Activity Report 2011

Team PARKAS

Parallélisme de Kahn Synchrone

RESEARCH CENTER
Paris - Rocquencourt

THEME
Embedded and Real Time Systems

Table of contents

Team PARKAS

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

The PARKAS team was established on April 1st 2011. It is located at École Normale Supérieure. The team is also a member of the Département d'Informatique of ENS.

1. Members

Research Scientist

Albert Cohen [Senior Researcher INRIA, HdR]

Faculty Members

Marc Pouzet [Team leader, ENS, Professor at UPMC, HdR]
Jean Vuillemin [ENS, Director of DI until August 2011, HdR]
Louis Mandel [Université Paris Sud]

PhD Students

Sean Halle [University of California Santa Cruz, INRIA contract, until June 2011]
Konrad Trifunovic [Université Paris Sud, INRIA contract, until July 2011]
Cedric Auger [Université Paris Sud, MESR scholarship]
Cupertino Miranda [Université Paris Sud, Portuguese FCT grant]
Boubacar Diouf [Université Paris Sud, ATER, until December 2011]
Léonard Gérard [Université Paris Sud, AMN]
Cedric Pasteur [Université Paris Diderot, AMX]
Ramakrishna Upadrasta [Université Paris Sud, MESR scholarship]
Jean-Yves Vet [UPMC, External student, CEA DAM]
Tobias Grosser [Google Doctoral Fellowship, from August 2011]
Riyadh Baghdadi [UPMC, ENS contract, from October 2011]
Adrien Guatto [ENS contract, from October 2011]

Post-Doctoral Fellows

Sven Verdoolaege [Until June 2011]
Timothy Bourke [Until September 2011]
Mehdi Doggi [ENS, From December 2011]
Antoniu Pop [From October 2011]

Others

Camille Gallet [Engineer, September to December 2011]
Ivan Llopard [External engineer, CEA LETI, From August 2011]
Riyadh Baghdadi [UPMC, Master 2 Intern, April to September 2011]
Adrien Guatto [UPMC, Master 2 Intern, April to September 2011]
Brice Gelineau [École Polytechnique, Master 1 Intern, April to July 2011]
Boris Arnoux [École Polytechnique and Telecom ParisTech, Master 2 Intern, July to December 2011]

2. Overall Objectives

2.1. Highlights

- Tobias Grosser has been awarded a Google European Doctoral Fellowship, a highly competitive 3 years scholarship of 120k €(14 recipients in 2011).

3. Scientific Foundations

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 [19], 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 [19], [30], initialization analysis [31] and causality analysis [33]). 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 [29], [28], and modular code generation [20], [14].

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.

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.

[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/
[4] http://www.3ds.com/products/catia/portfolio/dymola/overview/
[5] http://rml.lri.fr/

The development of REACTIVEML was started by Louis Mandel during his PhD thesis [49], [46] 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 [15] 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 [50].

REACTIVEML has been used for simulating routing protocols in ad-hoc networks [45] and large scale sensor networks [61]. 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 [60]. 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 [22], [23]. 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 [67]. It was algorithmically expensive and limited to periodic systems. In [26], 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 [47], as was a formalization of the theory in COQ [27]. 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 [51].

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 [54] (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).

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 [34].

[6] More precisely a subset of OCAML without objects or functors.

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 [43], [37], [62], [35] 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 [24], [36], [57], [56]. 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 [25], [24], we ported these techniques to the GCC compiler [55], [64], [63], 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 [57], [56], [58], [6]. 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).

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 [52], [53]. 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. Application Domains

The goal of the PARKAS 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 (signal processing, numerical, non-numerical); (3) the simulation of (a possibly huge number of) embedded systems in close interaction (e.g., simulation of factories, 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 [39], synchronous concurrency [13], and the expression of the two in functional languages [18], [49]. To reach our goal, we plan to leverage a large body of formal principles: language design, semantics, type theory, concurrency models, synchronous circuits and algorithms (code generation, optimization, polyhedral compilation).

The activity of the project is in the field of formal methods and compilation techniques for the development of computing and communicating reactive applications.

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.

The theoretical and practical interest of synchronous languages [12] is well established (Airbus A340 and A380 civil airplanes, EDF nuclear plants, railway signaling, system-on-chip for consumer electronics, etc.). There are also a variety of modeling/programming tools for the development of embedded systems (Simulink/StateFlow [7] by MathWorks, the academic tools Scilab/Scicos [8] or Ptolemy II [9]). These tools allow to program and simulate both the (discrete) controller and its physical (continuous) environment and are thus more widely applicable than synchronous languages have been. Nonetheless, they suffer from semantics issues [17], the non-reproducibility of simulations [21] and inefficiencies in generated code, which precludes their use in the most critical software applications (e.g., civil avionics and railways) which are controlled by independent certification authorities. To remedy these problems and to apply synchronous languages to a wider range of applications, we believe that two main issues must be addressed:

1. The ability to program and simulate, from a *single source code* both the system (e.g., the discrete controller) and its environment. The environment can be made of possibly huge numbers of other processes that are added and removed dynamically, and that possibly evolve in continuous time. This raises the question of the semantics and efficient simulation of the whole, the interaction with numerical solvers, and the static isolation of parts of the code which execute in bounded time and memory.
2. The ability to generate efficient parallel code from a synchronous specification, targeting modern architectures including shared-memory multi-core processors, non-uniform or distributed memory

[7] http://www.mathworks.com/products/simulink
[8] http://www-rocq.inria.fr/scicos/
[9] http://ptolemy.berkeley.edu/ptolemyII/

many-core architectures, tiled processor arrays, and heterogeneous systems with hardware accelerators (like GPUs). This also raises the question of language annotations to describe architecture and behavioral (e.g., real-time) constraints while preserving modular composition. It requires language and compiler support for relaxed synchronous models with jittering or buffered communication.

