RESEARCH CENTER

FIELD
Algorithmics, Programming, Software and Architecture

Activity Report 2014

Section Software

Edition: 2015-03-24

ARIC Project-Team

5. New Software and Platforms

5.1. Overview

AriC software realizations are accessible from the web page http://www.ens-lyon.fr/LIP/AriC/ware. We describe below only those which progressed in 2014.

5.2. GNU MPFR

Participant: Vincent Lefèvre [correspondant].

GNU MPFR is an efficient multiple-precision floating-point library with well-defined semantics (copying the good ideas from the IEEE-754 standard), in particular correct rounding in 5 rounding modes. GNU MPFR provides about 80 mathematical functions, in addition to utility functions (assignments, conversions...). Special data (*Not a Number*, infinities, signed zeros) are handled like in the IEEE-754 standard.

MPFR was one of the main pieces of software developed by the old SPACES team at Loria. Since late 2006, with the departure of Vincent Lefèvre to Lyon, it has become a joint project between the Caramel (formerly SPACES then CACAO) and the AriC (formerly Arénaire) project-teams. MPFR has been a GNU package since 26 January 2009.

An MPFR-MPC developers meeting took place from 20 to 22 January 2014 in Nancy. There was no new release this year, but various developments were done in the trunk.

The main work done in the AriC project-team:

- Changed the behavior of the `mpfr_set_exp` function to avoid undefined behavior in some cases (this change mainly impacted the internal usage).
- Bug fixes and various improvements (portability, efficiency, etc.).
- The `mpfr_sum` function is being rewritten (`new-sum` branch); see Section 6.2.8 .

URL: http://www.mpfr.org/

GNU MPFR is on the Black Duck Open Hub community platform for free and open source software: https://www.openhub.net/p/gnu-mpfr

- ACM: D.2.2 (Software libraries), G.1.0 (Multiple precision arithmetic), G.4 (Mathematical software).
- AMS: 26-04 Real Numbers, Explicit machine computation and programs.
- APP: no longer applicable (copyright transferred to the Free Software Foundation).
- License: LGPL version 3 or later.
- Type of human computer interaction: C library, callable from C or other languages via third-party interfaces.
- OS/Middleware: any OS, as long as a C compiler is available.
- Required library or software: GMP.
- Programming language: C.
- Documentation: API in texinfo format (and other formats via conversion); algorithms are also described in a separate document.

5.3. Exhaustive Tests for the Correct Rounding of Mathematical Functions

Participant: Vincent Lefèvre.

The search for the worst cases for the correct rounding (hardest-to-round cases) of mathematical functions (exp, log, sin, cos, etc.) in a fixed precision (mainly double precision) using Lefèvre's algorithm is implemented by a set of utilities written in Perl, with calls to Maple/intpakX for computations on intervals and with C code generation for fast computations. It also includes a client-server system for the distribution of intervals to be tested and for tracking the status of intervals (fully tested, being tested, aborted).

The Perl scripts have been improved (in particular, for the interaction with Grid Engine).

5.4. FPLLL: A Lattice Reduction Library

Participant: Damien Stehlé [correspondant].

fplll contains several algorithms on lattices that rely on floating-point computations. This includes implementations of the floating-point LLL reduction algorithm, offering different speed/guarantees ratios. It contains a "wrapper" choosing the estimated best sequence of variants in order to provide a guaranteed output as fast as possible. In the case of the wrapper, the succession of variants is oblivious to the user. It also includes a rigorous floating-point implementation of the Kannan-Fincke-Pohst algorithm that finds a shortest non-zero lattice vector, and the BKZ reduction algorithm.

The fplll library is used or has been adapted to be integrated within several mathematical computation systems such as Magma, Sage, and PariGP. It is also used for cryptanalytic purposes, to test the resistance of cryptographic primitives.

This year, several improvements to the BKZ (block Korkine Zolotarev) algorithm have been implemented. Further, the library is now hosted on `github`.

URL: https://github.com/dstehle/fplll

- ACM: D.2.2 (Software libraries), G.4 (Mathematical software)
- APP: Procedure started
- License: LGPL v2.1
- Type of human computer interaction: C++ library callable, from any C++ program.
- OS/Middleware: any, as long as a C++ compiler is available.
- Required library or software: MPFR and GMP.
- Programming language: C++.
- Documentation: available in html format on **URL:**https://github.com/dstehle/fplll

5.5. Sipe

Participant: Vincent Lefèvre.

Sipe is a mini-library in the form of a C header file, to perform radix-2 floating-point computations in very low precisions with correct rounding, either to nearest or toward zero. The goal of such a tool is to do proofs of algorithms/properties or computations of tight error bounds in these precisions by exhaustive tests, in order to try to generalize them to higher precisions. The currently supported operations are addition, subtraction, multiplication (possibly with the error term), fused multiply-add/subtract (FMA/FMS), and miscellaneous comparisons and conversions. Sipe provides two implementations of these operations, with the same API and the same behavior: one based on integer arithmetic, and a new one based on floating-point arithmetic.

New in 2014:

- `sipe_to_mpfr` function;
- support for `__float128` from GCC/libquadmath (implementing the binary128 format);
- some corrections.

URL: https://www.vinc17.net/research/sipe/

- ACM: D.2.2 (Software libraries), G.4 (Mathematical software).
- AMS: 26-04 Real Numbers, Explicit machine computation and programs.
- License: LGPL version 2.1 or later.
- Type of human computer interaction: C header file.
- OS/Middleware: any OS.
- Required library or software: GCC compiler.
- Programming language: C.
- Documentation: comment at the beginning of the code and Research report Inria RR-7832.

5.6. Gfun

Participant: Bruno Salvy.

Gfun is a Maple package for the manipulation of linear recurrence or differential equations. It provides tools for guessing a sequence or a series from its first terms; for manipulating rigorously solutions of linear differential or recurrence equations, using the equation as a data-structure. This year, the implementation effort was focused on speeding up the guessing routines in the case of sequences with symbolic parameters that come up in general hypergeometric identities.

CARAMEL Project-Team

5. New Software and Platforms

5.1. Introduction

A major part of the research done in the CARAMEL team is published within software. On the one hand, this enables everyone to check that the algorithms we develop are really efficient in practice; on the other hand, this gives other researchers — and us of course — basic software components on which they — and we — can build other applications.

5.2. GNU MPFR

Participant: Paul Zimmermann [contact].

GNU MPFR is one of the main pieces of software developed by the CARAMEL team. Since end 2006, it has become a joint project between CARAMEL and the ARÉNAIRE project-team (now ARIC, INRIA Grenoble - Rhône-Alpes). GNU MPFR is a library for computing with arbitrary precision floating-point numbers, together with well-defined semantics, and is distributed under the LGPL license. All arithmetic operations are performed according to a rounding mode provided by the user, and all results are guaranteed correct to the last bit, according to the given rounding mode.

No new release was made in 2014. However a developers meeting was organized in January 20 to 22 in Nancy, together with the developers of GNU MPC.

5.3. GNU MPC

Participant: Paul Zimmermann [contact].

GNU MPC is a floating-point library for complex numbers, which is developed on top of the GNU MPFR library, and distributed under the LGPL license. It is co-written with Andreas Enge (LFANT project-team, INRIA Bordeaux - Sud-Ouest). A complex floating-point number is represented by $x + iy$, where x and y are real floating-point numbers, represented using the GNU MPFR library. The GNU MPC library provides correct rounding on both the real part x and the imaginary part y of any result. GNU MPC is used in particular in the TRIP celestial mechanics system developed at IMCCE (*Institut de Mécanique Céleste et de Calcul des Éphémérides*), and by the Magma and Sage computational number theory systems.

Version 1.0.2 (Fagus silvatica) was released in January, with a few bug fixes, some related to the use in our own work related to the computation of Igusa class polynomials.

5.4. Finite Fields

Participants: Pierrick Gaudry, Emmanuel Thomé [contact], Luc Sanselme.

$\texttt{mp}\mathbb{F}_q$ is (yet another) library for computing in finite fields. The purpose of $\texttt{mp}\mathbb{F}_q$ is not to provide a software layer for accessing finite fields determined at runtime within a computer algebra system like Magma, but rather to give a very efficient, optimized code for computing in finite fields precisely known at *compile time*. $\texttt{mp}\mathbb{F}_q$ can adapt to finite fields of any characteristic and any extension degree. However, one of the targets being the use in cryptology, $\texttt{mp}\mathbb{F}_q$ somehow focuses on prime fields and on fields of characteristic two.

When it was first written in 2007, $\texttt{mp}\mathbb{F}_q$ established reference marks for fast elliptic curve cryptography: the authors improved over the fastest examples of key-sharing software in genus 1 and 2, both over binary fields and prime fields. A stream of academic works followed the idea behind $\texttt{mp}\mathbb{F}_q$ and improved over such timings, notably by Scott, Aranha, Longa, Bos, Hisil, Costello.

The library's purpose being the *generation* of code rather than its execution, the working core of mp$\mathbb{F}_q$ consists of roughly 18,000 lines of Perl code, which generate most of the C code. mp$\mathbb{F}_q$ is distributed at http://mpfq.gforge.inria.fr/.

In 2014, mp$\mathbb{F}_q$ has undergone some sanitization work, related to embedded assembly, build system, coverage test, and processor feature support. The fact that mp$\mathbb{F}_q$ is used in CADO-NFS has played an important role in fostering these changes to the mp$\mathbb{F}_q$ code. Future plans regarding the linear algebra code in CADO-NFS are expected to rely on the arithmetic part being implemented in mp$\mathbb{F}_q$. Preliminary work in this direction has been implemented by Luc Sanselme. Preliminary code by Hamza Jeljeli and Bastien Vialla from LIRMM, Montpellier, based on RNS arithmetic (Residue Number System) is also to be integrated in this context. We therefore expect more work in this area in the coming months, eventually leading to a new release.

5.5. gf2x

Participants: Pierrick Gaudry, Emmanuel Thomé [contact], Paul Zimmermann.

GF2X is a software library for polynomial multiplication over the binary field, developed together with Richard Brent (Australian National University, Canberra, Australia). It holds state-of-the-art implementation of fast algorithms for this task, employing different algorithms in order to achieve efficiency from small to large operand sizes (Karatsuba and Toom-Cook variants, and eventually Schönhage's or Cantor's FFT-like algorithms). GF2X takes advantage of specific processor instructions (SSE, PCLMULQDQ).

The current version of GF2X is 1.1, released in May 2012 under the GNU GPL. Since 2009, GF2X can be used as an auxiliary package for the widespread software library NTL, as of version 5.5. GF2X is also packaged in the Debian Linux distribution.

In 2014, the development version of GF2X has been updated to include some minor cleanups.

An LGPL-licensed portion of GF2X is also part of the CADO-NFS software package.

5.6. CADO-NFS

Participants: Cyril Bouvier, Alain Filbois, Pierrick Gaudry, Alexander Kruppa, Thomas Richard, Emmanuel Thomé [contact], Paul Zimmermann.

CADO-NFS is a program to factor integers using the Number Field Sieve algorithm (NFS), originally developed in the context of the ANR-CADO project (November 2006 to January 2010).

NFS is a complex algorithm which contains a large number of sub-algorithms. The implementation of all of them is now complete, but still leaves some places to be improved. Compared to existing implementations, the CADO-NFS implementation is already a reasonable player. Several factorizations have been completed using our implementation.

Since 2009, the source repository of CADO-NFS is publicly available for download, and is referenced from the software page at http://cado-nfs.gforge.inria.fr/. A major new release, CADO-NFS 2.1, was published in July 2014, with a bug-fix release (2.1.1) in October. Among the main improvements, the polynomial selection now runs in two stages, several unit tests have been added, various small speed-ups and bug fixes.

More and more people use CADO-NFS to perform medium to large factorizations. In February, Fabien Perigaud and Cédric Pernet from Cassidian Cybersecurity reverse-engineered a ransomware, which in the end boiled down to factoring numbers with CADO-NFS.

5.7. Belenios

Participants: Pierrick Gaudry, Stéphane Glondu [contact].

In collaboration with the CASSIS team, we develop an open-source private and verifiable electronic voting protocol, named BELENIOS. Our system is an evolution of an existing system, Helios, developed by Ben Adida, and used e.g., by UCL and the IACR association in real elections. The main differences with Helios are the following ones:

- In Helios, the ballot box publishes the encrypted ballots together with their corresponding voters. This raises a privacy issue in the sense that whether someone voted or not shall not necessarily be publicized on the web. Publishing this information is in particular forbidden by CNIL's recommendation. BELENIOS no longer publishes voters' identities, still guaranteeing correctness of the tally.
- Helios is verifiable except that one has to trust that the ballot box will not add ballots. The addition of ballots is particularly hard to detect as soon as the list of voters is not public. We have therefore introduced an additional authority that provides credentials that the ballot box can verify but not forge [18], [23].

This new version has been implemented by Stéphane Glondu [0]. The first public release has been done in January 2014. In the last public release (April 2014), BELENIOS still uses a major component of the Helios system, the booth. Since then, the booth has been reimplemented but is not yet part of a public release. This development version of BELENIOS has been used in December 2014 for selecting photos of LORIA's calendar (187 persons voted for 0 to 6 pictures, within a set of 52 choices).

5.8. CMH

Participant: Emmanuel Thomé [contact].

In collaboration with the LFANT project-team, INRIA Bordeaux – Sud-Ouest, we develop the CMH software package and library, which holds code for computing Igusa class polynomials. Those characterize principally polarized abelian varieties of dimension 2 having complex multiplication by the ring of integers of a quartic CM field.

The source repository of CMH is publicly available for download, and is referenced from the software page at http://cmh.gforge.inria.fr/.

Version 1.0 has been released in March 2014, simultaneously with the publication of a computation record.

5.9. Platforms

5.9.1. CATREL cluster

Installed in 2013, the CATREL computer cluster now plays an essential role in providing the team with the necessary resources to achieve significant computations, which illustrate well the efficiency of the algorithms developed in our research, together with their implementations.

[0] http://belenios.gforge.inria.fr/

CASCADE Project-Team (section vide)

CRYPT Team (section vide)

GALAAD2 Team

5. New Software and Platforms

5.1. Mathemagix, a free computer algebra environment

Participant: Bernard Mourrain.

http://www.mathemagix.org/

algebra, univariate polynomial, multivariate polynomial, matrices, series, fast algorithm, interpreter, compiler, hybrid software.

MATHEMAGIX is a free computer algebra system which consists of a general purpose interpreter, which can be used for non-mathematical tasks as well, and efficient modules on algebraic objects. It includes the development of standard libraries for basic arithmetic on dense and sparse objects (numbers, univariate and multivariate polynomials, power series, matrices, etc., based on FFT and other fast algorithms). These developments, based on C++, offer generic programming without losing effectiveness, via the parameterization of the code (*template*) and the control of their instantiations.

The language of the interpreter is imperative, strongly typed and high level. A compiler of this language is available. A special effort has been put on embedding of existing libraries written in other languages like C or C++. An interesting feature is that this extension mechanism supports template types, which automatically induce generic types inside Mathemagix. Connections with GMP, MPFR for extended arithmetic, LAPACK for numerical linear algebra are currently available in this framework.

The project aims at building a bridge between symbolic computation and numerical analysis. It is structured by collaborative software developments of different groups in the domain of algebraic and symbolic-numeric computation.

In this framework, we are working more specifically on the following components:

- REALROOT: a set of solvers using subdivision methods to isolate the roots of polynomial equations in one or several variables; continued fraction expansion of roots of univariate polynomials; Bernstein basis representation of univariate and multivariate polynomials and related algorithms; exact computation with real algebraic numbers, sign evaluation, comparison, certified numerical approximation.
- SHAPE: tools to manipulate curves and surfaces of different types including parameterized, implicit with different type of coefficients; algorithms to compute their topology, intersection points or curves, self-intersection locus, singularities, ...

These packages are integrated from the former library SYNAPS (SYmbolic Numeric APplicationS) dedicated to symbolic and numerical computations. There are also used in the algebraic-geometric modeler AXEL.

Collaborators: Grégoire Lecerf, Joris van der Hoeven and Philippe Trébuchet.

5.2. Axel, a geometric modeler for algebraic objects

Participants: Nicolas Douillet, Anaïs Ducoffe [contact], Valentin Michelet, Bernard Mourrain, Hung Nguyen, Meriadeg Perrinel.

http://axel.inria.fr.

computational algebraic geometry, curve, implicit equation, intersection, parameterization, resolution, surface, singularity, topology

We are developing a software called AXEL (Algebraic Software-Components for gEometric modeLing) dedicated to algebraic methods for curves and surfaces. Many algorithms in geometric modeling require a combination of geometric and algebraic tools. Aiming at the development of reliable and efficient implementations, AXEL provides a framework for such combination of tools, involving symbolic and numeric computations.

The software contains data structures and functionalities related to algebraic models used in geometric modeling, such as polynomial parameterizations, B-splines, implicit curves and surfaces. It provides algorithms for the treatment of such geometric objects, such as tools for computing intersection points of curves or surfaces, for detecting and computing self-intersection points of parameterized surfaces, for implicitization, for computing the topology of implicit curves, for meshing implicit (singular) surfaces, etc.

The developments related to isogeometric analysis have been integrated as dedicated plugins. Optimization techniques and solvers for partial differential equations developed by R. Duvigneau (OPALE) have been connected.

The new version of the algebraic-geometric modelers based on the DTK platform is still developed in order to provide a better modularity and a better interface to existing computation facilities and geometric rendering interface. This software is intended to be multi-platform, and jobs are running nightly on the Continous Integration platform https://ci.inria.fr/ of Inria, performing builds and tests on Virtual Machines of different OS such as Fedora, Ubuntu, Windows.

AXEL is written in C++ and thanks to a wrapping system using SWIG, its data structures and algorithms can be integrated into C# programs, as well as Python and Java programs. This wrapper was used to integrate AXEL into the CAD software TopSolid, developed by Missler Company and written in C#. But it also enables AXEL to embed a Python interpreter.

Other functionalities were also added or improved: the scientific visualization was improved and it is now possible to create dynamic geometric model in AXEL.

The software is distributed as a source package, as well as binary packages for Linux, MacOSX and Windows. It is hosted at http://dtk.inria.fr/axel with some of its plugins developed on Inria's gforge server (http://gforge.inria.fr) The first version of the software has been downloaded more than 15000 times, since it is available. A new version, AXEL 2.3.1, was released at the end of this year.

Collaboration with Gang Xu (Hangzhou Dianzi University, China), Julien Wintz (Dream), Elisa Berrini (MyCFD, Sophia), Angelos Mantzaflaris (GISMO library, Linz, Austria) and Laura Saini (Post-Doc GALAAD/Missler, TopSolid).

GEOMETRICA Project-Team

5. New Software and Platforms

5.1. CGAL, the Computational Geometry Algorithms Library

Participants: Jean-Daniel Boissonnat, Olivier Devillers, Marc Glisse, Aymeric Pellé, Monique Teillaud, Mariette Yvinec.

With the collaboration of Pierre Alliez, Hervé Brönnimann, Manuel Caroli, Pedro Machado Manhães de Castro, Frédéric Cazals, Frank Da, Christophe Delage, Andreas Fabri, Julia Flötotto, Philippe Guigue, Michael Hemmer, Samuel Hornus, Clément Jamin, Menelaos Karavelas, Sébastien Loriot, Abdelkrim Mebarki, Naceur Meskini, Andreas Meyer, Sylvain Pion, Marc Pouget, François Rebufat, Laurent Rineau, Laurent Saboret, Stéphane Tayeb, Jane Tournois, Radu Ursu, and Camille Wormser http://www.cgal.org

CGAL is a C++ library of geometric algorithms and data structures. Its development has been initially funded and further supported by several European projects (CGAL, GALIA, ECG, ACS, AIM@SHAPE) since 1996. The long term partners of the project are research teams from the following institutes: Inria Sophia Antipolis - Méditerranée, Max-Planck Institut Saarbrücken, ETH Zürich, Tel Aviv University, together with several others. In 2003, CGAL became an Open Source project (under the LGPL and QPL licenses).

The transfer and diffusion of CGAL in industry is achieved through the company GEOMETRY FACTORY (http://www.geometryfactory.com). GEOMETRY FACTORY is a *Born of Inria* company, founded by Andreas Fabri in January 2003. The goal of this company is to pursue the development of the library and to offer services in connection with CGAL (maintenance, support, teaching, advice). GEOMETRY FACTORY is a link between the researchers from the computational geometry community and the industrial users.

The aim of the CGAL project is to create a platform for geometric computing supporting usage in both industry and academia. The main design goals are genericity, numerical robustness, efficiency and ease of use. These goals are enforced by a review of all submissions managed by an editorial board. As the focus is on fundamental geometric algorithms and data structures, the target application domains are numerous: from geological modeling to medical images, from antenna placement to geographic information systems, etc.

The CGAL library consists of a kernel, a list of algorithmic packages, and a support library. The kernel is made of classes that represent elementary geometric objects (points, vectors, lines, segments, planes, simplices, isothetic boxes, circles, spheres, circular arcs...), as well as affine transformations and a number of predicates and geometric constructions over these objects. These classes exist in dimensions 2 and 3 (static dimension) and d (dynamic dimension). Using the template mechanism, each class can be instantiated following several representation modes: one can choose between Cartesian or homogeneous coordinates, use different number types to store the coordinates, and use reference counting or not. The kernel also provides some robustness features using some specifically-devised arithmetic (interval arithmetic, multi-precision arithmetic, static filters...).

A number of packages provide geometric data structures as well as algorithms. The data structures are polygons, polyhedra, triangulations, planar maps, arrangements and various search structures (segment trees, d-dimensional trees...). Algorithms are provided to compute convex hulls, Voronoi diagrams, Boolean operations on polygons, solve certain optimization problems (linear, quadratic, generalized of linear type). Through class and function templates, these algorithms can be used either with the kernel objects or with user-defined geometric classes provided they match a documented interface.

Finally, the support library provides random generators, and interfacing code with other libraries, tools, or file formats (ASCII files, QT or LEDA Windows, OpenGL, Open Inventor, Postscript, Geomview...). Partial interfaces with Python, SCILAB and the Ipe drawing editor are now also available.

GEOMETRICA is particularly involved in general maintenance, in the arithmetic issues that arise in the treatment of robustness issues, in the kernel, in triangulation packages and their close applications such as alpha shapes, in mesh generation and related packages. Two researchers of GEOMETRICA are members of the CGAL Editorial Board, whose main responsibilities are the control of the quality of CGAL, making decisions about technical matters, coordinating communication and promotion of CGAL.

CGAL is about 700,000 lines of code and supports various platforms: GCC (Linux, Mac OS X, Cygwin...), Visual C++ (Windows), Intel C++. A new version of CGAL is released twice a year, and it is downloaded about 10000 times a year. Moreover, CGAL is directly available as packages for the Debian, Ubuntu and Fedora Linux distributions.

More numbers about CGAL: there are now 12 editors in the editorial board, with approximately 20 additional developers. The user discussion mailing-list has more than 1000 subscribers with a relatively high traffic of 5-10 mails a day. The announcement mailing-list has more than 3000 subscribers.

5.1.1. High-dimensional kernel Epick_d

Participant: Marc Glisse.

We implemented a new high-dimensional kernel taking advantage of the progress that was made in dimensions 2 and 3. It is meant to be used with a reimplementation of high-dimensional triangulations (in progress).

5.1.2. Number type Mpzf

Participant: Marc Glisse.

We added a new exact ring number type that can represent all finite double floating-point numbers. It makes building a Delaunay triangulation 8 times faster than with earlier CGAL releases in some degenerate cases.

5.1.3. CGALmesh: a Generic Framework for Delaunay Mesh Generation

Participants: Jean-Daniel Boissonnat, Mariette Yvinec.

In collaboration with Pierre Alliez (EPI Titane), ClémentJamin (EPI Titane)

CGALmesh is the mesh generation software package of the Computational Geometry Algorithm Library (CGAL). It generates isotropic simplicial meshes – surface triangular meshes or volume tetrahedral meshes – from input surfaces, 3D domains as well as 3D multi-domains, with or without sharp features. The underlying meshing algorithm relies on restricted Delaunay triangulations to approximate domains and surfaces, and on Delaunay refinement to ensure both approximation accuracy and mesh quality. CGALmesh provides guarantees on approximation quality as well as on the size and shape of the mesh elements. It provides four optional mesh optimization algorithms to further improve the mesh quality. A distinctive property of CGALmesh is its high flexibility with respect to the input domain representation. Such a flexibility is achieved through a careful software design, gathering into a single abstract concept, denoted by the oracle, all required interface features between the meshing engine and the input domain. We already provide oracles for domains defined by polyhedral and implicit surfaces. [27] [53]

5.1.4. Periodic Meshes

Participants: Aymeric Pellé, Monique Teillaud.

There is a growing need for a 3D periodic mesh generator for various fields, such as material engineering or modeling of nano-structures. We are writing a software package answering this need, and which will be made publicly available in the open source library CGAL. The software is based on the CGAL 3D volume mesh generator package and the CGAL 3D periodic triangulations package. [42] [63]

5.2. Gudhi library

Participants: Jean-Daniel Boissonnat, Marc Glisse, Clément Maria, Mariette Yvinec.

With the collaboration of David Salinas

https://project.inria.fr/gudhi/software/

The GUDHI open source library will provide the central data structures and algorithms that underly applications in geometry understanding in higher dimensions. It is intended to both help the development of new algorithmic solutions inside and outside the project, and to facilitate the transfert of results in applied fields. The first release of the GUDHI library includes: – Data structures to represent, construct and manipulate simplicial complexes; – Algorithms to compute persistent homology and multi-field persistent homology.

GRACE Project-Team

5. New Software and Platforms

5.1. CADO-NFS-DLOG

F. Morain is one of the developers of CADO-NFS (available at http://cado-nfs.gforge.inria.fr/), which now includes new algorithms for discrete logarithm computations over finite fields.

5.2. Fast Compact Diffie–Hellman software

Working with C. Costello (Microsoft Research) and H. Hisil (Yasar), B. Smith contributed to the development of a competitive, high-speed, open implementation of the Diffie–Hellman protocol (described in [21]), targeting the 128-bit security level on Intel platforms. The source code is freely available at http://research.microsoft.com/en-us/downloads/ef32422a-af38-4c83-a033-a7aafbc1db55/ and http://hhisil.yasar.edu.tr/files/hisil20140318compact.tar.gz.

5.3. Platforms

5.3.1. ACTIS: Contribution to Sage

In the beginning of 2014,D. Augot and C. Pernet submitted an IJD proposal (ingénieur jeune diplomé) to Inria, called Projet Actis (Algorithmic Coding Theory In Sage). The aim of this project is to vastly improve the state of the error correcting library in Sage. The existing library does not present a good and usable API, and the provided algorithms are very basic, irrelevant, and outdated. We thus have two directions for improvement: renewing the APIs to make them actually usable by researchers, and incorporating efficient programs for decoding, like J. Nielsen's CodingLib, which contains many new algorithms.

We hired D. Lucas on October 1st; he has started implementing various basic things, in a standalone manner. We plan to publish these snippets of code to the Sage community in January 2015. Our plan is to interact a lot with the Sage community, to ensure that our new APIs will cover most of the needs of various communities.

LFANT Project-Team

4. New Software and Platforms

4.1. Pari/Gp

Participants: Karim Belabas [correspondent], Bill Allombert, Henri Cohen, Andreas Enge, Hamish Ivey-Law.

http://pari.math.u-bordeaux.fr/

PARI/GP is a widely used computer algebra system designed for fast computations in number theory (factorisation, algebraic number theory, elliptic curves, ...), but it also contains a large number of other useful functions to compute with mathematical entities such as matrices, polynomials, power series, algebraic numbers, etc., and many transcendental functions.

