top of reliable DWARF information.

At the time of writing, we have a tool that can validate DWARF unwind tables generated by mainstream compilers; the approach is effective, we found a problem in Clang table generation and several in GLIBC inline-assembly snippets. We also designed and implemented a tool that can synthesise DWARF unwind tables from binary that lacks them (e.g. because the compiler did not generate them - immediate applications: JITs assembly, inline assembly, ...). Additionally we have designed and implemented a ahead-of-time compiler of DWARF unwind tables to assembly, and an ad-hoc unwinder integrated with the defacto standard unwinder libuwind. It can speed up unwinding by a factor between 25x and 60x (depending on application), with a 2.5x size overhead for unwind information.

Discussion is in progress to get these tools included in mainstream tool (e.g. the GNU profiler Perf).

6.5. Sundials/ML: OCaml interface to Sundials Numeric Solvers

Participants: Timothy Bourke, Marc Pouzet.

This year we made major updates to the Sundials/ML OCaml interface to support v3.1.x of the Sundials Suite of numerical solvers.

This release adds support for the new generic matrix and linear solver interfaces. Major work was required to add these new modules, update the existing solver interfaces, and ensure backwards compatibility with Sundials to v2.7.0 (which is still the version installed by Debian stable). We also improved our treatment of integer types used in indexing, refactored the Dls and Sls matrix modules, improved our generation of performance stats (by adding confidence intervals), made the configure script more robust, and untangle the mass-solver and Jacobian interfaces of the ARKODE solver.

6.6. Zélus

Participants: Timothy Bourke, Marc Pouzet.

This year, we made a major revision of the language and compiler, called now the version 2. The language now deal with higher order functions. All the static analyses, type inference, causality inference and the initialization analysis has been extended. The code generation has also been improved, in particular the interface with the numeric solver. Several larger examples have been written.

A paper that present the overall approach followed in ZELUS has been published [12].

6.7. Deterministic Concurrency: A Clock-Synchronised Shared Memory Approach

Participant: Marc Pouzet.

Synchronous programming (SP) provides deterministic concurrency. So far, however, communication has been constrained to a set of primitive clock-synchronised shared memory (scm) data types, such as data-flow registers, streams and signals with restricted read and write accesses that limit modularity and behavioural abstractions. In the paper [23], we propose an extension to the SP theory which retains the advantages of deterministic concurrency, but allows communication to occur at higher levels of abstraction than currently supported by SP data types. Our approach is as follows. To avoid data races, each csm type publishes a policy interface for specifying the admissibility and precedence of its access methods. Each instance of the csm type has to be policy-coherent, meaning it must behave deterministically under its own policy—a natural requirement if the goal is to build deterministic sys- tems that use these types. In a policy-constructive system, all access methods can be scheduled in a policy-conformant way for all the types without deadlocking. In this paper, we show that a policy-constructive program exhibits deterministic concurrency in the sense that all policy-conformant interleavings produce the same input-output behaviour. Policies are conservative and support the csm types existing in current SP languages. This work is a follower of a old work we did in 2009, published at LCTES about scheduling policies.

6.8. Compiling synchronous languages for multi-processor implementations

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

This work was performed with industrial partners in the context of the ASSUME project.

We have continued to improve our front-end tools for a use case provided by Airbus. This tool now generates three kinds of monolithic Lustre program, which are taken as an input of the Lopht tool (AOSTE team), which in turn generates an executable for the Kalray MPPA. In particular, one of the code generators is based on the hyper-period-expansion transformation, which unrolls the computation and generates a single step function running at the slowest period. This transformation allows managing the multi-periodic aspect of the application at the source level. Together with the work of the AOSTE team and Airbus, it allows us to execute the full application on a MPPA (TRL-5 Airbus certification level).