Having these two objectives in mind, we adopt a *language-centric approach*, focusing our efforts on the development of languages together with dedicated type systems and compilation techniques. We insist on concrete implementations as they allow us to validate and experiment with our proposals. This experimental point of view can be related to the Ptolemy project of Edward Lee in Berkeley [16]. Our technical approach is distinguished, however, by a focus on the synchronous model and a preference for typed functional programming.

There is also the observation that the synchronous model is expressive enough to account for other concurrency models (e.g., discrete-event and continuous [41], asynchronous communication [38]). Basing our study on the data-flow synchronous model will make it easier to design a sound, common semantics for all the language features. It will also simplify the remaining test and verification steps; this approach has proved to be crucial for the industrial success of the synchronous languages. Finding efficient ways to perform the early simulation of mixed discrete-continuous programs is also essential in the design flow of today's embedded systems. The creation of such simulation tools and associated tools for parallel code generation is a central and distinguishing feature of our proposal.

5. Software

5.1. Lucid Synchrone

Participant: Marc Pouzet [contact].

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). 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.

5.2. ReactiveML

Participants: Mehdi Dogguy, Louis Mandel [contact], 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 (Objective Caml).

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://www.lri.fr/~mandel/rml. 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 University Paris 6 and for the simulation of sensor networks at France Telecom and Verimag (CNRS, Grenoble).

5.3. Heptagon

Participants: Cédric Pasteur [contact], Brice Gelineau, Léonard Gérard, Adrien Guatto, Cédric Pasteur, Marc Pouzet.

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. 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, java and VHDL code.

Heptagon is jointly developed by Gwenael Delaval and Alain Girault from the INRIA POP ART team (Grenoble).

5.4. Lucy-n

Participants: Louis Mandel [contact], Adrien Guatto, Marc Pouzet.

http://www.lri.fr/~mandel/lucy-n

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.

5.5. ML-Sundials

Participants: Timothy Bourke, Marc Pouzet [contact].

ML-Sundials library provides an Ocaml interface to the Sundials numerical suite [10] (version 2.4.0). This library is used for solving and initial value problem and includes a zero-crossing detection mechanism. Only the CVODE solver with serial nvectors is currently supported. The structure and naming conventions largely follow the original libraries, both for ease of reading the existing documentation and for converting existing source code, but several changes have been made for programming convenience, namely:

- solver sessions are configured through algebraic data types rather than through multiple function calls,
- error conditions are signalled by exceptions rather than return codes (including in user-supplied callback routines),
- closures (partial applications of higher-order functions) are used to share user data between callback routines, and,
- explicit free commands are not necessary nor provided since Ocaml is a garbage-collected language.

[10] https://computation.llnl.gov/casc/sundials/main.html

The library is in use in a new synchronous hybrid language we are currently developping.

5.6. GCC

Participants: Albert Cohen [contact], Tobias Grosser, Antoniu Pop, Konrad Trifunovic, Feng Li, Riyadh Baghdadi, Cupertino Miranda.

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, parallel data-flow programming, and automatic parallelization to a data-flow runtime or architecture. This experiment borrows key design elements to synchronous data-flow languages.

Tobias Grosser is the maintainer of the Graphite optimization pass of GCC.

5.7. isl

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

http://freshmeat.net/projects/isl

Licence: LGPLv2.1+

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 also includes an ILP solver based on generalized basis reduction. isl also supports affine transformations for polyhedral compilation.

6. New Results

6.1. Compilation techniques for synchronous languages

- The paper *Modular Static Scheduling of Synchronous Data-flow Programs* by M. Pouzet and P. Raymond (VERIMAG Grenoble) has been selected among the two best papers at EMSOFT 2009. An extended version is published in a special issue of the *Journal of Design Automation for Embedded Systems*, in 2010. This work solves a 20 years problem raised by P. Raymond in 88 [59].
- Significant extensions have been provided to the Heptagon language. C. Pasteur contributed a memory optimization pass, combining a static analysis with user-given annotations (a paper on this subject is submitted to LCTES'2012), and the integration of discrete controller synthesis, developed and used by the SARDES team at INRIA Rhones-Alpes. Other work-in-progress extensions and experiments conducted inside the PARKAS team include the generation of VHDL code, parallel code generation, and n-synchronous code generation from Lucy-n.

6.2. Semantics and Implementation of Hybrid System Modelers

Hybrid systems modelers have become the corner stone of embedded system development, with Simulink [11] a de facto standard and Modelica [12] a new player. They allow both discrete controllers and their continuous environments to be expressed in a *single language*. Despite the availability of such tools, there remain a number of issues related to the lack of reproducibility of simulations and to the separation of the continuous part, which has to be exercised by a numerical solver, from the discrete part, which must be guaranteed not to evolve during a step. Such tools still raise a number of issues that, we believe, require more fundamental understanding.

In collaboration with Albert Benveniste and Benoit Caillaud (INRIA Rennes) we have proposed using non standard analysis as a semantic domain for hybrid systems. Non standard analysis is an extension of classical analysis in which infinitesimals can be manipulated as first class citizens. This allows us to provide a denotational semantics and a constructive semantics for hybrid systems, thus establishing simulation engines on a firm mathematical basis. In passing, we cleanly separate the job of the numerical analyst (solving differential equations) from that of the computer scientist (generating execution schemes).