- PARI is a C library, allowing fast computations.
- GP is an easy-to-use interactive shell giving access to the PARI functions.
- `gp2c`, the GP-to-C compiler, combines the best of both worlds by compiling GP scripts to the C language and transparently loading the resulting functions into GP; scripts compiled by `gp2c` will typically run three to four times faster.
- Version of PARI/GP: 2.7.2
- Version of `gp2c`: 0.0.9
- License: GPL v2+
- Programming language: C

4.2. GNU MPC

Participants: Andreas Enge [correspondent], Mickaël Gastineau [CNRS], Philippe Théveny [INRIA project-team ARIC], Paul Zimmermann [INRIA project-team CARAMEL].

http://mpc.multiprecision.org/.

GNUMPC is a C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result. It is built upon and follows the same principles as GNUMPFR.

It is a prerequisite for the GNU compiler collection GCC since version 4.5, where it is used in the C and Fortran front ends for constant folding, the evaluation of constant mathematical expressions during the compilation of a program. Since 2011, it is an official GNU project.

2012 has seen the first release of the major version 1.0.

- Version: 1.0.2 *Fagus silvatica*
- License: LGPL v3+
- ACM: G.1.0 (Multiple precision arithmetic)
- AMS: 30.04 Explicit machine computation and programs
- APP: Dépôt APP le 2003-02-05 sous le numéro IDDN FR 001 060029 000 R P 2003 000 10000
- Programming language: C

4.3. MPFRCX

Participant: Andreas Enge.

http://mpfrcx.multiprecision.org/

MPFRCX is a library for the arithmetic of univariate polynomials over arbitrary precision real (MPFR) or complex (MPC) numbers, without control on the rounding. For the time being, only the few functions needed to implement the floating point approach to complex multiplication are implemented. On the other hand, these comprise asymptotically fast multiplication routines such as Toom-Cook and the FFT.

- Version: 0.4.2 *Cassava*
- License: LGPL v2.1+
- Programming language: C

4.4. CM

Participant: Andreas Enge.

http://cm.multiprecision.org/

The CM software implements the construction of ring class fields of imaginary quadratic number fields and of elliptic curves with complex multiplication via floating point approximations. It consists of libraries that can be called from within a C program and of executable command line applications. For the implemented algorithms, see [8].

- Version: 0.2 *Blindhühnchen*
- License: GPL v2+
- Programming language: C

4.5. AVIsogenies

Participants: Damien Robert [correspondent], Gaëtan Bisson, Romain Cosset [INRIA project-team CARAMEL].

http://avisogenies.gforge.inria.fr/.

AVISOGENIES (Abelian Varieties and Isogenies) is a MAGMA package for working with abelian varieties, with a particular emphasis on explicit isogeny computation.

Its prominent feature is the computation of (ℓ, ℓ)-isogenies between Jacobian varieties of genus-two hyperelliptic curves over finite fields of characteristic coprime to ℓ; practical runs have used values of ℓ in the hundreds.

It can also be used to compute endomorphism rings of abelian surfaces, and find complete addition laws on them.

- Version: 0.6
- License: LGPL v2.1+
- Programming language: Magma

4.6. APIP

Participant: Jérôme Milan.

http://www.lix.polytechnique.fr/~milanj/apip/apip.xhtml

APIP, Another Pairing Implementation in PARI, is a library for computing standard and optimised variants of most cryptographic pairings.

The following pairings are available: Weil, Tate, ate and twisted ate, optimised versions (à la Vercauteren–Hess) of ate and twisted ate for selected curve families.

The following methods to compute the Miller part are implemented: standard Miller double-and-add method, standard Miller using a non-adjacent form, Boxall et al. version, Boxall et al. version using a non-adjacent form.

The final exponentiation part can be computed using one of the following variants: naive exponentiation, interleaved method, Avanzi–Mihailescu's method, Kato et al.'s method, Scott et al.'s method.

Part of the library has been included into PARI/GP proper.

- Version: 2012-10-17
- License: GPL v2+
- Programming language: C with libpari

4.7. CMH

Participants: Andreas Enge, Emmanuel Thomé [INRIA project-team CARAMEL].

http://cmh.gforge.inria.fr/

CMH computes Igusa class polynomials, parameterising two-dimensional abelian varieties (or, equivalently, Jacobians of hyperelliptic curves of genus 2) with given complex multiplication.

- Version: 1.0
- License: GPL v3+
- Programming language: C

4.8. Cubic

Participant: Karim Belabas.

http://www.math.u-bordeaux1.fr/~belabas/research/software/cubic-1.2.tgz

CUBIC is a stand-alone program that prints out generating equations for cubic fields of either signature and bounded discriminant. It depends on the PARI library. The algorithm has quasi-linear time complexity in the size of the output.

- Version: 1.2
- License: GPL v2+
- Programming language: C

4.9. Euclid

Participant: Pierre Lezowski.

http://www.math.u-bordeaux1.fr/~plezowsk/euclid/index.php.

Euclid is a program to compute the Euclidean minimum of a number field. It is the practical implementation of the algorithm described in [38]. Some corresponding tables built with the algorithm are also available. Euclid is a stand-alone program depending on the PARI library.

- Version: 1.2
- License: LGPL v2+
- Programming language: C

4.10. KleinianGroups

Participant: Aurel Page.

http://www.normalesup.org/~page/Recherche/Logiciels/logiciels.html

KLEINIANGROUPS is a Magma package that computes fundamental domains of arithmetic Kleinian groups.

- Version: 1.0
- License: GPL v3+
- Programming language: Magma

POLSYS Project-Team

5. New Software and Platforms

5.1. FGb

Participant: Jean-Charles Faugère [contact].

FGb is a powerful software for computing Gröbner bases.It includes the new generation of algorihms for computing Gröbner bases polynomial systems (mainly the F4, F5 and FGLM algorithms). It is implemented in C/C++ (approximately 250000 lines), standalone servers are available on demand. Since 2006, FGb is dynamically linked with Maple software (version 11 and higher) and is part of the official distribution of this software.
See also the web page http://www-polsys.lip6.fr/~jcf/Software/FGb/index.html.

5.2. GBLA

Participants: Jean-Charles Faugère [contact], Brice Boyer.

- ACM: I.1.2 Algebraic algorithms
- Programming language: C/C++

GBLA a new open source C library for linear algebra dedicated to Gröbner bases computations (see http://www-polsys.lip6.fr/~jcf/Software/index.html).

5.3. RAGlib

Participant: Mohab Safey El Din [contact].

RAGLib is a Maple library for solving over the reals polynomial systems and computing sample points in semi-algebraic sets.

5.4. Epsilon

Participant: Dongming Wang [contact].

Epsilon is a library of functions implemented in Maple and Java for polynomial elimination and decomposition with (geometric) applications.

5.5. SLV

Participant: Elias Tsigaridas [contact].

SLV is a software package in C that provides routines for isolating (and subsequently refine) the real roots of univariate polynomials with integer or rational coefficients based on subdivision algorithms and on the continued fraction expansion of real numbers. Special attention is given so that the package can handle polynomials that have degree several thousands and size of coefficients hundrends of Megabytes. Currently the code consists of $\sim 5\,000$ lines.

- ACM: I.1.2 Algebraic algorithms
- Programming language: C/C++

SECRET Project-Team

5. New Software and Platforms

5.1. New Software

5.1.1. CFS Implementation

Participants: Grégory Landais, Nicolas Sendrier.

https://gforge.inria.fr/projects/cfs-signature/

Reference implementation of parallel CFS (reinforced version of the digital signature scheme CFS [93] due to Matthieu Finiasz [95]). Two variants are proposed, one with a « bit-packing » finite field arithmetic and an evolution with a « bit-slicing » finite-field arithmetic (collaboration with Peter Schwabe). For 80 bits of security the running time for producing one signature with the « bit-packing » variant is slightly above one second. This is high but was still the fastest so far. The evolution with the « bit-slicing » arithmetic produces the same signature in about 100 milliseconds.

5.1.2. Collision Decoding

Participants: Grégory Landais, Nicolas Sendrier.

https://gforge.inria.fr/projects/collision-dec/

Implementation of two variants of information set decoding, Stern-Dumer [97], [94] and MMT [96]. To our knowledge it is the best full-fledged open-source implementation of generic decoding of binary linear codes. It is the best generic attack against code-based cryptography. This software has the best score for breaking existing publicly available challenges (see http://pqcrypto.org/wild-challenges.html).

SPECFUN Project-Team

5. New Software and Platforms

5.1. SSReflect

SSReflect is a language extension of the Coq system and was originally written by G. Gonthier for his formal proof of the Four-Color Theorem [0]. In the team, A. Mahboubi and E. Tassi participate to its development, maintenance, distribution, documentation, and user support. A new version (1.5) was released in March 2014. The proof language now offers fine-grained control on type-classes inference and offers new proof commands to ease forward reasoning. In particular the 'have' tactic now supports new modifiers to ease stating generalized formulas as well as hoisting out deeply nested forward steps.

5.2. The Mathematical Components library

The Mathematical Components library is a set of Coq libraries that cover the mechanization of the proof of the Odd Order Theorem, with large contributions by A. Mahboubi and E. Tassi. After the formal proof was completed in September 2012, stable libraries had been distributed [0] with the SSReflect extension, while remaining parts of the libraries had remained under continued improvements in view of potential reuse. In March 2014, version 1.5 of library was released. With it, the library includes 16 more theory files, covering in particular field and Galois theory, advanced character theory, and a construction of algebraic numbers.

5.3. Coq

The way Coq processes theory files has been improved. When used as a batch compiler, Coq is now able to decouple the checking of statements and definitions from the checking of proofs. All proofs can be checked independently taking advantage of modern parallel hardware. When used interactively in conjunction with PIDE-based interfaces, Coq is now able to process the document asynchronously by delegating most of the task to external workers.

The Coq build process was also improved to better support the Windows platform and to enable third parties to provide pre-compiled plugins for such platform.

5.4. Coq/jEdit

Building on top of the asynchronous processing of Coq proofs, we have implemented a plugin that connects the jEdit generic text editor to Coq. This plugin is an adaptation of a similar plugin, written by M. Wenzel, for the Isabelle proof assistant. The interaction using this plugin is a significant change from existing user interfaces, making full use of Coq's asynchronous processing capabilities to provide richer feedback about the proof a user is editing.

The plugin was released as a beta in November 2014 and is available at http://pages.saclay.inria.fr/carst.tankink/jedit.html.

5.5. Other maintained software

We still actively maintain the following other software, which have not had a new release this year.

[0] http://www.msr-inria.fr/projects/mathematical-components/
[0] http://www.msr-inria.fr/projects/mathematical-components/

5.5.1. DDMF

(2007–): Web site consisting of interactive tables of mathematical formulas on elementary and special functions. The formulas are automatically generated by OCaml and computer-algebra routines. Users can ask for more terms of the expansions, more digits of the numerical values, proofs of some of the formulas, etc. See http://ddmf.msr-inria.inria.fr/1.9.1/ddmf. We count hundreds of user sessions per month. Source code distributed under CeCILL-B. A next release is under preparation: it will base on a different, more user-friendly rendering tool (MathJax) and will display more contents.

5.5.2. DynaMoW

(2007–): Programming tool for controlling the generation of mathematical websites that embed dynamical mathematical contents generated by computer-algebra calculations. Implemented in OCaml. See http://ddmf.msr-inria.inria.fr/DynaMoW/. Source code distributed under CeCILL-B.

5.5.3. Ring

(2004–): Coq normalization tool and decision procedure for expressions in commutative ring theories. Implemented in Coq and OCaml. Integrated in the standard distribution of the Coq proof assistant since 2005.

5.5.4. Mgfun

(1994–): Maple package for symbolic summation, integration, and other closure properties of multivariate special functions. Now distributed as part of Algolib, a collection of packages for combinatorics and manipulations of special functions, available at http://algo.inria.fr/libraries/. This software has been used this year for our formal proof of irrationality of $\zeta(3)$.

VEGAS Project-Team

4. New Software and Platforms

4.1. QI: Quadrics Intersection

QI stands for "Quadrics Intersection". QI is the first exact, robust, efficient and usable implementation of an algorithm for parameterizing the intersection of two arbitrary quadrics, given in implicit form, with integer coefficients. This implementation is based on the parameterization method described in [5] [29] and represents the first complete and robust solution to what is perhaps the most basic problem of solid modeling by implicit curved surfaces.

QI is written in C++ and builds upon the LiDIA computational number theory library [20] bundled with the GMP multi-precision integer arithmetic [19]. QI can routinely compute parameterizations of quadrics having coefficients with up to 50 digits in less than 100 milliseconds on an average PC; see [29] for detailed benchmarks.

Our implementation consists of roughly 18,000 lines of source code. QI has being registered at the Agence pour la Protection des Programmes (APP). It is distributed under a free for non-commercial use Inria license and will be distributed under the QPL license in the next release. The implementation can also be queried via a web interface [21].

Since its official first release in June 2004, QI has been downloaded six times a month on average and it has been included in the geometric library EXACUS developed at the Max-Planck-Institut für Informatik (Saarbrücken, Germany). QI is also used in a broad range of applications; for instance, it is used in photochemistry for studying the interactions between potential energy surfaces, in computer vision for computing the image of conics seen by a catadioptric camera with a paraboloidal mirror, and in mathematics for computing flows of hypersurfaces of revolution based on constant-volume average curvature.

4.2. Isotop: Topology and geometry of planar algebraic curves

ISOTOP is a Maple software for computing the topology of an algebraic plane curve, that is, for computing an arrangement of polylines isotopic to the input curve. This problem is a necessary key step for computing arrangements of algebraic curves and has also applications for curve plotting. This software has been developed since 2007 in collaboration with F. Rouillier from Inria Paris - Rocquencourt. It is based on the method described in [3] which incorporates several improvements over previous methods. In particular, our approach does not require generic position.

Isotop is registered at the APP (June 15th 2011). This version is competitive with other implementations (such as ALCIX and INSULATE developed at MPII Saarbrücken, Germany and TOP developed at Santander Univ., Spain). It performs similarly for small-degree curves and performs significantly better for higher degrees, in particular when the curves are not in generic position.

We are currently working on an improved version integrating our new bivariate polynomial solver.

4.3. CGAL: Computational Geometry Algorithms Library

Born as a European project, CGAL (http://www.cgal.org) has become the standard library for computational geometry. It offers easy access to efficient and reliable geometric algorithms in the form of a C++ library. CGAL is used in various areas needing geometric computation, such as: computer graphics, scientific visualization, computer aided design and modeling, geographic information systems, molecular biology, medical imaging, robotics and motion planning, mesh generation, numerical methods...

In computational geometry, many problems lead to standard, though difficult, algebraic questions such as computing the real roots of a system of equations, computing the sign of a polynomial at the roots of a system, or determining the dimension of a set of solutions. We want to make state-of-the-art algebraic software more accessible to the computational geometry community, in particular, through the computational geometric library CGAL. On this line, we contributed a model of the *Univariate Algebraic Kernel* concept for algebraic computations [23] (see Sections 8.2.2 and 8.4). This CGAL package improves, for instance, the efficiency of the computation of arrangements of polynomial functions in CGAL [30]. We are currently developing a model of the *Bivariate Algebraic Kernel* based on a new bivariate polynomial solver.

4.4. Fast_polynomial: fast polynomial evaluation software

The library *fast_polynomial*[0] provides fast evaluation and composition of polynomials over several types of data. It is interfaced for the computer algebra system *Sage* and its algorithms are documented [0]. This software is meant to be a first step toward a certified numerical software to compute the topology of algebraic curves and surfaces. It can also be useful as is and is submitted for integration in the computer algebra system *Sage*.

This software is focused on *fast online computation*, *multivariate evaluation*, *modularity*, and *efficiency*.

Fast online computation. The library is optimized for the evaluation of a polynomial on several point arguments given one after the other. The main motivation is numerical path tracking of algebraic curves, where a given polynomial criterion must be evaluated several thousands of times on different values arising along the path.

Multivariate evaluation. The library provides specialized fast evaluation of multivariate polynomials with several schemes, specialized for different types such as *mpz* big ints, *boost* intervals with hardware precision, *mpfi* intervals with any given precision, etc.

Modularity. The evaluation scheme can be easily changed and adapted to the user needs. Moreover, the code is designed to easily extend the library with specialization over new *C++* objects.

Efficiency. The library uses several tools and methods to provide high efficiency. First, the code uses templates, such that after the compilation of a polynomial for a specific type, the evaluation performance is equivalent to low-level evaluation. Locality is also taken into account: the memory footprint is minimized, such that an evaluation using the classical Hörner scheme will use $O(1)$ temporary objects and divide and conquer schemes will use $O(\log n)$ temporary objects, where n is the degree of the polynomial. Finally, divide and conquer schemes can be evaluated in parallel, using a number of threads provided by the user.

[0] http://trac.sagemath.org/sage_trac/ticket/13358
[0] http://arxiv.org/abs/1307.5655

ALF Project-Team

5. New Software and Platforms

5.1. Panorama

The ALF team is developing several software prototypes for research purposes: compilers, architectural simulators, programming environments ....

Among the prototypes developed in the project, this section reports only the softwares that had significant revisions in 2014. Among the softwares available from the project website and not reported here, **ATMI** **http://www.irisa.fr/alf/atmi**, a microarchitecture temperature model for processor simulation, **STiMuL** **http://www.irisa.fr/alf/stimul**, a temperature model for steady state studies, **ATC** **http://www.irisa.fr/alf/atc**, an address trace compressor, and **HAVEGE** **http://www.irisa.fr/alf/havege** an unpredictable random number generator.

5.2. TPCalc

Participant: Pierre Michaud.

microarchitecture simulation

TPCalc is a throughput calculator for microarchitecture studies concerned with multi-program workloads consisting of sequential programs. Because microarchitecture simulators are slow, it is difficult to simulate throughput experiments where a multicore executes many jobs that enter and leave the system. The usual practice of measuring instantaneous throughput on independent coschedules chosen more or less randomly is not a rigorous practice because it assumes that all the coschedules are equally important, which is not always true. TPCalc can compute the average throughput of a throughput experiment without actually doing the throughput experiment. The user first defines the workload heterogeneity (number of different job types), the multicore configuration (number of cores and symmetries). TPCalc provides a list of base coschedules. The user then simulates these coschedules, using some benchmarks of his/her choice, and feeds back to TPCalc the measured execution rates (e.g., instructions per cycle or instructions per second).TPCalc eventually outputs the average throughput. Several throughput metrics are available, corresponding to different workload assumptions. These metrics are described in our ACM TACO paper, a collaboration with Ghent University [15].

TPCalc is an open-source software written in C++. It runs on Unix-based systems (Linux, OS X ...). It is available for download at http://www.irisa.fr/alf/downloads/michaud/tpcalc.html.

5.3. Heptane

Participants: Isabelle Puaut, Damien Hardy.

WCET estimation

Status: Registered with APP (Agence de Protection des Programmes). Available under GNU General Public License v3, with number IDDN.FR.001.510039.000.S.P.2003.000.10600.

The aim of Heptane is to produce upper bounds of the execution times of applications. It is targeted at applications with hard real-time requirements (automotive, railway, aerospace domains). Heptane computes WCETs using static analysis at the binary code level. It includes static analyses of microarchitectural elements such as caches and cache hierarchies.

For more information, please contact Damien Hardy or Isabelle Puaut.

5.4. Tiptop

Participant: Erven Rohou.

Performance, hardware counters, analysis tool.

Status: Registered with APP (Agence de Protection des Programmes). Available under GNU General Public License v2, with number IDDN.FR.001.450006.000.S.P.2011.000.10800. Current version is 2.2, released March 2013.

Tiptop has been integrated in major Linux distributions, such as Fedora, Debian, Ubuntu.

Tiptop is a new simple and flexible user-level tool that collects hardware counter data on Linux platforms (version 2.6.31+). The goal is to make the collection of performance and bottleneck data as simple as possible, including simple installation and usage. In particular, we stress the following points.

- Installation is only a matter of compiling the source code. No patching of the Linux kernel is needed, and no special-purpose module needs to be loaded.
- No privilege is required, any user can run *tiptop* — non-privileged users can only watch processes they own, ability to monitor anybody's process opens the door to side-channel attacks.
- The usage is similar to *top*. There is no need for the source code of the applications of interest, making it possible to monitor proprietary applications or libraries. And since there is no probe to insert in the application, understanding of the structure and implementation of complex algorithms and code bases is not required.
- Applications do not need to be restarted, and monitoring can start at any time (obviously, only events that occur after the start of *tiptop* are observed).
- Events can be counted per thread, or per process.
- Any expression can be computed, using the basic arithmetic operators, constants, and counter values.
- A configuration file lets users define their prefered setup, as well as custom expressions.

Tiptop is written in C. It can take advantage of libncurses when available for pseudo-graphic display.

For more information, please contact Erven Rohou or visit http://tiptop.gforge.inria.fr.

5.5. Padrone

Participants: Erven Rohou, Emmanuel Riou.

Performance, profiling, dynamic optimization

Status: Ongoing development, early prototype. Registered with APP (Agence de Protection des Programmes).

Padrone is new platform for dynamic binary analysis and optimization. It provides an API to help clients design and develop analysis and optimization tools for binary executables. Padrone attaches to running applications, only needing the executable binary in memory. No source code or debug information is needed. No application restart is needed either. This is specially interesting for legacy or commercial applications, but also in the context of cloud deployment, where actual hardware is unknown, and other applications competing for hardware resources can vary. The profiling overhead is minimum.

Padrone is instrumental to the PhD developments of Nabil Hallou.

Padrone is written in C.

For more information, please contact Erven Rohou.

5.6. Barra

Participant: Sylvain Collange.

GPU simulator

Other Contributors : David Defour (Université de Perpignan), Alexandre Kouyoumdjian (Inria), Elie Gedeon (ENS Lyon), Fabrice Mouhartem (Inria)

Status : APP registration in progress. Available under the new BSD License

Research on throughput-oriented architectures demands accurate and representative models of GPU architectures in order to be able to evaluate new architectural ideas, explore design spaces and characterize applications. The Barra project [0] is a simulator of the NVIDIA Tesla GPU architecture.

Barra builds upon knowledge acquired through micro-benchmarking, in order to provide a baseline model representative of industry practice. The simulator provides detailed statistics to identify optimization opportunities and is fully customizable to experiment ideas of architectural modifications. Barra incorporates both a functional model and a cycle-level performance model.

Visit http://barra.gforge.inria.fr/ or contact Sylvain Collange.

[0] http://gforge.inria.fr/plugins/mediawiki/wiki/barra/index.php/Main_Page

ATEAMS Project-Team

4. New Software and Platforms

4.1. MicroMachinations

Participant: Riemer Van Rozen [correspondent].

Characterization: A-2-up3, SO-4, SM-2-up3, EM-3, SDL-3-up4, OC-DA-3-CD-3-MS-3-TPM-3.

WWW:

Objective: To create an integrated, live environment for modeling and evolving game economies. This will allow game designers to experiment with different strategies to realize game mechanics. The environment integrates with the SPIN model checker to prove properties (reachability, liveness). A runtime system for executing game economies allows MicroMachinations models to be embedded in actual games.

Users: Game designers

Impact: One of the important problems in game software development is the distance between game design and implementation in software. MicroMachinations has the potential to bridge this gap by providing live design tools that directly modify or create the desired software behaviors.

Competition: None.

Engineering: The front-end of MicroMachinations is built using the Rascal language workbench, including visualization, model checking, debugging and standard IDE features. The runtime library is implemented in C++ and will be evaluated in the context of industrial game design.

Publications: [11]

4.1.1. Novelties

- MMLib was finished to allow the execution of game economies directly within games. This supports "Live programming" of the behavior of games. The library has been used in the development of the real-life game "Johnny Jetstream", designed by IC3DMedia.

4.2. Naked Object Algebras

Participant: Tijs Van Der Storm [correspondent].

Characterization: A5, SO-4, SM-4, EM-4, SDL-4-up5, OC-DA-3-CD-3-MS-3-TPM-3.

WWW: https://github.com/cwi-swat/naked-object-algebras

Objective: Supporting modular and extensible language development.

Users: Programmers, language designers.

Impact: Object Algebras promise a new level of modularity and extensibility in the implementation of recursive data types. The NAO framework lifts this to the implementation of software languages, including the declarative declaration of concrete syntax.

Competition: Language prototyping tools.

Engineering: NAO consists of a few hundred lines of Java code. It has no external dependencies, except ANTLR for parsing.

Publications: [27], [33]

4.2.1. Novelties

- NAO has been used to develop an extensible variant of the QL questionnaire language [33].

4.3. Rascal

Participants: Paul Klint, Jurgen Vinju [correspondent], Tijs Van Der Storm, Pablo Inostroza Valdera, Davy Landman, Bert Lisser, Atze Van Der Ploeg, Vadim Zaytsev, Anastasia Izmaylova, Michael Steindorfer, Jouke Stoel, Ali Afroozeh, Ashim Shahi.

Characterization: A5, SO-4, SM-4, EM-4, SDL-4-up5, OC-DA-3-CD-3-MS-3-TPM-3.

WWW: http://www.rascal-mpl.org

Objective: Provide a completely integrated programming language parametric meta programming language for the construction of any kind of meta program for any kind of programming language: analysis, transformation, generation, visualization.

Users: Researchers in model driven engineering, programming languages, software engineering, software analysis, as well as practitioners that need specialized tools.

Impact: Rascal is making the mechanics of meta programming into a non-issue. We can now focus on the interesting details of the particular fact extraction, model, source analysis, domain analysis as opposed to being distracted by the engineering details. Simple things are easy in Rascal and complex things are manageable, due to the integration, the general type system and high-level programming features.

Competition: There is a plethora of meta programming toolboxes and frameworks available, ranging from plain parser generators to fully integrated environments. Rascal is distinguished because it is a programming language rather than a specification formalism and because it completely integrates different technical domains (syntax definition, term rewriting, relational calculus). For simple tools, Rascal competes with scripting languages and for complex tools it competes context-free general parser generators, with query engines based on relational calculus and with term rewriting and strategic programming languages.

Engineering: Rascal is about 100 kLOC of Java code, designed by a core team of three and with a team of around 8 PhD students and post-docs contributing to its design, implementation and maintenance. The goal is to work towards more bootstrapping and less Java code as the project continues.

Publications: [7], [6], [8], [5], [6]

4.3.1. Novelties

- Improvements of the language-parametric model to represent software projects (M3) [9].
- Performance improvements of the Rascal interpreter throughout.
- Further improvements to the compiler for Rascal, based on new language construct guarded coroutines.
- New language feature: keyword parameters. This will further allow simplificiation of the core language, as well as support better extensibility.
- Significant improvements to the Rascal static type checker.
- Further improvements to the new GLL parser (Iguana).
- Design of a new DSL for describing core banking infrastructure was started (ReBEL). Rascal was also used to develop a state machine DSL for use in embedded devices (Machino).

4.4. IDE Meta-tooling Platform

Participants: Jurgen Vinju [correspondent], Michael Steindorfer.

IMP, the IDE meta tooling platform is an Eclipse plugin developed mainly by the team of Robert M. Fuhrer at IBM TJ Watson Research institute. It is both an abstract layer for Eclipse, allowing rapid development of Eclipse based IDEs for programming languages, and a collection of meta programming tools for generating source code analysis and transformation tools.

Characterization: A5, SO-3, SM4-up5, EM-4, SDL-5, DA-2-CD-2-MS-2-TPM-2

WWW: https://github.com/impulse-org/

Objective: The IDE Meta Tooling Platform (IMP) provides a high-level abstraction over the Eclipse API such that programmers can extend Eclipse with new programming languages or domain specific languages in a few simple steps. IMP also provides a number of standard meta tools such as a parser generator and a domain specific language for formal specifications of configuration parameters.

