IN PARTNERSHIP WITH:
CNRS

Ecole normale supérieure de Paris

Activity Report 2014

Project-Team PARKAS

Parallélisme de Kahn Synchrone

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

RESEARCH CENTER
Paris - Rocquencourt

THEME
Embedded and Real-time Systems

Table of contents

Project-Team PARKAS

Keywords: Compiling, Embedded Systems, Parallelism, Programming Languages, Synchronous Languages

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

1. Members

Research Scientists
- Timothy Bourke [Inria, Starting Research position]
- Albert Cohen [Inria, Senior Researcher, HdR]
- Francesco Zappa Nardelli [Inria, Researcher, HdR]

Faculty Members
- Marc Pouzet [Team leader, Univ. Paris VI, Professor]
- Jean Vuillemin [ENS, Professor, HdR]

Engineers
- Thibaut Balabonski [granted by ANR WMC, until Aug. 2014]
- Jun Inoue [Inria, until Nov 2014, granted by Caisse des Dépôts et Consignations]
- Adarsh Keshan [Inria, until Oct 2014, granted by Min. du Redressement Productif]
- Feng Li [Inria, until Jun 2014, granted by Caisse des Dépôts et Consignations]
- Mircea Namolaru [Inria, granted by Caisse des Dépôts et Consignations]
- Chandan Reddy Gopal [Inria, from Mar 2014, granted by Caisse des Dépôts et Consignations]
- Sven Verdoolaege [Inria, granted by Min. de l'Economie]
- Zhen Zhang [Inria, granted by Min. de l'Economie]

PhD Students
- Riyadh Baghdadi [ENS Paris]
- Guillaume Baudart [ENS Paris]
- Ulysse Beaugnon [ENS Paris, from Mar 2014]
- Andi Drebes [Inria, from Oct 2014]
- Camille Gallet [CEA, until Feb 2016]
- Tobias Grosser [Inria, until Aug 2014, granted by Google UK Ltd]
- Adrien Guatto [ENS Paris]
- Nhat Minh Lê [ENS Paris]
- Robin Morisset [Inria]

Post-Doctoral Fellows
- Pejman Attar [Inria, until Oct 2014, granted by ANR WMC]
- Michael Kruse [ENS Paris, Min. de l'Economie]
- Antoine Madet [Inria, until Jun 2014, granted by Min. de l'Economie]

Administrative Assistants
- Anna Bednarik [Inria, until Oct. 2014]
- Assia Saadi [Inria, from Oct. 2014]

Others
- Quentin Bunel [Inria, Intern, from May 2014 until Jul 2014]
- Yabin Hu [China Nat. Univ. of Defense and Technology, Intern, from Jun 2014 until Aug 2014]
- Abhishek Jain [Inria, Intern, from May 2014]
- Vijay Keswani [Inria, Intern, from May 2014 until Jul 2014]
- Siddharth Prusty [Inria, Intern, from May 2014 until Jul 2014]
- Basile Clément [ENS Paris, from Mar 2014]

Louis Mandel [IBM Watson, USA]

2. Overall Objectives

2.1. Overall Objectives

The goal of the project is the design, semantics and compilation of languages for the implementation of provably safe and efficient computing systems. We are driven by the ideal of a unique source code used both to *program* and *simulate* a wide variety of systems, including (1) embedded real-time controllers (e.g., fly-by-wire, engine control); (2) computationally intensive applications (e.g., video); (3) the simulation of (a possibly huge number of) embedded systems in close interaction (e.g., simulation of electrical or sensor networks, train tracking). All these applications share the need for formally defined languages used both for simulation and the generation of target code. For that purpose, we design languages and experiment with compilers that transform mathematical specifications of systems into target code, that may execute on parallel (multi-core) architectures.

Our research team draws inspiration and focus from the simplicity and complementarity of the data-flow model of Kahn process networks, synchronous concurrency, and the expression of the two in functional languages. To reach our goal, we plan to leverage a large body of formal principles: language design, semantics, type theory, concurrency models (including recent works on the formalisation of relaxed memory models), synchronous circuits and algorithms (code generation, optimization, polyhedral compilation).

3. Research Program

3.1. Presentation and originality of the PARKAS team

Our project is founded on our expertise in three complementary domains: (1) synchronous functional programming and its extensions to deal with features such as communication with bounded buffers and dynamic process creation; (2) mathematical models for synchronous circuits; (3) compilation techniques for synchronous languages and optimizing/parallelizing compilers.

A strong point of the team is its experience and investment in the development of languages and compilers. Members of the team also have direct collaborations for several years with major industrial companies in the field and several of our results are integrated in successful products. Our main results are briefly summarized below.

3.1.1. Synchronous functional programming

In [30], Paul Caspi and Marc Pouzet introduced *synchronous Kahn networks* as those Kahn networks that can be statically scheduled and executed with bounded buffers. This was the origin of the language LUCID SYNCHRONE, [1] [2] an ML extension of the synchronous language LUSTRE with higher-order features, dedicated type systems (clock calculus as a type system [30], [41], initialization analysis [42] and causality analysis [44]). The language integrates original features that are not found in other synchronous languages: such as combinations of data flow, control flow, hierarchical automata and signals [40], [39], and modular code generation [31], [26].

In 2000, Marc Pouzet started to collaborate with the SCADE team of Esterel-Technologies on the design of a new version of SCADE. [3] Several features of LUCID SYNCHRONE are now integrated into SCADE 6, which has been distributed since 2008, including the programming constructs `merge`, `reset`, the clock calculus and the type system. Several results have been developed jointly with Jean-Louis Colaço and Bruno Pagano from Esterel-Technologies, such as ways of combining data-flow and hierarchical automata, and techniques for their compilation, initialization analysis, etc.

[1] http://www.di.ens.fr/~pouzet/lucid-synchrone
[2] The name is a reference to Lustre which stands for "Lucid Synchrone et Temps réel".
[3] http://www.esterel-technologies.com/products/scade-suite/

Dassault-Systèmes (Grenoble R&D center, part of Delmia-automation) developed the language LCM, a variant of LUCID SYNCHRONE that is used for the simulation of factories. LCM follows closely the principles and programming constructs of LUCID SYNCHRONE (higher-order, type inference, mix of data-flow and hierarchical automata). The team in Grenoble is integrating this development into a new compiler for the language Modelica. [4]

In parallel, the goal of REACTIVEML[5] was to integrate a synchronous concurrency model into an existing ML language, with no restrictions on expressiveness, so as to program a large class of reactive systems, including efficient simulations of millions of communicating processes (e.g., sensor networks), video games with many interactions, physical simulations, etc. For such applications, the synchronous model simplifies system design and implementation, but the expressiveness of the algorithmic part of the language is just as essential, as is the ability to create or stop a process dynamically.

The development of REACTIVEML was started by Louis Mandel during his PhD thesis [55], [53] and is ongoing. The language extends OCAML[6] with Esterel-like synchronous primitives — synchronous composition, broadcast communication, pre-emption/suspension — applying the solution of Boussinot [27] to solve causality issues.

Several open problems have been solved by Louis Mandel: the interaction between ML features (higher-order) and reactive constructs with a proper type system; efficient simulation that avoids busy waiting. The latter problem is particularly difficult in synchronous languages because of possible reactions to the absence of a signal. In the REACTIVEML implementation, there is no busy waiting: inactive processes have no impact on the overall performance. It turns out that this enables REACTIVEML to simulate millions of (logical) parallel processes and to compete with the best event-driven simulators [56].

REACTIVEML has been used for simulating routing protocols in ad-hoc networks [52] and large scale sensor networks [67]. The designer benefits from a real programming language that gives precise control of the level of simulation (e.g., each network layer up to the MAC layer) and programs can be connected to models of the physical environment programmed with LUTIN [66]. REACTIVEML is used since 2006 by the synchronous team at VERIMAG, Grenoble (in collaboration with France-Telecom) for the development of low-consumption routing protocols in sensor networks.

3.1.2. *Relaxing synchrony with buffer communication*

In the data-flow synchronous model, the clock calculus is a static analysis that ensures execution in bounded memory. It checks that the values produced by a node are instantaneously consumed by connected nodes (synchronous constraint). To program Kahn process networks with bounded buffers (as in video applications), it is thus necessary to explicitly place nodes that implement buffers. The buffers sizes and the clocks at which data must be read or written have to be computed manually. In practice, it is done with simulation or successive tries and errors. This task is difficult and error prone. The aim of the n-synchronous model is to automatically compute at compile time these values while insuring the absence of deadlock.