We have also improved the front-end tools for the use case provided by Safran. These tools ware integrated into the Heptagon compiler. Many improvements to the parser and many convenient program transformations (tuple and array destruction, equation clustering, ...) were implemented in the Heptagon compiler in order to treat this use case and enable the Lopht tool to extract the best performance. In particular, we have investigated the impact of inlining on the degree of parallelism exposed by the use-case application.

In addition to the work described above, we have defined a language extension for 1-synchronous clocks, strictly periodic clocks with a single activation. We show that we can derive a scheduling problem from the clock constraints in a program. However, solving these constraints by using an interesting cost functions (such as WCET load balancing across the different phases of a period) with an ILP does not scale for the two use cases. Thus, we used the fact that we do not need the optimal solution to fall back on heuristics, which finds a good solution within acceptable bounds. We have also investigated the effect of a non-determinism operator on the scheduling constraints, which gives extra freedom for choosing a schedule.

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

In collaboration with ANSYS, a compilation technique has been designed for compiling SCADE to multi-core [24].

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).

7.2. Bilateral Grants with Industry

In 2018 Francesco Zappa Nardelli was awarded a Google Research Fellowship to pursue the work on DWARF unwinding, about 50k euros.

8. Partnerships and Cooperations

8.1. National Initiatives

The Inria Project Lab (IPL) *Modeliscale* treats the modelling and analysis of Cyber-Physical Systems at large scale. The PARKAS team contributes their expertise in programming language design for reactive and hybrid systems to this multi-team effort.

8.1.1. ANR

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

8.1.2. FUI: Fonds unique interministériel

Modeliscale contract (AAP-24). Using Modelica at scale to model and simulate very large Cyber-Physical Systems. Principal industrial partner: Dassault-Systèmes. 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. H2020 Projects

Program: H2020 "Smart Anything Everywhere (SAE)" initiative

Project acronym: TETRAMAX

Project title: Technology Transfer via Multinational Application Experiments

Duration: September 2017 – August 2021

Coordinator: Rainer Leupers

Other partners: Rheinisch-Westfaelische Technische Hochschule Aachen, RWTH (Germany); AMG Technology Ood, AMGT (Bulgaria); Ruhr-Universitaet Bochum, RUB (Germany); Budapesti Muszaki Es Gazdasagtudomanyi Egyetem, BME (Hungary); Universitat Politecnica De Catalunya, UPC (Spain); Control Data Systems Srl, CDS (Romania); Chalmers Tekniska Hoegskola Ab, CHALMERS, (Sweden); Technische Universiteit Delft, TuDelft (Netherlands); The University Of Edinburgh, UEDIN, (United Kingdom); Fundingbox Accelerator Sp z o.o., FBOX, (Poland); Universiteit Gent, UGENT (Belgium); Vysoka Skola Banska -Technicka Univerzita Ostrava, IT4I, (Czech Republic); Institut Jozef Stefan, JSI, Slovenia, Techmo Spolka z o.o., TECHMO (Poland); Universita Di Pisa, PISA (Italy); Tallinna Tehnikaulikool, TTU (Estonia); Tty-Saatio,TUT (Finland); Think Silicon Ereyna Kai Technologia Anonymi, Etairia, THINKS (Greece); Technische Universitaet Muenchen, TUM (Germany); Sveuciliste U Zagrebu Fakultet Elektrotehnike I Racunarstva, UZAGREB, (Croatia); Zentrum Fur Innovation Und Technik In Nordrhein-Westfalen GmbH, ZENIT (Germany).

Abstract: The overall ambition of TETRAMAX is building and leveraging a European Competence Center Network in customized low-energy computing, providing easy access for SMEs and mid-caps to novel CLEC technologies via local contact points. This is a bidirectional interaction: SMEs can demand CLEC technologies and solutions via the network, and vice versa academic research institutions can actively and effectively offer their new technologies to European industries. Furthermore, TETRAMAX wants to support 50+ industry clients and 3rd parties with innovative technologies, using different kinds of Technology Transfer Experiments (TTX) to accelerate innovation within European industries and to create a competitive advantage in the global economy.