Users: Designers and implementers of IDEs for programming languages and domain specific languages. Also, designers and implementers of meta programming tools.

Impact: IMP is a popular among meta programmers especially for it provides the right level of abstraction.

Competition: IMP competes with other Eclipse plugins for meta programming (such as Model Driven Engineering tools), but its API is more general and more flexible. IMP is a programmers framework rather than a set of generators.

Engineering: IMP is a long-lived project of many contributors, which is managed as an Eclipse incubation project at `eclipse.org`. Currently we are moving the project to Github to explore more different ways of collaboration.

Publications: [2] [29]

4.4.1. Novelties

- Significant performance improvements to the IMP program database. Performance is now better than equivalent data structure libraries in Scala and Clojure.

4.5. Ensō

Participant: Tijs Van Der Storm [correspondent].

Characterization: A5, SO-4, SM-3-up-4, EM-2-up-4, SDL-4, OC-DA-4-CD-4-MS-4-TPM-4

WWW: http://www.enso-lang.org

Objective: Together with Prof. Dr. William R. Cook of the University of Texas at Austin, and Alex Loh, Tijs van der Storm has been designing and implementing a new programming system, called Ensō. Ensō is theoretically sound and practical reformulation of model-based development. It is based on model-interpretation as opposed to model transformation and code generation. Currently, the system already supports models for schemas (data models), web applications, context-free grammars, diagram editors and security.

Users: All programmers.

Impact: Ensō has the potential to revolutionize the activity of programming. By looking at model driven engineering from a completely fresh perspective, with as key ingredients interpreters and partial evaluation, it may make higher level (domain level) program construction and maintenance as effective as normal programming.

Competition: Ensō competes as a programming paradigm with model driven engineering tools and generic programming and languages that provide syntax macros and language extensions.

Engineering: Ensō is a completely self-hosted system in 7000 lines of code.

Publications: [14], [16], [13]

CAIRN Project-Team

5. New Software and Platforms

5.1. Panorama

With the ever raising complexity of embedded applications and platforms, the need for efficient and customizable compilation flows is stronger than ever. This need of flexibility is even stronger when it comes to research compiler infrastructures that are necessary to gather quantitative evidence of the performance/energy or cost benefits obtained through the use of reconfigurable platforms. From a compiler point of view, the challenges exposed by these complex reconfigurable platforms are quite significant, since they require the compiler to extract and to expose an important amount of coarse and/or fine grain parallelism, to take complex resource constraints into consideration while providing efficient memory hierarchy and power management.

Because they are geared toward industrial use, production compiler infrastructures do not offer the level of flexibility and productivity that is required for compiler and CAD tool prototyping. To address this issue, we have designed an extensible source-to-source compiler infrastructure that takes advantage of leading edge model-driven object-oriented software engineering principles and technologies.

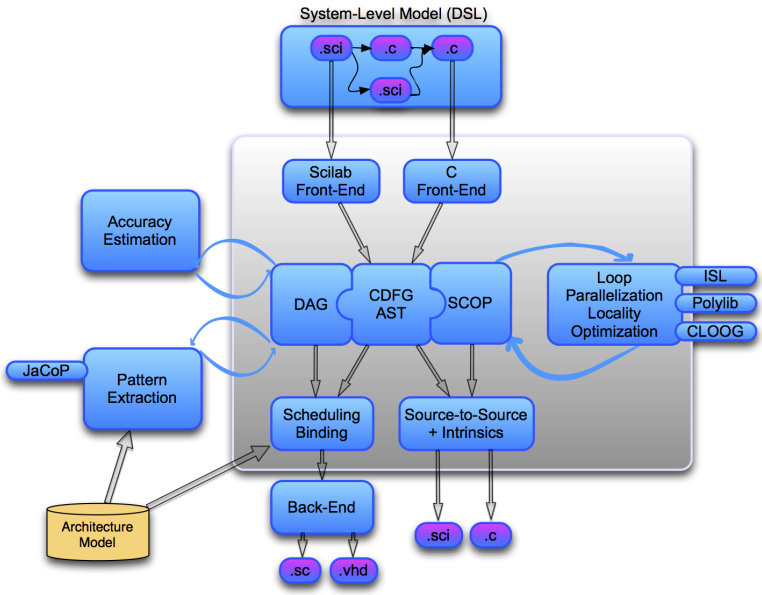

Figure 2. CAIRN*'s general software development framework.*

Figure 2 shows the global framework that is being developed in the group. Our compiler flow mixes several types of intermediate representations. The baseline representation is a simple tree-based model enriched with control flow information. This model is mainly used to support our source-to-source flow, and serves as the backbone for the infrastructure. We use the extensibility of the framework to provide more advanced representations along with their corresponding optimizations and code generation plug-ins. For example, for our pattern selection and accuracy estimation tools, we use a data dependence graph model in all basic

blocks instead of the tree model. Similarly, to enable polyhedral based program transformations and analysis, we introduced a specific representation for affine control loops that we use to derive a Polyhedral Reduced Dependence Graph (PRDG). Our current flow assumes that the application is specified as a system level hierarchy of communicating tasks, where each task is expressed using C (or Scilab in the short future), and where the system level representation and the target platform model are defined using Domain Specific Languages (DSL).

Gecos (Generic Compiler Suite) is the main backbone of CAIRN's flow. It is an open source Eclipse-based flexible compiler infrastructure developed for fast prototyping of complex compiler passes. Gecos is a 100% Java based implementation and is based on modern software engineering practices such as Eclipse plugin or model-driven software engineering with EMF (Eclipse Modeling Framework). As of today, our flow offers the following features:

- An automatic floating-point to fixed-point conversion flow (for HLS and embedded processors). **ID.Fix** is an infrastructure for the automatic transformation of software code aiming at the conversion of floating-point data types into a fixed-point representation. http://idfix.gforge.inria.fr.
- A polyhedral-based loop transformation and parallelization engine (mostly targeted at HLS). http://gecos.gforge.inria.fr. It was used for source-to-source transformations in the context of Nano2012 projects in collaboration with STMicroelectronics.
- A custom instruction extraction flow (for ASIP and dynamically reconfigurable architectures). **Durase** and **UPaK** are developed for the compilation and the synthesis targeting reconfigurable platforms and the automatic synthesis of application specific processor extensions. They use advanced technologies, such as graph matching and graph merging together with constraint programming methods.
- Several back-ends to enable the generation of VHDL for specialized or reconfigurable IPs, and SystemC for simulation purposes (e.g., fixed-point simulations).

5.2. Gecos

Participants: Steven Derrien [corresponding author], Nicolas Simon, Antoine Morvan.

Keywords: source-to-source compiler, model-driven software engineering, retargetable compilation.

The Gecos (Generic Compiler Suite) project is a source-to-source compiler infrastructure developed in the Cairn group since 2004. It was designed to enable fast prototyping of program analysis and transformation for hardware synthesis and retargetable compilation domains.

Gecos is 100% Java based and takes advantage of modern model driven software engineering practices. It uses the Eclipse Modeling Framework (EMF) as an underlying infrastructure and takes benefits of its features to make it easily extensible. Gecos is open-source and is hosted on the Inria gforge at http://gecos.gforge.inria.fr.

The Gecos infrastructure is still under very active development, and serves as a backbone infrastructure to projects of the group. Part of the framework is jointly developed with Colorado State University and since 2012 it is used in the context of the ALMA European project. Recent developments in Gecos have focused on polyhedral loop transformations and efficient SIMD code generation for fixed point arithmetic data-types as a part of the ALMA project. Significant efforts were also put to provide a coarse-grain parallelization engine targeting the data-flow actor model in the context of the COMPA ANR project.

5.3. ID.Fix: Infrastructure for the Design of Fixed-point Systems

Participants: Olivier Sentieys [corresponding author], Romuald Rocher, Nicolas Simon.

Keywords: fixed-point arithmetic, source-to-source code transformation, accuracy optimization, dynamic range evaluation

The different techniques proposed by the team for fixed-point conversion are implemented on the ID.Fix infrastructure. The application is described with a C code using floating-point data types and different pragmas, used to specify parameters (dynamic, input/output word-length, delay operations) for the fixed-point conversion. This tool determines and optimizes the fixed-point specification and then, generates a C code using fixed-point data types (ac_fixed) from Mentor Graphics. The infrastructure is made-up of two main modules corresponding to the fixed-point conversion (ID.Fix-Conv) and the accuracy evaluation (ID.Fix-Eval)

The different developments carried out in 2014 allowed to have a complete compatibility with GeCos. The structure of each node in the graph has been changed to simplify the graph modifications. The Octave software has been added instead of Matlab for LTI and recursive systems conversion. A development has started to replace Matlab/Octave tool by a C code algorithm to reduce optimization time. In the context of the ANR DEFIS project, the ID.Fix tool has been reorganized to be integrated in the DEFIS toolflow.

In 2014, ID.Fix has been demonstrated during University Booth at IEEE/ACM DATE.

5.4. UPaK: Abstract Unified Pattern-Based Synthesis Kernel for Hardware and Software Systems

Participants: Christophe Wolinski [corresponding author], François Charot.

Keywords: compilation for reconfigurable systems, pattern extraction, constraint-based programming.

We are developing (with strong collaboration of Lund University, Sweden and Queensland University, Australia) UPaK *Abstract Unified Pattern Based Synthesis Kernel for Hardware and Software Systems* [117]. The preliminary experimental results obtained by the UPak system show that the methods employed in the systems enable a high coverage of application graphs with small quantities of patterns. Moreover, high application execution speed-ups are ensured, both for sequential and parallel application execution with processor extensions implementing the selected patterns. UPaK is one of the basis for our research on compilation and synthesis for reconfigurable platforms. It is based on the HCDG representation of the Polychrony software designed at Inria-Rennes in the project-team Espresso.

5.5. DURASE: Automatic Synthesis of Application-Specific Processor Extensions

Participants: Christophe Wolinski [corresponding author], François Charot.

Keywords: compilation for reconfigurable systems, instruction-set extension, pattern extraction, graph covering, constraint-based programming.

We are developing a framework enabling the automatic synthesis of application specific processor extensions. It uses advanced technologies, such as algorithms for graph matching and graph merging together with constraints programming methods. The framework is organized around several modules.

- CoSaP: Constraint Satisfaction Problem. The goal of CoSaP is to decouple the statement of a constraint satisfaction problem from the solver used to solve it. The CoSaP model is an Eclipse plugin described using EMF to take advantage of the automatic code generation and of various EMF tools.
- HCDG: Hierarchical Conditional Dependency Graph. HCDG is an intermediate representation mixing control and data flow in a single acyclic representation. The control flow is represented as hierarchical guards specifying the execution or the definition conditions of nodes. It can be used in the Gecos compilation framework via a specific pass which translates a CDFG representation into an HCDG.
- Patterns: Flexible tools for identification of computational pattern in a graph and graph covering. These tools model the concept of pattern in a graph and provide generic algorithms for the identification of pattern and the covering of a graph. The following sub-problems are addressed: (sub)-graphs isomorphism, patterns generation under constraints, covering of a graph using a library of patterns. Most of the implemented algorithms use constraints programming and rely on the CoSaP module to solve the optimization problem.

5.6. PowWow: Power Optimized Hardware and Software FrameWork for Wireless Motes (AP-L-10-01)

Participants: Olivier Sentieys [corresponding author], Olivier Berder, Arnaud Carer, Steven Derrien.

Keywords: Wireless Sensor Networks, Low Power, Preamble Sampling MAC Protocol, Hardware and Software Platform

PowWow is an open-source hardware and software platform designed to handle wireless sensor network (WSN) protocols and related applications. Based on an optimized preamble sampling medium access (MAC) protocol, geographical routing and `protothread` library, PowWow requires a lighter hardware system than Zigbee [79] to be processed (memory usage including application is less than 10kb). Therefore, network lifetime is increased and price per node is significantly decreased.

CAIRN's hardware platform (see Figure 3) is composed of:

- The motherboard, designed to reduce power consumption of sensor nodes, embeds an MSP430 microcontroller and all needed components to process PowWow protocol except radio chip. JTAG, RS232, and I2C interfaces are available on this board.
- The radio chip daughter board is currently based on a TI CC2420.
- The coprocessing daughter board includes a low-power FPGA which allows for hardware acceleration for some PowWow features and also includes dynamic voltage scaling features to increase power efficiency. The current version of PowWow integrates an Actel IGLOO AGL250 FPGA and a programmable DC-DC converter. We have shown that gains in energy of up to 700 can be obtained by using FPGA acceleration on functions like CRC-32 or error detection with regards to a software implementation on the MSP430.
- Finally, a last daughter board is dedicated to energy harvesting techniques. Based on the energy management component LTC3108 from Linear Technologies, the board can be configured with several types of stored energy (batteries, micro-batteries, super-capacitors) and several types of energy sources (a small solar panel to recover photovoltaic energy, a piezoelectric sensor for mechanical energy and a Peltier thermal energy sensor).

Figure 3. CAIRN*'s PowWow motherboard with radio and energy-harvesting boards connected*

PowWow distribution also includes a generic software architecture using event-driven programming and organized into protocol layers (PHY, MAC, LINK, NET and APP). The software is based on Contiki [95], and more precisely on the `Protothread` library which provides a sequential control flow without complex state machines or full multi-threading.

To optimize the network regarding a particular application and to define a global strategy to reduce energy, PowWow offers the following extra tools: over-the-air reprogramming (and soon reconfiguration), analytical power estimation based on software profiling and power measurements, a dedicated network analyzer to probe and fix transmissions errors in the network. More information can be found at http://powwow.gforge.inria.fr.

5.7. Ziggie: a Platform for Wireless Body Sensor Networks

Participants: Olivier Sentieys, Olivier Berder, Arnaud Carer, Antoine Courtay [corresponding author], Robin Bonamy.

Keywords: Wireless Body Sensor Networks, Low Power, Gesture Recognition, Localization, Hardware and Software Platform

The Zyggie sensor node has been developed in the team to create an autonomous Wireless Body Sensor Network (WBSN) with the capabilities of monitoring body movements. The Zyggie platform is part of the BoWI project funded by CominLabs. Zyggie is composed of: an ATMEGA128RFA1 microcontroller, an MPU9150 Inertial Measurement Unit (IMU), an RF AS193 switch with two antennas, an LSP331AP barometer, a DC/DC voltage regulator with a battery charge controller, a wireless inductive battery charge controller, and some switches and control LEDs.

Figure 4. CAIRN*'s Ziggie platform for WBSN*

The IMU is composed of a 3-axis accelerometer, a 3-axis gyrometer and a 3-axis magnetometer. The IMU is communicating its data to the embedded microcontroller via an I2C protocol. We also developed our own MAC protocol for synchronization and data exchanges between nodes. The Zyggie platform is used in many PhD works for evaluating data fusion algorithms (RSSI + IMU data) (Zhongwei Zheng, UR1 and Alexis Aulery, UBS/UR1), low power computing algorithms (Alexis Aulery, UBS/UR1), wireless protocols (Viet Hoa Nguyen, UR1) and body channel characterization (Rizwan Masood, TB).

CAMUS Team

5. New Software and Platforms

5.1. PolyLib

Participant: Vincent Loechner.

PolyLib [0] is a C library of polyhedral functions, that can manipulate unions of rational polyhedra of any dimension. It was the first to provide an implementation of the computation of parametric vertices of a parametric polyhedron, and the computation of an Ehrhart polynomial (expressing the number of integer points contained in a parametric polytope) based on an interpolation method. Vincent Loechner is the maintainer of this software. It is distributed under GNU General Public License version 3 or later.

Apart from normal maintenance, it was parallelized using OpenMP with the support of Master student Adilla Susungi, funded by the ICPS team (ICube laboratory, University of Strasbourg).

5.2. APOLLO software and LLVM

Participants: Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño, Willy Wolff, Luis Esteban Campostrini, Matías Perez, Alexandra Jimborean, Philippe Clauss.

We are developing a framework called APOLLO (Automatic speculative POLyhedral Loop Optimizer) whose main concepts are based on our previous framework VMAD. However, several important implementation issues are now handled differently in order to improve the performance and usability of the framework, and also to open its evolution to new interesting perspectives. APOLLO is dedicated to automatic, dynamic and speculative parallelization of loop nests that cannot be handled efficiently at compile-time. It is composed of a static part consisting of specific passes in the LLVM compiler suite, plus a modified Clang frontend, and a dynamic part consisting of a runtime system. Its last extensions are presented in subsection 6.2 .

5.3. IBB source-to-source xfor compiler

Participants: Imen Fassi, Philippe Clauss, Cédric Bastoul.

Imen Fassi has developped a source-to-source xfor compiler called IBB (Iterate-But-Better) which is automatically translating any C source code containing xfor-loops into an equivalent source code where xfor-loops have been transformed into equivalent for-loops. The polyhedral code generator CLooG [27] is used to generate the corresponding code. The IBB compiler has been improved in some aspects in 2014: loop bounds can now be min and max functions, IBB uses the OpenScop format to encode statements and iteration domains.

The xfor structure is also currently incorporated in the polyhedral parser Clan [0], opening the door of xfor usage to polyhedral compilation tools. Additionally, an xfor programming wizard is currently being developed, providing automatic dependence analysis and code verification to users, thanks to the dependence analyzer Candl [0].

5.4. CLooG

Participant: Cédric Bastoul.

[0] http://icps.u-strasbg.fr/PolyLib
[0] http://icps.u-strasbg.fr/~bastoul/development/clan
[0] http://icps.u-strasbg.fr/~bastoul/development/candl

CLooG [0] is a free software and library to generate code (or an abstract syntax tree of a code) for scanning Z-polyhedra. That is, it finds a code (e.g. in C, FORTRAN...) that reaches each integral point of one or more parameterized polyhedra. CLooG has been originally written to solve the code generation problem for optimizing compilers based on the polyhedral model. Nevertheless it is used now in various area e.g. to build control automata for high-level synthesis or to find the best polynomial approximation of a function. CLooG may help in any situation where scanning polyhedra matters. While the user has full control on generated code quality, CLooG is designed to avoid control overhead and to produce a very effective code. CLooG is widely used (including by GCC and LLVM compilers), disseminated (it is installed by default by the main Linux distributions) and considered as the state of the art in polyhedral code generation.

5.5. OpenScop

Participant: Cédric Bastoul.

OpenScop [0] is an open specification that defines a file format and a set of data structures to represent a static control part (SCoP for short), i.e., a program part that can be represented in the polyhedral model. The goal of OpenScop is to provide a common interface to the different polyhedral compilation tools in order to simplify their interaction. To help the tool developers to adopt this specification, OpenScop comes with an example library (under 3-clause BSD license) that provides an implementation of the most important functionalities necessary to work with OpenScop.

5.6. Clan

Participants: Cédric Bastoul, Imen Fassi.

Clan [0] is a free software and library which translates some particular parts of high level programs written in C, C++, C# or Java into a polyhedral representation called OpenScop. This representation may be manipulated by other tools to, e.g., achieve complex analyses or program restructurations (for optimization, parallelization or any other kind of manipulation). It has been created to avoid tedious and error-prone input file writing for polyhedral tools (such as CLooG, LeTSeE, Candl etc.). Using Clan, the user has to deal with source codes based on C grammar only (as C, C++, C# or Java). Clan is notably the frontend of the two major high-level compilers Pluto and PoCC.

5.7. Clay

Participant: Cédric Bastoul.

Clay [0] is a free software and library devoted to semi-automatic optimization using the polyhedral model. It can input a high-level program or its polyhedral representation and transform it according to a transformation script. Classic loop transformations primitives are provided. Clay is able to check for the legality of the complete sequence of transformation and to suggest corrections to the user if the original semantics is not preserved. Clay is still experimental at this report redaction time but is already used during advanced compilation labs at Paris-Sud University and is one of the foundations of our ongoing work on simplifying code manipulation by programmers.

[0] http://www.cloog.org
[0] http://icps.u-strasbg.fr/~bastoul/development/openscop
[0] http://icps.u-strasbg.fr/~bastoul/development/clan
[0] http://icps.u-strasbg.fr/~bastoul/development/clay

COMPSYS Project-Team

5. New Software and Platforms

5.1. Introduction

This section lists and briefly describes the software developments conducted within Compsys. Most are tools that we extend and maintain over the years. They mainly concern three activities: a) the development of research tools, in general available on demand, linked to polyhedra and loop/array transformations, b) the development of tools linked to the start-up XTREMLOGIC (mostly done outside Compsys but partly inspired by work from Compsys), and c) the development of algorithms in the context of our collaborations with STMicroelectronics. These last developments have been stopped right now, since the end of the Mediacom project in 2012. They are described in previous Compsys activity reports: they concerned register allocation, SSA deconstruction, liveness analysis, intermediate representations, etc. They were done within the compilers of STMicroelectronics, not as stand-alone tools, so they are not available as the other tools.

Concerning tools based on a polyhedral representation of nested loops, many of them are now available. They have been developed and maintained over the years by different teams, after the introduction of Paul Feautrier's Pip, a tool for parametric integer linear programming. This "polyhedral model" view of codes is now widely accepted: it was or is used by Cairn, Parkas, and Camus Inria project-teams, PIPS at École des Mines de Paris, Suif from Stanford University, Compaan at Berkeley and Leiden, PiCo from the HP-Labs (continued as PicoExpress by Synfora and now Synopsis), the DTSE methodology at Imec, Sadayappan's group at Ohio State University, Rajopadhye's group at Colorado State's University, etc. In the last 10 years, several compiler groups have shown their interest in polyhedral methods, e.g., the Gcc group, IBM, and Reservoir Labs, a company that develops a compiler fully based on the polyhedral model and on the techniques that we (the french community) introduced for loop and array transformations. Polyhedra are also used in test and certification projects (Verimag, Lande, Vertecs). Now that these techniques are well-established and disseminated in and by other groups, we prefer to focus on the development of new techniques and tools, which are described here. Some of these tools can be used through a web interface on the Compsys tool demonstrator web page http://compsys-tools.ens-lyon.fr/.

Recent activity concerns the development, by Christophe Alias, of HLS compiler parts to be transferred to the XTREMLOGIC start-up (Zettice project) (see Section 7.2). An important effort of applied research and software development [12] has been achieved, resulting in the Dcc (DPN C Compiler) tool, outlined in Section 5.5 . Also, optimization developments (scalability, memory leaks, parallelization, etc) were performed on the PoCo compiler framework (see Section 5.6) and the Bee tool (see Section 5.7).

Also, several successive developments have been made related to termination tools. Our first implementation, RanK (see Section 5.9), was extended by other tools such as SToP (see Section 5.12) and, more recently Termite, (see Section 5.13).

5.2. Pip

Participants: Cédric Bastoul [professor, Strasbourg University and Inria/CAMUS], Paul Feautrier.

Paul Feautrier is the main developer of Pip (Parametric Integer Programming) since its inception in 1988. Basically, Pip is an "all integer" implementation of the Simplex, augmented for solving integer programming problems (the Gomory cuts method), which also accepts parameters in the non-homogeneous term. Pip is freely available under the GPL at http://www.piplib.org. It is widely used in the automatic parallelization community for testing dependences, scheduling, several kind of optimizations, code generation, and others. Beside being used in several parallelizing compilers, Pip has found applications in some unconnected domains, as for instance in the search for optimal polynomial approximations of elementary functions (see the Inria project Aric, previously known as Arénaire).

5.3. Cl@k

Participants: Fabrice Baray [Mentor Graphics, Former post-doc in Compsys], Alain Darte.

Cl@k (Critical LAttice Kernel) is a stand-alone optimization tool which computes or approximates the critical lattice for a given 0-symmetric polytope. (An admissible lattice is a lattice whose intersection with the polytope is reduced to 0; a critical lattice is an admissible lattice with minimal determinant). This tool is useful for the automatic derivation of array mappings that enable memory reuse, based on the notions of admissible lattice and of modular allocation (linear mapping plus modulo operations). It has been developed in 2005-2006 by Fabrice Baray, former post-doc Inria under Alain Darte's supervision.

Its application to array contraction has been implemented by Christophe Alias in a tool called Bee (see Section 5.7). More information is available at http://compsys-tools.ens-lyon.fr/clak/. The Cl@k tool is unfortunately outdated today (it is hard, if not impossible, to recompile it) and would need to be re-implemented. An extension of its underlying theory is also in progress.

5.4. Syntol

Participant: Paul Feautrier.

Syntol is a modular process network scheduler. The source language is C augmented with specific constructs for representing communicating regular process (CRP) systems. The present version features a syntax analyzer, a semantic analyzer to identify DO loops in C code, a dependence computer, a modular scheduler, and interfaces for CLooG (loop generator developed by C. Bastoul) and Cl@k (see Sections 5.3 and 5.7). The dependence computer now handles casts, records (`structures`), and the modulo operator in subscripts and conditional expressions. The latest developments are, firstly, a new code generator, and secondly, several experimental tools for the construction of bounded parallelism programs.

- The new code generator, based on the ideas of Boulet and Feautrier [20], generates a counter automaton that can be presented as a C program, as a rudimentary VHDL program at the RTL level, as an automaton in the Aspic input format, or as a drawing specification for the DOT tool.
- Hardware synthesis can only be applied to bounded parallelism programs. Our present aim is to construct threads with the objective of minimizing communications and simplifying synchronization. The distribution of operations among threads is specified using a placement function, which is found using techniques of linear algebra and combinatorial optimization.

5.5. Dcc

Participants: Christophe Alias, Alexandru Plesco [XtremLogic].

Dcc (DPN C Compiler) compiles a C program annotated with pragmas to a data-aware process network (DPN), a regular process network close to a circuit description that makes explicit the I/O transfers and the synchronizations. Dcc features throughput optimization, communication vectorization, and automatic parallelization.

Dcc is registered at the APP ("agence de protection des programmes") and has been transferred to the XTREMLOGIC start-up under an Inria licence. It uses a patented technology [12] and serves as a *front-end* for the HLS suite of the XTREMLOGIC start-up. Dcc has been implemented by Christophe Alias, using the PoCo compiler infrastructure (Section 5.6) and the Bee tool (Section 5.7). It represents more than 3000 lines of C++ code.

5.6. PoCo

Participant: Christophe Alias.

PoCo is a polyhedral compilation framework providing many features to prototype program analysis and optimizations in the polyhedral model:

- C parsing and extraction of the polyhedral representation.
- Symbolic layer on the state-of-art polyhedral libraries Polylib (set operations on polyhedra) and Piplib (parameterized ILP, see Section 5.2).
- Dependence analysis (PRDG, array dataflow analysis), array region analysis, array liveness analysis.
- C and VHDL code generation.

PoCo is registered at the APP ("agence de protection des programmes") and has been transferred to the XTREMLOGIC start-up under an Inria licence. The Bee, Chuba, and RanK tools presented thereafter make an extensive use of PoCo abstractions. PoCo has been developed by Christophe Alias. It represents more than 19000 lines of C++ code.

5.7. Bee

Participants: Christophe Alias, Alain Darte.

Bee is a source-to-source optimizer that resizes and reallocates optimally the arrays used by a program under scheduling constraints. Bee applies a fine-grain lifetime analysis for arrays. Then, the mathematical optimization of the Cl@k tool (Section 5.3) finds the array allocation (expressed as an affine lattice). Finally, Bee derives the actual array allocation and generates the C code accordingly. Bee was – to our knowledge – the first complete (i.e., with an element-wise lifetime analyzer integrated with an allocator) array contraction tool. Bee allows to allocate and to size the channels in process networks, providing a global affine schedule. This feature is fundamental in HLS (see Section 3.1.2) and more generally in automatic parallelization where communication buffers must be allocated and sized. An online demonstrator is available at http://compsys-tools.ens-lyon.fr/bee/index.html.