Technically, it allows processes to be composed whenever they can be synchronized through a bounded buffer [32], [33]. The new flexibility is obtained by relaxing the clock calculus by replacing the equality of clocks by a sub-typing rule. The result is a more expressive language which still offers the same guarantees as the original. The first version of the model was based on clocks represented as ultimately periodic binary words [73]. It was algorithmically expensive and limited to periodic systems. In [37], an abstraction mechanism is proposed which permits direct reasoning on sets of clocks that are defined as a rational slope and two shifts. An implementation of the n-synchronous model, named LUCY-N, was developed in 2009 [54], as was a formalization of the theory in COQ [38]. We also worked on low-level compiler and runtime support to parallelize the execution of relaxed synchronous systems, proposing a portable intermediate language and runtime library called ERBIUM [57].

[4] http://www.3ds.com/products/catia/portfolio/dymola/overview/
[5] http://rml.lri.fr/
[6] More precisely a subset of OCAML without objects or functors.

This work started as a collaboration between Marc Pouzet (LIP6, Paris, then LRI and Inria Proval, Orsay), Marc Duranton (Philips Research then NXP, Eindhoven), Albert Cohen (Inria Alchemy, Orsay) and Christine Eisenbeis (Inria Alchemy, Orsay) on the real-time programming of video stream applications in set-top boxes. It was significantly extended by Louis Mandel and Florence Plateau during her PhD thesis [61] (supervised by Marc Pouzet and Louis Mandel). Low-level support has been investigated with Cupertino Miranda, Philippe Dumont (Inria Alchemy, Orsay) and Antoniu Pop (Mines ParisTech). Further directions of research and experimentation have been and are being followed through the theses of Léonard Gérard, Adrien Guatto and Nhat Minh Lê.

3.1.3. Polyhedral compilation and optimizing compilers

Despite decades of progress, the best parallelizing and optimizing compilers still fail to extract parallelism and to perform the necessary optimizations to harness multi-core processors and their complex memory hierarchies. *Polyhedral compilation* aims at facilitating the construction of more effective optimization and parallelization algorithms. It captures the flow of data between individual instances of statements in a loop nest, allowing to accurately model the behavior of the program and represent complex parallelizing and optimizing transformations. Affine multidimensional scheduling is one of the main tools in polyhedral compilation [45]. Albert Cohen, in collaboration with Cédric Bastoul, Sylvain Girbal, Nicolas Vasilache, Louis-Noël Pouchet and Konrad Trifunovic (LRI and Inria Alchemy, Orsay) has contributed to a large number of research, development and transfer activities in this area.

The relation between polyhedral compilation and data-flow synchrony has been identified through data-flow array languages [51], [50], [68], [46] and the study of the scheduling and mapping algorithms for these languages. We would like to deepen the exploration of this link, embedding polyhedral techniques into the compilation flow of data-flow, relaxed synchronous languages.

Our previous work led to the design of a theoretical and algorithmic framework rooted in the polyhedral model of compilation, and to the implementation of a set of tools based on production compilers (Open64, GCC) and source-to-source prototypes (PoCC, http://pocc.sourceforge.net). We have shown that not only does this framework simplify the problem of building complex loop nest optimizations, but also that it scales to real-world benchmarks [34], [47], [64], [63]. The polyhedral model has finally evolved into a mature, production-ready approach to solve the challenges of maximizing the scalability and efficiency of loop-based computations on a variety of high performance and embedded targets.

After an initial experiment with Open64 [35], [34], we ported these techniques to GCC [62], [70], [69] and LLVM [49], applying them to multi-level parallelization and optimization problems, including vectorization and exploitation of thread-level parallelism. Independently, we made significant progress in the design of effective optimization heuristics, working on the interactions between the semantics of the compiler's intermediate representation and the structure of the optimization space [64], [63], [65], [23], [60]. These results open opportunities for complex optimizations that target larger problems, such as the scheduling and placement of process networks, or the offloading of computational kernels to hardware accelerators (such as GPUs). A new framework has been designed, centered on the Integer Set Library (isl, http://freecode.com/projects/isl) and implemented through multiple compiler interfaces (Graphite in GCC, Polly in LLVM) and a source-to-source research compiler (PPCG) [72], [36], [48], [71]. This new framework underlies our collaborative research activities in the CARP and COPCAMS European projects, as well as emerging transfer projects through the TETRACOM European coordination action and bilateral industry contracts in preparation.

3.1.4. Automatic compilation of high performance circuits

For both cost and performance reasons, computing systems tightly couple parts realized in hardware with parts realized in software. The boundary between hardware and software keeps moving with the underlying technology and the external economic pressure. Moreover, thanks to FPGA technology, hardware itself has become programmable. There is now a pressing need from industry for hardware/software co-design, and for tools which automatically turn software code into hardware circuits, or more usually, into hybrid code that simultaneously targets GPUs, multiple cores, encryption ASICs, and other specialized chips.

Departing from customary C-to-VHDL compilation, we trust that sharper results can be achieved from source programs that specify bit-wise time/space behavior in a rigorous synchronous language, rather than just the I/O behavior in some (ill-specified) subset of C. This specification allows the designer to also program the (asynchronous) environment in which to operate the entire system, and to profile/measure/control each variable of the design.

At any time, the designer can edit a single specification of the system, from which both the software and the hardware are automatically compiled, and guaranteed to be compatible. Once correct (functionally and with respect to the behavioral specification), the application can be automatically deployed (and tested) on a hard/soft hybrid co-design support.

Key aspects of the advocated methodology were validated by Jean Vuillemin in the design of a PAL2HDTV video sampler [58], [59]. The circuit was automatically compiled from a synchronous source specification, decorated and guided by a few key hints to the hardware back-end, that targetted an FPGA running at real-time video specifications: a tightly-packed highly-efficient design at 240MHz, generated 100% automatically from the application specification source code, and including all run-time/debug/test/validate ancillary software. It was subsequently commercialized on FPGA by LetItWave, and then on ASIC by Zoran. This successful experience underlines our research perspectives on parallel synchronous programming.

4. Application Domains

4.1. Provably safe and efficient computing systems

The project addresses the design, semantics and implementation of programming languages together with compilation techniques to develop provably safe and efficient computing systems. Traditional applications can be found in safety critical embedded systems with hard real-time constraints such as avionics (e.g., fly-by-wire command), railways (e.g., on board control, engine control), nuclear plants (e.g., emergency control of the plant). While embedded applications have been centralized, they are now massively parallel and physically distributed (e.g., sensor networks, train tracking, distributed simulation of factories) and they integrate computationally intensive algorithms (e.g., video processing) with a mix of hard and soft real-time constraints. Finally, systems are heterogeneous with discrete devices communicating with physical ones (e.g., interface between analog and digital circuits). Programming and simulating a whole system from a unique source code, with static guarantees on the reproducibility of simulations together with a compiler to generate target embedded code is a scientific and industrial challenge of great importance.

5. New Software and Platforms

5.1. Lucid Synchrone

Participant: Marc Pouzet [contact].

Synchronous languages, type and clock inference, causality analysis, compilation

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.

It is distributed under binary form, at URL http://www.di.ens.fr/~pouzet/lucid-synchrone/.

The language was used, from 1996 to 2006 as a laboratory to experiment various extensions of the language Lustre. Several programming constructs (e.g. merge, last, mix of data-flow and control-structures like automata), type-based program analysis (e.g., typing, clock calculus) and compilation methods, originaly introduced in Lucid Synchrone are now integrated in the new SCADE 6 compiler developped at Esterel-Technologies and commercialized since 2008.

Three major release of the language has been done and the current version is V3 (dev. in 2006). As of 2014, 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.

5.2. ReactiveML

Participant: Guillaume Baudart [contact].

Programming language, synchronous reactive programming, concurrent systems, dedicated type-systems.

With Louis Mandel (IBM Watson, USA) and Cédric Pasteur.

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.

ReactiveML is distributed at URL http://reactiveml.org. The compiler is distributed under the terms of the Q Public License and the library is distributed under the terms of the GNU Library General Public License. The development of ReactiveML started at the University Paris 6 (from 2002 to 2006).