8.2.2. Collaborations in European Programs, Except FP7 & H2020

Program: ITEA3

Project acronym: 14014 ASSUME

Project title: Affordable Safe & Secure Mobility Evolution

Duration: September 2015 – December 2018

Coordinator: Dumitru Potop Butucaru

Other partners: *France:* Airbus, École Normale Supérieure (ENS), Esterel Technologies, Kalray SA, Safran Aircraft Engines SAS SNECMA, Safran Electronics & Defense Sagem, Sorbonne Université, Thales; *Germany:* AbsInt Angewandte Informatik GmbH, Assystem Germany GmbH, BTC Embedded Systems AG, Daimler AG, FZI Forschungszentrum Informatik, Karlsruhe Institute of Technology (KIT), Kiel University, Model Engineering Solutions GmbH, OFFIS, Robert Bosch GmbH, Technical University of Munich; *Netherlands:* Eindhoven University of Technology, NXP Semiconductors Netherlands BV, Recore Systems BV, TNO, University of Twente, VDL Enabling Transport Solutions, Verum Software Tools BV; *Sweden:* Arcticus Systems AB, FindOut Technologies AB, KTH (Royal Institute of Technology), Mälardalen University, Scania; *Turkey:* Arçelik, Ericsson Ar-Ge, Ford Otosan, Havelsan, KoçSistem, UNIT Information Technologies R&D Ltd.

Abstract: Future mobility solutions will increasingly rely on smart components that continuously monitor the environment and assume more and more responsibility for a convenient, safe and reliable operation. Currently the single most important roadblock for this market is the ability to come up with an affordable, safe multi-core development methodology that allows industry to deliver trustworthy new functions at competitive prices. ASSUME will provide a seamless engineering methodology, which addresses this roadblock on the constructive and analytic side.

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).

8.3.2.1. Indo-French Center of Applied Mathematics

POLYFLOW

Title: Polyhedral Compilation for Data-Flow Programming Languages

International Partner (Institution - Laboratory - Researcher):

IISc Bangalore (India) - Uday Kumar Reddy Bondhugula

Duration: 2016 - 2018

Start year: 2016

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.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Selection

9.1.1.1. Member of the Conference Program Committees

- T. Bourke served on the PC of the International Conference on Embedded Software (EMSOFT 2018).
- T. Bourke served on the PC for the Journées Francophones des Langages Applicatifs (JFLA 2018).
- T. Bourke served on the PC of the American Modelica Conference 2018.
- T. Bourke served on the PC of the Japanese Modelica Conference 2018.
- T. Bourke served on the PC of the International Workshop on Software and Compilers for Embedded Systems (SCOPES 2018).
- F. Zappa Nardelli will serve on the PC of OOPSLA 2019 (International Conference on Object-Oriented Programming, Systems, Languages & Applications).
- M. Pouzet served on the PC of the International Workshop on Software and Compilers for Embedded Systems (SCOPES 2018).
- M. Pouzet served on the PC of the International Conference on Embedded Software (EMSOFT 2018).
- M. Pouzet served on the PC of the International Conference on Principles and Practice of Declarative Programming (PPDP 2018).

- M. Pouzet served on the PC of the International Forum on specification & Design Languages (FDL 2018).

9.1.1.2. Reviewer

- T. Bourke reviewed for the International Joint Conference on Automated Reasoning (IJCAR 2018)
- T. Bourke reviewed for the International Conference on Interactive Theorem Proving (ITP 2018)
- T. Bourke reviewed for the International Symposium on Principles and Practice of Declarative Programming (PPDP 2018)

9.1.2. Journal

9.1.2.1. Reviewer - Reviewing Activities

- T. Bourke was a reviewer for Science of Computer Programming.