- In late 2010, we presented in *49th Conference on Design and Control* in 2010 [11] the use of non standard semantics as a semantical ground for a hybrid synchronous language.
- Since the, we have extended this work in the following directions: 1/ a synchronous Kahn semantics for hybrid programs. Programs are viewed as synchronous ones running on an infinitely fast discrete base clock of the form $BaseClock(\partial) = \{n\partial \mid n \in {}^{\star}\mathbb{N}\}$, with infinitesimal step ∂ and ${}^{\star}\mathbb{N}$ the non-standard extension of $\mathbb{N}$. 2/ the definition of a standardization principle that gives sufficient conditions for a hybrid program to be standardizable. Under these conditions, the semantics corresponds to the semantics using *super-dense time* [44], [40], [42] for hybrid systems defined in [10]. 3/ a large amount of experimentations with Simulink to illustrate some of its pitfalls concerning in particular the treatment of zero-crosing cascades. This work is detailled in a long paper appearing in the *Journal of Computer Science and Systems* [1], in 2011.
- Starting from a minimal, yet full-featured, Lustre-like synchronous language, we have proposed a conservative extension where data-flow equations can be mixed with ordinary differential equations (ODEs) with possible reset. A type system is proposed to statically distinguish discrete computations from continuous ones and to ensure that signals are used in their proper domains. The extended data-flow language is realized through a source-to-source transformation into a synchronous subset, which can then be compiled using existing tools into routines that are both efficient and bounded in their use of memory. These routines are orchestrated with a single off-the-shelf numerical solver using a simple but precise algorithm which treats causally-related cascades of zero-crossings. We have validated the viability of the approach through experiments with the SUNDIALS [13] library. The basis of this work has been presented at the *ACM International Conference on Languages, Compilers, Theory of Embedded Systems*, 2011 [3].
- This work shows that it is possible to define a language which combines both the expressiveness of synchronous a synchronous language and that of ODEs where continuous solvers are approximated by a black-box solver. The most noticiable result was to *recycle* several techniques developed for synchronous languages: Kahn semantics, compilation techniques, static analysis. During year 2011, we extended the basic language with with hierarchical automata. This work has been presented at the *ACM International Conference on Embedded Software*, 2011 [2].
- In parallel with these theoretical works, M. Pouzet and T. Bourke have developed during year 2011 a new synchronous language and its compiler. The language, called ZELUS, extends a synchronous language with ODEs. It is first-order, functional and which mixes continuous-time and discrete-time signals. The expressiveness is that of (the first-order subset of) Lucid Synchrone (e.g., type inference

[11] http://www.mathworks.fr/products/simulink/index.html
[12] https://modelica.org/
[13] https://computation.llnl.gov/casc/sundials/main.html

and polymorphism, mix of data-flow and hierarchical automata) and ODEs with possible reset. Continuous trajectories are computed by a black-box numerical solver and we made our experiments with SUNDIALS.

6.3. N-Synchronous Languages

The n-synchronous model introduced a way to compose streams which have *almost the same clock* and can be synchronized through the use of a finite buffer.

We have designed the language Lucy-n to program in this model of computation [4], [48]. This language is similar to the first order synchronous data-flow language Lustre in which a buffer operator is added. A dedicated type system allows to check that programs can be executed in bounded memory and to compute the buffers sizes needed. Technically it is done through the introduction of a subtyping constraint at each bufferization point.

To solve the subtyping constraints we have defined an algorithm that uses an improved version of the state-of-the-art abstraction. We have proved the correctness properties of this new abstraction in Coq.

We also worked on new typing algorithms that do not use clock abstraction and thus allows to model Latency Insensitive Design [5] in Lucy-n [48].

- A. Guatto, together with L. Mandel and M. Pouzet, worked on the code generation for Lucy-n. They investigated two approaches. The first one was to use the schedules and buffer sizes computed by the compiler to generate a classical Lustre program. The second approach was to define a dynamic scheduling protocol similar to the ones used in latency insensitive designs.
- L. Mandel, in collaboration with F. Plateau (Prove&Run), developed a new resolution constraint algorithm for the clocking of Lucy-n programs [4]. Even if the new algorithm is less efficient that the one using abstraction, it has the advantage to be more precise and thus to accept more programs.
- L. Mandel, F. Plateau and M. Pouzet have extended the Lucy-n language with a new operator to be able to model Latency Insensitive Designs. Thanks to the new resolution constraint algorithm, the Lucy-n compiler is able to compute static schedules for such designs [5].

6.4. Synchronous Circuits

- Followed up on J. Vuillemin's result that the XOR variant of non-deterministic automata can be efficiently minimized [65], we explore this newly opened branch of computational automata theory. One contribution is a *Decision Diagram* for Boolean functions which has minimal dimension: this is appealing for both the verification and synthesis of memory-less circuits. Parts of this recent work were presented at the Boole ANR cooperation [8] and Synchron 2011.
- An extension of Boolean Decison Diagrams to integer representation and operations is given in [66]: we pursue software experimentations with arithmetics on such gigantic (yet sparse) numbers.
- The paper [32] was translated to English from the 1974 original and published in honor of Gilles Kahn.

6.5. Reactive Programming

ReactiveML is an extension of OCaml with synchronous concurrency, based on synchronous parallel composition and broadcast of signals. The goal is to provide a general model of deterministic concurrency inside a general purpose functional language to program reactive systems. It is particularly suited to program discrete simulations, for instance of sensor networks. The current focus of the research is being able to simulate huge systems, composed of millions of agents, by extending the current purely sequential implementation in order to be able to take advantage of multi-core and distributed architectures.

A first experiment consisted in creating a parallel runtime without any modification to the language. As the OCaml language on which ReactiveML is based (the ReactiveML compiler generates OCaml code) does not allow to create parallel programs communicating via shared memory, the new runtime was written in the F# language, part of Microsoft .Net environment. As the language is almost source to source compatible with OCaml, the ReactiveML compiler was left untouched. Several parallel runtimes were written, using traditional task scheduling techniques like work stealing or directly using light task mechanisms available in F#. This experiment demonstrated many speedup opportunities by parallelizing the runtime but also highlighted several problems and limitations of the language. Although this experiment was very useful in understanding the stakes of parallelizing ReactiveML, the performance gap between OCaml and F# (OCaml generates sequential code that is about 10 times slower) makes this version of the runtime of little practical use. We then proposed an extension of ReactiveML called clock domains. It consists in creating local notions of instants that are invisible from the outside. This extension should solve most of the problems raised by the previous experiment and help the parallelization of the language. The sequential runtime was adapted to this extension and a distributed version, using processes communicating via message passing, is currently being developed.