The language was mainly used for the simulation of mobile ad hoc networks at the Pierre and Marie Curie University and for the simulation of sensor networks at France Telecom and Verimag (CNRS, Grenoble). A new application to mixed music programming has been developed.

5.3. Heptagon

Participants: Adrien Guatto, Marc Pouzet [contact].

Synchronous languages, compilation, optimizing compilation, parallel code generation, behavioral synthesis.

With Cédric Pasteur, Léonard Gérard, and Brice Gelineau.

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.

Some extensions have already been made, most notably automata, a parallel code generator with Futures, support for correct and efficient in-place array computations. It's currently used to experiment with linear typing for arrays and also to introduce a concept of asynchronous parallel computations. The compiler developed in our team generates C, C++, java and VHDL code.

Transfer activities based on our experience in Heptagon are taking place through the "Fiabilité and Sûreté de Fonctionnement" project at IRT SystemX, led by Alstom Transport, since 2013.

Heptagon is jointly developed with Gwenael Delaval and Alain Girault from the Inria POP ART team (Grenoble). Gwenael Delaval is developing the controller synthesis tool BZR (http://bzr.inria.fr/) above Heptagon. Both software are distributed under a GPL licence.

5.4. Lucy-n: an n-synchronous data-flow programming language

Participants: Albert Cohen, Adrien Guatto, Marc Pouzet.

With Louis Mandel (IBM Watson, USA).

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.

The language compiler and associated tools are available in a binary form at http://www.lri.fr/~mandel/lucy-n.

In 2013, a complete re-implementation has been started. This new version will take into account the new features developed during the PhD of Adrien Guatto. Parallel code generation for this new version also involves compilation and runtime system research in collaboration with Nhat Minh Lê and Robin Morisset.

5.5. ML-Sundials

Participants: Timothy Bourke, Jun Inoue, Marc Pouzet [contact].

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.

OCaml versions of the standard examples usually have an overhead of about 50% compared to the original C versions, and almost never more than 100%.

The current version of Sundials/ML comprises about 30,000 lines of OCaml (plus 10,000 lines of api documentation) and 12,000 lines of C (plus 1000 lines of commentary). In comparison to our previous development (called ML-Sundials), the current version includes a major rewrite of the 'nvector' interface to allow easier generalisation to parallel and custom vectors (both of which have now been implemented), a rewrite of the linear solver interfaces, a redesign of the linear solver interface (now including the ability to specify linear solvers in OCaml), and the inclusion of the CVODES and IDAS solvers.

Sundials/ML allows the use of the state-of-the-art Sundials numerical simulation library from OCaml programs. We use it within PARKAS for the Zélus compiler (documented elsewhere) and our ongoing experiments with Modelica. The binding is, however, complete and general purpose. It can potentially replace the less complete libraries underlying three or four open source projects.

The Sundials/ML source code has now been released under a BSD-3 license. It is available on github and through opam.

5.6. Zélus

Participants: Timothy Bourke, Marc Pouzet [contact].

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.

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.

This year we reimplemented several basic numeric solver algorithms after a careful analysis of the Simulink versions together with the binding to SUNDIALS CVODE. This was necessary to enable detailed comparsions between our tool and Simulink (the de facto industrial standard in this domain). We also improved the algorithm for zero-crossing detection, simplified and streamlined the back-end interface.

We developed several new examples to aid in the development, debugging, and dissemination of our work together with various talks and demonstrations. These included a simple backhoe model (which served as a introducing example in the HSCC paper), an adaptive control example from Astrom and Wittenmark's text, and a model of Zeno behaviour based on a zig-zagging object (presented at Synchron).

Zélus was been released officially in 2013 with several complete documented examples on http://zelus.di.ens.fr. Work continued in 2014 with many refinements to the compilation passes. The runtime has also been improved and simplified.

5.7. GCC

Participants: Albert Cohen [contact], Tobias Grosser, Feng Li, Riyadh Baghdadi, Nhat Minh Lê.

Compilation, optimizing compilation, parallel data-flow programming automatic parallelization, polyhedral compilation. http://gcc.gnu.org

Licence: GPLv3+ and LGPLv3+

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% free software, free in the sense that it respects the user's freedom.

PARKAS contributes to the polyhedral compilation framework, also known as Graphite. We also distribute an experimental branch for a stream-programming extension of OpenMP called OpenStream (used in numerous research activites and grants). This effort borrows key design elements to synchronous data-flow languages.

Tobias Grosser is one of main contributors of the Graphite optimization pass of GCC.

5.8. isl

Participants: Sven Verdoolaege [contact], Tobias Grosser, Albert Cohen.

Presburger arithmetic, integer linear programming, polyhedral library, automatic parallelization, polyhedral compilation. http://freshmeat.net/projects/isl

Licence: MIT

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.

isl has become the de-facto standard for every recent polyhedral compilation project. Thanks to a license change from LGPL to MIT, its adoption is also picking up in industry.

5.9. ppcg

Participants: Sven Verdoolaege [contact], Tobias Grosser, Riyadh Baghdadi, Albert Cohen.

Presburger arithmetic, integer linear programming, polyhedral library, automatic parallelization, polyhedral compilation. http://freshmeat.net/projects/ppcg

Licence: MIT

More tools are being developed, based on isl. 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).

5.10. Tool support for the working semanticist

Participants: Basile Clément, Francesco Zappa Nardelli [contact].

Languages, semantics, tool support, theorem prouvers.

We are working on tools to support large scale semantic definitions, for programming languages and architecture specifications. For that we develop two complementary tools, Ott and Lem.

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:

1. a LaTeX source file that defines commands to build a typeset version of the definition;
2. a Coq version of the definition;
3. an Isabelle version of the definition; and
4. 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.

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.

In collaboration with Peter Sewell (Cambridge University) and Scott Owens (University of Kent).

The current version of Ott is about 30000 lines of OCaml. The tool is available from http://moscova.inria.fr/~zappa/software/ott (BSD licence). It is widely used in the scientific community.

The development version of Lem is available from http://www.cs.kent.ac.uk/people/staff/sao/lem/.

In addition to the usual bug-fixes, in 2014 we have investigated several approaches to interactively explore a semantics definition, with the aim of building a toolbox to debug operational semantics and to attempt to falsify expected properties. This code is not yet released.

5.11. Cmmtest: a tool for hunting concurrency compiler bugs

Participants: Francesco Zappa Nardelli [contact], Robin Morisset, Pejman Attar.

Languages, concurrency, memory models, C11/C++11, 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:

1. compiles the program using the compiler and compiler optimisations that are being tested;
2. runs the compiled program in an instrumented execution environment that logs all memory accesses to global variables and synchronisations;
3. 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.

Cmmtest is available from http://www.di.ens.fr/~zappa/projects/cmmtest/ and a list of bugs reported thanks to cmmtest is available from http://www.di.ens.fr/~zappa/projects/cmmtest/gcc-bugs.html.

In 2014 Cmmtest has been used by the ThreadSanitizer team at Google to debug some subtle false positive race reports, due to the compiler introducing memory accesses.

6. New Results

6.1. Highlights of the Year

The paper *ReactiveML, a reactive extension to ML* of Mandel and Pouzet has been declared to be the *most influential paper of PPDP (Principles and Practice of Declaractive Programming) 2005*. A previous version of the paper, submitted to JFLA'05, has been declared to be "une contribution marquante parmi les articles publiés aux JFLA".

6.2. Quasi-synchrony

Participants: Guillaume Baudart, Timothy Bourke, Marc Pouzet.

We study the implementation of critical control applications on the so-called *quasi-periodic* distributed architectures. These architectures, used in civil avionics (e.g., Airbus A380), consist of a collection of distributed processors running with *quasi-periodic* clocks, that is, un-synchronized physical clocks subject to bounded jitterring. The theory of quasi-synchrony has been introduced by Paul Caspi in the 2000' [29]. Loosely Time-Triggered Architectures (LTTA) denotes such architectures with the prototocol used to implement a synchronous program on top of it.

Over the last ten year two protocols were considered: (1) *Back-Pressure* LTTA [25] based on a acknowledgement mechanism reminiscent of elastic circuit [43]. (2) *Time-Based* LTTA [28] which uses timing constraints of the architecture to mimic a synchronous execution.