9.1.3. Invited Talks

- T. Bourke was invited to present a seminar at the Collège de France in January 2018: *La vérification formelle d'un compilateur Lustre*.
- T. Bourke was invited to present an LSV Seminar (Laboratoire Spécification et Vérification) at the ENS Cachan in January 2018: *Compiling a Synchronous Language with Timers for Simulation*
- T. Bourke was invited to present to the Société Informatique de France doctoral seminar at the École normale supérieure in June 2018: *10 lignes de logique pour 1 ligne de code (correct)*
- F. Zappa Nardelli was invited to present to the DeepSpec Workshop in July 2018 on *Debugging Debug Information*.
- T. Bourke and M. Pouzet participated by invitation in the Shonan Meeting 136 on *Functional Stream Libraries and Fusion: What's next?*, in Japan in October 2018.
- M. Pouzet was invited to give a lecture at the International summer school at Marktoberdorf, in July 2018.

9.1.4. Research Administration

- F. Zappa Nardelli will chair the part-time assistant professor recruitment committee at École Polytechinque.

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

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

Master: T. Bourke participated in reviewing the L3 and 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 in progress : Lélio Brun, 3rd year, supervised by T. Bourke and M. Pouzet.

PhD in progress : Chandan Reddy, 3rd year, supervised by A. Cohen.

PhD : Jie Zhao, 3rd year, supervised by A. Cohen, defended in December 2018.

PhD in progress : Basile Clément, 1st year, supervised by F. Zappa Nardelli and A. Cohen.

9.2.3. Juries

- Francesco Zappa Nardelli was jury member of the PhD thesis of Francois Ginraud, Grenoble, Jan 2018.
- T. Bourke was an examiner for the thesis of Jiangchao LIU at the École normale supérieure in February 2018.
- T. Bourke was an examiner for the thesis of Hai NGUYEN VAN at the Université Paris-Sud in September 2018.
- T. Bourke was an examiner for the thesis of Narjes JOMAA at the Université de Lille in December 2018.
- F. Zappa Nardelli was an examiner for the thesis of Jie Zhao at the École normale supérieure in December 2018.

9.3. Popularization

9.3.1. Internal or external Inria responsibilities

- F. Zappa Nardelli is member of the CES of Inria.

10. Bibliography

Major publications by the team in recent years

[1] 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

[2] 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

[3] T. BOURKE, J.-L. COLAÇO, B. PAGANO, C. PASTEUR, M. POUZET. *A Synchronous-based Code Generator For Explicit Hybrid Systems Languages*, in "International Conference on Compiler Construction (CC)", London, United Kingdom, LNCS, July 2015, https://hal.inria.fr/hal-01242732

[4] L. GÉRARD, A. GUATTO, C. PASTEUR, M. POUZET. *A modular memory optimization for synchronous data-flow languages: application to arrays in a lustre compiler*, in "Proceedings of the 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems", Beijing, China, ACM, June 2012, pp. 51–60 [*DOI :* 10.1145/2248418.2248426], https://hal.inria.fr/hal-00728527

[5] J. C. JUEGA, S. VERDOOLAEGE, A. COHEN, J. I. GÓMEZ, C. TENLLADO, F. CATTHOOR. *Patterns for parallel programming on GPUs*, in "Patterns for parallel programming on GPUs", F. MAGOULÈS (editor), Saxe-Cobourg, 2013, vol. Evaluation of State-of-the-Art Parallelizing Compilers Generating CUDA Code for Heterogeneous CPU/GPU Computing, ISBN 978-1-874672-57-9, https://hal.archives-ouvertes.fr/hal-01257261

[6] L. MANDEL, F. PLATEAU, M. POUZET. *Static Scheduling of Latency Insensitive Designs with Lucy-n*, in "FMCAD 2011 - Formal Methods in Computer Aided Design", Austin, TX, United States, October 2011, https://hal.inria.fr/hal-00654843