In collaboration with P. Attar (INRIA), F. Boussinot (INRIA) and J.-F. Susini (CNAM), L. Mandel worked on the design of DSL [9], a script language for the orchestration of concurrent programs. L .Mandel developed in ReactiveML and JoCaml an interpreter for the DSL language.

6.6. Polyhedral Compilation

L.-N. Pouchet presented fundamental advances in the construction and linear optimization of multidimensional affine transformation spaces at the POPL 2011 conference, in collaboration with A. Cohen and colleagues from Paris Sud University, Louisiana State University and Ohio State University.

Our team is actively integrating the polyhedral optimization framework in two production compilers: the *Graphite* framework in GCC and the *Polly* framework in LLVM. We are also working towards using the polyhedral framework to target GPGPU and manycore architectures, and to generate aggressively optimized code starting from high level languages. New isl-based version of Graphite and Polly have been contributed, enabling state-of-the-art affine transformations in GCC and LLVM, respectively. Dramatic performance improvements are expected in 2012, impacting the upcoming GCC 4.8 and LLVM 3.1, respectively.

K. Trifunovic, F. Li and A. Cohen, in collaboration with R. Ladelsky from IBM Research Haifa, presented a paper about some of these progresses at the GROW 2011 workshop [7].

Among the challenges that arise when adapting the polyhedral framework to production compilers, Riyadh Baghdadi has been working on memory-based dependences. Part of it is a practical compiler construction issue, where upstream passes such as the transformation to three-address code and PRE/CSE introduce new scalar variables leading to additional memory-based dependences. The other difficulty is to identify a profitable tradeoff between memory expansion (privatization, renaming) and parallelism. Memory-based dependences not only increase the complexity of the optimization but most importantly, they reduce the degree of freedom available to express effective loop nest transformations, limiting the overall effectiveness of the polyhedral framework. We designed and implemented a technique that solves this problem by allowing a compiler to relax the constraint of memory-based dependences on loop nest transformations and that does not incur the memory footprint overhead of scalar and array expansion. The proposed technique is based on the concept of polyhedral live range interval interference. While previous polyhedral optimization techniques could not achieve any speedups in benchmarks with scalar variables. This technique enabled a speedup of up to $16\times$ on numerical kernels from the *Polybench* benchmark suite (on a 24-core machine).

6.7. Parallel Data-Flow Programming

A. Pop defended his PhD thesis in September at MINES ParisTech, on a data-flow, streaming extension of OpenMP. Semantical, compilation and runtime system aspects have been covered in depth. Early results were published at HiPEAC 2011 and presented by A. Pop, in collaboration with A. Cohen. Follow-up work include the maturation of the proposed semantics and language extensions, with a thorough implementation and experimentation.

In parallel, F. Li, in collaboration with A. Pop and A. Cohen, explores the automatic compilation of SSA programs into dynamic data-flow parallelism, and the integration of streaming dependences to extend the method to non-scalar data flow. A paper has been published at WIR 2011 (workshop associated to CGO 2011), and a comprehensive, modular compilation method for arbitrary control flow, will be presented at MULTIPROG 2012 (associated with HiPEAC 2012).

Classical compilation techniques, found in Lustre, Scade, Lucid Synchrone, and all the dataflow synchronous languages, generate very efficient sequential code. Thus our main goal is to allow parallel code generation without changing the generation of the sequential code. To this matter, we introduced in the dataflow synchronous setting the famous asynchronous calls bundled with futures, which date back to MultiLisp designed by R. Halstead in the early 1980. It allows to separate the request of a computation from the actual use of its result. This approach has two main advantages. First, the compilation of these asynchronous calls is implemented by a simple wrapper encapsulating the called sequential code. It permits full compatibility with existing generated code. The futures are treated like usual values, so except for the asynchronous calls, we use the known sequential code generation. Second, this asynchronous calls and futures are only annotations and may be fully erased without changing the semantics of the program.

L. Gérard implements this proposition in our Heptagon compiler. The first backend was done in Java, as a proof of concept, using the threads and futures of the Java language. More efficient back-ends are being explored, using OpenMP stream-computing extensions and the TStar data-flow primitives.

7. Contracts and Grants with Industry

7.1. Grants with Industry

- Google European Doctoral Fellowship for Tobias Grosser, 120k €.

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. INRIA Action d'Envergure Synchronics

Participants: Albert Cohen, Marc Pouzet [contact], Louis Mandel.

This project is funded by INRIA for 4 years and started in Jan. 2008. The coordinators are A. Girault (INRIA Rhône Alpes) and M. Pouzet. http://synchronics.inria.fr/

The goal of the project is to propose new languages for the development of embedded systems allowing *from a unique source* to both simulate the system with its environment and generate code. It capitalizes on recent extensions of data-flow synchronous languages (Lucid Synchrone, ReactiveML), a relaxed form of synchrony, and means to mix discrete and continous systems inside the synchronous model of time.

The project focuses on language extensions to increase modularity, dedicated type systems to ensure safety properties, efficient compilation and the mix of discrete and continuous time.

Partners: INRIA Rhône Alpes (Gwenaël Delaval, Alain Girault, Bertrand Jeannet), IRISA (Benoit Caillaud), VERIMAG (Erwan Jahier, Pascal Raymond), INRIA Rocquencourt (Albert Cohen, Marc Pouzet, Louis Mandel)

8.1.2. PARTOUT

Participants: Mehdi Dogguy, Louis Mandel [contact], Cédric Pasteur, Marc Pouzet.