During year 2014, we have entirelly reformulated the model of LTTA using synchronous semantics and principles. Compared to previous formalizations based on Petri nets [24], this new presentation is is simpler and more uniform with the same theoretical model used for both the application and the protocol ((1) or (2)). Moreover, it is easier to consider mixed protocols (a whole application with part based on time-based communication and others based on back-pressure). Besides this, we also proposed a new and more flexible Time-Based LTTA, allowing for pipelining by not reconstructing global synchronization, unlike what was done in previous Time-Based LTTA.

6.3. Hybrid Synchronous Languages

Participants: Guillaume Baudart, Timothy Bourke, Marc Pouzet.

During year 2014, we mainly worked on two directions: (a) the design and implementation of causality analysis for hybrid systems modelers; (b) the design and implementation of a new compilation technique producing imperative sequential code.

This research is conducted in collaboration with Albert Benveniste and Benoit Caillaud (Hycomes team at Inria, Rennes), Jean-Louis Colaco, Cédric Pasteur and Bruno Pagano from the SCADE core team of Esterel-Technologies/ANSYS.

Causality analysis In this work, we address the static detection of causality loops for a hybrid modeling language that combines synchronous Lustre-like data-flow equations with Ordinary Differential Equations (ODEs). We introduce the operator *last(x)* for the left-limit of a signal *x*. This operator is used to break causality loops and permits a uniform treatment of discrete and continuous state variables. The semantics relies on non-standard analysis, defining an execution as a sequence of infinitesimally small steps. A signal is deemed *causally correct* when it can be computed sequentially and only progresses by infinitesimal steps outside of discrete events. The causality analysis takes the form of a simple type system. In well-typed programs, signals are proved continuous during integration.

This analysis has been presented at [4] and is fully implemented in the hybrid synchronous language Zélus.

A Synchronous-based Code Generator For Explicit Hybrid Systems Languages The generation of sequential code is important for simulations to be efficient and to produce target embedded code. While sequential code generation in hybrid modeling tools is routinely used for efficient simulation, it is little or not used for producing target embedded code in critical applications submitted to strong safety requirements. This is a break in the development chain: parts of the applications must be rewritten into either sequential or synchronous programs, and all properties verified on the source model cannot be trusted and have to be re-verified on the target code.

In this work, we present a novel approach for the code generation of a hybrid systems modeling language. By building on top of an existing synchronous language and compiler, it reuses almost all the existing infrastructure with only a few modifications. Starting from an existing synchronous data-flow language extended with Ordinary Differential Equations (ODEs), we detail the translation to sequential code. The translation is expressed as a sequence of source-to-source transformations. A generic intermediate language is introduced to represent transition functions which are turned into C code. The versatility of the compiler organisation is illustrated by considering two classical targets: generation of simulation code complying with the FMI standard and linking with an off-the-shelf numerical solver (Sundials CVODE).

This new code generation has been implemented in two different compilers: the Zélus research prototype and the industrial SCADE Suite KCG code generator, at Esterel-Technologies/ANSYS. Here, SCADE is conservatively extended with ODEs, following previous works by Benveniste et al. and implemented in Zélus. In the SCADE compiler, it was possible to reuse almost all the existing infrastructure like static checking, intermediate languages, and optimisations, with few modifications. The extension to account for hybrid features represents only 5% additional lines of code, which is surprisingly low. Moreover, the proposed language extension is conservative in that regular synchronous functions are compiled as before—the same synchronous code is used both for simulation and for execution on target platforms.

This full-scale validation confirm the interest in building a hybrid systems modeler on top of a existing synchronous language. Moreover, the precise definition of code generation, built on a proven compiler infrastructure of a synchronous language avoids the rewriting of control software and may also increase the confidence in what is simulated.

This work will be presented at the *International Conference on Compiler Construction (CC)*, in April 2015.

6.4. Fidelity in Real-Time Programming

Participants: Guillaume Baudart, Timothy Bourke.

Synchronous languages are a rigorous approach to programming, analyzing, and implementing embedded systems. Real-time aspects are typically handled by discretizing time using either (implicit) ticks or (explicit) named signals, and later verifying that the (necessarily bounded) execution time of a reaction is strictly less than the period of the fastest timing signal. This approach has many advantages: it separates logical behaviour from implementation concerns, yields a simple and precise programming model, and abstracts from eventual run-time environments. For an important subclass of embedded protocols and controllers, however, we believe it advantageous to add constructions that deal more concretely with real-time constraints.

We are pursuing these ideas in the enriched timing model provided by the Zélus programming language (detailed elsewhere). We continue to study the extension and application of this language to the modelling, simulation, analysis, and implementation of real-time embedded software.

This year we developed three case studies: quasi-synchronous architectures (from last year), loosely time-triggered architectures (detailed elsewhere), and a small embedded controller. These case studies motivate and drive our research and implicitly define the subclass of embedded systems that we aim to treat. They have each been modelled in Zélus and can be simulated with the existing compiler.

We made progress on defining a subset of Zélus that is ammenable to discretization techniques for more flexible simulation. A first version of an appropriate algorithm has been sketched and partially implemented. Work continues on developing it with the idea of incorporating it into the Zélus compiler and using it to treat our case studies.

6.5. Mechanization of AODV loop freedom proof

Participant: Timothy Bourke.

The Ad hoc On demand Distance Vector (AODV) routing protocol is described in RFC3561. It allows the nodes in a Mobile Ad hoc Network (MANET) to know where to forward messages so that they eventually reach their destinations. The nodes of such networks are *reactive systems* that cooperate to provide a global service (the sending of messages from node to node) satisfying certain correctness properties (namely ‘loop freedom’—that messages are never sent in circles).

This year I finalized both the framework for network invariant proofs [20] and its application to the AODV protocol [21] and submitted them for inclusion in the *Archive of Formal Proof*, an online and open-source repository of formal developements in the Isabelle proof assistant (indexed as a journal). I presented results on the framework at the Vienna ‘Summer of Logic’ [6] and my colleagues presented the application in Sydney [5]. Together with an intern at NICTA and Sydney, my colleagues and I made preliminary investigations into extending the framework and model with timing details. A journal version of the ITP paper has been submitted.

In collaboration with Peter Höfner (NICTA) and Robert J. van Glabbeek (UNSW/NICTA).

6.6. Reasoning about C11 Program Transformations

Participants: Francesco Zappa Nardelli, Thibaut Balabonski, Robin Morisset.

We have shown that the weak memory model introduced by the 2011 C and C++ standards does not permit many of common source-to-source program transformations (such as expression linearisation and "roach motel" reordering) that modern compilers perform and that are deemed to be correct. As such it cannot be used to define the semantics of intermediate languages of compilers, as, for instance, LLVM aimed to. We consider a number of possible local fixes, some strengthening and some weakening the model. We have evaluated the proposed fixes by determining which program transformations are valid with respect to each of the patched models. We have provided formal Coq proofs of their correctness or counterexamples as appropriate.

A paper on this work has been accepted in [13]. In collaboration with Viktor Vafeiadis (MPI-SWS, Germany).

6.7. Language design on top of JavaScript

Participant: Francesco Zappa Nardelli.

This research project aims at improving the design of the JavaScript language. In [22] we propose a typed extension of JavaScript combining dynamic types, concrete types and like types to let developers pick the level of guarantee that is appropriate for their code. We have implemented our type system and we have explored the performance and software engineering benefits.

With Gregor Richards and Jan Vitek (Purdue University).

6.8. Tiling for Stencils

Participants: Tobias Grosser, Sven Verdoolaege, Albert Cohen.

This research project aims with optimizing time-iterated stencil operations.

Iterative stencil computations are important in scientific computing and more and more also in the embedded and mobile domain. Recent publications have shown that tiling schemes that ensure concurrent start provide efficient ways to execute these kernels. Diamond tiling and hybrid-hexagonal tiling are two tiling schemes that enable concurrent start. Both have different advantages: diamond tiling has been integrated in a general purpose optimization framework and uses a cost function to choose among tiling hyperplanes, whereas the greater flexibility with tile sizes for hybrid-hexagonal tiling has been exploited for effective generation of GPU code.

We undertook a comparative study of these two tiling approaches and proposed a hybrid approach that combines them. We analyzed the effects of tile size and wavefront choices on tile-level parallelism, and formulate constraints for optimal diamond tile shapes. We then extended, for the case of two dimensions, the diamond tiling formulation into a hexagonal tiling one, which offers both the flexibility of hexagonal tiling and the generality of the original diamond tiling implementation. We also showed how to compute tile sizes that maximize the compute-to-communication ratio, and apply this result to compare the best achievable ratio and the associated synchronization overhead for diamond and hexagonal tiling.