Bee is registered at the APP ("agence de protection des programmes") and has been transferred to the XTREMLOGIC start-up under an Inria licence. It is also used as an external tool by the compiler Gecos developed in the Cairn team at Irisa. Bee has been implemented by Christophe Alias, using the PoCo compiler infrastructure (see Section 5.6). It represents more than 2400 lines of C++ code.

5.8. Chuba

Participants: Christophe Alias, Alain Darte, Alexandru Plesco [Compsys/Zettice].

Chuba is a source-level optimizer that improves a C program in the context of the high-level synthesis (HLS) of hardware. Chuba is an implementation of the work described in the PhD thesis of Alexandru Plesco. The optimized program specifies a system of multiple communicating accelerators, which optimizes the data transfers with the external DDR memory. The program is divided into blocks of computations obtained thanks to tiling techniques, and, in each block, data are fetched by block to reduce the penalty due to line changes in the DDR accesses. Four accelerators achieve data transfers in a macro-pipeline fashion so that data transfers and computations (performed by a fifth accelerator) are overlapped.

So far, the back-end of Chuba is specific to the HLS tool C2H but the analysis is quite general and adapting Chuba to other HLS tools should be possible. Besides, it is interesting to mention that the program analysis and optimizations implemented in Chuba address a problem that is also very relevant in the context of GPGPUs. The underlying theory and corresponding experiments are described in [17].

Chuba has been implemented by Christophe Alias, using the PoCo compiler infrastructure (see Section 5.6). It represents more than 900 lines of C++ code.

5.9. RanK

Participants: Christophe Alias, Alain Darte, Paul Feautrier, Laure Gonnord [Compsys].

RanK is a software tool that can prove the termination of a program (in some cases) by computing a *ranking function*, i.e., a mapping from the operations of the program to a well-founded set that *decreases* as the computation advances. In case of success, RanK can also provide an upper bound of the worst-case time complexity of the program as a symbolic affine expression involving the input variables of the program (parameters), when it exists. In case of failure, RanK tries to prove the non-termination of the program and then to exhibit a counter-example input. This last feature is of great help for program understanding and debugging. The theory underlying RanK was presented at SAS'10 [15].

The input of RanK is an integer automaton, computed by C2fsm (see Section 5.11), representing the control structure of the program to be analyzed. RanK uses the Aspic tool (see Section 5.10), developed by Laure Gonnord during her PhD thesis, to compute automaton invariants. RanK has been used to discover successfully the worst-case time complexity of many benchmarks kernels of the community (see the WTC benchmark suite at http://compsys-tools.ens-lyon.fr/wtc/index.html). It uses the libraries Piplib (Section 5.2) and Polylib.

RanK has been implemented by Christophe Alias, using the compiler infrastructure PoCo (Section 5.6). It represents more than 3000 lines of C++. The tool has been presented at the CSTVA'13 workshop [16]. An online demonstrator is available at http://compsys-tools.ens-lyon.fr/rank.

5.10. Aspic

Participant: Laure Gonnord.

Aspic is an invariant generator for general counter automata. Used with C2fsm (see Section 5.11), it can be used to derivate invariants for numerical C programs, and also to prove safety. It is also part of the WTC toolsuite (see http://compsys-tools.ens-lyon.fr/wtc/index.html), a set of examples to demonstrate the capability of the RanK tool (see Section 5.9) for evaluating worse-case time complexity (number of transitions when executing an automaton).

Aspic implements the theoretical results of Laure Gonnord's PhD thesis on acceleration techniques and has been maintained since 2007.

5.11. C2fsm

Participant: Paul Feautrier.

C2fsm is a general tool that converts an arbitrary C program into a counter automaton. This tool reuses the parser and pre-processor of Syntol (see Section 5.4), which has been extended to handle `while` and `do while` loops, `goto`, `break`, and `continue` statements. C2fsm reuses also part of the code generator of Syntol and has several output formats, including FAST (the input format of Aspic, see Section 5.10), a rudimentary VHDL generator, and a DOT generator which draws the output automaton. In contrast to the FAST format, an ad hoc format, FLOW, uses a relational representation and retains non-affine constructs. C2fsm is also able to do elementary transformations on the automaton, such as eliminating useless states, transitions and variables, simplifying guards, or selecting cut-points, i.e., program points on loops that can be used by RanK (see Section 5.9) to prove program termination.

5.12. SToP

Participants: Christophe Alias, Guillaume Andrieu [University of Lille], Laure Gonnord [Compsys].

SToP (Scalable Termination of Programs) is the implementation of the modular termination technique presented at the TAPAS'12 workshop [18]. It takes as input a large irregular C program and conservatively checks its termination. To do so, SToP generates a set of small programs whose termination implies the termination of the whole input program. Then, the termination of each small program is checked thanks to RanK (see Section 5.9). In case of success, SToP infers a ranking (schedule) for the whole program. This schedule can be used in a subsequent analysis to optimize the program.

SToP represents more than 2000 lines of C++. The first results are available at http://compsys-tools.ens-lyon.fr/stop.

5.13. Termite

Participants: Laure Gonnord, Gabriel Radanne [ENS Rennes], David Monniaux [CNRS/VERIMAG].

Termite (Termination of C programs) is the implementation of our new algorithm "Counter-example based generation of ranking functions" (see Section 6.4). Based on LLVM and Pagai (a tool that generates invariants), the tool automatically generates a ranking function for each *head of loop*. Its implementation is under consolidation, it will be publicly available soon.

Termite represents 3000 lignes of OCaml.

5.14. Simplifiers

Participant: Paul Feautrier.

The aim of the `simple` library is to simplify Boolean formulas on affine inequalities. It works by detecting redundant inequalities in the representation of the subject formula as an ordered binary decision diagram (OBDD), see details in [23]. It uses PIP (see Section 5.2) for testing the feasibility – or unfeasibility – of a conjunction of affine inequalities.

The library is written in Java and is presented as a collection of class files. For experimentation, several front-ends have been written. They differ mainly in their input syntax, among which are a C like syntax, the Mathematica and SMTLib syntaxes, and an ad hoc Quast (quasi-affine syntax tree) syntax. See the first results at http://compsys-tools.ens-lyon.fr/stop.

DREAMPAL Team

4. New Software and Platforms

4.1. New Software and Platforms

Download page : https://gforge.inria.fr/frs/?group_id=3646

4.1.1. HoMade

HoMade is a softcore processor that we have started developing in 2012. The current version is reflective (i.e., the program it executes is self-modifiable), and statically configurable; dynamically reconfigurable multi-processors are the next steps. Users have to add to it the functionality they need in their applications via IPs. We have also being developing a library of IPs for the most common processor functions (ALU, registers, ...). All the design is in VHDL except for some schematic specifications.

The V5 version of HoMade has been developed in the Spring 2014. It has been used by ∼140 4th-year computer science students at Univ. Lille enrolled in the hardware architecture course (https://sites.google.com/site/tpm1aev/home). The new features of V5 are listed in Section 5.2 .

4.1.2. JHomade

JHomade is a software suite written in JAVA, including compilers and tools for the HoMade processor. It allows us to compile HiHope programs to Homade machine code and load the resulting binaries on FPGA boards. It was first released in 2013. The second version in 2014 includes several new features, like a C-frontend, a binary decoder and a code-generator for VHDL simulation. New features of the HiHope language are described in more detail in Section 5.3 .

4.1.3. Kcheck

Kcheck is a tool for the symbolic execution of programs in arbitrary languages defined in the $\mathbb{K}$ framework (http://k-framework.org), such as C and Java as well as the languages HiHope and Homade machine-code languages developed in out team. It also allow users to formally verify programs against specifications written in Reachability Logic, a specification formalism that can be seen as a language-independent Hoare logic. More information about the theory underlying Kcheck is given in Section 5.5 .

In 2014 we have developed a new and improved version of our tool, in order to keep up with the new modular infrastructure of the $\mathbb{K}$ framework. An online interface has been developed and is available at https://fmse.info.uaic.ro/tools/kcheck/. We have also started (since Nov. 2014) a development in the Coq proof assistant in order to obtained certificates for the program verifications performed by our tool.

GCG Team

5. New Software and Platforms

5.1. Givy

Givy is a runtime currently developed as part of the phd thesis of François Gindraud. It is designed for architectures with distributed memories, with the Kalray MPPA as the main target. It will execute dynamic data-flow task graphs, annotated with memory dependencies. It will automatically handle scheduling and placement of tasks (using the memory dependency hints), and generate memory transfers between distributed memory nodes when needed by using a software cache coherence protocol. Most of the work this year was done on implementing and testing a memory allocator with specific properties that is a building block of the whole runtime. This memory allocator is also tuned to work on the MPPA and its constraints, turning with very little memory and being efficient in the context of multithreaded calls.

5.2. Tirex

The Tirex Intermediate Representation has previously been generated from within both the Path64 and GCC compilers. In order to increase the usability of Tirex and to decrease the amount of required code maintenance that is induced by compiler evolutions a Tirex-generator has been written that is capable of creating the Tirex representation of a program based on its corresponding assembler code.

5.3. LLVM plugins

Work has been started on multiple plugins for the LLVM compiler framework that implement the code optimisations that have been elaborated by the team. While being work in progress this already provides us with crucial information for program analysis such as data-dependencies.

PAREO Project-Team

5. New Software and Platforms

5.1. ATerm

Participant: Pierre-Etienne Moreau [correspondant].

ATerm (short for Annotated Term) is an abstract data type designed for the exchange of tree-like data structures between distributed applications.

The ATerm library forms a comprehensive procedural interface which enables creation and manipulation of ATerms in C and Java. The ATerm implementation is based on maximal subterm sharing and automatic garbage collection.

We are involved (with the CWI) in the implementation of the Java version, as well as in the garbage collector of the C version. The Java version of the ATerm library is used in particular by *Tom*.

The ATerm library is documented, maintained, and available at the following address: http://www.meta-environment.org/Meta-Environment/ATerms.

5.2. Tom

Participants: Jean-Christophe Bach, Christophe Calvès, Horatiu Cirstea, Pierre-Etienne Moreau [correspondant].

Since 2002, we have developed a new system called *Tom* [27], presented in [11], [12]. This system consists of a pattern matching compiler which is particularly well-suited for programming various transformations on trees/terms and XML documents. Its design follows our experiments on the efficient compilation of rule-based systems [24]. The main originality of this system is to be language and data-structure independent. This means that the *Tom* technology can be used in a C, C++ or Java environment. The tool can be seen as a Yacc-like compiler translating patterns into executable pattern matching automata. Similarly to Yacc, when a match is found, the corresponding semantic action (a sequence of instructions written in the chosen underlying language) is triggered and executed. *Tom* supports sophisticated matching theories such as associative matching with neutral element (also known as list-matching). This kind of matching theory is particularly well-suited to perform list or XML based transformations for example.

In addition to the notion of *rule*, *Tom* offers a sophisticated way of controlling their application: a strategy language. Based on a clear semantics, this language allows to define classical traversal strategies such as *innermost*, *outermost*, *etc.* Moreover, *Tom* provides an extension of pattern matching, called *anti-pattern matching*. This corresponds to a natural way to specify *complements* (*i.e.*, what should not be there to fire a rule). *Tom* also supports the definition of cyclic graph data-structures, as well as matching algorithms and rewriting rules for term-graphs.

Tom is documented, maintained, and available at http://tom.loria.fr as well as at http://gforge.inria.fr/projects/tom.

POSTALE Team

4. New Software and Platforms

4.1. New Software

4.1.1. MyNRC: image-oriented library for allocation and manipulation of SIMD 1D, 2D and 3D structures

Participant: Lionel Lacassagne.

MyNRC is multi-plateform library that can handle SSE, AVX, Neon and ST VECx registers.

4.1.2. CovTrack: agile realtime multi-target tracking algorithm

Participants: Michèle Gouiffès, Lionel Lacassagne, Florence Laguzet, Andrés Romero.

4.1.3. tmLQCD for Blue Gene/Q

Participant: Michael Kruse [correspondant].

tmLQCD is a program suite for lattice quantum chromodynamics (Lattice QCD) using the chirally twisted Wilson quarks to reduce discretization artifacts. This software is in productive use by the European Twisted Mass Collaboration (ETMC).

As to not waste precious computation time on the supercomputers it is running on, it is important to optimize the code in order to run as fast as possible. tmLQCD has already been customized for Intel Xeon processors, the Blue Gene/L and Blue Gene/P from IBM. For the latter's successor, the Blue Gene/Q, more profound changes to the program are necessary. With these changes, tmLQCD reaches a peak performance of up to 55% of the machines theoretical floating point performance.

The Blue Gene/Q optimized tmLQCD is available at: http://github.com/Meinersbur/tmLQCD

4.1.4. Molly

Participant: Michael Kruse [correspondant].

Using Polly extension, the LLVM compiler framework is able to automatically parallelize general programs for shared memory threading for by exploiting the powerful analysis and transformations of the polyhedral model.

Molly adds the ability to manage distributed memory using the polyhedral model and is therefore able to automatically parallelize even for the largest of today's supercomputer. Once the distribution of data between the computer's nodes is known, Molly determines the values that are required to be transferred between the nodes and chunks them into as few messages as possible. It also keeps tracks of the buffers required by the MPI interface. Transfers are asynchronous such that further computations take place while the data is being transferred.

Molly has not yet been officially released.

4.1.5. Dohko (http://dohko.io/)

Participant: Alessandro Ferreira Leite [correspondant].

Automating multi-cloud configuration is a difficult task. The difficulties are mostly due to clouds' heterogeneity and the lack of tools to coordinate cloud computing configurations automatically. As a result, virtual machine image (VMI) is the common approach to configure cloud environment. Although VMI can handle functional properties like minimum disk size, operating system, and software packages, it leads to a high number of configuration options, increasing the difficulty to select one that matches users' requirements. Moreover, the usage of VMI usually results in vendor lock-in. Furthermore, VMI leaves for the users the work of selecting a resource to deploy the image and for orchestrating them accordingly, i.e., the work of selecting and instantiating the VMI in each cloud. In addition, VMI migration across multiple clouds normally has a high cost due to network traffic. Finally, in case of cloud's failure, it may be difficult for users to re-create the failed environment in another cloud, since the image will be inaccessible.

Therefore, to overcome these issues, we developed a configuration management tool for cloud computing. Our tool, called Dohko, allows the users to configure a multi-cloud computing environment, following a declarative strategy. In this case, the users describe their applications and requirements and use our tool to select the resources and to set up the whole computing environment automatically, taking into account temporal and functional dependencies between the resources. Moreover, following a software product line (SPL) engineering method, Dohko captures the knowledge of configuring cloud environments in form of reusable assets. In this case, a product is a cloud computing environment that meets the user requirements, where the requirements can be either based on high or low-level descriptions. Examples of low-level descriptions include: virtualization type, disk technology, sustainable performance, among others, whereas high-level descriptions include the number of CPU cores, the RAM memory size, and the maximum monetary cost per hour. In this context, a cloud computing environment also matches cloud's configuration constraints. Besides that, thanks to the usage of an extended feature model (EFM) with attributes, our approach enables the description of the whole computing environment (i.e., hardware and software) independent of cloud provider and without requiring the usage of virtual machine image. In this case, it relies on an off-the-shelf constraint satisfaction problem (CSP) solver to implement the feature model and to select the resources.

By employing a declarative strategy, Dohko could execute a biological sequence comparison application in two distinct cloud providers (i.e., Amazon EC2 and Google Compute Engine) considering a single and a multi-cloud scenario, demanding minimal users' intervention to instantiate the whole cloud environment, as well as to execute the application. In particular, our solution tackles the lack of middleware prototypes that can support different scenarios when using services from many clouds. Moreover, it meets the functional requirements identified for multiple cloud-unaware systems [136] such as: (a) it provides a way to describe functional and non-functional requirements through the usage of an SPL engineering method; (b) it can aggregate services from distinct clouds; (c) it provides a homogeneous interface to access services of multiple clouds; (d) it allows the service selection of the clouds; (e) it can deploy its components across many clouds; (f) it provides automatic procedures for deployments; (g) it utilizes an overlay network to connect and to organize the resources; (h) it does not impose any constraint for the connected clouds.

4.2. Platforms

4.2.1. Fast linear system solvers in public domain libraries (http://icl.cs.utk.edu/magma/)

Participant: Marc Baboulin [correspondant].

Hybrid multicore+GPU architectures are becoming commonly used systems in high performance computing simulations. In this research, we develop linear algebra solvers where we split the computation over multicore and graphics processors, and use particular techniques to reduce the amount of pivoting and communication between the hybrid components. This results in efficient algorithms that take advantage of each computational unit [12]. Our research in randomized algorithms yields to several contributions to propose public domain libraries PLASMA and MAGMA in the area of fast linear system solvers for general and symmetric indefinite systems. These solvers minimize communication by removing the overhead due to pivoting in LU and $LDLT$ factorization. Different approaches to reduce communication are compared in [2].

See also the web page http://icl.cs.utk.edu/magma/.

4.2.2. cTuning Framework (http://cTuning.org): Repository and Tools for Collective Characterization and Optimization of Computing Systems

Participant: Grigori Fursin [correspondant].

Designing, porting and optimizing applications for rapidly evolving computing systems is often complex, ad-hoc, repetitive, costly and error prone process due to an enormous number of available design and optimization choices combined with the complex interactions between all components. We attempt to solve this fundamental problem based on collective participation of users combined with empirical tuning and machine learning.

We developed cTuning framework that allows to continuously collect various knowledge about application characterization and optimization in the public repository at cTuning.org. With continuously increasing and systematized knowledge about behavior of computer systems, users should be able to obtain scientifically motivated advices about anomalies in the behavior of their applications and possible solutions to effectively balance performance and power consumption or other important characteristics.

Currently, we use cTuning repository to analyze and learn profitable optimizations for various programs, datasets and architectures using machine learning enabled compiler (MILEPOST GCC). Using collected knowledge, we can quickly suggest better optimizations for a previously unseen programs based on their semantic or dynamic features [10].

We believe that such approach will be vital for developing efficient Exascale computing systems. We are currently developing the new extensible cTuning2 framework for automatic performance and power tuning of HPC applications.

For more information, see the web page http://cTuning.org.

4.2.3. NT2 (http://www.github.com/MetaScale/nt2)

Participants: Pierre Esterie, Joël Falcou, Mathias Gaunard, Ian Masliah, Antoine Tran Tan.

NT2 is a C++ high level framework for scientific computing.[18]

4.2.4. Boost.SIMD (http://www.github.com/MetaScale/nt2)

Participants: Pierre Esterie, Joël Falcou, Mathias Gaunard.

Boost.SIMD provides a portable way to vectorize computation on Altivec, SSE or AVX while providing a generic way to extend the set of supported functions and hardwares.

TASC Project-Team

5. New Software and Platforms

5.1. Platforms

5.1.1. CHOCO

Participants: Nicolas Beldiceanu, Jean-Guillaume Fages, Xavier Lorca [correspondant], Thierry Petit, Charles Prud'Homme [main developer], Rémi Douence.

CHOCO is a Java discrete constraints library integrating within a same system *explanations*, *soft constraints* and *global constraints* (90000 lines of source code). In 2014 developments were focusing on the following aspects:

- For second consecutive year, CHOCO has participated at the MiniZinc Challenge, an annual competition of constraint programming solvers. In competition with 16 other solvers, CHOCO has won three bronze medals in three out of four categories (Free search, Parallel search and Open class).
- Five versions have been released all year long, the last one (v3.3.0, Dec. 17th) has the particularity to be promoted on Maven Central Repository. The major modifications were related to a simplification of the API but also improvement of the overall solver.
- A User Guide is now available: 164 pages describing how to use CHOCO, together with a new website.
- Finally, Charles Prud'homme and Jean-Guillaume Fages, the main contributors of CHOCO, have defended their Phd, publishing at the same time their work in the source code. In particular, an extension of CHOCO now provides support for constraints involving graph variables.

5.1.2. IBEX

Participants: Ignacio Araya, Clément Carbonnel, Gilles Chabert [correspondant], Benoit Desrochers, Luc Jaulin, Bertrand Neveu, Jordan Ninin, Ignacio Salas Donoso, Gilles Trombettoni.

IBEX (Interval-Based EXplorer) is a C++ library for solving nonlinear constraints over real numbers. The main feature of Ibex is its ability to build solver/paver strategies declaratively through the contractor programming paradigm. It also comes with a black-box solver and a global optimizer.

In 2014 the work on IBEX has focused on the following points.

- Global optimizer:
  - Rigorous mode in the global optimizer (certification of the feasibility of strict equality constraints for the minimum found). This includes Newton-based inflation iteration, Hansen test for underconstrained systems (see Global Optimization using Interval Analysis, E. Hansen, 1992).
  - Unconstrained local search algorithm (quasi-Newton method with trust regions).
  - Rejection test based on first-order conditions (see First Order Rejection Tests For Multiple-Objective Optimization, A. Goldsztejn et al. [42]).
  - Multiple selection technique in exploration (see A new multisection technique in interval methods for global optimization, L.G. Casado, Computing, 2000)
- Contractors:
  - Existentially-quantified constraints, (see Contractor Programming, [8]).
  - Mohc contractor, (see Exploiting Monotonicity in Interval Constraint Propagation, I. Araya et al., [41]).

  - Q-intersection, (see Q-intersection Algorithms for Constraint-Based Robust Parameter Estimation, C. Carbonnel et al., AAAI 2014, [27]).
  - Contractor based on pixel maps (started in Oct 2014, still in progress, see Using set membership methods for robust underwater robot localization, PhD, J. Sliwka).
- Miscellanous
  - Everyday code improvement (around 400 commits in 2014).
  - Symbolic processing features (symbol occurrence splitting, function construction from strings, progress in differentiation with vector/matrix operations).
  - numerous bug fixes (especially in the inner arithmetic routines).

5.1.3. Global Constraint Catalog

Participants: Nicolas Beldiceanu [correspondant], Mats Carlsson, Sophie Demassey, Helmut Simonis.

The global constraint catalog presents and classifies global constraints and describes different aspects with meta data. It consist of

1. a pdf version that can be downloaded from http://sofdem.github.io/gccat/ (at item *working version*) containing 431 constraints, 4070 pages and 1000 figures,
2. an on line version accessible from the previous address,
3. meta data describing the constraints (buton *PL* for each constraint, e.g., alldifferent.pl),
4. an online service (i.e, a *constraint seeker*) which provides a web interface to search for global constraints, given positive and negative ground examples.

This year developments were focusing on:

1. maintaining the content of the catalogue,
2. making more easy the navigation within the pdf version,
3. continuing the redesign of the figures using TikZ: 200 figures were converted and 100 figures remain to be converted, and adding new illustrations (150 figures).
4. updating the web version of the catalogue (see http://sofdem.github.io/gccat/).

5.1.4. AIUR

Participant: Florian Richoux [correspondant].

AIUR (Artificial Intelligence Using Randomness) is an AI for $StarCraft: BroodWar^{tm}$.

The main idea is to be unpredictable by making some stochastic choices. The AI starts a game with a "mood" randomly picked up among 5 moods, dictating some behaviors (aggressive, fast expand, macro-game, ...). In addition, some other choices (productions, timing attacks, early aggressions, ...) are also taken under random conditions.

Learning is an essential part of AIUR. For this, it uses persistent I/O files system to record which moods are efficient against a given opponent, in order to modify the probability distribution for the mood selection. The current system allows both on-line and off-line learning.

AIUR is an open source program under GNU GPL v3 licence, written in C++ (18.000 lines of code). Source and documentations are available at github.com/AIUR-group/AIUR. AIUR finished 4^{th} to $StarCraft^{tm}$ AI competitions organized at the conferences AIIDE 2014 and CIG 2014.

5.1.5. GHOST

Participant: Florian Richoux [correspondant].

GHOST (General meta-Heuristic Optimization Solving Tool) is a template C++11 library designed for $StarCraft: BroodWar^{tm}$, under the terms of the GNU GPL v3 licence and is about 7500 lines long. GHOST implements a meta-heuristic solver aiming to solve any kind of combinatorial and optimization RTS-related problems represented by a CSP/COP [36]. The solver handles dedicated geometric and assignment constraints in a way that is compatible with very strong real time requirements. The source code as well as documentation pages are available at github.com/richoux/GHOST.

This framework is a deep extension of an ad-hoc solver. Although GHOST has been developed recently (during Summer 2014), it got itself quickly noticed by a French video-game developing company. We are starting discussion about a technology transfer of GHOST.

AOSTE Project-Team

5. New Software and Platforms

5.1. TimeSquare

Participants: Nicolas Chleq, Julien Deantoni, Frédéric Mallet [correspondant].

TimeSquare is a software environment for the modeling and analysis of timing constraints in embedded systems. It relies specifically on the Time Model of the MARTE UML profile (see section 3.2), and more accurately on the associated Clock Constraint Specification Language (CCSL) for the expression of timing constraints.

TimeSquare offers five main functionalities:

1. graphical and/or textual interactive specification of logical clocks and relative constraints between them;
2. definition and handling of user-defined clock constraint libraries;
3. automated simulation of concurrent behavior traces respecting such constraints, using a Boolean solver for consistent trace extraction;
4. call-back mechanisms for the traceability of results (animation of models, display and interaction with waveform representations, generation of sequence diagrams...).
5. compilation to pure java code to enable embedding in non eclipse applications or to be integrated as a time and concurrency solver within an existing tool.

In practice TimeSquare is a set of plug-ins developed for the Eclipse modeling framework. The software is registered by the *Agence pour la Protection des Programmes*, under number IDDN.FR.001.170007.000.S.P.2009.001.10600. It can be downloaded from the site http://timesquare.inria.fr/. It has been integrated in the OpenEmbeDD ANR RNTL platform and recently in the Gemoc Studio.

5.2. K-Passa

Participants: Jean-Vivien Millo [correspondant], Robert de Simone.

This software is dedicated to the simulation, analysis, and static scheduling of Event/Marked Graphs, SDF and KRG extensions. A graphical interface allows to edit the Process Networks and their time annotations (*latency,* ...). Symbolic simulation and graph-theoretic analysis methods allow to compute and optimize static schedules, with best throughputs and minimal buffer sizes. In the case of KRG the (ultimately k-periodic) routing patterns can also be provided and transformed for optimal combination of switching and scheduling when channels are shared. KPASSA also allows for import/export of specific description formats such as UML-MARTE, to and from our other TimeSquare tool.

The tool was originally developed mainly as support for experimentations following our research results on the topic of Latency-Insensitive Design. This research was conducted and funded in part in the context of the CIM PACA initiative, with initial support from ST Microelectronics and Texas Instruments.

KPASSA is registered by the Agence pour la Protection des Programmes, under the number IDDN.FR.001.310003.000.S.P.2009.000.20700. It can be downloaded from the site http://www-sop.inria.fr/aoste/index.php?page=software/kpassa.

5.3. SynDEx

Participants: Yves Sorel [correspondant], Meriem Zidouni.

SynDEx is a system level CAD software implementing the AAA methodology for rapid prototyping and for optimizing distributed real-time embedded applications. Developed in OCaML it can be downloaded free of charge, under Inria copyright, from the general SynDEx site http://www.syndex.org.

The AAA methodology is described in section 3.3 . Accordingly, SYNDEX explores the space of possible allocations (spatial distribution and temporal scheduling), from application elements to architecture resources and services, in order to match real-time requirements; it does so by using schedulability analyses and heuristic techniques. Ultimately it generates automatically distributed real-time code running on real embedded platforms. The last major release of SYNDEX (V7) allows the specification of multi-periodic applications.