[7] R. MORISSET, P. PAWAN, F. ZAPPA NARDELLI. *Compiler testing via a theory of sound optimisations in the C11/C++11 memory model*, in "PLDI 2013 - 34th ACM SIGPLAN conference on Programming language design and implementation", Seattle, WA, United States, ACM, June 2013, pp. 187-196 [*DOI :* 10.1145/2491956.2491967], https://hal.inria.fr/hal-00909083

[8] A. POP, A. COHEN. *OpenStream: Expressiveness and Data-Flow Compilation of OpenMP Streaming Programs*, in "ACM Transactions on Architecture and Code Optimization", 2013, vol. 9, n° 4, Selected for presentation at the HiPEAC 2013 Conference [*DOI :* 10.1145/2400682.2400712], https://hal.inria.fr/hal-00786675

[9] J. SEVCIK, V. VAFEIADIS, F. ZAPPA NARDELLI, S. JAGANNATHAN, P. SEWELL. *CompCertTSO: A Verified Compiler for Relaxed-Memory Concurrency*, in "Journal of the ACM (JACM)", 2013, vol. 60, n° 3, pp. art. 22:1-50 [*DOI :* 10.1145/2487241.2487248], https://hal.inria.fr/hal-00909076

[10] V. VAFEIADIS, T. BALABONSKI, S. CHAKRABORTY, R. MORISSET, F. ZAPPA NARDELLI. *Common compiler optimisations are invalid in the C11 memory model and what we can do about it*, in "POPL 2015 - 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages", Mumbai, India, January 2015, https://hal.inria.fr/hal-01089047

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] J. ZHAO. *A Combined Language and Polyhedral Approach for Heterogeneous Parallelism*, PSL Research University, December 2018, https://tel.archives-ouvertes.fr/tel-01988073

Articles in International Peer-Reviewed Journals

[12] A. BENVENISTE, T. BOURKE, B. CAILLAUD, J.-L. COLAÇO, C. PASTEUR, M. POUZET. *Building a Hybrid Systems Modeler on Synchronous Languages Principles*, in "Proceedings of the IEEE", September 2018, vol. 106, n° 9, pp. 1568 - 1592 [*DOI :* 10.1109/JPROC.2018.2858016], https://hal.inria.fr/hal-01879026

[13] T. BOURKE, J. INOUE, M. POUZET. *Sundials/ML: Connecting OCaml to the Sundials Numeric Solvers*, in "Electronic Proceedings in Theoretical Computer Science", December 2018, vol. 285, pp. 101-130, https://arxiv.org/abs/1812.11668 [*DOI :* 10.4204/EPTCS.285.4], https://hal.inria.fr/hal-01967659

[14] S. RAMAKRISHNAN, A. GOHLKE, F. LI, J. COLEMAN, W. XU, J. E. ROTHMAN, F. PINCET. *High-Throughput Monitoring of Single Vesicle Fusion Using Freestanding Membranes and Automated Analysis*, in "Langmuir", May 2018, vol. 34, n° 20, pp. 5849-5859 [*DOI :* 10.1021/ACS.LANGMUIR.8B00116], https://hal.sorbonne-universite.fr/hal-01954048

[15] F. ZAPPA NARDELLI, J. BELYAKOVA, A. PELENITSYN, B. CHUNG, J. BEZANSON, J. VITEK. *Julia Subtyping: A Rational Reconstruction*, in "Proceedings of the ACM on Programming Languages", 2018, vol. 27, OOPSLA, Article 113 [*DOI :* 10.1145/3276483], https://hal.inria.fr/hal-01882137

[16] O. ZINENKO, S. HUOT, C. BASTOUL. *Visual Program Manipulation in the Polyhedral Model*, in "ACM Transactions on Architecture and Code Optimization", March 2018, vol. 15, n° 1, pp. 1 - 25 [*DOI :* 10.1145/3177961], https://hal.inria.fr/hal-01744426