One particularly exciting result is the ability to apply tiling to periodic data domains. These computations are prevalent in computational sciences, particularly in partial differential equation solvers. We proposed a fully automatic technique suitable for implementation in a compiler or in a domain-specific code generator for such computations. Dependence patterns on periodic data domains prevent existing algorithms from finding tiling opportunities. Our approach augments a state-of-the-art parallelization and locality-enhancing algorithm from the polyhedral framework to allow time-tiling of stencil computations on periodic domains. Experimental results on the swim SPEC CPU2000fp benchmark show a speedup of $5\times$ and $4.2\times$ over the highest SPEC performance achieved by native compilers on Intel Xeon and AMD Opteron multicore SMP systems, respectively. On other representative stencil computations, our scheme provides performance similar to that achieved with no periodicity, and a very high speedup is obtained over the native compiler. We also report a mean speedup of about $1.5\times$ over a domain-specific stencil compiler supporting limited cases of periodic boundary conditions. To the best of our knowledge, it has been infeasible to manually reproduce such optimizations on swim or any other periodic stencil, especially on a data grid of two-dimensions or higher.

These works resulted in a number of high-profile publications, including a nommination for a best paper award, and culminated with the PhD thesis defense of Tobias Grosser.

6.9. Portable representation for polyhedral compilation

Participants: Riyadh Baghdadi, Michael Kruse, Chandan Reddy, Tobias Grosser, Sven Verdoolaege, Albert Cohen.

Programming accelerators such as GPUs with low-level APIs and languages such as OpenCL and CUDA is difficult, error prone, and not performance-portable. Automatic parallelization and domain specific languages have been proposed to hide this complexity and to regain some performance portability. We proposed PENCIL, a subset of GNU C99 with specific programming rules. A compiler for a Domain-Specific Language (DSL) may use it as a target language, a domain expert may use it as a portable implementation language facilitating the parallelization of real-world applications, and an optimization expert may use PENCIL to accelerate legacy applications.

The design of PENCIL is simultaneously a key research result and a milestone for parallelizing compiler engineering/design. Aspects of its static-analysis-friendly, formal semantics are highly original, for the language's ability to preserve expressiveness and modularity without jeopardizing a (polyhedral) compiler's ability to perform aggressive transformations. We validated its potential as a front-end to a state-of-the-art polyhedral compiler, extending its applicability to dynamic, data dependent control flow and non-affine array accesses. We illustrated this PENCIL-enabled flow on the generation of highly optimized OpenCL code, considering a set of standard benchmarks (Rodinia and SHOC), image processing kernels, and DSL embedding scenarios for linear algebra (BLAS) and signal processing radar applications (SPEAR-DE). We ran experimental results on a variety of platforms, including an AMD Radeon HD 5670 GPU, an Nvidia GTX470 GPU, and an ARM Mali-T604 GPU.

This work is conducted in collaboration with partners from ARM, RealEyes (a computer vision company) and Imperial College.

6.10. Correct and efficient runtime systems

Participants: Nhat Minh Lê, Robin Morisset, Adrien Guatto, Albert Cohen.

Complementing our different compilation efforts for synchronous and task-parallel data-flow languages, we studied the implementation of Kahn process networks, a deterministic parallel programming model, on shared memory multiprocessors. This model is based on a familiar abstraction: blocking communication through bounded, in-order, single-producer single-consumer queues.

We proposed two novel algorithms that construct such blocking queues on top of concurrent ring buffers and user-land scheduling components. We implemented our algorithms in C11, taking advantage of the relaxed memory model of the language, and prove the correctness of this implementation.

We used these algorithms in a complete runtime system for Kahn process networks with applications ranging from linear algebra kernels to stream computing. In particular, our implementations of the Cholesky and LU factorizations outperform state-of-the-art parallel linear algebra libraries on commodity x86 hardware.

6.11. A Functional Synchronous Language with Integer Clocks

Participants: Adrien Guatto, Albert Cohen, Louis Mandel, Marc Pouzet.

Synchronous languages in the vein of Lustre are first-order functional languages dedicated to stream processing. Lustre compilers use a type-like static analysis, the clock calculus, to reject programs that cannot be implemented as finite state machines. The broad idea is to assign to each element of a stream a logical computation date in a global, discrete time scale. When this analysis succeeds, the types obtained guide the code generation phase of the compiler, which produces transition functions. In practice, these functions consists in simple, bounded memory C code featuring only assignments and conditional statements.

This research work explores a variation on Lustre and its compilation. Our proposal is twofold. First, we add a new construct that creates a local time scale whose internal steps are invisible from the outside. Second, we change the clock calculus to allow several elements of a stream to be computed during the same time step. The resulting type system comes with a soundness proof, which relies on an elementary form of step-indexed realizability, and with a code generation scheme adapted to the new setting, and featuring nested loops in the target code.

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR

ANR WMC project (program "jeunes chercheuses, jeunes chercheurs"), 2012–2016, 200 Keuros. F. Zappa Nardelli is the main investigator.

ANR Boole project (program "action blanche"), 2009-2014.

ANR CAFEIN, 2013-2015. Marc Pouzet.

7.1.2. 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).

ManycoreLabs contract (Briques Génériques du Logiciel Embarqué). Partenaire principal: Kalray. Inria contacts are Albert Cohen (PARKAS, Paris), Alain Darte (COMPSYS, Lyon), Fabrice Rastello (CORSE, Grenoble).

7.1.3. Others

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

7.2. European Initiatives

7.2.1. FP7 & H2020 Projects

7.2.1.1. COPCAMS

Type: ARTEMIS JU

Defi: NC

Instrument: ASP

Objectif: NC

Duration: April 2013 - March 2016

Coordinator: Christian Fabre

Partner: CEA LETI, Grenoble, France

Inria contact: Albert Cohen

Abstract: Cognitive cameras on manycore platforms

7.2.1.2. EMC2

Type: ARTEMIS JU

Defi: NC

Instrument: AIPP

Objectif: NC

Duration: April 2014 - March 2917

Coordinator: Werner Weber

Partner: Infineon, Munich ,Germany

Inria contact: Albert Cohen

Abstract: Embedded multicrical systems on multicores

7.2.1.3. ITEA2

Type: ITEA2

Defi: NC

Instrument: NC

Objectif: NC

Duration: September 2012 - November 2015

Coordinator: Daniel Bouskela (EDF)

Partner: Dassault-Systèmes, EDF, Modelon, DLR (Germany)

Inria contact: Benoit Caillaud, Marc Pouzet

Abstract: Model Driven Physical Systems Operation

7.3. International Initiatives

7.3.1. Inria Associate Teams

7.3.1.1. POLYFLOW

Title: Polyhedral Compilation for Data-Flow Programming Languages

International Partner (Institution - Laboratory - Researcher):

IISc Bangalore (INDE)

Duration: 2013 - 2015/12

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

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 such as C, where computation is specified in terms of statements with zero or more nested loops and other control structures around them. 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 system analysis, modeling and design, in embedded reactive control. They also underline the construction of many domain-specific languages and compiler intermediate representations. The copy and execution semantics of data-flow languages impose a different set of challenges. We plan to bridge this gap by studying techniques that could enable extraction of a polyhedral representation from data-flow programs, transform them, and synthesize them from their equivalent polyhedral representation.

7.4. International Research Visitors

7.4.1. Visits of International Scientists

Prof. Cesare Tinelli, was invited by ENS in the PARKAS team.

Date: June 2014 (one month)

Institution: Iowa State University, USA.

7.4.1.1. Internships

Siddharth Prusty Siddharth

Date: May 2014 - Jul 2014

Institution: IITK (India)

Vijay Keswani Vijay

Date: May 2014 - Jul 2014

Institution: IITK (India)

Quentin Bunel

Date: May 2014 - Jul 2014

Institution: UPMC (France)

Abhishek Jain

Date: May 2014 - Jul 2014 and Dec 2014 - Jan 2015

Institution: IITD (India)

Yabin Hu

Date: Jun 2014 - Jul 2014

Institution: China Nat. Univ. of Defense and Technology (China)

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific events organisation

8.1.1.1. General chair, scientific chair

Albert Cohen was the general chair of PPoPP 2015, and will be the general chair of PLDI 2017 (associated with ECRC).