This project is funded by ANR (program DEFIS). http://www-sop.inria.fr/mimosa/PARTOUT

It started on january 2009 for 4 years; the coordinator is Frédéric Boussinot from INRIA Indes.

Partners: INRIA Indes, CNAM, LRI.

The goal of the project PARTOUT is, from a programming language point of view, to study the impact on programming of the globalization of parallelism which now covers all the spectrum of informatics, ranging from multicore architectures and distributed systems, up to applications deployed on the Web.

8.1.3. Mediacom Project

Participants: Albert Cohen [contact], Ramakrishna Upadrasta.

Partners: INRIA CompSys, ALF, Arenaire.

Mediacom is one of the projects of the Nano2012 collaboration framework between STMicroelectronics and INRIA, 09/2009–12/2012. Mediacom is a collaboration between the compilation group of STMicroelectronics HED, led by Christian Bertin, and the INRIA CompSys, ARENAIRE, ALF and PARKAS (formerly ALCHEMY) groups. We are working on portable concurrent intermediate languages, inspired by data-flow synchronous languages and polyhedral compilation, and on just-in-time parallelization algorithms.

8.2. European Initiatives

8.2.1. HiPEAC network of excellence

HiPEAC is a network of excellence on High-Performance Embedded Architectures and Compilers. It was first established as an FP6 network in 2004, and renewed as an FP7 4 years later. INRIA is one of the partners of the network. Albert Cohen leads the Compiler Platform cluster (9 research clusters in total). 02/2008–01/2012.

8.2.2. TERAFLUX integrated project

The TERAFLUX project is funded under the FP7 FET pro-active program on teradevice computing, 01/2010-12/2013. Albert Cohen is responsible for WP4. Our work addresses data-flow synchronous parallel programming, polyhedral compilation for data-flow programs, and compiler support for data-driven multi-threaded architectures with hundreds of computing cores. We contribute compilation algorithms and experimental language designs, with prototypes based on LUCID SYNCHRONE and direct contributions to GCC through the design of data-flow synchronous extensions of OpenMP. One of our goals is to transfer results of the project to production tools, including GCC and simulation platforms for many-core processors. A standardization effort (supported by INRIA's D2T) aims for the adoption of the language extensions by the OpenMP Architecture Review Board.

8.2.3. PHARAON specific targeted research project

The PHARAON project is funded on the embedded systems strategic objective, 09/2011–08/2014. Albert Cohen is responsible for WP5. Our work addresses data-flow synchronous programming for multiprocessor systems-on-chip, with an emphasis on an embedded development methodology and tools to optimize energy consumption and facilitate the correct-by-construction refinement of a functional specification. The Heptagon and Streaming OpenMP platforms of the team are used in the project. PHARAON is led by Thales Communications and Security.

8.2.4. CARP specific targeted research project

The CARP project is funded on the computing systems strategic objective, 12/2011–11/2014. Our work addresses polyhedral automatic parallelization for vector accelerators, with an emphasis on extending the scope of polyhedral compilation and integrating vectorization and specialization techniques. isl is an important component of this work, along with a new source-to-source compilation framework being developed in the project. CARP is led by Imperial College, and our team collaborates closely with ARM Cambridge in the specification of a portable parallel intermediate language to facilitate automatic parallelization and vectorization.

8.2.5. Collaborations in European Programs, except FP7

8.2.5.1. Euro-TM COST action

This new action started in April 2011. It aims at consolidating European research on transactional memory, by coordinating the research groups working on the development of complementary, interdisciplinary aspects of Transactional Memories, including theoretical foundations, algorithms, hardware and operating system support, language integration and development tools, and applications. Our participation is focused on the interaction between data-flow and transactional memory models.

8.3. International Initiatives

8.3.1. Visits of International Scientists

8.3.1.1. International guests of the PARKAS seminars

- November 2011: Alex Nicolau, UCI. *Variability, Accuracy, and Performance evaluation*.
- September 2011: Daisuke Ishii, JSPS, National Institute of Informatics (Tokyo) and ProVal team, INRIA Saclay. *An Execution Algorithm for a Hybrid Modeling Language HydLa*.
- August 2011: Peter Gammie, the Australian National University and National ICT Australia. *Verified Synthesis of Knowledge-Based Programs in Finite Synchronous Environments*.
- June 2011: Jan Vitek, Purdue University. *Virtualizing Real-time Embedded Systems with Java*.
- May 2011: Walid Najjar, University of California Riverside. *Performance, Productivity and Programmability: The Coming of Age of FPGA Code Accelerators*.
- March 2011: Eunjung Park, University of Delaware. *Predictive Modeling in a Polyhedral Optimization Space*.

8.3.1.2. Other visits

- John Cavazos, University of Delaware, visited us in January and July 2011. We have been collaborating on statistical methods in polyhedral compilation since John's postdoc at the University of Edinburgh. A joint paper was published at CGO 2011. Albert Cohen is on the PhD thesis committee of his student, Eunjung Park.
- P. Sadayappan, Ohio State University, visited us in December 2011. We collaborate for a long time on polyhedral compilation methods and tools. A joint paper was published at POPL 2011. P. Sadayappan has been the Master thesis advisor of Tobias Grosser and he is participating to the direction of his PhD thesis, and he hosts a former student, Louis-Noël Pouchet for more than 1 year as a postdoc.
- John Plaice, University of New South Wales, visited us in December 2011. He is a long-time synchronous programming and functional programming expert. His Translucid language experiment could be the basis for a collaboration on efficient compilation of array-based computations and synchronous language expressiveness.