Application algorithms can be edited graphically as directed acyclic task graphs (DAG) where each edge represents a data dependence between tasks, or they may be obtained by translations from several formalisms such as Scicos (http://www.scicos.org), Signal/Polychrony (http://www.irisa.fr/espresso/Polychrony/download.php), or UML2/MARTE models (http://www.omg.org/technology/documents/profile_catalog.htm).

Architectures are represented as graphical block diagrams composed of programmable (processors) and non-programmable (ASIC, FPGA) computing components, interconnected by communication media (shared memories, links and busses for message passing). In order to deal with heterogeneous architectures it may feature several components of the same kind but with different characteristics.

Two types of non-functional properties can be specified for each task of the algorithm graph. First, a period that does not depend on the hardware architecture. Second, real-time features that depend on the different types of hardware components, ranging amongst *execution and data transfer time, memory, etc.*. Requirements are generally constraints on deadline equal to period, latency between any pair of tasks in the algorithm graph, dependence between tasks, etc.

Exploration of alternative allocations of the algorithm onto the architecture may be performed manually and/or automatically. The latter is achieved by performing real-time multiprocessor schedulability analyses and optimization heuristics based on the minimization of temporal or resource criteria. For example while satisfying deadline and latency constraints they can minimize the total execution time (makespan) of the application onto the given architecture, as well as the amount of memory. The results of each exploration is visualized as timing diagrams simulating the distributed real-time implementation.

Finally, real-time distributed embedded code can be automatically generated for dedicated distributed real-time executives, possibly calling services of resident real-time operating systems such as Linux/RTAI or Osek for instance. These executives are deadlock-free, based on off-line scheduling policies. Dedicated executives induce minimal overhead, and are built from processor-dependent executive kernels. To this date, executives kernels are provided for: TMS320C40, PIC18F2680, i80386, MC68332, MPC555, i80C196 and Unix/Linux workstations. Executive kernels for other processors can be achieved at reasonable cost following these examples as patterns.

5.4. Lopht

Participants: Thomas Carle, Manel Djemal, Dumitru Potop Butucaru [correspondant].

The Lopht (Logical to Physical Time Compiler) has been designed as an implementation of the AAA methodology. Like SynDEx, Lopht relies on off-line allocation and scheduling techniques to allow real-time implementation of dataflow synchronous specifications onto multiprocessor systems. But there are several originality points: a stronger focus on efficiency, which results in the use of a compilation-like approach, a focus on novel target architectures (many-core chips and time-triggered embedded systems), and the possibility to handle multiple, complex non-functional requirements covering real-time (release dates and deadlines possibly different from period, major time frame, end-to-end flow constraints), ARINC 653 partitioning, the possibility to preempt or not each task, and finally SynDEx-like allocation.

Improved efficiency is attained through the use of classical and novel data structures and optimization algorithms pertaining to 3 fields: synchronous language compilation, classical compiler theory, and real-time scheduling. A finer representation of execution conditions allows us to make a better use of double

resource reservation and thus improve latency and throughput. The use of software pipelining allows the improvement of computation throughput. The use of post-scheduling optimizations allows a reduction in the number of preemptions. The focus on novel architectures means that architecture descriptions need to define novel communication media such as the networks-on-chips (NoCs), and that real-time characteristics must include those specific to a time-triggered execution model, such as the Major Time Frame (MTF). Attaining efficiency alse requires a fine-grain description of more classical platform resources, such as the multi-bank RAMs, to allow efficient allocation during scheduling.

Significant contributions to the Lopht tool have been brought by T. Carle (the extensions concerning time-triggered platforms), M. Djemal (the extensions concerning many-core platforms), and Zhen Zhang under the supervision of D. Potop Butucaru. The tool has been used and extended during the PARSEC project. It is currently used in the direct collaboration with Airbus Defence and Space and the CNES, in the IRT SystemX/FSF project, and in the CAPACITES project. It has been developed in OCaml.

5.5. SAS

Participants: Daniel de Rauglaudre [correspondant], Yves Sorel.

The SAS (Simulation and Analysis of Scheduling) software allows the user to perform the schedulability analysis of periodic task systems in the monoprocessor case.

The main contribution of SAS, when compared to other commercial and academic softwares of the same kind, is that it takes into account the exact preemption cost between tasks during the schedulability analysis. Beside usual real-time constraints (precedence, strict periodicity, latency, etc.) and fixed-priority scheduling policies (Rate Monotonic, Deadline Monotonic, Audsley^{++}, User priorities), SAS additionaly allows to select dynamic scheduling policy algorithms such as Earliest Deadline First (EDF). The resulting schedule is displayed as a typical Gantt chart with a transient and a permanent phase, or as a disk shape called "dameid", which clearly highlights the idle slots of the processor in the permanent phase.

For a schedulable task system under EDF, when the exact preemption cost is considered, the period of the permanent phase may be much longer than the least commun multiple (LCM) of the periods of all tasks, as often found in traditional scheduling theory. Specific effort has been made to improve display in this case. The classical utilization factor, the permanent exact utilization factor, the preemption cost in the permanent phase, and the worst response time for each task are all displayed when the system is schedulable. Response times of each task relative time can also be displayed (separately).

SAS is written in OCaML, using CAMLP5 (syntactic preprocessor) and OLIBRT (a graphic toolkit under X). Both are written by Daniel de Rauglaudre. It can be downloaded from the site http://pauillac.inria.fr/~ddr/sas-dameid/.

CONVECS Project-Team

5. New Software and Platforms

5.1. The CADP Toolbox

Participants: Hubert Garavel [correspondent], Frédéric Lang, Radu Mateescu, Wendelin Serwe.

We maintain and enhance CADP (*Construction and Analysis of Distributed Processes* – formerly known as *CAESAR/ALDEBARAN Development Package*) [1], a toolbox for protocols and distributed systems engineering [0]. In this toolbox, we develop and maintain the following tools:

- CAESAR.ADT [41] is a compiler that translates LOTOS abstract data types into C types and C functions. The translation involves pattern-matching compiling techniques and automatic recognition of usual types (integers, enumerations, tuples, etc.), which are implemented optimally.
- CAESAR [47], [46] is a compiler that translates LOTOS processes into either C code (for rapid prototyping and testing purposes) or finite graphs (for verification purposes). The translation is done using several intermediate steps, among which the construction of a Petri net extended with typed variables, data handling features, and atomic transitions.
- OPEN/CAESAR [42] is a generic software environment for developing tools that explore graphs on the fly (for instance, simulation, verification, and test generation tools). Such tools can be developed independently of any particular high level language. In this respect, OPEN/CAESAR plays a central role in CADP by connecting language-oriented tools with model-oriented tools. OPEN/CAESAR consists of a set of 16 code libraries with their programming interfaces, such as:
  - CAESAR_GRAPH, which provides the programming interface for graph exploration,
  - CAESAR_HASH, which contains several hash functions,
  - CAESAR_SOLVE, which resolves Boolean equation systems on the fly,
  - CAESAR_STACK, which implements stacks for depth-first search exploration, and
  - CAESAR_TABLE, which handles tables of states, transitions, labels, etc.

  A number of on-the-fly analysis tools have been developed within the OPEN/CAESAR environment, among which:
  - BISIMULATOR, which checks bisimulation equivalences and preorders,
  - CUNCTATOR, which performs steady-state simulation of continuous-time Markov chains,
  - DETERMINATOR, which eliminates stochastic nondeterminism in normal, probabilistic, or stochastic systems,
  - DISTRIBUTOR, which generates the graph of reachable states using several machines,
  - EVALUATOR, which evaluates MCL formulas,
  - EXECUTOR, which performs random execution,
  - EXHIBITOR, which searches for execution sequences matching a given regular expression,
  - GENERATOR, which constructs the graph of reachable states,
  - PROJECTOR, which computes abstractions of communicating systems,
  - REDUCTOR, which constructs and minimizes the graph of reachable states modulo various equivalence relations,

[0] http://cadp.inria.fr

  - SIMULATOR, XSIMULATOR, and OCIS, which enable interactive simulation, and
  - TERMINATOR, which searches for deadlock states.

- BCG (*Binary Coded Graphs*) is both a file format for storing very large graphs on disk (using efficient compression techniques) and a software environment for handling this format. BCG also plays a key role in CADP as many tools rely on this format for their inputs/outputs. The BCG environment consists of various libraries with their programming interfaces, and of several tools, such as:
  - BCG_CMP, which compares two graphs,
  - BCG_DRAW, which builds a two-dimensional view of a graph,
  - BCG_EDIT, which allows the graph layout produced by BCG_DRAW to be modified interactively,
  - BCG_GRAPH, which generates various forms of practically useful graphs,
  - BCG_INFO, which displays various statistical information about a graph,
  - BCG_IO, which performs conversions between BCG and many other graph formats,
  - BCG_LABELS, which hides and/or renames (using regular expressions) the transition labels of a graph,
  - BCG_MIN, which minimizes a graph modulo strong or branching equivalences (and can also deal with probabilistic and stochastic systems),
  - BCG_STEADY, which performs steady-state numerical analysis of (extended) continuous-time Markov chains,
  - BCG_TRANSIENT, which performs transient numerical analysis of (extended) continuous-time Markov chains, and
  - XTL (*eXecutable Temporal Language*), which is a high level, functional language for programming exploration algorithms on BCG graphs. XTL provides primitives to handle states, transitions, labels, *successor* and *predecessor* functions, etc.

    For instance, one can define recursive functions on sets of states, which allow evaluation and diagnostic generation fixed point algorithms for usual temporal logics (such as HML [51], CTL [37], ACTL [39], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [45] and providing a unified access to a graph partitioned in fragments distributed over a set of remote machines, possibly located in different countries. The PBG format is supported by several tools, such as:
  - PBG_CP, PBG_MV, and PBG_RM, which facilitate standard operations (copying, moving, and removing) on PBG files, maintaining consistency during these operations,
  - PBG_MERGE (formerly known as BCG_MERGE), which transforms a distributed graph into a monolithic one represented in BCG format,
  - PBG_INFO, which displays various statistical information about a distributed graph.

- The connection between explicit models (such as BCG graphs) and implicit models (explored on the fly) is ensured by OPEN/CAESAR-compliant compilers, e.g.:
  - BCG_OPEN, for models represented as BCG graphs,
  - CAESAR.OPEN, for models expressed as LOTOS descriptions,
  - EXP.OPEN, for models expressed as communicating automata,
  - FSP.OPEN, for models expressed as FSP [55] descriptions,
  - LNT.OPEN, for models expressed as LNT descriptions, and
  - SEQ.OPEN, for models represented as sets of execution traces.

The CADP toolbox also includes TGV (*Test Generation based on Verification*), which has been developed by the VERIMAG laboratory (Grenoble) and the VERTECS project-team at Inria Rennes – Bretagne-Atlantique.

The CADP tools are well-integrated and can be accessed easily using either the EUCALYPTUS graphical interface or the SVL [43] scripting language. Both EUCALYPTUS and SVL provide users with an easy and uniform access to the CADP tools by performing file format conversions automatically whenever needed and by supplying appropriate command-line options as the tools are invoked.

5.2. The TRAIAN Compiler

Participants: Hubert Garavel [correspondent], Frédéric Lang, Wendelin Serwe.

We develop a compiler named TRAIAN for translating LOTOS NT descriptions into C programs, which will be used for simulation, rapid prototyping, verification, and testing.

The current version of TRAIAN, which handles LOTOS NT types and functions only, has useful applications in compiler construction [44], being used in all recent compilers developed by CONVECS.

The TRAIAN compiler can be freely downloaded from the CONVECS Web site [0].

5.3. The PIC2LNT Translator

Participants: Radu Mateescu, Gwen Salaün [correspondent].

We develop a translator named PIC2LNT from an applied π-calculus (see § 6.1) to LNT, which enables the analysis of concurrent value-passing mobile systems using CADP.

PIC2LNT is developed by using the SYNTAX tool (developed at Inria Paris-Rocquencourt) for lexical and syntactic analysis together with LOTOS NT for semantical aspects, in particular the definition, construction, and traversal of abstract trees.

The PIC2LNT translator can be freely downloaded from the CONVECS Web site [0].

5.4. The PMC Partial Model Checker

Participants: Radu Mateescu, Frédéric Lang.

We develop a tool named PMC (*Partial Model Checker*, see § 6.4), which performs the compositional model checking of dataless MCL formulas on networks of communicating automata described in the EXP language.

PMC can be freely downloaded from the CONVECS Web site [0].

[0] http://convecs.inria.fr/software/traian
[0] http://convecs.inria.fr/software/pic2lnt
[0] http://convecs.inria.fr/software/pmc

HYCOMES Team

5. New Software and Platforms

5.1. Mica: A Modal Interface Compositional Analysis Toolbox

Participant: Benoît Caillaud.

http://www.irisa.fr/s4/tools/mica/

Mica is an Ocaml library developed by Benoît Caillaud implementing the Modal Interface algebra published in [5], [4]. The purpose of Modal Interfaces is to provide a formal support to contract based design methods in the field of system engineering. Modal Interfaces enable compositional reasoning methods on I/O reactive systems.

In Mica, systems and interfaces are represented by extension. However, a careful design of the state and event heap enables the definition, composition and analysis of reasonably large systems and interfaces. The heap stores states and events in a hash table and ensures structural equality (there is no duplication). Therefore complex data-structures for states and events induce a very low overhead, as checking equality is done in constant time.

Thanks to the Inter module and the mica interactive environment, users can define complex systems and interfaces using Ocaml syntax. It is even possible to define parameterized components as Ocaml functions.

Mica is available as an open-source distribution, under the CeCILL-C Free Software License Agreement (http://www.cecill.info/licences/Licence_CeCILL-C_V1-en.html).

5.2. Flipflop and TnF-C++: Test and Flip Net Synthesis Tools for the Automated Synthesis of Surgical Procedure Models

Participant: Benoît Caillaud.

http://tinyurl.com/oql6f3y

Flipflop is a Test and Flip net synthesis tool implementing a linear algebraic polynomial time algorithm. Computations are done in the $\mathbb{Z}/2\mathbb{Z}$ ring. Test and Flip nets extend Elementary Net Systems by allowing test to zero, test to one and flip arcs. The effect of flip arcs is to complement the marking of the place. While the net synthesis problem has been proved to be NP hard for Elementary Net Systems, thanks to flip arcs, the synthesis of Test and Flip nets can be done in polynomial time. Test and flip nets have the required expressivity to give concise and accurate representations of surgical processes (models of types of surgical operations). Test and Flip nets can express causality and conflict relations. The tool takes as input either standard XES log files (a standard XML file format for process mining tools) or a specific XML file format for surgical applications. The output is a Test and Flip net, solution of the following synthesis problem: Given a finite input language (log file), compute a net, which language is the least language in the class of Test and Flip net languages, containing the input language.

TnF-C++ is a robust and portable re-implementation of Flipflop, developed in 2014 and integrated in the S3PM toolchain. Both software have been designed in the context of the S3PM project on surgical procedure modeling and simulation (see section 7.1).

MUTANT Project-Team

5. New Software and Platforms

5.1. Antescofo

Participants: Arshia Cont, Jean-Louis Giavitto, Florent Jacquemard, José Echeveste.

Antescofo is a modular polyphonic Score Following system as well as a Synchronous Programming language for musical composition. The module allows for automatic recognition of music score position and tempo from a realtime audio Stream coming from performer(s), making it possible to synchronize an instrumental performance with computer realized elements. The synchronous language within Antescofo allows flexible writing of time and interaction in computer music.

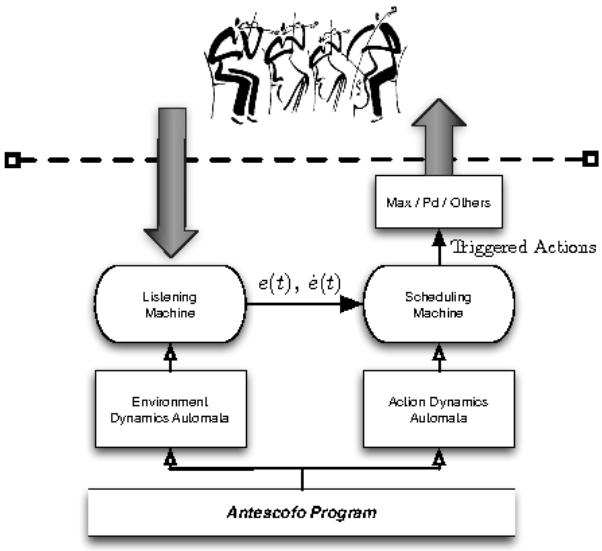

Figure 4. General scheme of Antescofo virtual machine

A complete new version of Antescofo has been released in November 2014 on Ircam Forumnet. This version includes major improvements over the previous version as a result of MuTant's research and development. Its development has benefited from of intensive interactions with composers, especially Julia Blondeau, José-Miguel Fernandez, and Marco Stroppa.

One major and sensible improvement is a total review of *Antescofo*'s realtime machine listening as a result of [12], which allows robust recognition in highly polyphonic and noisy environments with the help of novel notions of *Probabilistic Time Coherency*. This improvement allowed team members to participate in collaborative work with Paris Orchestra among others.

The 2014 release of *Antescofo* also includes new anticipative synchronization strategies. In the context of the PhD of José Echeveste, they are systematically studied with the help of the Orchestre de Paris and the composer Marco Stroppa. This work won the best presentation award at ICMC 2014.

The new internal architecture unifies the handling of external (musical) events and the handling of internal (logical) events in a framework able to manage multiple time frames (relative, absolute or computed). The notion of synchronization has been extended to be able to synchronize on the update of a variable in addition to the update of the listening machine. This mechanism offers new opportunities, especially in the context of open scores and improvised music [16].

The 2014 version targets the Max and PureData (Pd) environments on Mac, but also on Linux on Windows (Pd version). A standalone version is used to simulate a performance in Ascograph and to offer batch processing capabilities as well as a testing framework.

5.2. Ascograph: Antescofo Visual Editor

Participants: Thomas Coffy [ADT], Arshia Cont.

The Antescofo programming language can be extended to visual programing to better integrate existing scores and to allow users to construct complex and embedded temporal structures that are not easily integrated into text. This project is held since October 2012 thanks to Inria ADT Support.

AscoGraph, the Antescofo graphical score editor released in 2013, provides a autonomous Integrated Development Environment (IDE) for the authoring of Antescofo scores. Antescofo listening machine, when going forward in the score during recognition, uses the message passing paradigm to perform tasks such as automatic accompaniment, spatialization, etc. The Antescofo score is a text file containing notes (chord, notes, trills, ...) to follow, synchronization strategies on how to trigger actions, and electronic actions (the reactive language). This editor shares the same score parsing routines with Antescofo core, so the validity of the score is checked on saving while editing in AscoGraph, with proper parsing errors handling. Graphically, the application is divided in two parts (see Figure 2). On the left side, a graphical representation of the score, using a timeline with tracks view. On the right side, a text editor with syntax coloring of the score is displayed. Both views can be edited and are synchronized on saving. Special objects such as "curves", are graphically editable: they are used to provide high-level variable automation facilities like breakpoints functions (BPF) with more than 30 interpolations possible types between points, graphically editable.

An important feature of AscoGraph is the score import from MusicXML or MIDI files, which make the complete workflow of the composition of a musical piece much easier than before.

AscoGraph is strongly connected with Antescofo core object (using OSC over UDP): when a score is edited and modified it is automatically reloaded in Antescofo, and on the other hand, when Antescofo follows a score (during a concert or rehearsal) both graphical and textual view of the score will scroll and show the current position of Antescofo.

AscoGraph is released under Open-Source MIT license and has been released publicly along with new Antescofo architecture during IRCAM Forum 2013. Recent development was published in [11].

5.3. Antescofo Timed Test Platform

Participants: Florent Jacquemard, Clément Poncelet.

The frequent use of Antescofo in live and public performances with human musicians implies strong requirements of temporal reliability and robustness to unforeseen errors in input. To address these requirements and help the development of the system and authoring of pieces by users, we are developing a platform for the automation of testing the behavior of Antescofo on a given score, with of focus on timed behavior. It makes possible to automate the following main tasks:

1. (1) generation of relevant input data for testing, with the sake of exhaustiveness,
2. (2) computation of the corresponding expected output, according to a formal specification of the expected behavior of the system on a given mixed score,
3. (3) black-box execution of the input test data,
4. (4) comparison of expected and real output and production of a test verdict.

The input and output data are timed traces (sequences of events together with inter-event durations).

Our platform uses state of the art techniques and tools for *model-based testing* of embedded systems [31]. Some models of The environment of the system (the musicians) and the expected behavior of the system are both represented by formal models. We have developed a compiler for producing automatically such models, in an intermediate representation language (IR), from mixed scores. The IR are in turn converted into Timed Automata and passed, to the model-checker Uppaal.

Uppaal is used, with its extension Cover, for the above generation Task (1). Following some *coverage criteria*, this tools makes a systematic exploration of the state space of the model. We propose also an alternative approach for the generation of input traces by *fuzzing* of an ideal trace obtained from the score (a trace represented a perfectly timed performance of the score).

Task (2) is also performed by Uppaal, by simulation, using the model of the system and the generated test input.

Moreover, we have implemented several tools for Tasks (3) and (4), corresponding to different boundaries for the implementation under test (black box): e.g. the interpreter of Antescofo's synchronous language alone, or with tempo detection, or the whole system.

PARKAS Project-Team

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.

SPADES Team

5. New Software and Platforms

5.1. Prototypes

5.1.1. Logical Causality

Participant: Gregor Goessler.

We are developing LoCa, a prototype tool written in Scala that implements the analysis of logical causality described in 6.3.3 . LoCa currently supports causality analysis in BIP and networks of timed automata. The core analysis engine is implemented as an abstract class, such that support for other models of computation (MoC) can be added by instantiating the class with the basic operations of the MoC.

5.1.2. Cosyma

Participant: Gregor Goessler.

We have developed CoSyMA, a tool for automatic controller synthesis for incrementally stable switched systems based on multi-scale discrete abstractions. The tool accepts a description of a switched system represented by a set of differential equations and the sampling parameters used to define an approximation of the state-space on which discrete abstractions are computed. The tool generates a controller — if it exists — for the system that enforces a given safety or time-bounded reachability specification.

5.1.3. The SIAAM virtual machine

Participant: Jean-Bernard Stefani.

The SIAAM abstract machine is an object-based realization of the Actor model of concurrent computation. Actors can exchange arbitrary object graphs in messages while still enjoying a strong isolation property. It guarantees that each actor can only directly access objects in its own local heap, and that information between actors can only flow via message exchange. The SIAAM machine has been implemented for Java as a modified Jikes virtual machine. The resulting SIAAM software comprises:

- A modified Jikes RVM that implements actors and actor isolation as specified by the SIAAM machine.
- A set of static analyses build using the Soot Java optimization framework for optimizing the execution of the SIAAM/Jikes virtual machine, and for helping programmers diagnose potential performance issues.
- A formal proof using the Coq proof assistant of the SIAAM isolation property.

The SIAAM machine is the subject of Quentin Sabah's PhD thesis [67].

5.1.4. pyCPA_TCA

Participant: Sophie Quinton.

We are developing pyCPA_TCA, a pyCPA plugin for Typical Worst-Case Analysis as described in Section 6.2.2 . pyCPA is an open-source Python implementation of Compositional Performance Analysis developed at TU Braunschweig, which allows in particular response-time analysis. pyCPA_TCA is an extension of this tool that is co-developed by Sophie Quinton and Zain Hammadeh at TU Braunschweig. It allows in particular the computation of weakly-hard guarantees for real-time tasks, i.e. number of deadline misses out of a sequence of executions. So far, pyCPA_TCA is restricted to uniprocessor systems of independent tasks, scheduled according to static priority scheduling.

TEA Project-Team

5. New Software and Platforms

5.1. The Eclipse project POP

Participants: Loïc Besnard, Thierry Gautier, Paul Le Guernic, Jean-Pierre Talpin.

The distribution of project POP [0] is a major achievement of the ESPRESSO project. The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. It was finalised in the frame of project OPEES, as a case study: by passing the POLARSYS qualification kit as a computer aided simulation and verification tool. This qualification was implemented by CS Toulouse in conformance with relevant generic (platform independent) qualification documents. Polychrony is now distributed by the Eclipse project POP on the platform of the POLARSYS industrial working group. Team TEA aims at continuing its dissemination to academic partners, as to its principles and features, and industrial partners, as to the services it can offer.

Technically, project POP is composed of the Polychrony toolset, under GPL license, and its Eclipse framework, under EPL license.

The Polychrony toolset. The Polychrony toolset is an Open Source development environment for critical/embedded systems. It is based on Signal, a real-time polychronous dataflow language. It provides a unified model-driven environment to perform design exploration by using top-down and bottom-up design methodologies formally supported by design model transformations from specification to implementation and from synchrony to asynchrony. It can be included in heterogeneous design systems with various input formalisms and output languages.

The Polychrony toolset provides a formal framework:

- to validate a design at different levels, by the way of formal verification and/or simulation,
- to refine descriptions in a top-down approach,
- to abstract properties needed for black-box composition,
- to assemble heterogeneous predefined components (bottom-up with COTS),
- to generate executable code for various architectures.

The Polychrony toolset contains three main components and an experimental interface to GNU Compiler Collection (GCC):

- The Signal toolbox, a batch compiler for the Signal language, and a structured API that provides a set of program transformations. The Signal toolbox can be installed without other components. The Signal toolbox is distributed under GPL V2 license.
- The Signal GUI, a Graphical User Interface to the Signal toolbox (editor + interactive access to compiling functionalities). The Signal GUI is distributed under GPL V2 license.
- The SME/SSME platform, a front-end to the Signal toolbox in the Eclipse environment. The SME/SSME platform is distributed under EPL license.
- GCCst, a back-end to GCC that generates Signal programs (not yet available for download).

In 2013, to be able to use the Signal GUI both as a specific tool and as a graphical view under Eclipse, the code of the Signal GUI has been restructured in three parts: a common part used by both tools (28 classes), a specific part for the Signal GUI (2 classes), a specific part for Eclipse (2 classes). Such a structuration facilitates the maintenance of the products.

[0] *Polychrony on POLARSYS (POP)*, an Eclipse project in the POLARSYS Industry Working Group, 2013. https://www.POLARSYS.org/projects/POLARSYS.pop

The Polychrony toolset also provides:

- libraries of Signal programs,
- a set of Signal program examples,
- user oriented and implementation documentations,
- facilities to generate new versions.

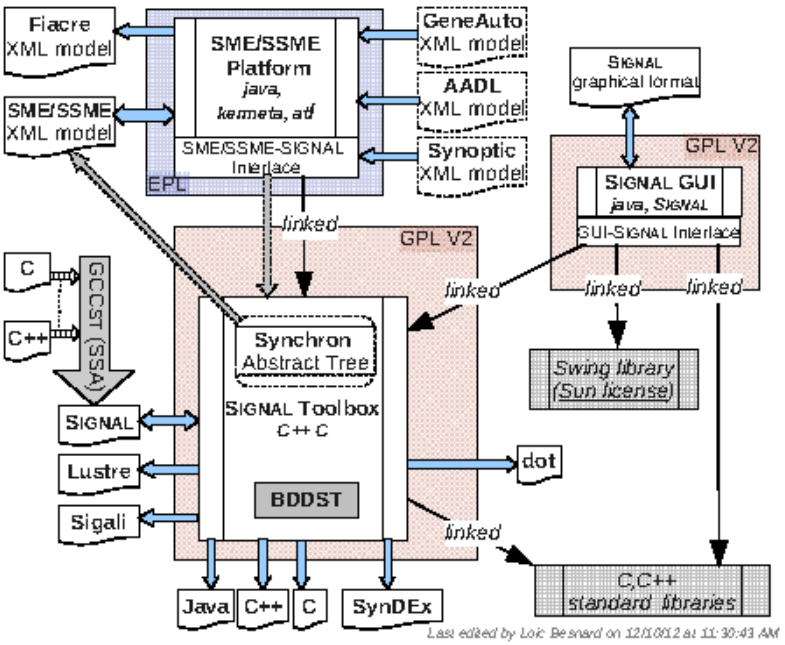

Figure 1. The Polychrony toolset high-level architecture

Dassault Systèmes, supplies a commercial implementation of Polychrony, called RT-Builder, used for industrial scale projects.