8.1.2. Scientific events selection

8.1.2.1. Chair of conference program committee

Albert Cohen was the program chair of CC 2014, the TPC chair of the DAC 2014 ESS1 subcommittee, and the program chair of ARCS 2015.

8.1.2.2. Member of the conference program committee

Albert Cohen was a member of the PC of PLDI 2014, of the ERC of PLDI 2015 and PACT 2015, of the ERC of ASPLOS 2014, PPoPP 2014 and ICS 2014.

Marc Pouzet was a member of the PC of DAC (Design Automation Conference) 2014, EOOLT (Equation-Based Object-Oriented Modeling Languages and Tools) 2014, AFADL (Approches Formelles dans l'Assistance au Développement de Logiciels) 2014.

Francesco Zappa Nardelli was a member of the PC of PPoPP (Principles and Practice of Concurrent Programming) 2014 and ESOP (European symposium on programming) 2014.

8.1.3. Journal

8.1.3.1. Member of the editorial board

Albert Cohen is an associate editor of ACM TACO and IJPP (Springer).

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

Master: A. Cohen: “Operating Systems Principles and Programming” (M1), 38h, École Polytechnique, France

Master: A. Cohen & F. Zappa Nardelli, “Semantics, languages and algorithms for multicore programming”, Lecture, 9h+14h, M2, MPRI: Ecole normale supérieure and Université Paris Diderot, France

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

Licence: A. Cohen, “Components of a Computing System Introduction to Computer Architecture and Operating Systems” (L3), 44h, École Polytechnique, France

Licence : F. Zappa Nardelli: “Introduction à l’informatique” (L3), TDs, 40h, École Polytechnique, France

Licence : M. Pouzet: “Operating Systems” (L3), Lectures, ENS, France.

Licence : J. Vuillemin & T. Bourke, “Digital Systems”, Lectures and TDs, ENS, France

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

8.2.2. Supervision

HdR : F. Zappa Nardelli, Reasoning between programming languages and architectures, ENS, January 2014

PhD : F. Li, Compiling for a multithreaded dataflow architecture: algorithms, tools, and experience. Université Pierre et Marie Curie - Paris VI, May 2014

PhD in progress : Guillaume Baudart, 2nd year, supervised by T. Bourke and M. Pouzet

PhD in progress : Adrien Guatto, 4th year, supervised by T. Bourke and A. Cohen and M. Pouzet

PhD in progress : Robin Morisset, 2nd year, supervised by F. Zappa Nardelli

M2 Internship : Ulysse Beaugnon (MPRI), supervised by A. Cohen and M. Pouzet.

M2 Internship : Basile Clement (MPRI), supervised by A. Cohen and M. Pouzet.

8.2.3. Juries

- Albert Cohen was a reviewer in the PhD thesis committees of Jarryd Beck (2015, U. New South Wales, Australia), Pranav Tendulkar (2014, U. Grenoble), Cedric Nugteren (2014, T. U. Eindhoven, The Netherlands).
- Albert Cohen was a president in the PhD thesis committees of Sofiane Lagraa (2014, U. Grenoble), Bruno Bodin (2014, UPMC).
- Albert Cohen was examiner in the PhD thesis committee of Victor Lomüller (2014, U. Grenoble).
- Marc Pouzet was president of the Jury for the habilitation of Anne Bouillard, April 8, 2014.
- Marc Pouzet was jury member of the PhD. thesis of Arlen Cox, Universtity of Colorado, Boulder, November 17, 2014.
- Marc Pouzet was member of the committee for the AERES evaluation of the Lab. of SupAero (ISAE), Toulouse, Nov. 2014.
- Marc Pouzet was member of the Prix de thèse du GDR GPL (Génie de la Programmation et du Logiciel).

8.3. Popularization

T. Bourke hosted a tutorial by Makarius Wenzel at ENS Ulm on the Isabelle proof assistant.

9. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] F. LI. *Compiling for a multithreaded dataflow architecture : algorithms, tools, and experience*, Université Pierre et Marie Curie - Paris VI, May 2014, https://tel.archives-ouvertes.fr/tel-00992753

[2] F. ZAPPA NARDELLI. *Reasoning between Programming Languages and Architectures*, ENS Paris - Ecole Normale Supérieure de Paris, January 2014, Habilitation à diriger des recherches, https://hal.inria.fr/tel-01110117

Articles in International Peer-Reviewed Journals

[3] R. GIORGI, R. M. BADIA, F. BODIN, A. COHEN, P. EVRIPIDOU, P. FARABOSCHI, B. FECHNER, G. R. GAO, A. GARBADE, R. GAYATRI, S. GIRBAL, D. GOODMAN, B. KHAN, S. KOLIAÏ, J. LANDWEHR, N. MINH, F. LI, M. LUJÀN, A. MENDELSON, L. MORIN, N. NAVARRO, T. PATEJKO, A. POP, P. TRANCOSO, T. UNGERER, I. WATSON, S. WEIS, S. ZUCKERMAN, M. VALERO. *TERAFLUX: Harnessing dataflow in next generation teradevices*, in "Microprocessors and Microsystems", 2014, pp. 976-990, Available online 18 April 2014 [*DOI :* 10.1016/J.MICPRO.2014.04.001], https://hal.inria.fr/hal-00992721

International Conferences with Proceedings

[4] A. BENVENISTE, B. CAILLAUD, B. PAGANO, M. POUZET. *A type-based analysis of causality loops in hybrid modelers*, in "HSCC '14: International Conference on Hybrid Systems: Computation and Control", Berlin, Germany, Proceedings of the 17th international conference on Hybrid systems: computation and control (HSCC '14), ACM Press, April 2014, 13 p. [*DOI :* 10.1145/2562059.2562125], https://hal.inria.fr/hal-01093388

[5] T. BOURKE, R. J. VAN GLABBEEK, P. HÖFNER. *A Mechanized Proof of Loop Freedom of the (Untimed) AODV Routing Protocol*, in "ATVA 2014: Automated Technology for Verification and Analysis", Sydney, Australia, Lecture Notes in Computer Science, Springer, November 2014, vol. 8837, 17 p. [*DOI :* 10.1007/978-3-319-11936-6_5], https://hal.inria.fr/hal-01092360

[6] T. BOURKE, R. J. VAN GLABBEEK, P. HÖFNER. *Showing Invariance Compositionally for a Process Algebra for Network Protocols*, in "ITP 2014: Interactive Theorem Proving", Vienna, Austria, Lecture Notes in Computer Science, Springer, July 2014, vol. 8558, 16 p. [*DOI :* 10.1007/978-3-319-08970-6_10], https://hal.inria.fr/hal-01092348

[7] A. DELPEUCH, A. PRELLER. *From Natural Language to RDF Graphs with Pregroups*, in "EACL'2014: 14th Conference of the European Chapter of the Association for Computational Linguistics", Gothenburg, Sweden, EACL, April 2014, pp. 55-62, http://hal-lirmm.ccsd.cnrs.fr/lirmm-00992381

[8] T. GROSSER, A. COHEN, J. HOLEWINSKI, P. SADAYAPPAN, S. VERDOOLAEGE. *Hybrid Hexagonal/Classical Tiling for GPUs*, in "Intl. Symp. on Code Generation and Optimization (CGO)", Orlando, FL, United States, February 2014, https://hal.inria.fr/hal-00911177

[9] N. HILI, C. FABRE, S. DUPUY-CHESSA, D. RIEU, I. LLOPARD. *Model-Based Platform Composition for Embedded System Design*, in "2014 IEEE 8th International Symposium on Embedded Multicore/Manycore SoCs", Aizu-Wakamatsu, Japan, University of Aizu, September 2014, https://hal.inria.fr/hal-01071208

[10] I. LLOPARD, A. COHEN, C. FABRE, N. HILI. *A Parallel Action Language for Embedded Applications and its Compilation Flow*, in "17th International Workshop on Software and Compilers for Embedded Systems", Sankt Goar, Germany, Proceedings of the 17th International Workshop on Software and Compilers for Embedded Systems, June 2014, pp. 118-127 [*DOI :* 10.1145/2609248.2609257], https://hal.inria.fr/hal-01001900