8.3.1.3. Supervision of post-docs, theses and Internships

- Marc Pouzet supervised the 12-month post-doc of Timothy Bourke, from the University of New South Wales, from September 2010 to October 2011, and funded by the large scale initiative SYNCHRONICS of INRIA. Timothy worked on the semantics and implementation of hybrid modelers.
- Albert Cohen co-advised the PhD thesis of Sean Halle, from the University of California Santa Cruz, defended in June 2011. Sean Halle worked on parallel programming models and analytical performance models.

9. Dissemination

9.1. Animation of the scientific community

9.1.1. Event organization

- Albert Cohen is the General Chair of the 7th HiPEAC conference, January 2012. Louis Mandel is member of the local arrangements committee.

  The conference is pionneering a new "journal first" publication model, and has been completely reorganized into a large networking event with 27 parallel events, an industry exhibit with 50 companies booths and posters, and a European project exhibit with 46 projects from the computing systems, embedded systems strategic objectives of the FP7 and from the FET programme.

  475 registered participants, 34 ACM TACO journal papers will be presented (the record number for previous HiPEAC conferences was 210 participants).
- Albert Cohen co-organized the 9th Symposium on Code Generation and Optimization, as a finance chair, April 2011. CGO is co-sponsored by ACM and IEEE, in Chamonix http://www.cgo.org/cgo2011. 240 participants and 120 paper submissions, an increase of 10% and 30% over the record numbers for the conference.
- Marc Pouzet organised the 18th annual edition of the international workshop SYNCHRON http://synchron2011.di.ens.fr/, located in Damarie-les-lys, Nov. 28 - Dec. 2, 2011. This annual workshop is devoted to recent development in synchronous languages and its applications.

  This year, we have invited collegues from IRCAM (Paris) and GRAME (Lyon) for a half day presentation of their work on programming languages for music.

  The workshop has been a great success with 80 participant whereas the workshop usually have 50.

9.1.2. Editorial boards

- Jean Vuillemin is on the board of 5 international journals.
- Marc Pouzet is associate editor of the EURASIP Journal on Embedded systems (http://www.hindawi.com/journals/es/.
- Marc Pouzet is "directeur de collection" for Hermes publisher.
- Albert Cohen is on the editorial board of the International Journal on Parallel Programming (IJPP, http://www.springer.com/computer/theoretical+computer+science/journal/10766).

9.1.3. Program committees

- M. Pouzet is a member of the program committee of the following conferences: Design, Automation & Test in Europe (DATE 2011 and DATE 2012); Approches Formelles dans l'Assistance au Développement de Logiciels (AFADL 2011 and AFADL 2012); Conference on Real-Time and Network Systems (RTNS 2011); Modélisation des systèmes Réactifs (MSR 2011); ACM Conf. on the Principles and Applications of Declarative Programming (PADL 2012), located with POPL;
- M. Pouzet is an expert reviewer for the Design Automation Conference (DAC) in 2011 and 2012.
- A. Cohen is a co-chair of the DAC 2012 ESS1&2 TPC.
- A. Cohen is a member of the program committee of the following conferences: PLDI 2011, LCTES 2011, PPOPP 2011, ICS 2012, CC 2013.
- A. Cohen is a member of the program committee of the following workshops: IMPACT 2011, GROW 2011, WIR 2011, IMPACT 2012, INTERACT 2012, GPGPU 2012.
- L. Mandel, member of the program committee of the Journées Francophones des Langages Applicatifs (JFLA 2012).

9.1.4. Participation to Thesis Committees

- M. Pouzet was a reviewer of the following PhD theses: Tayeb Bouhadiba (Université de Grenoble; dir: Florence Maraninchi; September 16th, 2010); Daniel Reynaud (INPL, Nancy; dir: Jean-Yves Marion; October 15th, 2010); Antony Coadou (Université de Nice; dir: Robert de Simone; December 3th, 2010).
- M. Pouzet was president of the PhD jury of Nicolas Vallée (July 12, 2011), Univ. Paris-Diderot; Nicolas Pouillon (Sept. 26, 2011), Université Pierre et Marie Curie.
- A. Cohen was a reviewer of the following PhD theses: Per Larsen (August 2011, DTU, Copenhagen); Paul Carpenter, (September 2011, UPC Barcelona); Serge Guelton (September 2011, UBO and Telecom Bretagne); Benoît Pradelle (December 2011, Université Louis Pasteur, Strasbourg); Nicolas Benoît (January 2012, Université de Versailles St-Quentin).
- A. Cohen was a committee member of the following PhD theses: Matthias Eriksson (May 2011, Linköping U.); Antoniu Pop (September 2011, MINES ParisTech); Asma Charfi (December 2011, Université Paris Sud);
- A. Cohen was on the Prelim committee of Eunjung Park (December 2011, University of Delaware).

9.1.5. Invited Presentations

- L. Mandel and M. Pouzet, were invited to give a lecture at IRCAM in the Master of "Acoustique, Traitement du signal, Informatique, Appliqués à la Musique". Nov. 2010 and Nov. 2011.
- M. Pouzet was invited speaker at Complex Systems Design & Management (CSDM), Paris, in Dec. 2011. The title of the talk was: *Programming hybrid systems with synchronous languages*.
- M. Pouzet was invited speaker by Hilding Elmquist, CTO at Dassault-Systèmes and the founder of the Modelica Language, at the 69th Modelica Design Meeting, Munich, Germany, in Jan. 2011. M. Pouzet gave two one-hour talks. The titles were *Lucid Synchrone, a Functional Synchronous Language* and *Design, Semantics and Compilation for a Hybrid Synchronous Language*.
- A. Cohen was a keynote speaker at the CTHPC 2011 workshop in Tai Chung, Taiwan, titled "Polyhedral Compilation Runs Out of (Static) Control!", July 2011.
- A. Cohen gave an invited presentation at the National Tsing Hua University, Hsin Chu, Taiwan, titled "Work-Streaming: a Combined Language, Compiler and Runtime Approach to Deterministic, Scalable and Efficient Parallel Computing", July 2011.