As part of its open-source release, the Polychrony toolset not only comprises source code libraries but also an important corpus of structured documentation, whose aim is not only to document each functionality and service, but also to help a potential developer to package a subset of these functionalities and services, and adapt them to developing a new application-specific tool: a new language front-end, a new back-end compiler. This multi-scale, multi-purpose documentation aims to provide different views of the software, from a high-level structural view to low-level descriptions of basic modules. It supports a distribution of the software "by apartment" (a functionality or a set of functionalities) intended for developers who would only be interested by part of the services of the toolset.

The Eclipse POP Framework. We have developed a meta-model and interactive editor of Polychrony in Eclipse. Signal-Meta is the meta-model of the Signal language implemented with Eclipse/Ecore. It describes all syntactic elements specified in [0]: all Signal operators (e.g. arithmetic, clock synchronization), model (e.g. process frame, module), and construction (e.g. iteration, type declaration).

[0] *SIGNAL V4-Inria version: Reference Manual*. Besnard, L., Gautier, T. and Le Guernic, P. http://www.irisa.fr/espresso/Polychrony, 2009

The meta-model primarily aims at making the language and services of the Polychrony environment available to inter-operation and composition with other components (e.g. AADL, Simulink, GeneAuto) within an Eclipse-based development toolchain. Polychrony now comprises the capability to directly import and export Ecore models instead of textual Signal programs, in order to facilitate interaction between components within such a toolchain.

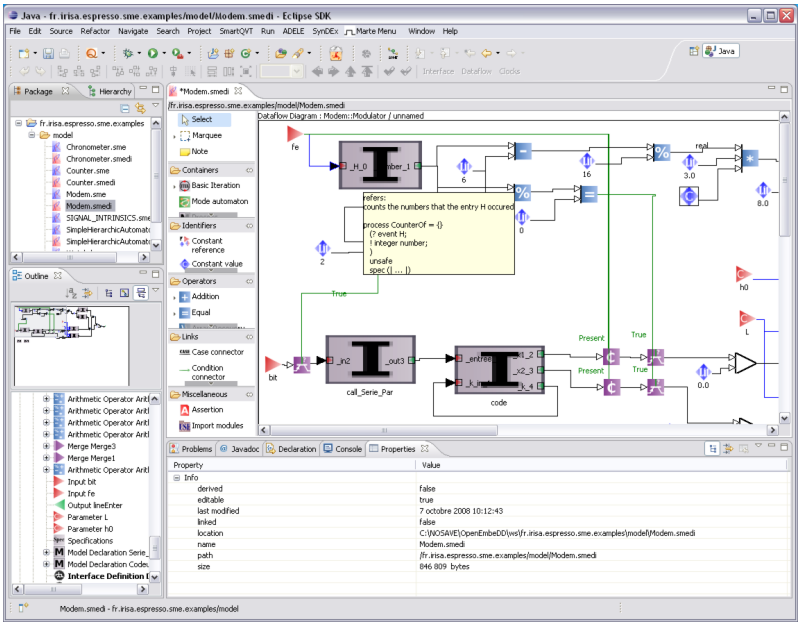

Figure 2. The Eclipse POP Environment

It also provides a graphical modelling framework allowing to design applications using a component-based approach. Application architectures can be easily described by just selecting components via drag and drop, creating some connections between them and specifying their parameters as component attributes. Using the modelling facilities provided with the Topcased framework, we have created a graphical environment for Polychrony called SME (Signal-Meta under Eclipse). To highlight the different parts of the modelling in Signal, we split the modelling of a Signal process in three diagrams: one to model the interface of the process, one to model the computation (or dataflow) part, and one to model all explicit clock relations and dependences. The SME environment is available through the ESPRESSO update site [0]. A new meta-model of Signal, called SSME (Syntactic Signal-Meta under Eclipse), closer to the Signal abstract syntax, has been defined and integrated in the Polychrony toolset.

It should be noted that the Eclipse Foundation does not host code under GPL license. So, the Signal toolbox useful to compile Signal code from Eclipse is hosted on our web server. For this reason, the building of the Signal toolbox, previously managed under Eclipse, has now been exported. The interface of the Signal toolbox for Eclipse is now managed using the CMake tool like the Signal toolbox and the Signal GUI.

5.2. Integrated Modular Avionics design using Polychrony

Participants: Loïc Besnard, Thierry Gautier, Paul Le Guernic, Jean-Pierre Talpin.

[0] *Polychrony Update Site for Eclipse plug-ins*. http://www.irisa.fr/espresso/Polychrony/update, 2009.

The Apex interface, defined in the ARINC standard [0], provides an avionics application software with the set of basic services to access the operating-system and other system-specific resources. Its definition relies on the Integrated Modular Avionics approach (IMA). A main feature in an IMA architecture is that several avionics applications (possibly with different critical levels) can be hosted on a single, shared computer system. Of course, a critical issue is to ensure safe allocation of shared computer resources in order to prevent fault propagations from one hosted application to another. This is addressed through a functional partitioning of the applications with respect to available time and memory resources. The allocation unit that results from this decomposition is the *partition*.

A partition is composed of *processes* which represent the executive units (an ARINC partition/process is akin to a Unix process/task). When a partition is activated, its owned processes run concurrently to perform the functions associated with the partition. The process scheduling policy is priority preemptive. Each partition is allocated to a processor for a fixed time window within a major time frame maintained by the operating system. Suitable mechanisms and devices are provided for communication and synchronization between processes (e.g. *buffer*, *event*, *semaphore*) and partitions (e.g. *ports* and *channels*). The specification of the ARINC 651-653 services in Signal [4] is now part of the Polychrony distribution and offers a complete implementation of the Apex communication, synchronization, process management and partitioning services. Its Signal implementation consists of a library of generic, parameterizable Signal modules.

5.3. Safety-Critical Java Level 1 Code generation from Dataflow Graph Specifications

Participants: Adnan Bouakaz, Thierry Gautier, Jean-Pierre Talpin.

We have proposed a dataflow design model [2] of SCJ/L1 applications [0] in which handlers (periodic and aperiodic actors) communicate only through lock-free channels. Hence, each mission is modeled as a dataflow graph. The presented dataflow design model comes with a development tool integrated in the Eclipse IDE for easing the development of SCJ/L1 applications and enforcing the restrictions imposed by the design model. It consists of a GMF editor where applications are designed graphically and timing and buffering parameters can be synthesized. Indeed, abstract affine scheduling is first applied on the dataflow subgraph, that consists only of periodic actors, to compute timeless scheduling constraints (e.g. relation between the speeds of two actors) and buffering parameters. Then, symbolic fixed-priority schedulability analysis (i.e., synthesis of timing and scheduling parameters of actors) considers both periodic and aperiodic actors.

Through a model-to-text transformation, using Acceleo, the SCJ code for missions, interfaces of handlers, and the mission sequencer is automatically generated in addition to the annotations needed by the memory checker. Channels are implemented as cyclic arrays or cyclical asynchronous buffers; and a fixed amount of memory is hence reused to store the infinite streams of tokens. The user must provide the SCJ code of all the `handleAsyncEvent()` methods. We have integrated the SCJ memory checker [0] in our tool so that potential dangling pointers can be highlighted at compile-time. To enhance functional determinism, we would like to develop an ownership type system to ensure that actors are strongly isolated and communicate only through buffers.

[0] *ARINC Report 651-1: Design Guidance for Integrated Modular Avionics*. Airlines Electronic Engineering Committee, 1997

[0] *Safety critical Java technology specification*. JSR-302, Year = 2010

[0] *Static checking of safety critical Java annotations*. Tang, D. Plsek, A. and Vitek, J. International Workshop on Java Technologies for Real-Time and Embedded Systems, 2010

ANTIQUE Team

5. New Software and Platforms

5.1. The Apron Numerical Abstract Domain Library

Participants: Antoine Miné [correspondent], Bertrand Jeannet [team PopArt, Inria-RA].

The APRON library is dedicated to the static analysis of the numerical variables of a program by abstract interpretation. Its goal is threefold: provide ready-to-use numerical abstractions under a common API for analysis implementers, encourage the research in numerical abstract domains by providing a platform for integration and comparison of domains, and provide a teaching and demonstration tool to disseminate knowledge on abstract interpretation.

The APRON library is not tied to a particular numerical abstraction but instead provides several domains with various precision versus cost trade-offs (including intervals, octagons, linear equalities and polyhedra). A specific C API was designed for domain developers to minimize the effort when incorporating a new abstract domain: only few domain-specific functions need to be implemented while the library provides various generic services and fallback methods (such as scalar and interval operations for most numerical data-types, parametric reduced products, and generic transfer functions for non-linear expressions). For the analysis designer, the APRON library exposes a higher-level API with C, C++, OCaml, and Java bindings. This API is domain-neutral and supports a rich set of semantic operations, including parallel assignments (useful to analyze automata), substitutions (useful for backward analysis), non-linear numerical expressions, and IEEE floating-point arithmetic.

The APRON library is freely available on the web at http://apron.cri.ensmp.fr/library; it is distributed under the LGPL license and is hosted at InriaGForge. Packages exist for the Debian and Fedora Linux distributions. In order to help disseminate the knowledge on abstract interpretation, a simple inter-procedural static analyzer for a toy language is included. An on-line version is deployed at http://pop-art.inrialpes.fr/interproc/interprocweb.cgi.

The APRON library is developed since 2006 and currently consists of 130 000 lines of C, C++, OCaml, and Java.

Current and past external library users include the Constraint team (LINA, Nantes, France), the Proval/Démon team (LRI Orsay, France), the Analysis of Computer Systems Group (New-York University, USA), the Sierum software analysis platform (Kansas State University, USA), NEC Labs (Princeton, USA), EADS CCR (Paris, France), IRIT (Toulouse, France), ONERA (Toulouse, France), CEA LIST (Saclay, France), VERIMAG (Grenoble, France), ENSMP CRI (Fontainebleau, France), the IBM T.J. Watson Research Center (USA), the University of Edinburgh (UK).

Additionally, APRON is used internally by the team to assist the research on numeric domains and static analyses by enabling the development of fast prototypes. Specifically, in 2014, APRON has been used to support the design of piece-wise linear ranking function domains to infer termination and functional liveness properties in the implementation of the FUNCTION prototype analyzer, and to implement and experiment a new numeric domain for octagonal constraints with absolute values. It has also been used in the introductory course on program verification given by members of the team.

5.2. The Astrée Static Analyzer of Synchronous Software

Participants: Patrick Cousot [project scientific leader, correspondent], Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, Xavier Rival.

ASTRÉE is a static analyzer for sequential programs based on abstract interpretation [40], [35], [41], [36].

The ASTRÉE static analyzer [34], [44][1] www.astree.ens.fr aims at proving the absence of runtime errors in programs written in the C programming language.

ASTRÉE analyzes structured C programs, with complex memory usages, but without dynamic memory allocation nor recursion. This encompasses many embedded programs as found in earth transportation, nuclear energy, medical instrumentation, and aerospace applications, in particular synchronous control/command. The whole analysis process is entirely automatic.

ASTRÉE discovers all runtime errors including:

- undefined behaviors in the terms of the ANSI C99 norm of the C language (such as division by 0 or out of bounds array indexing);
- any violation of the implementation-specific behavior as defined in the relevant Application Binary Interface (such as the size of integers and arithmetic overflows);
- any potentially harmful or incorrect use of C violating optional user-defined programming guidelines (such as no modular arithmetic for integers, even though this might be the hardware choice);
- failure of user-defined assertions.

The analyzer performs an abstract interpretation of the programs being analyzed, using a parametric domain (ASTRÉE is able to choose the right instantiation of the domain for wide families of software). This analysis produces abstract invariants, which over-approximate the reachable states of the program, so that it is possible to derive an *over*-approximation of the dangerous states (defined as states where any runtime error mentioned above may occur) that the program may reach, and produces alarms for each such possible runtime error. Thus the analysis is sound (it correctly discovers *all* runtime errors), yet incomplete, that is it may report false alarms (i.e., alarms that correspond to no real program execution). However, the design of the analyzer ensures a high level of precision on domain-specific families of software, which means that the analyzer produces few or no false alarms on such programs.

In order to achieve this high level of precision, ASTRÉE uses a large number of expressive abstract domains, which allow expressing and inferring complex properties about the programs being analyzed, such as numerical properties (digital filters, floating-point computations), Boolean control properties, and properties based on the history of program executions.

ASTRÉE has achieved the following two unprecedented results:

- **A340–300.** In Nov. 2003, ASTRÉE was able to prove completely automatically the absence of any RTE in the primary flight control software of the Airbus A340 fly-by-wire system, a program of 132,000 lines of C analyzed in 1h20 on a 2.8 GHz 32-bit PC using 300 MB of memory (and 50mn on a 64-bit AMD Athlon 64 using 580 MB of memory).
- **A380.** From Jan. 2004 on, ASTRÉE was extended to analyze the electric flight control codes then in development and test for the A380 series. The operational application by Airbus France at the end of 2004 was just in time before the A380 maiden flight on Wednesday, 27 April, 2005.

These research and development successes have led to consider the inclusion of ASTRÉE in the production of the critical software for the A350. ASTRÉE is currently industrialized by AbsInt Angewandte Informatik GmbH and is commercially available.

5.3. The AstréeA Static Analyzer of Asynchronous Software

Participants: Patrick Cousot [project scientific leader, correspondent], Radhia Cousot, Jérôme Feret, Antoine Miné, Xavier Rival.

ASTRÉEA is a static analyzer prototype for parallel software based on abstract interpretation [42], [43], [37]. It started with support from THÉSÉE ANR project (2006–2010) and is continuing within the ASTRÉEA project (2012–2015).

The ASTRÉEA prototype www.astreea.ens.fr is a fork of the ASTRÉE static analyzer (see 5.2) that adds support for analyzing parallel embedded C software.

ASTRÉEA analyzes C programs composed of a fixed set of threads that communicate through a shared memory and synchronization primitives (mutexes, FIFOs, blackboards, etc.), but without recursion nor dynamic creation of memory, threads nor synchronization objects. ASTRÉEA assumes a real-time scheduler, where thread scheduling strictly obeys the fixed priority of threads. Our model follows the ARINC 653 OS specification used in embedded industrial aeronautic software. Additionally, ASTRÉEA employs a weakly-consistent memory semantics to model memory accesses not protected by a mutex, in order to take into account soundly hardware and compiler-level program transformations (such as optimizations). ASTRÉEA checks for the same run-time errors as ASTRÉE, with the addition of data-races.

Compared to ASTRÉE, ASTRÉEA features: a new iterator to compute thread interactions, a refined memory abstraction that takes into account the effect of interfering threads, and a new scheduler partitioning domain. This last domain allows discovering and exploiting mutual exclusion properties (enforced either explicitly through synchronization primitives, or implicitly by thread priorities) to achieve a precise analysis.

ASTRÉEA is currently being applied to analyze a large industrial avionic software: 1.6 MLines of C and 15 threads, completed with a 2,500-line model of the ARINC 653 OS developed for the analysis. The analysis currently takes a few tens of hours on a 2.9 GHz 64-bit intel server using one core and generates around 1,050 alarms. The low computation time (only a few times larger than the analysis time by ASTRÉE of synchronous programs of a similar size and structure) shows the scalability of the approach (in particular, we avoid the usual combinatorial explosion associated to thread interleavings). Precision-wise, the result, while not as impressive as that of ASTRÉE, is quite encouraging. The development of ASTRÉEA continues within the scope of the ASTRÉEA ANR project.

5.4. ClangML: A binding with the CLANG C-frontend

Participants: François Berenger [Correspondent], Devin Mccoughlin, Pippijn Van Steenhoeven.

CLANGML is an OCaml binding with the CLANG front-end of the LLVM compiler suite. Its goal is to provide an easy to use solution to parse a wide range of C programs, that can be called from static analysis tools implemented in OCaml, which allows to test them on existing programs written in C (or in other idioms derived from C) without having to redesign a front-end from scratch. CLANGML features an interface to a large set of internal AST nodes of CLANG, with an easy to use API. Currently, CLANGML supports all C language AST nodes, as well as a large part of the C nodes related to C++ and Objective-C.

It has been applied to the parsing of the Minix micro-kernel as well as of other C programs.

CLANGML has been implemented in C++, OCaml and Camlp4. It has been released as an open source contribution on GitHUB and as an OPAM package.

5.5. FuncTion: An Abstract Domain Functor for Termination

Participants: Caterina Urban, Antoine Miné [Correspondent].

FUNCTION is a research prototype static analyzer to analyze the termination and functional liveness properties of programs. It accepts programs in a small non-deterministic imperative language. It is also parameterized by a property: either termination, or a recurrence or a guarantee property (according to the classification by Manna and Pnueli of program properties). It then performs a backward static analysis that automatically infers sufficient conditions at the beginning of the program so that all executions satisfying the conditions also satisfy the property. FUNCTION is based on an extension to liveness properties of the framework to analyze termination by abstract interpretation proposed by Patrick Cousot and Radhia Cousot in [39]. FUNCTION infers ranking functions using piecewise-defined abstract domains. Several domains are available to partition the ranking function, including intervals, octagons, and polyhedra. Two domains are also available to represent the value of ranking functions: a domain of affine ranking functions, and a domain of ordinal-valued ranking functions (which allows handling programs with unbounded non-determinism).

The analyzer is written in OCaml and implemented on top of the APRON library. It can be used on-line through a web interface: http://www.di.ens.fr/~urban/FuncTion.html.

FUNCTION participated to SV-COMP 2014 (3rd Competition on Software Verification, demonstration section) and is also selected to participate to SV-COMP 2015 in the termination category [31].

5.6. HOO: Heap Abstraction for Open Objects

Participant: Arlen Cox [Correspondent].

JSAna with HOO is a static analyzer for JavaScript programs. The primary component, HOO, which is designed to be reusable by itself, is an abstract domain for a dynamic language heap. A dynamic language heap consists of open, extensible objects linked together by pointers. Uniquely, HOO abstracts these extensible objects, where attribute/field names of objects may be unknown. Additionally, it contains features to keeping precise track of attribute name/value relationships as well as calling unknown functions through desynchronized separation.

As a library, HOO is useful for any dynamic language static analysis. It is designed to allow abstractions for values to be easily swapped out for different abstractions, allowing it to be used for a wide-range of dynamic languages outside of JavaScript.

5.7. The MemCADstatic analyzer

Participants: Xavier Rival [correspondent], François Berenger, Huisong Li, Antoine Toubhans.

Shape analysis. MEMCADis a static analyzer that focuses on memory abstraction. It takes as input C programs, and computes invariants on the data structures manipulated by the programs. It can also verify memory safety. It comprises several memory abstract domains, including a flat representation, and two graph abstractions with summaries based on inductive definitions of data-structures, such as lists and trees and several combination operators for memory abstract domains (hierarchical abstraction, reduced product). The purpose of this construction is to offer a great flexibility in the memory abstraction, so as to either make very efficient static analyses of relatively simple programs, or still quite efficient static analyses of very involved pieces of code. The implementation consists of over 30 000 lines of ML code, and relies on the CLANGML front-end. The current implementation comes with over 300 small size test cases that are used as regression tests.

5.8. The OpenKappa Modeling Plateform

Participants: Pierre Boutillier [Paris VII], Monte Brown [Harvard Medical School], Vincent Danos [University of Edinburgh], Jérôme Feret [Correspondent], Luca Grieco, Walter Fontana [Harvard Medical School], Russ Harmer [ENS Lyon], Jean Krivine [Paris VII].

Causal traces, Model reduction, Rule-based modeling, Simulation, Static analysis. OPENKAPPA is a collection of tools to build, debug and run models of biological pathways. It contains a compiler for the Kappa Language [50], a static analyzer [49] (for debugging models), a simulator [48], a compression tool for causal traces [47], [45], and a model reduction tool [4], [46], [53].

OPENKAPPA is developed since 2007 and, the OCaml version currently consists of 46 000 lines of OCaml. Software are available in OCaml and in Java. Moreover, an Eclipse pluggin is available.A compiler from CellDesigner into Kappa has been released in 2013.

OPENKAPPA is freely available on the web at http://kappalanguage.org under the LGPL license. Discussion groups are also available on line.

Current external users include the ETH Zürich, the UNAM-Genomics Mexico team. It is used as pedagocical material in graduate lessons at Harvard Medical School, and at the Interdisciplinary Approaches to Life science (AIV) Master Program (Université de Médecine Paris-Descartes).

5.9. QUICr set abstract domain

Participant: Arlen Cox [Correspondent].

QUICr is an OCaml library that implements a parametric abstract domain for sets. It is constructed as a functor that accepts any numeric abstract domain that can be adapted to the interface and produces an abstract domain for sets of numbers combined with numbers. It is relational, flexible, and tunable. It serves as a basis for future exploration of set abstraction.

5.10. Translation Validation

Participant: Xavier Rival [correspondent].

Abstract interpretation, Certified compilation, Static analysis, Translation validation, Verifier. The main goal of this software project is to make it possible to certify automatically the compilation of large safety critical software, by proving that the compiled code is correct with respect to the source code: When the proof succeeds, this guarantees that no compiler bug did cause incorrect code to be generated. Furthermore, this approach should allow to meet some domain specific software qualification criteria (such as those in DO-178 regulations for avionics software), since it allows proving that successive development levels are correct with respect to each other i.e., that they implement the same specification. Last, this technique also justifies the use of source level static analyses, even when an assembly level certification would be required, since it establishes separately that the source and the compiled code are equivalent.

The compilation certification process is performed automatically, thanks to a prover designed specifically. The automatic proof is done at a level of abstraction which has been defined so that the result of the proof of equivalence is strong enough for the goals mentioned above and so that the proof obligations can be solved by efficient algorithms.

The current software features both a C to Power-PC compilation certifier and an interface for an alternate source language frontend, which can be provided by an end-user.

5.11. Zarith

Participants: Antoine Miné [Correspondent], Xavier Leroy [Inria Paris-Rocquencourt], Pascal Cuoq [CEA LIST].

ZARITH is a small (10K lines) OCaml library that implements arithmetic and logical operations over arbitrary-precision integers. It is based on the GNU MP library to efficiently implement arithmetic over big integers. Special care has been taken to ensure the efficiency of the library also for small integers: small integers are represented as Caml unboxed integers and use a specific C code path. Moreover, optimized assembly versions of small integer operations are provided for a few common architectures.

ZARITH is an open-source project hosted at OCamlForge (http://forge.ocamlcore.org/projects/zarith) and distributed under a modified LGPL license.

ZARITH is currently used in the ASTRÉE analyzer to enable the sound analysis of programs featuring 64-bit (or larger) integers. It is also used in the Frama-C analyzer platform developed at CEA LIST and Inria Saclay.

CELTIQUE Project-Team

4. New Software and Platforms

4.1. Javalib

Participants: Frédéric Besson [correspondant], David Pichardie, Pierre Vittet, Laurent Guillo.

Javalib is an efficient library to parse Java .class files into OCaml data structures, thus enabling the OCaml programmer to extract information from class files, to manipulate and to generate valid .class files.
See also the web page http://sawja.inria.fr/.

- Version: 2.3
- Programming language: Ocaml

4.2. SAWJA

Participants: Frédéric Besson [correspondant], David Pichardie, Pierre Vittet, Laurent Guillo.

Sawja is a library written in OCaml, relying on Javalib to provide a high level representation of Java bytecode programs. It name comes from Static Analysis Workshop for JAva. Whereas Javalib is dedicated to isolated classes, Sawja handles bytecode programs with their class hierarchy and with control flow algorithms.

Moreover, Sawja provides some stackless intermediate representations of code, called JBir and A3Bir. The transformation algorithm, common to these representations, has been formalized and proved to be semantics-preserving.
See also the web page http://sawja.inria.fr/.

- Version: 1.5
- Programming language: Ocaml

4.3. Jacal

Participants: Frédéric Besson [correspondant], Thomas Jensen, David Pichardie, Delphine Demange.

Static program analysis, Javacard, Certification, AFSCM

`Jacal` is a JAvaCard AnaLyseur developed on top of the SAWJA (see Section 4.2) platform. This proprietary software verifies automatically that Javacard programs conform with the security guidelines issued by the AFSCM (Association Française du Sans Contact Mobile). `Jacal` is based on the theory of abstract interpretation and combines several object-oriented and numeric analyses to automatically infer sophisticated invariants about the program behaviour. The result of the analysis is thereafter harvest to check that it is sufficient to ensure the desired security properties.

4.4. Timbuk

Participant: Thomas Genet [correspondant].

Timbuk is a library of OCAML functions for manipulating tree automata. More precisely, Timbuk deals with finite bottom-up tree automata (deterministic or not). This library provides the classical operations over tree automata (intersection, union, complement, emptiness decision) as well as exact or approximated sets of terms reachable by a given term rewriting system. This last operation can be certified using a checker extracted from a Coq specification. The checker is now part of the Timbuk distribution. Timbuk distribution now also provides a CounterExample Guided Abstraction Refinement (CEGAR) tool for tree automata completion. The CEGAR part is based on the Buddy BDD library. Timbuk also provides an implementation of Lattice Tree Automata to (efficiently) represent built-in values such as integers, strings, etc. in recognized tree languages. See also the web page http://www.irisa.fr/celtique/genet/timbuk/.

- Version: 3.1
- Programming language: Ocaml

4.5. JSCert

Participants: Alan Schmitt [correspondant], Martin Bodin.

The JSCert project aims to really understand JavaScript. JSCert itself is a mechanised specification of JavaScript, written in the Coq proof assistant, which closely follows the ECMAScript 5 English standard. JSRef is a reference interpreter for JavaScript in OCAML, which has been proved correct with respect to JSCert and tested with the Test 262 test suite.

We plan to build other verification and analysis projects on top of JSCert and JSRef, in particular the certification of derivations in program logics or static analyses.

This project is an ongoing collaboration between Inria and Imperial College. More information, including the source code, is available at http://jscert.org/.

DEDUCTEAM Exploratory Action

5. New Software and Platforms

5.1. Dedukti

Dedukti is a proof-checker for the $\lambda\Pi$-calculus modulo. As it can be parametrized by an arbitrary set of rewrite rules, defining an equivalence relation, this calculus can express many different theories. Dedukti has been created for this purpose: to allow the interoperability of different theories.

Dedukti's core is based on the standard algorithm [29] for type-checking semi-full pure type systems and implements a state-of-the-art reduction machine inspired from Matita's [28] and modified to deal with rewrite rules.

Dedukti's input language features term declarations and definitions (opaque or not) and rewrite rule definitions. A basic module system allows the user to organize its project in different files and compile them separately.

Dedukti now features matching modulo beta for a large class of patterns called Miller's patterns, allowing for more rewriting rules to be implemented in Dedukti.

Dedukti has been developed by Mathieu Boespflug, Olivier Hermant, Quentin Carbonneaux and Ronan Saillard. It is composed of about 2500 lines of OCaml.

5.2. Coqine, Holide, Focalide and Sigmaid

Dedukti comes with four companion tools: Holide, an embedding of HOL proofs through the OpenTheory format [41], Coqine, an embedding of Coq proofs, Focalide, an embedding of FoCaLiZe certified programs, and Sigmaid, a type-checker for the simply-typed ς-calculus with subtyping and a translator to Dedukti. All of the OpenTheory standard library and a part of Coq's and FoCaLiZe's libraries are checked by Dedukti.