[11] L. MANDEL, C. PASTEUR. *Reactivity of Cooperative Systems*, in "Static Analysis (SAS)", Munich, Germany, Lecture Notes in Computer Science, Springer, September 2014, vol. 8723, 17 p. [*DOI :* 10.1007/978-3-319-10936-7_14], https://hal.inria.fr/hal-01093169

[12] K. STOCK, M. KONG, T. GROSSER, L.-N. POUCHET, F. RASTELLO, J. RAMANUJAM, P. SADAYAPPAN. *A Framework for Enhancing Data Reuse via Associative Reordering*, in "PLDI '14 - 35th ACM SIGPLAN Conference on Programming Language Design and Implementation", Edinburgh, United Kingdom, ACM, June 2014, pp. 65-76 [*DOI :* 10.1145/2594291.2594342], https://hal.inria.fr/hal-01016093

[13] 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

[14] S. VERDOOLAEGE, S. GUELTON, T. GROSSER, A. COHEN. *Schedule Trees*, in "IMPACT - 4th Workshop on Polyhedral Compilation Techniques, associated with HiPEAC", Vienna, Austria, ACM, January 2014, https://hal.inria.fr/hal-00911894

National Conferences with Proceedings

[15] A. GUATTO, L. MANDEL. *Réseaux de Kahn à rafales et horloges entières*, in "JFLA 2014 - Vingt-cinquièmes Journées Francophones des Langages Applicatifs", Fréjus, France, January 2014, https://hal.inria.fr/hal-00919281

[16] L. MANDEL, C. PASTEUR. *Exécution efficace de programmes ReactiveML*, in "JFLA 2014 - Vingt-cinquièmes Journées Francophones des Langages Applicatifs", Fréjus, France, January 2014, https://hal.inria.fr/hal-00919271

Research Reports

[17] G. BAUDART, A. BENVENISTE, A. BOUILLARD, P. CASPI. *A Unifying View of Loosely Time-Triggered Architectures*, March 2014, n^{o} RR-8494, 14 p. , https://hal.inria.fr/hal-00955496

[18] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *On the index of multi-mode DAE Systems (also called Hybrid DAE Systems)*, Inria ; ENS, November 2014, n^{o} RR-8630, 30 p. , https://hal.inria.fr/hal-01084069

[19] L. MANDEL, C. PASTEUR. *Reactivity of Cooperative Systems: Application to ReactiveML – extended version*, June 2014, n^{o} RR-8549, 29 p. , https://hal.inria.fr/hal-01010349

Other Publications

[20] T. BOURKE. *Mechanization of the Algebra for Wireless Networks (AWN)*, August 2014, 186 p. , Entry in the Archive of Formal Proofs (ISSN: 2150-914x), https://hal.inria.fr/hal-01104031

[21] T. BOURKE, P. HÖFNER. *Loop freedom of the (untimed) AODV routing protocol*, October 2014, 496 p. , Entry in the Archive of Formal Proofs (ISSN: 2150-914x), https://hal.inria.fr/hal-01104033

[22] G. RICHARDS, F. ZAPPA NARDELLI, J. VITEK. *Concrete Types for JavaScript*, 2014, forthcoming, https://hal.inria.fr/hal-00909092

References in notes

[23] R. BAGHDADI, A. COHEN, S. VERDOOLAEGE, K. TRIFUNOVIĆ. *Improved Loop Tiling based on the Removal of Spurious False Dependences*, in "ACM Transactions on Architecture and Code Optimization", 2013, vol. 9, n^{o} 4, Selected for presentation at the HiPEAC 2013 Conf [*DOI :* 10.1145/2400682.2400711], https://hal.inria.fr/hal-00786674

[24] A. BENVENISTE, A. BOUILLARD, P. CASPI. *A unifying view of loosely time-triggered architectures*, in "EMSOFT", 2010, pp. 189–198

[25] A. BENVENISTE, P. CASPI, M. DI NATALE, C. PINELLO, A. SANGIOVANNI-VINCENTELLI, S. TRIPAKIS. *Loosely time-triggered architectures based on communication-by-sampling*, in "EMSOFT", 2007, pp. 231–239

[26] D. BIERNACKI, J.-L. COLAÇO, G. HAMON, M. POUZET. *Clock-directed Modular Code Generation of Synchronous Data-flow Languages*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)", Tucson, Arizona, June 2008

[27] F. BOUSSINOT, R. DE SIMONE. *The SL synchronous language*, in "IEEE Transaction on Software Engineering", 1996

[28] P. CASPI, A. BENVENISTE. *Time-robust discrete control over networked loosely time-triggered architectures*, in "CDC", 2008, pp. 3595–3600

[29] P. CASPI. *The Quasi-Synchronous Approach to Distributed Control Systems*, VERIMAG, Crysis Project, May 2000, n^{o} CMA/009931, The Cooking Book

[30] P. CASPI, M. POUZET. *Synchronous Kahn Networks*, in "ACM SIGPLAN International Conference on Functional Programming (ICFP)", Philadelphia, Pensylvania, May 1996

[31] P. CASPI, M. POUZET. *A Co-iterative Characterization of Synchronous Stream Functions*, in "Coalgebraic Methods in Computer Science (CMCS'98)", Electronic Notes in Theoretical Computer Science, March 1998, Extended version available as a VERIMAG tech. report no. 97–07

[32] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *Synchroning Periodic Clocks*, in "ACM International Conference on Embedded Software (EMSOFT'05)", Jersey city, New Jersey, USA, September 2005

[33] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Synchronous Kahn Networks: a Relaxed Model of Synchrony for Real-Time Systems*, in "ACM International Conference on Principles of Programming Languages (POPL'06)", Charleston, South Carolina, USA, January 2006

[34] A. COHEN, S. GIRBAL, D. PARELLO, M. SIGLER, O. TEMAM, N. VASILACHE. *Facilitating the Search for Compositions of Program Transformations*, in "Intl. Conf. on Supercomputing (ICS'05)", Boston, Massachusetts, June 2005, pp. 151–160

[35] A. COHEN, S. GIRBAL, O. TEMAM. *A Polyhedral Approach to Ease the Composition of Program Transformations*, in "Euro-Par'04", Pisa, Italy, LNCS, Springer-Verlag, August 2004, n^o 3149, pp. 292–303

[36] A. COHEN, T. GROSSER, P. H. J. KELLY, J. RAMANUJAM, P. SADAYAPPAN, S. VERDOOLAEGE. *Split Tiling for GPUs: Automatic Parallelization Using Trapezoidal Tiles to Reconcile Parallelism and Locality, avoiding Divergence and Load Imbalance*, in "GPGPU 6 - Sixth Workshop on General Purpose Processing Using GPUs", Houston, United States, March 2013, https://hal.inria.fr/hal-00786812

[37] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. *Abstraction of Clocks in Synchronous Data-flow Systems*, in "The Sixth ASIAN Symposium on Programming Languages and Systems (APLAS)", Bangalore, India, December 2008

[38] A. COHEN, L. MANDEL, F. PLATEAU, M. POUZET. *Relaxing Synchronous Composition with Clock Abstraction*, 2009, Workshop on Hardware Design using Functional languages (HFL 09) - ETAPS, http://hal.inria.fr/hal-00645333

[39] J.-L. COLAÇO, G. HAMON, M. POUZET. *Mixing Signals and Modes in Synchronous Data-flow Systems*, in "ACM International Conference on Embedded Software (EMSOFT'06)", Seoul, South Korea, October 2006

[40] J.-L. COLAÇO, B. PAGANO, M. POUZET. *A Conservative Extension of Synchronous Data-flow with State Machines*, in "ACM International Conference on Embedded Software (EMSOFT'05)", Jersey city, New Jersey, USA, September 2005

[41] J.-L. COLAÇO, M. POUZET. *Clocks as First Class Abstract Types*, in "Third International Conference on Embedded Software (EMSOFT'03)", Philadelphia, Pennsylvania, USA, october 2003

[42] J.-L. COLAÇO, M. POUZET. *Type-based Initialization Analysis of a Synchronous Data-flow Language*, in "International Journal on Software Tools for Technology Transfer (STTT)", August 2004, vol. 6, n^o 3, pp. 245–255

[43] J. CORTADELLA, M. KISHINEVSKY. *Synchronous Elastic Circuits with Early Evaluation and Token Counterflow*, in "DAC", 2007, pp. 416-419