9.2. Interaction with the scientific community

9.2.1. Prizes and distinctions

Since 2007, Marc Pouzet is a junior member of the IUF (*"Institut Universitaire de France"*), that distinguishes each year a few French university professors for the high quality of their research activities.

9.2.2. Collective responsibilities within INRIA

- M. Pouzet is an elected member of the INRIA Evaluation Committee.
- A. Cohen is an elected member of the INRIA Scientific Council.
- A. Cohen is a member of the BCP of INRIA Rocquencourt.
- A. Cohen is the scientific representative for the International Relations of INRIA for INRIA Rocquencourt, and a member of the COST GTRI committee.
- J. Vuillemin was head of Computer Science at ENS from 2007 to 2011.

9.2.3. Collective responsibilities outside INRIA

- J. Vuillemin is a 2011 member of the International Scientific Advisory Board, National ICT Australia.
- J. Vuillemin chairs the 2011 Scientific Advisory Board of I2R in Singapore.
- J. Vuillemin received the “Grand prix 2008 des sciences de l’Informatique et de leurs applications”, from the French Academy of Sciences and foundation EADS.
- J. Vuillemin was "Directeur Scientifique" at INRIA from 2005 to 2007.
- L. Mandel is in charge of the International Selection of ENS with J. Bertrane (ENS) for the computer science department.
- M. Pouzet was president of the hiring committee for a Chaire position funded by CNRS and École Polytechnique, in the computer science department of École Polytechnique, in january 2011.
- M. Pouzet was a member of the hiring committee for an Assistant Professor position at École Nationale Supérieure des Techniques Avancées (ENSTA), in May 2011; a Full Professor position in Université de Nice, in May 2011.
- A. Cohen was a member of two hiring committees for Associate Professor positions at UPMC in May and September 2011.
- A. Cohen jointly created and operates the Electrical Engineering multidisciplinary program at at École Polytechnique (Master 1 level), with Yvan Bonnassieu (physics department) and Frédéric Bonnans (applied mathematics department).

9.3. Industrial Dissemination

- M. Pouzet is scientific advisor for the compagny Esterel-Technology on the evolution of SCADE (start in nov. 2011).
- A. Cohen initiated and manages a 20 man-year R&D project in production-ready polyhedral compilation within GCC (Graphite project, since GCC 4.5), in collaboration with AMD, IBM Research, and several GCC developers.

9.4. Teaching

Licence: “Projet de programmation” (L3), L. Mandel (23h), Université Paris-Sud 11, France
Licence: “Systèmes et réseaux” (L3), M. Pouzet (24h), L. Mandel (24h), École Normale Supérieure, France
Licence: “Langages de programmation et compilation” (L3), L. Mandel (24h), École Normale Supérieure, France
Licence: “Components of a Computing System Introduction to Computer Architecture and Operating Systems” (L3), A. Cohen (44h), École Polytechnique, France
Master 1 École Polytechnique: “Operating Systems Principles and Programming” (M1), A. Cohen (38h), École Polytechnique, France
Master Parisien de Recherche en Informatique (MPRI): “Synchronous systems” (M2), M. Pouzet (12h), J. Vuillemin (12h), L. Mandel (6h), A. Cohen (6h), École Normale Supérieure and Université Paris Diderot, France
Master sur les Systèmes Industriels Complexes (COMASIC): Univ. Paris-Sud, École Polytechnique, INSTN. Synchronous Languages (24h), M. Pouzet.

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journal

[1] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *Non-Standard Semantics of Hybrid Systems Modelers*, in "Journal of Computer and System Sciences (JCSS)", 2011, vol. Special issue in honor of Amir Pnueli, Accepted for publication., http://www.di.ens.fr/~pouzet/bib/modelica_JCSS_revised.pdf.

International Conferences with Proceedings

[2] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *A Hybrid Synchronous Language with Hierarchical Automata: Static Typing and Translation to Synchronous Code*, in "ACM SIGPLAN/SIGBED Conference on Embedded Software (EMSOFT'11)", Taipei, Taiwan, October 2011, http://www.di.ens.fr/~pouzet/bib/emsoft11.pdf.

[3] A. BENVENISTE, T. BOURKE, B. CAILLAUD, M. POUZET. *Divide and recycle: types and compilation for a hybrid synchronous language*, in "ACM SIGPLAN/SIGBED Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES'11)", Chicago, USA, April 2011, http://www.di.ens.fr/~pouzet/bib/lctes11.pdf.

[4] L. MANDEL, F. PLATEAU. *Typage des horloges périodiques en Lucy-n*, in "Vingt deuxièmes Journées Francophones des Langages Applicatifs (JFLA 2011)", La Bresse, France, January 2011, http://www.lri.fr/~mandel/papiers/MandelPlateau-JFLA-2011.pdf.

[5] L. MANDEL, F. PLATEAU, M. POUZET. *Static Scheduling of Latency Insensitive Designs with Lucy-n*, in "Formal Methods in Computer Aided Design (FMCAD 2011)", Austin, TX, USA, October 2011, http://www.lri.fr/~mandel/lucy-n/fmcad11/.

[6] L.-N. POUCHET, U. BONDHUGULA, C. BASTOUL, A. COHEN, J. RAMANUJAM, P. SADAYAPPAN, N. VASILACHE. *Loop Transformations: Convexity, Pruning and Optimization*, in "38þ Symp. on Principles of Programming Languages (POPL'11)", Austin, Texas, January 2011.

[7] K. TRIFUNOVIĆ, A. COHEN, R. LADELSKI, F. LI. *Elimination of Memory-Based Dependences for Loop-Nest Optimization and Parallelization: Evaluation of a Revised Violated Dependence Analysis Method on a Three-Address Code Polyhedral Compiler*, in "3 rd GCC Research Opportunities Workshop (GROW'11, associated with CGO)", Chamonix, France, April 2011.