A preliminary version of Coqine supports the following features of Coq: the raw Calculus of Constructions, inductive types, and fixpoint definitions. Coqine is currently being rewritten to support universes. Coqine has been developed by Mathieu Boespflug, Guillaume Burel, and Ali Assaf.

Holide supports all the features of HOL, including ML-polymorphism, constant definitions, and type definitions. It is able to translate all of the OpenTheory standard theory library. Holide has been developed by Ali Assaf.

Focalide has been improved to support FoCaLiZe proofs found by Zenon using the Dedukti backend for Zenon developped by Frédéric Gilbert. This backend has been improved by a simple typing mechanism in order to work with Focalide. Focalide has also been updated again to work with the latest version of FoCaLiZe.

Sigmaid implements a shallow embedding of the simply-typed ς-calculus of Abadi and Cardelli with subtyping in the $\lambda\Pi$-calculus modulo. This translation has been proved[21] to preserve the typing judgments and the semantics of the simply-typed ς-calculus and tested on the examples of Abadi and Cardelli.

Focalide and Sigmaid have been developed by Raphaël Cauderlier.

Translators from Version 2.0 of the SMT-LIB standard and from the SMT-solver veriT have been initiated. They are currently developed by Frédéric Gilbert.

5.3. iProver Modulo

iProver Modulo is an extension of the automated theorem prover iProver originally developed by Konstantin Korovin at the University of Manchester. It implements Ordered polarized resolution modulo, a refinement of the Resolution method based on Deduction modulo. It takes as input a proposition in predicate logic and a clausal rewriting system defining the theory in which the formula has to be proved. Normalization with respect to the term rewriting rules is performed very efficiently through translation into OCaml code, compilation and dynamic linking. Experiments have shown that Ordered polarized resolution modulo dramatically improves proof search compared to using raw axioms. iProver modulo is also able to produce proofs that can be checked by Dedukti, therefore improving confidence. iProver modulo is written in OCaml, it consists of 1,200 lines of code added to the original iProver.

A tool that transforms axiomatic theories into polarized rewriting systems, thus making them usable in iProver Modulo, has also been developed. Autotheo supports several strategies to orient the axioms, some of them being proved to be complete, in the sense that Ordered polarized resolution modulo the resulting systems is refutationally complete, some others being merely heuristics. In practice, autotheo takes a TPTP input file and transforms the axioms into rewriting rules, and produces an input file for iProver Modulo.

iProver Modulo and autotheo have been developed by Guillaume Burel. iProver Modulo is released under a GPL license.

5.4. Super Zenon and Zenon Modulo

Several extensions of the *Zenon* automated theorem prover (developed by Damien Doligez at *Inria* in the *Gallium* team) to Deduction modulo have been studied. These extensions intend to be applied in the context of the automatic verification of proof rules and obligations coming from industrial applications formalized using the *B* method.

The first extension, developed by Mélanie Jacquel and David Delahaye, is called *Super Zenon* and is an extension of *Zenon* to superdeduction, which can be seen as a variant of Deduction modulo. This extension is a generalization of previous experiments [42] together with Catherine Dubois and Karim Berkani (*Siemens*), where *Zenon* has been used and extended to superdeduction to deal with the *B* set theory and automatically prove proof rules of *Atelier B*. This generalization consists in allowing us to apply the extension of *Zenon* to superdeduction to any first order theory by means of a heuristic that automatically transforms axioms of the theory into rewrite rules. This work is described in [13] [35], which also proposes a study of the possibility of recovering intuition from automated proofs using superdeduction.

The second extension, developed by Pierre Halmagrand, David Delahaye, Damien Doligez, and Olivier Hermant, is called *Zenon Modulo* and is an extension of *Zenon* to Deduction modulo. Compared to *Super Zenon*, this extension allows us to deal with rewrite rules both over propositions and terms. Like *Super Zenon*, *Zenon Modulo* is able to deal with any first order theory by means of a similar heuristic. To assess the approach of *Zenon Modulo*, we have applied this extension to the first order problems coming from the TPTP library. An increase of the number of proved problems has been observed, with in particular a significant increase in the category of set theory. Over these problems of the TPTP library, we have also observed a significant proof size reduction, which confirms this aspect of Deduction modulo. These results are gathered into two publications [33], [34].

The third extension, developed by Guillaume Bury and David Delahaye, is an extension of *Zenon* to (rational and integer) linear arithmetic (using the simplex algorithm), that has been integrated to *Zenon Modulo* by Guillaume Bury and Pierre Halmagrand, in order to be applied in the framework of the *B* set theory to the verification of proof obligations of *Atelier B* [17]. Experiments have been conducted over the benchmarks of the *BWare* project, and it turns out that more than 95% of the proof obligations are proved thanks to this extension.

5.5. Zipperposition (and extensions) and Logtk

Zipperposition is an implementation of the superposition method; it relies on the library Logtk for basic logic data structures and algorithms. Zipperposition is designed as a testbed for extensions to superposition, and can currently deal with polymorphic typed logic, integer arithmetic, and total orderings; an extension to handle structural induction is being worked on by Simon Cruanes.

Those pieces of software also depend on many smaller tools and libraries developped by Simon Cruanes in OCaml. In particular, efficient iterators were key to implementing arithmetic rules successfully, and a lightweight extension to the standard library has been developped steadily and released regularly.

5.6. CoLoR

CoLoR is a Coq library on rewriting theory and termination of more than 83,000 lines of code [2]. It provides definitions and theorems for:

- Mathematical structures: relations, (ordered) semi-rings.
- Data structures: lists, vectors, polynomials with multiple variables, finite multisets, matrices, finite graphs.
- Term structures: strings, algebraic terms with symbols of fixed arity, algebraic terms with varyadic symbols, pure and simply typed λ-terms.
- Transformation techniques: conversion from strings to algebraic terms, conversion from algebraic to varyadic terms, arguments filtering, rule elimination, dependency pairs, dependency graph decomposition, semantic labelling.
- Termination criteria: polynomial interpretations, multiset ordering, lexicographic ordering, first and higher order recursive path ordering, matrix interpretations.

CoLoR is distributed under the CeCILL license. It is currently developed by Frédéric Blanqui and Kim-Quyen Ly, but various people participated to its development since 2006 (see the website for more information).

5.7. HOT

HOT is an automated termination prover for higher-order rewrite systems based on the notion of computability closure and size annotation [31]. It won the 2012 competition in the category “higher-order rewriting union beta”. The sources (5000 lines of OCaml) are not public. It is developed by Frédéric Blanqui.

5.8. Moca

Moca is a construction functions generator for OCaml data types with invariants.

It allows the high-level definition and automatic management of complex invariants for data types. In addition, it provides the automatic generation of maximally shared values, independently or in conjunction with the declared invariants.

A relational data type is a concrete data type that declares invariants or relations that are verified by its constructors. For each relational data type definition, Moca compiles a set of construction functions that implements the declared relations.

Moca supports two kinds of relations:

- predefined algebraic relations (such as associativity or commutativity of a binary constructor),
- user-defined rewrite rules that map some pattern of constructors and variables to some arbitrary user’s define expression.

The properties that user-defined rules should satisfy (completeness, termination, and confluence of the resulting term rewriting system) must be verified by a programmer’s proof before compilation. For the predefined relations, Moca generates construction functions that allow each equivalence class to be uniquely represented by their canonical value.

Moca is distributed under QPL. It is written in OCaml (14,000 lines) It is developed by Frédéric Blanqui, Pierre Weis (EPI Pomdapi) and Richard Bonichon (CEA).

5.9. Rainbow

Rainbow is a set of tools for automatically verifying the correctness of termination certificates expressed in the CPF format used in the termination competition. It contains:

- a tool `xsd2coq` for generating Coq data types for representing XML files valid wrt some XML Schema,
- a tool `xsd2ml` for generating OCaml data types and functions for parsing XML files valid wrt some XML Schema,
- a tool for translating a CPF file into a Coq script,
- a standalone Coq certified tool for verifying the correctness of a CPF file.

Rainbow is distributed under the CeCILL license. It is developed in OCaml (10,000 lines) and Coq (19,000 lines). It is currently developed by Frédéric Blanqui and Kim-Quyen Ly. See the website for more information.

5.10. mSAT

mSAT is a modular, proof-producing, SAT and SMT core based on Alt-Ergo Zero, written in OCaml. The solver accepts user-defined terms, formulas and theory, making it a good tool for experimenting. This tool produces resolution proofs as trees in which the leaves are user-defined proof of lemmas.

An encoding of tableaux rules as a theory for SMT solvers has been implemented and tested in mSAT. mSat has also been extended to implement model constructing satisfiability calculus, a variant of SMT solvers in which assignment of variables to values are propagated along with the usual boolean assignment of litterals.

ESTASYS Exploratory Action

5. New Software and Platforms

5.1. The Plasma Statistical Model Checker

Participants: Axel Legay [Coordinator], Sean Sedwards, Benoît Boyer, Louis-Marie Traonouez, Kevin Corre.

5.1.1. PLASMA

Statistical model checking (SMC) is a fast emerging technology for industrial scale verification and optimisation problems. In recognition of this, our group is developing a *P*latform for *L*earning and *A*dvanced *S*tatistical *M*odel checking *A*lgorithms: PLASMA.

PLASMA (see https://project.inria.fr/plasma-lab/) was conceived to have high performance and be extensible, using a proprietary virtual machine [48]. Since SMC requires only an executable semantics and is not constrained by decidability, we can easily implement different modelling languages and logics. Our involvement in the DANSE [0] and DALi [0] European projects has also made us aware of the need to provide efficient verification for externally implemented simulators. We thus devised PLASMA-lab, a modular SMC library that allows external users to tightly integrate their own code with our efficient SMC algorithms and integrated development environment [47]. PLASMA-lab has now been successfully integrated with DESYRE [0], Scilab [0] and MATLAB [0].

The PLASMA-lab architecture is now the basis of our free-standing tool, [0] which includes all the modelling languages, logics and algorithms developed by our group. In particular, we have recently developed cutting edge algorithms for rare events [50], [49], [26], nondeterminism [28], [34], [37] and learning [14], [41].

5.2. Quail

Participants: Axel Legay [Coordinator], Fabrizio Biondi [Coordinator], Jean Quilbeuf.

Privacy is a central for Systems of Systems and interconnected objects. We propose QUAIL, a tool that can be used to quantify privacy of components. QUAIL is the only tool able to perform an arbitrary-precision quantitative analysis of the security of a system depending on private information. Thanks to its Markovian semantics model, QUAIL computes the correlation between the system's observable output and the private information, obtaining the amount of bits of the secret that the attacker will infer by observing the output. QUAIL is open source and can be downloaded at https://project.inria.fr/quail/.

QUAIL is able to evaluate the safety of randomized protocols depending on secret data, allowing to verify a security protocol's effectiveness. QUAIL can also be used to find previously unknown security vulnerabilities in software systems and security protocols. The tool can verify whether a protocol is protecting its secret in a perfect way, and quantify how much the secret is exposed to being revealed otherwise.

QUAIL has been used to quantify whether voting protocols respect the anonymity of the voters, proving that preference ranking voting schemes are more secure than single preference ones. It has also been applied to the security of smart grids and a number of classic examples like dining cryptographers, authentication protocols and grades protocol.

[0] http://www.danse-ip.eu
[0] http://www.ict-dali.eu
[0] http://www.ales.eu.com
[0] http://www.scilab.org
[0] http://www.mathworks.com
[0] https://project.inria.fr/plasma-lab

Since its initial release in 2013, QUAIL's algorithm has been improved employing abstract trace exploration and statistical estimation techniques, making it thousands of times faster than the initial version and outperforming other comparable analysis tools on most use cases.

5.3. PyEcdar

Participants: Axel Legay [Coordinator], Louis-Marie Traonouez [Coordinator].

One of the main difficulties with Systems of Systems is to describe the connection and interactions between the components. We propose PyEcdar as a solution to this problem. PyEcdar (https://project.inria.fr/pyecdar/) is a free software that analyses timed games and timed specifications. The goal of the tool is to allow a fast prototyping of new analysis techniques. It currently allows to solve timed games based on timed automata models. These can be extended with adaptive features to represent dynamicity and to model software product lines.

The tool has been originally developed to analyze the robustness of timed specifications, in extension of the tool Ecdar (http://people.cs.aau.dk/~adavid/ecdar/). As Ecdar, it allows to compose components specifications based on Timed I/O Automata (TIOA), and it implements timed game algorithms for checking consistency and compatibility. Additionally, it features original methods for checking the robustness of these specifications.

The tool has been later extended to analyse adaptive systems. It therefore implements original algorithms for checking featured timed games against requirements expressed in the timed AdaCTL logic.

The tool is written in Python with around 3'000 lines of code. It uses a Python console as user interface, from which it can load TIOA components from XML files written in the UPPAAL format (http://www.uppaal.org/), and design complex system by combining the components using a simple algebra. Then, it can analyze these systems, transform them and save them in a new XML file.

GALLIUM Project-Team

5. New Software and Platforms

5.1. OCaml

Participants: Damien Doligez [correspondant], Alain Frisch [LexiFi], Jacques Garrigue [Nagoya University], Fabrice Le Fessant, Xavier Leroy, Luc Maranget, Gabriel Scherer, Mark Shinwell [Jane Street], Leo White [OCaml Labs, Cambridge University], Jeremy Yallop [OCaml Labs, Cambridge University].

OCaml, formerly known as Objective Caml, is our flagship implementation of the Caml language. From a language standpoint, it extends the core Caml language with a fully-fledged object and class layer, as well as a powerful module system, all joined together by a sound, polymorphic type system featuring type inference. The OCaml system is an industrial-strength implementation of this language, featuring a high-performance native-code compiler for several processor architectures (IA32, AMD64, PowerPC, ARM, ARM64) as well as a bytecode compiler and interactive loop for quick development and portability. The OCaml distribution includes a standard library and a number of programming tools: replay debugger, lexer and parser generators, documentation generator, and compilation manager.

Web site: http://caml.inria.fr/

5.2. CompCert C

Participants: Xavier Leroy [correspondant], Sandrine Blazy [EPI Celtique], Jacques-Henri Jourdan.

The CompCert C verified compiler is a compiler for a large subset of the C programming language that generates code for the PowerPC, ARM and x86 processors. The distinguishing feature of Compcert is that it has been formally verified using the Coq proof assistant: the generated assembly code is formally guaranteed to behave as prescribed by the semantics of the source C code. The subset of C supported is quite large, including all C types except `long double`, all C operators, almost all control structures (the only exception is unstructured `switch`), and the full power of functions (including function pointers and recursive functions but not variadic functions). The generated PowerPC code runs 2–3 times faster than that generated by GCC without optimizations, and only 7% (resp. 12%) slower than GCC at optimization level 1 (resp. 2).

Web site: http://compcert.inria.fr/

5.3. The diy tool suite

Participants: Luc Maranget [correspondant], Jade Alglave [Microsoft Research, Cambridge], Jacques-Pascal Deplaix, Susmit Sarkar [University of St Andrews], Peter Sewell [University of Cambridge].

The **diy** suite provides a set of tools for testing shared memory models: the **litmus** tool for running tests on hardware, various generators for producing tests from concise specifications, and **herd**, a memory model simulator. Tests are small programs written in x86, Power or ARM assembler that can thus be generated from concise specification, run on hardware, or simulated on top of memory models. Test results can be handled and compared using additional tools.

Web site: http://diy.inria.fr/

5.4. Zenon

Participant: Damien Doligez.

Zenon is an automatic theorem prover based on the tableaux method. Given a first-order statement as input, it outputs a fully formal proof in the form of a Coq proof script. It has special rules for efficient handling of equality and arbitrary transitive relations. Although still in the prototype stage, it already gives satisfying results on standard automatic-proving benchmarks.

Zenon is designed to be easy to interface with front-end tools (for example integration in an interactive proof assistant), and also to be easily retargeted to output scripts for different frameworks (for example, Isabelle and Dedukti).

Web site: http://zenon-prover.org/

MARELLE Project-Team

5. New Software and Platforms

5.1. Coq

Participants: Enrico Tassi, Benjamin Grégoire.

Coq is developed mainly in the project-team $\pi.r^2$ with contributions from many other individuals. Enrico Tassi and Benjamin Grégoire are regular contributors. In particular for 2014, Benjamin Grégoire provided advice on connecting virtual machine execution with other aspects of the Coq system and Enrico Tassi worked on a new interactive mode that supports a *document* view of the proof script, with faster user experience. Enrico Tassi also worked on improvements for the use of Coq on Windows.

5.2. Easycrypt

Participants: Gilles Barthe [IMDEA Software Institute], François Dupressoir [IMDEA Software Institute], Benjamin Grégoire [correspondant], César Kunz [IMDEA Software Institute], Benedikt Schmid [IMDEA Software Institute], Pierre-Yves Strub [IMDEA Software Institute].

EasyCrypt is a toolset for reasoning about relational properties of probabilistic computations with adversarial code. Its main application is the construction and verification of game-based cryptographic proofs. EasyCrypt can also be used for reasoning about differential privacy.

5.3. zoocrypt

Participants: Gilles Barthe [IMDEA Software Institute], François Dupressoir [IMDEA Software Institute], Benjamin Grégoire [correspondant], César Kunz [IMDEA Software Institute], Benedikt Schmid [IMDEA Software Institute], Pierre-Yves Strub [IMDEA Software Institute].

ZooCrypt (see http://www.easycrypt.info/zoocrypt/) is an automated tool for analyzing the security of padding-based public-key encryption schemes (i.e. schemes built from trapdoor permutations and hash functions). This years we extended the tool to be able to deal with schemes based on cyclic groups and bilinear maps.

5.4. CoqApprox

Participants: Nicolas Brisebarre [CNRS], Mioara Joldes, Érik Martin-Dorel, Micaela Mayero [Iut de Villetaneuse], Jean-Michel Muller, Ioana Paşca [Iut de Nimes], Laurence Rideau, Laurent Théry [correspondant].

We develop a formalization of rigorous polynomial approximation using Taylor models inside the Coq proof assistant, with a special focus on genericity and efficiency for the computations. In 2014, this library has been included in CoqInterval, distributed by the Toccata research team.

5.5. Ssreflect and Mathematical Components

Participants: Yves Bertot, Cyril Cohen, Laurence Rideau, Enrico Tassi [correspondant], Laurent Théry.

Most of the formal proofs developed in our team are integrated in the Ssreflect extension of the Coq system and the Mathematical Components library. Work this year has concentrated on providing new versions of ssreflect that are compatible with the evolutions of Coq (to prepare for the upcoming release) and integrating our results in the description of real numbers. We also laid the foundations for a book explaining the structure and principles at work in the Math-components library.

MEXICO Project-Team

5. New Software and Platforms

5.1. Software

5.1.1. Software

5.1.1.1. Mole/Cunf: unfolders for Petri Nets

Participant: Stefan Schwoon [correspondant].

Mole computes, given a safe Petri net, a finite prefix of its unfolding. It is designed to be compatible with other tools, such as PEP and the Model-Checking Kit, which are using the resulting unfolding for reachability checking and other analyses. The tool Mole arose out of earlier work on Petri nets. Details on Mole can be found at http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/. Mole served as an experimentation platform for several of our papers in recent years, most recently [33].

In the context of MExICo, we have created a new tool called Cunf [82], which is able to handle contextual nets, i.e. Petri nets with read arcs [80].While in principle every contextual net can be transformed into an equivalent Petri net and then unfolded using Mole, Cunf can take advantage of their special features to do the job faster and produce a smaller unfolding. Cunf has recently been extended with a verification component that takes advantage of these features; More details can be found at http://www.lsv.ens-cachan.fr/~rodrigue/tools/cunf/. Moreover, Cunf has been integrated into the CosyVerif environment (see section 5.2.1). Cunf has also participated in the Model Checking Contest held at the Petri Nets conference in 2013 and 2014.

5.1.1.2. TOURS: Testing On Unfolded Reactive Systems

Participant: Hernán Ponce de León [correspondant].

The MOLE - based testing tool TOURS [42] has been developed in 2014 with the help of intern Konstantinos Athanasiou, jointly supervised by Hernán Ponce de León and Stefan Schwoon of the MExICo team at LSV); it has served successfully to experiment the partial-order based testing methodology on a scalable benchmark example (elevator control).

5.1.1.3. COSMOS : a Statistical Model Checker for the Hybrid Automata Stochastic Logic

Participant: Benoît Barbot [correspondant].

COSMOS is a statistical model checker for the Hybrid Automata Stochastic Logic (HASL). HASL employs Linear Hybrid Automata (LHA), a generalization of Deterministic Timed Automata (DTA), to describe accepting execution paths of a Discrete Event Stochastic Process (DESP), a class of stochastic models which includes, but is not limited to, Markov chains. As a result HASL verification turns out to be a unifying framework where sophisticated temporal reasoning is naturally blended with elaborate reward-based analysis. COSMOS takes as input a DESP (described in terms of a Generalized Stochastic Petri Net), an LHA and an expression Z representing the quantity to be estimated. It returns a confidence interval estimation of Z; recently, it has been equipped with functionalities for rare event analysis. COSMOS is written in C++ and is freely available to the research community.

Details on COSMOS can be found at http://www.lsv.ens-cachan.fr/~barbot/cosmos/

5.2. Platforms

5.2.1. Platform CosyVerif

CosyVerif (http://www.cosyverif.org/) is a platform dedicated to the formal specification and verification of dynamic systems. It allows to specify systems using several formalisms (such as automata and Petri nets), and to run verification tools on these models. CosyVerif integrates several tools, that are mainly developed by researchers of the MeFoSyLoMa group (a Parisian verification group, http://www.mefosyloma.fr/).

The platform is client/server based. The modeler creates models on the client side, either programmatically, or in a dedicated graphical editor. Tools are then executed on the server side.

CosyVerif is available as installable bundles, that embed the client, the server, and also the tools. It is also usable through a public server hosted within the laboratory.

The platform offers a common language for the description of the models, in order to create interoperability between clients and tools. It also provides a way to define easily new formalisms within the platform, and to manipulate models that are instances of these formalisms. To the best of our knowledge, no other verification framework presents such a feature.

CosyVerif targets three different kinds of users:

- Students use this platform in two M2 courses in modeling and verification courses. *Citer les deux cours*
- Tool developers, that are usually researchers, use the platform to distribute their tools, and have a demonstration version easily available. They also use CosyVerif for tutorials in conferences or workshops *Citer Petri nets 2014*.
- Industrial case studies have used the platform since its creation to prove properties on systems in various fields, such as: transportation systems, scheduling, hardware, robotics, databases, banking systems, home automation...

The platform is managed by a steering committee consisting of researchers and engineers of three laboratories: LIP6, LIPN, LSV. This committee decides strategic orientations as well as technical choices.

This year, we have fully redesigned the platform, with two goals in mind: first, to use technologies that target better our users; and second, to provide more functionalities.

- We switched to lightweight web technologies, in order to ease the deployment and use of CosyVerif. For the users, it means that they can access a graphical editor within their web browser. They can also access the platform through an API, usable with any HTTP client.
- We improved the language for formalisms and models in order to allow the modular definition of new formalisms. We switched from a class/instance paradigm to a prototype one, that allows to represent complex models in a both efficient and usable way.
- We extended the server to handle not only executions. It is now primarily a repository of formalismes, models, services and executions, that belong to users or project. It also handles the tools executions, and the collaborative edition of models.
- We started working on a system to help building packages for the various components of the platform (client, server, tools, ...), to ease its installation. It is used to create the bundles of CosyVerif, that are available to download. Another team (Secsi) of the LSV laboratory is interested in this system, and will support its development in 2015.

All the developed software are open source and free software tools.

Two engineers have worked this year on CosyVerif:

- Francis Hulin-Hubard, part-time (CNRS engineer);
- Alban Linard, full-time (Inria engineer).

CosyVerif has been used for teaching in two master programs (Universities Paris 6 and Paris 13/Villetaneuse) It has been used in a tutorial in the Petri Nets 2014 conference.

We are currently in the process of giving a better visibility to the project, by transforming it into a consortium. Our goal is to identify industrial fields where the tools of the platform can be applied successfully, by proposing services to the industry. The strength of the platform relies on the variety of techniques offered by the tools, that adapt to a wide range of problems. In order to increase the number of techniques, we have been joined by another partner from Geneva.

PARSIFAL Project-Team

5. New Software and Platforms

5.1. Abella

Participants: Kaustuv Chaudhuri [correspondant], Matteo Cimini [Indiana University], Dale Miller, Olivier Savary-Bélanger [Princeton University], Mary Southern [University of Minnesota], Yuting Wang [University of Minnesota].

Main web-site: http://abella-prover.org.

Abella is an interactive theorem prover for reasoning about data structures with binding constructs using the λ-tree approach to syntax. It consists of a sophisticated reasoning logic that supports induction, co-induction, and generic reasoning. Abella also supports the *two-level logic approach* by means of a specification logic based on the logic programming language λProlog.

In 2014, the following additions were made to the system.

- A new translation layer was added to Abella's specification layer, which was used to build an interface to the LF dependent type theory [61]. This extension was documented in the following paper: [27]. A number of examples of the use of this new specification language are available at the following URL: http://abella-prover.org/lf
- Two minor releases were made, versions 2.0.2 and 2.0.3, that fixed a number of bugs and added several convenience features. Consult the change log for more details.

Accompanying these additions were the following publications.

- A new comprehensive tutorial for the Abella system has been accept to appear in the *Journal of Formalized Reasoning* [31].
- The new tactical plugin architecture and the dynamic contexts plugin of Abella in the following paper: [26].
- The use of co-induction and higher-order relations to formalize the meta-theory of various bisimulation-up-to techniques for common process calculi: [19].

5.2. Bedwyr

Participants: Quentin Heath, Dale Miller [correspondant].

Main web-site: http://slimmer.gforge.inria.fr/bedwyr/.

Quentin Heath has continued to maintain and enhance this model checking system. In particular, the tabling mechanism has been extended and formalized to a greater extent. The tabling mechanism is now able to use Horn clause lemmas in order to increase the power of the table. For example, given this enhancement it is possible to tell Bedwyr that if a given board position (in some game) has a winning strategy then symmetric versions of that board also have winning strategies. Thus, when a given board position is recognized as winning, then table will understand that all symmetric versions of that board are winning.

Significant energies have also gone into trying to understand how cyclic proofs (recognized using the tabling mechanism) can be turned into certifiable proof evidence. Good results are currently developed for treating bisimulation and non-reachability: in these cases, cyclic proofs are used to supply invariants for induction and co-induction.

5.3. Psyche

Participants: Stéphane Graham-Lengrand [correspondant], Assia Mahboubi, Jean-Marc Notin.

Psyche (*Proof-Search factorY for Collaborative HEuristics*) is a modular proof-search engine whose first version, 1.0, was released in 2012:
http://www.lix.polytechnique.fr/~lengrand/Psyche/

The engine implements the ideas developed in the section "Trustworthy implementations of theorem proving techniques" above, and was the object of the system description [56].

Psyche's proof-search mechanism is simply the incremental construction of proof-trees in the polarized and focused sequent calculus. Its architecture organizes an interaction between a trusted universal kernel and smart plugins that are meant be efficient at solving certain kinds of problems:

The kernel contains the mechanisms for exploring the proof-search space in a sound and complete way, taking into account branching and backtracking. The output of Psyche comes from the (trusted) kernel and is therefore correct by construction. The plugins then drive the kernel by specifying how the branches of the search space should be explored, depending on the kind of problem that is being treated. The quality of the plugin is how fast it drives the kernel towards the final answer.