[44] P. CUOQ, M. POUZET. *Modular Causality in a Synchronous Stream Language*, in "European Symposium on Programming (ESOP'01)", Genova, Italy, April 2001

[45] P. FEAUTRIER. *Some Efficient Solutions to the Affine Scheduling Problem, Part II, multidimensional time*, in "Intl. J. of Parallel Programming", December 1992, vol. 21, n^o 6, pp. 389-420, See also Part I, one dimensional time, 21(5):315–348

[46] A. GAMATIÉ, E. RUTTEN, H. YU, P. BOULET, J.-L. DEKEYSER. *Synchronous Modeling and Analysis of Data Intensive Applications*, in "EURASIP Journal on Embedded Systems", 2008

[47] S. GIRBAL, N. VASILACHE, C. BASTOUL, A. COHEN, D. PARELLO, M. SIGLER, O. TEMAM. *Semi-Automatic Composition of Loop Transformations for Deep Parallelism and Memory Hierarchies*, in "Intl. J. of Parallel Programming", June 2006, vol. 34, n^{o} 3, pp. 261–317, Special issue on Microgrids

[48] T. GROSSER, A. COHEN, J. HOLEWINSKI, P. SADAYAPPAN, S. VERDOOLAEGE. *Hybrid Hexagonal/Classical Tiling for GPUs*, in "Intl. Symp. on Code Generation and Optimization (CGO)", Orlando, FL, United States, February 2014, https://hal.inria.fr/hal-00911177

[49] T. GROSSER, A. GRÖSSLINGER, C. LENGAUER. *Polly - Performing Polyhedral Optimizations on a Low-Level Intermediate Representation*, in "Parallel Processing Letters", 2012, vol. 22, n^{o} 4

[50] A.-C. GUILLOU, F. QUILLERÉ, P. QUINTON, S. RAJOPADHYE, T. RISSET. *Hardware Design Methodology with the Alpha Language*, in "FDL'01", Lyon, France, September 2001

[51] H. LEVERGE, C. MAURAS, P. QUINTON. *The* ALPHA *language and its use for the design of systolic arrays*, in "J. of VLSI Signal Processing", 1991, vol. 3, pp. 173–182

[52] L. MANDEL, F. BENBADIS. *Simulation of Mobile Ad hoc Network Protocols in ReactiveML*, in "Proceedings of Synchronous Languages, Applications, and Programming (SLAP'05)", Edinburgh, Scotland, Electronic Notes in Theoretical Computer Science, April 2005, Workshop ETAPS 2005

[53] L. MANDEL. *Conception, Sémantique et Implantation de ReactiveML : un langage à la ML pour la programmation réactive*, Université Paris 6, 2006

[54] L. MANDEL, F. PLATEAU, M. POUZET. *Lucy-n: a n-Synchronous Extension of Lustre*, in "10th International Conference on Mathematics of Program Construction (MPC'10)", Manoir St-Castin, Québec, Canada, Springer LNCS, June 2010

[55] L. MANDEL, M. POUZET. *ReactiveML, a Reactive Extension to ML*, in "ACM International Conference on Principles and Practice of Declarative Programming (PPDP)", Lisboa, July 2005

[56] F. MARANINCHI, N. BERTHIER, O. BEZET, G. FUNCHAL. *Writing Simulators with Synchronous Languages*, 2008, Synchron 2008: International Open Workshop on Synchronous Programming

[57] C. MIRANDA, A. POP, P. DUMONT, A. COHEN, M. DURANTON. *Erbium: A Deterministic, Concurrent Intermediate Representation to Map Data-Flow Tasks to Scalable, Persistent Streaming Processes*, in "Intl. Conf. on Compilers Architectures and Synthesis for Embedded Systems (CASES'10)", October 2010

[58] J.-B. NOTE, M. SHAND, J. VUILLEMIN. *Realtime video pixel matching*, in "International Conference on Field Programmable Logic and Applications", 2006, pp. 507 – 512

[59] J.-B. NOTE, J. VUILLEMIN. *Towards automatically compiling efficient FPGA hardware*, in "International Workshop on Design and Functional Languages", IEEE, 2007, pp. 115 – 124

[60] E. PARK, J. CAVAZOS, L.-N. POUCHET, C. BASTOUL, A. COHEN, P. SADAYAPPAN. *Predictive Modeling in a Polyhedral Optimization Space*, in "International Journal of Parallel Programming", 2013, vol. 41, n^o 5, pp. 704–750 [*DOI :* 10.1007/s10766-013-0241-1], https://hal.inria.fr/hal-00918653

[61] F. PLATEAU. *Modèle n-synchrone pour la programmation de réseaux de Kahn à mémoire bornée*, Université Paris-Sud 11Orsay, France, 6 janvier 2010, https://www.lri.fr/~mandel/lucy-n/~plateau/these/

[62] S. POP, A. COHEN, C. BASTOUL, S. GIRBAL, G.-A. SILBER, N. VASILACHE. *GRAPHITE: Loop Optimizations Based on the Polyhedral Model for GCC*, in "Proc. of the 4þ GCC Developper's Summit", Ottawa, Canada, June 2006

[63] L.-N. POUCHET, C. BASTOUL, A. COHEN, J. CAVAZOS. *Iterative Optimization in the Polyhedral Model: Part II, Multidimensional Time*, in "ACM Conf. on Programming Language Design and Implementation (PLDI'08)", Tucson, Arizona, June 2008

[64] L.-N. POUCHET, C. BASTOUL, A. COHEN, N. VASILACHE. *Iterative Optimization in the Polyhedral Model: Part I, One-Dimensional Time*, in "Intl. Symp. on Code Generation and Optimization (CGO'07)", San Jose, California, March 2007

[65] L.-N. POUCHET, U. BONDHUGULA, C. BASTOUL, A. COHEN, J. RAMANUJAM, P. SADAYAPPAN. *Combined Iterative and Model-driven Optimization in an Automatic Parallelization Framework*, in "ACM Supercomputing Conf. (SC'10)", New Orleans, Lousiana, November 2010, 11 p.

[66] P. RAYMOND, Y. ROUX, E. JAHIER. *Lutin: a language for specifying and executing reactive scenarios*, in "EURASIP Journal on Embedded Systems", 2008, vol. 2008, Article ID 753821

[67] L. SAMPER, F. MARANINCHI, L. MOUNIER, L. MANDEL. *GLONEMO: Global and Accurate Formal Models for the Analysis of Ad hoc Sensor Networks*, in "Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense'06)", Nice, France, May 2006

[68] J. SOULA, P. MARQUET, J.-L. DEKEYSER, A. DEMEURE. *Compilation principle of a specification language dedicated to signal processing*, in "Intl. Conf. on Parallel Computing Technologies", Novosibirsk, Russia, LNCS, Springer-Verlag, September 2001, vol. 2127, pp. 358–370

[69] K. TRIFUNOVIĆ, A. COHEN, D. EDELSOHN, F. LI, T. GROSSER, H. JAGASIA, R. LADELSKI, S. POP, J. SJÖDIN, R. UPADRASTA. *GRAPHITE Two Years After: First Lessons Learned From Real-World Polyhedral Compilation*, in "GCC Research Opportunities Workshop (GROW'10)", Pisa, Italy, January 2010

[70] K. TRIFUNOVIĆ, D. NUZMAN, A. COHEN, A. ZAKS, I. ROSEN. *Polyhedral-Model Guided Loop-Nest Auto-Vectorization*, in "Parallel Architectures and Compilation Techniques (PACT'09)", Raleigh, North Carolina, September 2009

[71] S. VERDOOLAEGE, S. GUELTON, T. GROSSER, A. COHEN. *Schedule Trees*, in "IMPACT - 4th Workshop on Polyhedral Compilation Techniques, associated with HiPEAC", Vienna, Austria, ACM, January 2014, https://hal.inria.fr/hal-00911894

[72] S. VERDOOLAEGE, J. C. JUEGA, A. COHEN, J. I. GÓMEZ, C. TENLLADO, F. CATTHOOR. *Polyhedral Parallel Code Generation for CUDA*, in "ACM Transactions on Architecture and Code Optimization", 2013,

vol. 9, n^{o} 4, Selected for presentation at the HiPEAC 2013 Conf. [*DOI :* 10.1145/2400682.2400713], https://hal.inria.fr/hal-00786677

[73] J. VUILLEMIN. *On Circuits and Numbers*, Digital, Paris Research Laboratory, 1993