International Conferences with Proceedings

[17] T. BOURKE, L. BRUN, M. POUZET. *Towards a verified Lustre compiler with modular reset*, in "21st International Workshop on Software and Compilers for Embedded Systems (SCOPES 2018)", Sankt Goar, Germany, Proceedings of the 21st International Workshop on Software and Compilers for Embedded Systems (SCOPES 2018), ACM Press, May 2018, 4 p. [*DOI :* 10.1145/3207719.3207732], https://hal.inria.fr/hal-01817949

[18] B. CHUNG, P. LI, F. ZAPPA NARDELLI, J. VITEK. *KafKa: Gradual Typing for Objects*, in "ECOOP 2018 - 2018 European Conference on Object-Oriented Programming", Amsterdam, Netherlands, July 2018, https://hal.inria.fr/hal-01882148

[19] P. FEAUTRIER, A. COHEN, A. DARTE. *On polynomial Code Generation*, in "IMPACT 2018", Manchester, United Kingdom, January 2018, https://hal.inria.fr/hal-01958096

[20] J. SOUYRIS, K. DIDIER, D. POTOP-BUTUCARU, G. IOOSS, T. BOURKE, A. COHEN, M. POUZET. *Automatic Parallelization from Lustre Models in Avionics*, in "ERTS2 2018 - 9th European Congress Embedded Real-Time Software and Systems", Toulouse, France, Proceedings of the 9th European Congress on Embedded Real-Time Software and Systems (ERTS2), 3AF - Association Aéronautique Astronautique de France and SEE - Société de l'électricité, de l'électronique et des technologies de l'information et de la communication and SIA - Société de Ingénieurs de l'Automobile, January 2018, pp. 1-4, https://hal.inria.fr/hal-01714054

[21] J. ZHAO, M. KRUSE, A. COHEN. *A polyhedral compilation framework for loops with dynamic data-dependent bounds*, in "CC'18 - 27th International Conference on Compiler Construction", Vienna, Austria, ACM Press, February 2018 [*DOI :* 10.1145/3178372.3179509], https://hal.inria.fr/hal-01720368

[22] O. ZINENKO, S. VERDOOLAEGE, C. REDDY, J. SHIRAKO, T. GROSSER, V. SARKAR, A. COHEN. *Modeling the conflicting demands of parallelism and Temporal/Spatial locality in affine scheduling*, in "CC'18 - 27th International Conference on Compiler Construction", Vienna, Austria, ACM Press, February 2018 [*DOI :* 10.1145/3178372.3179507], https://hal.inria.fr/hal-01751823

Conferences without Proceedings

[23] J. AGUADO, M. MENDLER, M. POUZET, P. ROOP, R. VON HANXLEDEN. *Deterministic Concurrency: A Clock-Synchronised Shared Memory Approach*, in "ESOP 2018 - European Symposium on Programming", Thessaloniki, Greece, April 2018, https://hal.archives-ouvertes.fr/hal-01960404

[24] J.-L. COLAÇO, B. PAGANO, C. PASTEUR, M. POUZET. *Scade 6: from a Kahn Semantics to a Kahn Implementation for Multicore*, in "Forum on specification & Design Languages (FDL)", Munich, Germany, September 2018, https://hal.archives-ouvertes.fr/hal-01960410

Research Reports

[25] K. DIDIER, D. POTOP-BUTUCARU, G. IOOSS, A. COHEN, J. SOUYRIS, P. BAUFRETON, A. GRAILLAT. *Parallelisation efficace de larges applications temps-reel*, Inria Paris, June 2018, n° RR-9180, https://hal.inria.fr/hal-01810176

[26] O. ZINENKO, L. CHELINI, T. GROSSER. *Declarative Transformations in the Polyhedral Model*, Inria ; ENS Paris - Ecole Normale Supérieure de Paris ; ETH Zurich ; TU Delft ; IBM Zürich, December 2018, n^{o} RR-9243, https://hal.inria.fr/hal-01965599

References in notes

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