[8] J. VUILLEMIN. *The dimension of Boolean Functions*, in "Boole Conference 2011", Luminy, ANR, 2011, vol. 3.

Other Publications

[9] P. ATTAR, F. BOUSSINOT, L. MANDEL, J.-F. SUSINI. *Proposal for a Dynamic Synchronous Language*, May 2011, http://hal.archives-ouvertes.fr/hal-00590420.

References in notes

[10] R. ALUR, C. COURCOUBETIS, N. HALBWACHS, T. A. HENZINGER, P.-H. HO, X. NICOLLIN, A. OLIVERO, J. SIFAKIS, S. YOVINE. *The Algorithmic Analysis of Hybrid Systems*, in "Theor. Comput. Sci.", 1995, vol. 138, n^{o} 1, p. 3-34.

[11] A. BENVENISTE, B. CAILLAUD, M. POUZET. *The Fundamentals of Hybrid Systems Modelers*, in "49th IEEE International Conference on Decision and Control (CDC)", Atlanta, Georgia, USA, December 15-17 2010, http://www.di.ens.fr/~pouzet/bib/cdc10.pdf.

[12] A. BENVENISTE, P. CASPI, S. EDWARDS, N. HALBWACHS, P. LE GUERNIC, R. DE SIMONE. *The synchronous languages 12 years later*, in "Proceedings of the IEEE", January 2003, vol. 91, n^{o} 1.

[13] G. BERRY. *Real time programming: Special purpose or general purpose languages*, in "Information Processing", 1989, vol. 89, p. 11-17.

[14] 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.

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

[16] C. BROOKS, E. A. LEE, X. LIU, S. NEUENDORFFER, Y. ZHAO, H. ZHENG. *Heterogeneous Concurrent Modeling and Design in Java*, Memorandum UCB/ERL M04/27, EECS, University of California, Berkeley, CA USA 94720, July 2004.

[17] P. CASPI, A. CURIC, A. MAIGNAN, C. SOFRONIS, S. TRIPAKIS. *Translating Discrete-Time Simulink to Lustre*, in "ACM Transactions on Embedded Computing Systems", 2005, Special Issue on Embedded Software.

[18] P. CASPI, G. HAMON, M. POUZET. *Synchronous Functional Programming with Lucid Synchrone*, in "Real-Time Systems: Models and verification — Theory and tools", ISTE, 2007, English version of Lucid Synchrone. Published during year 2007.

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

[20] 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 at www.lri.fr/∼pouzet.

[21] A. CHAPOUTOT. *Simulation abstraite : une analyse statique de modèles Simulink*, École Polytechnique, December 2008.

[22] 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.

[23] 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.

[24] 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, p. 151–160.

[25] 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, p. 292–303.

[26] 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.

[27] 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://www.lri.fr/~plateau/hfl09/.

[28] 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.

[29] 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.

[30] 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.

[31] 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, p. 245–255.

[32] B. COURCELLE, G. KAHN, J. VUILLEMIN. *Algorithms for equivalence and reduction to minimal form for a class of simple recursive equations*, in "From Semantics to Computer Science, Essays in Honour of Gilles Kahn", 2009, p. 169 – 184.

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

[34] 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, p. 389-420, See also Part I, one dimensional time, 21(5):315–348.

[35] 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.

[36] 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, p. 261–317, Special issue on Microgrids..

[37] 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.

[38] N. HALBWACHS, S. BAGHDADI. *Synchronous modeling of asynchronous systems*, in "EMSOFT'02", Grenoble, LNCS 2491, Springer Verlag, October 2002.

[39] G. KAHN. *The semantics of a simple language for parallel programming*, in "IFIP 74 Congress", North Holland, Amsterdam, 1974.

[40] E. A. LEE, H. ZHENG. *Operational Semantics of Hybrid Systems*, in "HSCC", 2005, p. 25-53.

[41] E. A. LEE, H. ZHENG. *Leveraging Synchronous Language Principles for Heterogeneous Modeling and Design of Embedded Systems*, in "EMSOFT", Salzburg, Austria, September 30-October 3 2007.

[42] E. A. LEE, H. ZHENG. *Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems*, in "EMSOFT", 2007, p. 114-123.

[43] 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, p. 173–182.

[44] O. MALER, Z. MANNA, A. PNUELI. *From timed to hybrid systems*, in "Real-Time: Theory in Practice", LNCS, Springer, 1992, vol. 600, eds. J.W. de Bakker and C. Huizing and W.-P. de Roever and G. Rozemberg.

[45] 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.

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

[47] 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.

[48] L. MANDEL, F. PLATEAU, M. POUZET. *Lucy-n: a n-Synchronous Extension of Lustre*, in "Tenth International Conference on Mathematics of Program Construction (MPC 2010)", Québec, Canada, June 2010, http://www.lri.fr/~mandel/papiers/MandelPlateauPouzet-MPC-10.pdf.

[49] 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.

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

[51] 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.

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

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

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

[55] 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.

[56] 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.

[57] 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.

[58] 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 pages.

[59] P. RAYMOND. *Compilation séparée de programmes Lustre*, Projet SPECTRE, IMAG, juillet 1988.

[60] 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.

[61] 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.

[62] 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, p. 358–370.

[63] 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.

[64] 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.

[65] J. VUILLEMIN, N. GAMA. *Compact normal form for regular languages as xor automata*, in "14th International Conference, CIAA 2009", Sydney, Australia, LLNCS, Springer-Verlag, 2009, vol. 5642, p. 24 – 33.

[66] J. VUILLEMIN. *Efficient data structure and algorithms for sparse integers, sets and predicates*, in "19th IEEE Symposium on Computer Arithmetic", IEEE, 2009, p. 7 – 14.

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