In 2014, major developments were achieved in Psyche, whose version 2.0 was released on 20th September 2014. It is now equipped with the machinery to handle quantifiers and quantifier-handling techniques. Concretely, it uses meta-variables to delay the instantiation of existential variables, and constraints on meta-variables are propagated through the various branches of the search-space, in a way that allows local backtracking. The kernel, of about 800 l.o.c., is purely functional.

PI.R2 Project-Team

4. New Software and Platforms

4.1. COQ (http://coq.inria.fr)

Participants: Bruno Barras [Inria Saclay], Yves Bertot [Marelle team, Sophia], Pierre Boutillier, Xavier Clerc [SED team], Pierre Courtieu [CNAM], Maxime Dénès [Gallium team, Rocquencourt], Julien Forest [CNAM], Stéphane Glondu [CARAMEL team, Nancy Grand Est], Benjamin Grégoire [Marelle team, Sophia], Vincent Gross [Consultant at NBS Systems], Hugo Herbelin [correspondant], Pierre Letouzey, Assia Mahboubi [SpecFun team, Saclay], Julien Narboux [University of Strasbourg], Jean-Marc Notin [Ecole Polytechnique], Christine Paulin [Toccata team, Saclay], Pierre-Marie Pédrot, Loïc Pottier [Marelle team, Sophia], Matthias Puech, Yann Régis-Gianas, François Ripault, Matthieu Sozeau, Arnaud Spiwack [Mines Paritech], Pierre-Yves Strub [IMDEA, Madrid], Enrico Tassi [Marelle team, Sophia], Benjamin Werner [Ecole Polytechnique].

4.1.1. Version 8.5

Version 8.5 was expected to be released after the summer of 2014, but this got delayed until the Coq Programming Language workshop mid-January 2015.

Coq 8.5 is a major release of the Coq proof assistant, including 5 major new features:

- Parallel development and compilation, inside files and across files, by Enrico Tassi (Inria SpecFun, then Marelle), a result of the Paral-ITP ANR project.
- Availability of all the features of Arnaud Spiwack's new proof engine, with more expressive, clearer semantics, multigoal tactics, deep backtracking,
- A compilation scheme from Coq to OCaml to native code by Maxime Dénès and Benjamin Grégoire (Inria Marelle, then University of Pennsylvania, then Inria Gallium), considerably improving on the previous virtual machine implementation by B. Grégoire.
- A Universe Polymorphic extension by Matthieu Sozeau that allows universe-generic developments, as required by the Homotopy Type Theory library for example,
- Primitive projections for records by Matthieu Sozeau, with significant efficiency improvements.

Coq 8.5 also includes many improvements at different levels: the primitive tactics, the tactic language, the specification language, the tools associated to Coq, etc. For a full list of changes, the reader is invited to look at http://coq.inria.fr or at the files CHANGES of the Coq archive.

4.1.2. Evaluation algorithms

The new unfolding algorithm for global constants that was proposed by Pierre Boutillier is ready for use in Coq 8.5.

4.1.3. Internal representation of projections

A new internal representation of record projections has been implemented in the 8.5 release by Matthieu Sozeau. During the stabilisation of this feature, we added a backwards compatibility layer that allows users to switch seamlessly to the new representation, keeping the same user-level interface for primitive and non-primitive projections (the record types and values being unchanged). This new representation adds eta-conversion of records defined with primitive projections to the definitional equality of Coq, enlarging the set of conversion problems that can be automatically handled by the system.

4.1.4. Universes

The new universe polymorphism system by Matthieu Sozeau is part of the 8.5 release. The implementation has been stabilised, benchmarked and tested heavily in the last year, with much input from the Homotopy Type Theory development team. In [27], Matthieu Sozeau and Nicolas Tabareau presented the system formally. It has since been extended with user-friendly features like named universes and commands to display the status of universe constraints. With the help from Maxime Dénès (Gallium Team), the native compilation system has also been extended to fully support universe polymorphism.

4.1.5. Internal architecture of the Coq software

Pierre Letouzey, Pierre-Marie Pédrot and Xavier Clerc have continued to work at improving the quality of the OCaml code which composes Coq :

- Many modules have been revised, in particular with cleaner naming conventions.
- Almost all uses of the generic OCaml comparison have been chased and transformed into specific code, thereby avoiding many potential bugs with advanced structures, while improving performances at the same time.
- The codes handling OCaml exceptions have been reworked to avoid undue interceptions of critical exceptions.
- Issues involving exceptions are now quite simpler to debug, thanks to easy-to-obtain backtraces.

4.1.6. Efficiency

Pierre-Marie Pédrot has been working on the overall optimisation of Coq, by tracking hotspots in the code. Coq trunk is currently much more efficient than its v8.4 counterpart, and is about as quick as v8.3, while having been expanded with a lot of additional features.

4.1.7. Documentation generation

Yann Régis-Gianas continued the development of a new version of coqdoc, the documentation generator of Coq. This new implementation is based on the interaction protocol with the Coq system and should be more robust with respect to the evolution of Coq.

4.1.8. Maintenance and coordination

The maintenance and coordination of Coq has been jointly done by Hugo Herbelin, Pierre Boutillier, Pierre Letouzey, Matthieu Sozeau, Pierre-Marie Pédrot, in relation with the other participants to the development.

A Coq working group is organised every two months (5 times a year). From the end of October, a Coq lunch holds weekly welcoming any person interested in the development of Coq in general. Discussions about the development happen, in particular, on `coq-dev@inria.fr` and http://coq.inria.fr/bugs.

4.1.9. The Coq extraction

In 2014, Pierre Letouzey built an extension of the Coq extraction that targets directly one of the internal layers of the OCaml compiler. This way, it is possible to avoid the generation of OCaml concrete syntax by the extraction, followed by a parsing phase when the OCaml compiler is launched on the extracted code. Our extension is able to shortcut these two phases. The interest is twofold. First, it seriously reduces the amount of code that should be considered as critical during a program development via extraction. Secondly, with this approach we are able to directly compile and run certain extracted examples, and internalise the result back into Coq, leading to a new promising command Extraction Compute. This extension is currently quite experimental.

4.1.10. Parametricity for the Coq proof assistant

During his stay in the πr^2 team, Marc Lasson developed a plugin for parametricity theory in the Coq proof assistant.

Parametricity theory was originally introduced by John Reynolds in his seminal paper about polymorphic λ-calculus (also known as System F). It is used to formalise the opacity of abstract datatypes in programming languages that provide idioms to handle types generically. Polymorphic functions cannot inspect their arguments with an abstract type, and have to use them uniformly. The main tool of parametricity theory is that of logical relations, which are relations between programs of the same type that are defined by induction on the structure of types.

Marc Lasson's work consisted in developing a parametricity theory for the terms of Coq. The result of this work is a new plugin for the proof assistant that computes logical relations as well as the proof witnesses that programs satisfy these logical relations. It is available on github http://github.com/mlasson/paramcoq.

The purpose of this plugin is to allow to use parametric arguments in Coq proofs, the main direct application is the certification of parametric programs. Thanks to powerful expressiveness of the proof assistant, this plugin will allow future users to use parametric arguments to a larger scale. Although parametricity theory was originally developed for studying programs, the fact that we can use it in a proof assistant enables new uses in other contexts, such as the formalisation of mathematics and the meta-theory of proof assistants).

In [24], Marc Lasson showed that parametricity may also be useful to derive properties about the groupoidal interpretation of Type Theory. It was known that the equality types (also known as identity types) of type theory carry the algebraic structure of ω-groupoids (which is a higher-dimensional version of groups). Parametricity theory allows us to prove that the terms witnessing these algebraic laws are canonical, in the sense that there is only one way to implement them (up to higher-order equalities).

4.1.11. Formalisation in Coq

Hugo Herbelin's type-theoretic construction of semi-simplicial sets [9] has been formalised in Coq.

Matthieu Sozeau and Nicolas Tabareau formalised a setoid model of type theory in Coq http://github.com/mattam82/groupoid. They are working on extending this work to the groupoid model using the latest tools available in Coq 8.5.

Frédéric Loulergue collaborates with Frédéric Dabrowski and Thomas Pinsard (Univ. Orléans) to verify in Coq the compilation pass [21] for a language with nested atomic sections and thread escape to a language with only threads and locks, building on [45].

4.1.12. Systematic development of programs for parallel and cloud computing

During his stay in the πr^2 team, Frédéric Loulergue continues to collaborate with Kento Emoto (Kyushu Institute of Technology), Zhenjiang Hu (National Institute for Informatics, Japan), Julien Tesson (Univ. Paris-Est Créteil), Wadoud Bousdira (Univ. Orléans), Kiminori Matsuzaki (Kochi University of Technology) and Vitor Rodrigues (Rochester Institute of Technology) to develop the SyDPaCC framework (http://traclifo.univ-orleans.fr/SyDPaCC).

The goal of this framework is to ease the systematic development of correct parallel programs, in particular large scale data-intensive applications. In Coq, users write inefficient (sequential) functional programs and through (partly automated) program transformations based on the theory of list homomorphisms [32], bulk synchronous parallel homomorphisms [59] and semi-ring homomorphisms [48], an efficient sequential version is obtained. This version can then be automatically parallelised thanks to type class instance resolution and instances relating specific functions to their parallel counterparts. The parallel versions of the programs are written with a Coq axiomatisation of Bulk Synchronous Parallel ML (BSML) primitives. To obtain the final code, these Coq programs are extracted towards OCaml with calls to a parallel implementation of the BSML library.

As the SyDPaCC framework currently mixes certified code extracted from Coq and unverified code, Frédéric Loulergue and Pierre Letouzey are working on an extended extraction that generates, when possible, OCaml asserts for preconditions on function arguments. The next version of the generate-test-aggregate library of SyDPaCC will use Marc Lasson's plugin for parametricity to prove a "theorem for free": currently only instantiations of this theorem for each provided generator are proved.

4.1.13. Proofs of algorithms on graphs

Jean-Jacques Lévy's current research is to review basic algorithms and make their formal proofs of correctness in Why3 + Coq. Filliâtre and Pottier already started this research, but we plan to focus on graph algorithms, with concerns on the feasability of these formal proofs and on the design of good libraries on top of Coq or Ssreflect. The goal is not to disprove these algorithms which are most probably correct, but to develop a theory of tools for proving algorithms with proof assistants and provers. Standard techniques use assertions in Hoare logic or TLA or any other logic, which are written on paper. With the recent development of good computer proof-assistants and the fantastic progress of SMT provers, the goal of providing algorithms with their correctness proofs checked by computer seems possible. The plan of this research is to use Why3, Coq, Ssreflect on standard computing systems, and also to motivate a few students to work on this project. The challenge would be to compete with Filliâtre, Pottier and Monate's group at CEA (France), or Fournet, Swamy and Pierce at Microsoft Research or Univ. of Pennsylvania. We want to demonstrate that the use of SMT provers can be well coupled with the one of interactive provers as already done in Why3 and in F* with refined types in probable future. The expected outcome would be to extend to larger programs and real software. But this seems quite ambitious at present time, since large scale needs more technology as showed by Gonthier for his long proofs of mathematical theorems, and since the world of programming is much less structured than the world of mathematics.

We completed proofs of the following major algorithms as exposed in Sedgewick's book: sorting, searching, depth-first search in graphs. This work is performed in collaboration with Chen Ran at Iscas (Institute of Software, Chinese Academy of Sciences). Proofs can be found at http://jeanjacqueslevy.net/why3 (see also [10]).

4.2. Other software developments

In collaboration with François Pottier (Inria Gallium), Yann Régis-Gianas maintained Menhir, an LR parser generator for OCaml.

Yann Régis-Gianas has been developing the "Hacking Dojo", with the help of Alexandre Ly (master student of Paris Diderot). a web platform to automatically grade programming exercises. The platform is now used in several courses of the University Paris Diderot.

In collaboration with Grégoire Duchêne (master student at Paris Diderot), Yann Régis-Gianas developed Tamasheq, a fully-customisable interpreter for the OCaml programming language. Users of this interpreter can write plugins to instrument the interpretation of an OCaml program with visualisation, interactive debugging or logging. A paper is in preparation.

Yves Guiraud has updated the Catex tool for Latex, whose purpose is to automate the production of string diagrams from algebraic expressions (http://www.pps.univ-paris-diderot.fr/~guiraud/catex/catex.zip).

Yves Guiraud has developed the Python library Cox for the computation of coherent presentations of Artin monoids, after [18] (http://www.pps.univ-paris-diderot.fr/~guiraud/cox/cox.zip).

Yves Guiraud collaborates with Samuel Mimram (LIX) to develop the prototype Rewr that implements the homotopical completion-reduction procedure of [6] (http://www.pps.univ-paris-diderot.fr/~smimram/rewr).

Eric Finster has developed a new proof assistant, called Orchard, which aims to pursue the emerging connections between type theory and higher category theory by providing an environment in which to explicitly manipulate higher categorical diagrams using a notation based on a collection of shapes called opetopes. Opetopes have strong connections to concepts from computer science: they have a natural interpretation as a series of canonical indexed inductive types, and thus can be implemented and reasoned about using standard techniques from functional programming. The goal of the Orchard project is to forge links between the homotopical ideas of homotopy type theory, and the higher categorical ideas coming from higher-dimensional rewriting theory by providing a common language in which to reason about both. A preliminary implementation is available at https://github.com/ericfinster/orchard.

SUMO Project-Team

5. New Software and Platforms

5.1. Sigali

Participants: Hervé Marchand, Nicolas Berthier.

Sigali is a model-checking tool that operates on ILTS (Implicit Labeled Transition Systems, an equational representation of an automaton), an intermediate model for discrete event systems. It offers functionalities for verification of reactive systems and discrete controller synthesis. It is developed jointly by the TEA and SUMO teams. The techniques used consist in manipulating the system of equations instead of the set of solutions, which avoids the enumeration of the state space. Each set of states is uniquely characterized by a predicate and the operations on sets can be equivalently performed on the associated predicates. Therefore, a wide spectrum of properties, such as liveness, invariance, reachability and attractivity, can be checked. Algorithms for the computation of predicates on states are also available. Sigali is connected with the Polychrony environment (Tea project-team) as well as the Matou environment (VERIMAG), thus allowing the modeling of reactive systems by means of Signal Specification or Mode Automata and the visualization of the synthesized controller by an interactive simulation of the controlled system. Sigali is registered at APP under the identification number IDDN.FR.001.370006.S.P.1999.000.10600.

Sigali is also integrated as part of the compiler of the language BZR (web site).

We are currently developing a new version of Sigali that will be able to handle numerical variables.

5.2. Tipex

Participants: Thierry Jéron, Hervé Marchand, Srinivas Pinisetty.

We are implementing a prototype tool named Tipex (TImed Properties Enforcement during eXecution) for the enforcement of timed properties, in collaboration with Ylies Falcone (LIG, Grenoble). Tipex is based on the theory and algorithms that we develop for the synthesis of enforcement monitors for properties specified by timed automata (TA). The prototype is developped in python, and uses the PyUPPAAL and DBMpyuppaal libraries of the UPPAAL tool. It is currently restricted to safety and co-safety timed property. The property provided as input to the tool is a TA that can be specified using the UPPAAL tool, and is stored in XML format. The tool synthesizes an enforcement monitor from this TA, which can then be used to enforce a sequence of timed events to satisfy the property. Experiments have been conducted on a set of case studies. This allowed to validate the architecture and feasibility of enforcement monitoring in a timed setting and to have a first assessment of performance (and to what extent the overhead induced by monitoring is negligible).

5.3. DAXML

Participant: Loïc Hélouët.

DAXML is an implementation of Distributed Active Documents, a formalism for data centric design of Web Services proposed by Serge Abiteboul. This implementation is based on a REST framework, and can run on a network of machines connected to internet and equipped with JAVA. This implementation was realized during the post doc of Benoit Masson in 2011. A demo of the software is available at this web page. This year, the source code of DAXML has been submitted at the APP, and a distribution with free ad-hoc licence will follow in 2015.

TEMPO Team

5. New Software and Platforms

5.1. SimSoC

We have continued to work on the SimSoC virtual prototyping framework distributed by Inria. Because of issues in the design of the Power Architecture simulator, we did a redesign of the Power simulator and a new implementation, so that we can simulate in the future both the Power Classic and Power Extended architectures in both 32 bits or 64 bits. We also contributed new extensions as described below.

TOCCATA Project-Team

5. New Software and Platforms

5.1. The Why3 system

Participants: Jean-Christophe Filliâtre [contact], Claude Marché, Guillaume Melquiond, Andrei Paskevich.

Criteria for Software Self-Assessment: A-3-up, SO-4, SM-4, EM-4, SDL-5, OC-4.[0]

Why3 is the next generation of *Why*. *Why3* clearly separates the purely logical specification part from generation of verification conditions for programs. It features a rich library of proof task transformations that can be chained to produce a suitable input for a large set of theorem provers, including SMT solvers, TPTP provers, as well as interactive proof assistants.

It is distributed as open source, under GPL license, at http://why3.lri.fr/. It is also distributed as part of major Linux distributions and in the OPAM packaging system http://opam.ocaml.org/packages/why3/why3.0.85/.

Why3 is used as back-end of our own tools *Krakatoa* and *Jessie*, but also as back-end of the GNATprove tool (Adacore company), and of the WP plugin of *Frama-C*. *Why3* has been used to develop and prove a significant part of the programs of our team gallery http://proval.lri.fr/gallery/index.en.html, and used for teaching (e.g., at the Master Parisien de Recherche en Informatique).

Why3 is used by other academic research groups, e.g. within the CertiCrypt/EasyCrypt project (http://easycrypt.gforge.inria.fr/) for certifying cryptographic programs. The *Why3* web site http://why3.lri.fr lists a few other works done by external researchers and relying on the use of *Why3*.

Two versions were released in 2014: 0.83 released in March and 0.84 in September, plus a few days later a bugfix version 0.85.

5.2. The Alt-Ergo theorem prover

Participants: Sylvain Conchon [contact], Évelyne Contejean, Alain Mebsout, Mohamed Iguernelala.

Criteria for Software Self-Assessment: A-3-up, SO-4, SM-4-up, EM-4, SDL-5, OC-4.

Alt-Ergo is an automated proof engine, dedicated to program verification, whose development started in 2006. It is fully integrated in the program verification tool chain developed in our team. It solves goals that are directly written ina *Why*'s annotation language; this means that *Alt-Ergo* fully supports first order polymorphic logic with quantifiers. *Alt-Ergo* also supports the standard [116] defined by the SMT-lib initiative.

It is currently used in our team to prove correctness of C and Java programs as part of the *Why* platform and the new *Why3* system. It is used as back-end prover in the environments Frama-C and CAVEAT for static analysis of C developed at CEA. In this context, Alt-Ergo has been qualified by Airbus and is integrated in the next generation of Airbus development process. Alt-Ergo is usable as a back-end prover in the SPARK verifier for ADA programs, since Oct 2010, and is also the main back-end prover of the new SPARK2014.

Alt-Ergo is integrated in several other tools and platforms: the Bware platform for discharging VCs generated by Atelier B, the EasyCrypt environment for verifying cryptographic protocols, the Pangolin programming language http://code.google.com/p/pangolin-programming-language/, etc.

Last but not least, Alt-Ergo is the solver used by the Cubicle model checker described below.

[0] self-evaluation following the guidelines (http://www.inria.fr/content/download/11783/409665/version/4/file/SoftwareCriteria-V2-CE.pdf) of the Software Working Group of Inria Evaluation Committee (http://www.inria.fr/institut/organisation/instances/commission-d-evaluation)

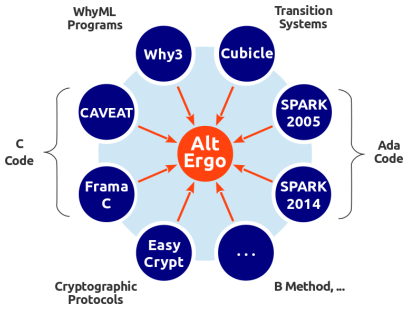

Figure 2.

Alt-Ergo is distributed as open source, under the CeCILL-C license, at URL http://alt-ergo.lri.fr/, and in the OPAM packaging system http://opam.ocaml.org/packages/alt-ergo/alt-ergo.0.95.2/. Latest public version is 0.99.1, released in Dec. 2014. Maintenance is done by the OcamlPro company http://alt-ergo.ocamlpro.com/.

5.3. The Cubicle model checker modulo theories

Participants: Sylvain Conchon [contact], Alain Mebsout.

Partners: A. Goel, S. Krstić (Intel Strategic Cad Labs in Hillsboro, OR, USA), F. Zaïdi (LRI, Université Paris-sud)

Cubicle is an open source model checker for verifying safety properties of array-based systems, which corresponds to a syntactically restricted class of parametrized transition systems with states represented as arrays indexed by an arbitrary number of processes. Cache coherence protocols and mutual exclusion algorithms are typical examples of such systems.

Cubicle model-checks by a symbolic backward-reachability analysis on infinite sets of states represented by specific simple formulas, called cubes. Cubicle is based on ideas introduced by MCMT (http://users.mat.unimi.it/users/ghilardi/mcmt/) from which, in addition to revealing the implementation details, it differs in a more friendly input language and a concurrent architecture. Cubicle is written in OCaml. Its SMT solver is a tightly integrated, lightweight and enhanced version of *Alt-Ergo*; and its parallel implementation relies on the Functory library.

Cubicle is distributed as open source, under the Apache license, at URL http://cubicle.lri.fr/, and in the OPAM packaging system http://opam.ocaml.org/packages/cubicle/cubicle.1.0.1/. Latest version is 1.0.1, released in Nov. 2014.

5.4. The Flocq library

Participants: Sylvie Boldo [contact], Guillaume Melquiond.

Criteria for Software Self-Assessment: A-2, SO-3, SM-3, EM-3, SDL-5, OC-4.

The Flocq library for the *Coq* proof assistant is a comprehensive formalization of floating-point arithmetic: core definitions, axiomatic and computational rounding operations, high-level properties [6]. It provides a framework for developers to formally certify numerical applications.

Flocq is currently used by the CompCert certified compiler for its support of floating-point computations.

It is distributed as open source, under a LGPL license, at http://flocq.gforge.inria.fr/. It was first released in 2010. Current version is 2.4.0 released in Sep. 2014.

5.5. The Gappa tool

Participant: Guillaume Melquiond [contact].

Criteria for Software Self-Assessment: A-3, SO-4, SM-4, EM-3, SDL-5, OC-4.

Given a logical property involving interval enclosures of mathematical expressions, Gappa tries to verify this property and generates a formal proof of its validity. This formal proof can be machine-checked by an independent tool like the *Coq* proof-checker, so as to reach a high level of confidence in the certification [82], [123].

Since these mathematical expressions can contain rounding operators in addition to usual arithmetic operators, Gappa is especially well suited to prove properties that arise when certifying a numerical application, be it floating-point or fixed-point. Gappa makes it easy to compute ranges of variables and bounds on absolute or relative roundoff errors.

Gappa is being used to certify parts of the mathematical libraries of several projects, including CRlibm, FLIP, and CGAL. It is distributed as open source, under a Cecill-B / GPL dual-license, at http://gappa.gforge.inria.fr/. Latest version is 1.1.2 released in October 2014.

Part of the work on this tool was done while in the Arénaire team (Inria Rhône-Alpes), until 2008.

5.6. The Interval package for Coq

Participants: Guillaume Melquiond [contact], Érik Martin-Dorel.

Criteria for Software Self-Assessment: A-3, SO-4, SM-3, EM-3, SDL-4, OC-4.

The Interval package provides several tactics for helping a *Coq* user to prove theorems on enclosures of real-valued expressions. The proofs are performed by an interval kernel which relies on a computable formalization of floating-point arithmetic in *Coq*.

Versions 1.0 and 2.0 were released in 2014. Version 2.0 integrates the CoqApprox library for computing Taylor models, so as to greatly improve performances when bounding univariate expressions [43].

It is distributed as open source, under a CeCILL-C license, at http://coq-interval.gforge.inria.fr/. Latest version is 2.0 released in November 2014.

Part of the work on this library was done while in the Mathematical Components team (Microsoft Research–Inria Joint Research Center).

5.7. The Coquelicot library for real analysis

Participants: Sylvie Boldo [contact], Catherine Lelay, Guillaume Melquiond.

Criteria for Software Self-Assessment: A-3, SO-4, SM-2, EM-3, SDL-4, OC-4.

The Coquelicot library is designed with three principles in mind. The first is the user-friendliness, achieved by implementing methods of automation, but also by avoiding dependent types in order to ease the stating and readability of theorems. This latter part was achieved by defining total function for basic operators, such as limits or integrals. The second principle is the comprehensiveness of the library. By experimenting on several applications, we ensured that the available theorems are enough to cover most cases. We also wanted to be able to extend our library towards more generic settings, such as complex analysis or Euclidean spaces. The third principle is for the Coquelicot library to be a conservative extension of the Coq standard library, so that it can be easily combined with existing developments based on the standard library. Moreover, we achieved this compatibility without adding any additional axiom.

The result is the Coquelicot library available at http://coquelicot.saclay.inria.fr. Latest version is 2.0.1 released in March 2014. It contains about 1,700 theorems and 37,000 lines of Coq.

5.8. The CFML tool for verifying OCaml code

Participant: Arthur Charguéraud [contact].

Criteria for Software Self-Assessment: A-2, SO-4, SM-2, EM-3, SDL-1, OC-4.

The *CFML* tool supports the verification of *OCaml* programs through interactive *Coq* proofs. *CFML* proofs establish the full functional correctness of the code with respect to a specification. They may also be used to formally establish bounds on the asymptotic complexity of the code. The tool is made of two parts: on the one hand, a characteristic formula generator implemented as an *OCaml* program that parses *OCaml* code and produces *Coq* formulae; and, on the other hand, a *Coq* library that provides notation and tactics for manipulating characteristic formulae interactively in *Coq*.

CFML is distributed under the LGPL license, and is available at http://arthur.chargueraud.org/softs/cfml/. It has been continuously extended since its first release in 2010. In particular, in 2014 support for the verification of asymptotic complexity bounds has been added.

5.9. Other Maintained Tools

5.9.1. The ALEA library for randomized algorithms

Participant: Christine Paulin-Mohring [contact].

Criteria for Software Self-Assessment: A-2, SO-3, SM-2, EM-3, SDL-4, OC-4.

The ALEA library is a *Coq* development for modeling randomized functional programs as distributions using a monadic transformation. It contains an axiomatisation of the real interval $[0, 1]$ and its extension to positive real numbers. It introduces definition of distributions and general rules for approximating the probability that a program satisfies a given property.

ALEA is used as a basis of the Certicrypt environment (MSR-Inria joint research center, Imdea Madrid, Inria Sophia-Antipolis) for formal proofs for computational cryptography [55]. It is also experimented in LABRI as a basis to study formal proofs of probabilistic distributed algorithms.

ALEA is distributed as open source, at http://www.lri.fr/~paulin/ALEA. Latest version is 8 released in May 2013. In particular, it includes a module to reason about random variables with values in positive real numbers.

5.9.2. Bibtex2html

Participants: Jean-Christophe Filliâtre [contact], Claude Marché.

Criteria for Software Self-Assessment: A-5, SO-3, SM-3, EM-3, SDL-5, OC-4.

Bibtex2html is a generator of HTML pages of bibliographic references. Distributed as open source since 1997, under the GPL license, at http://www.lri.fr/~filliatr/bibtex2html/. Latest version is 1.98 released in July 2014. Bibtex2html is also distributed as a package in most Linux distributions, and in the OPAM packaging system http://opam.ocaml.org/packages/bibtex2html/bibtex2html.1.98/.

We estimate that between 10000 and 100000 web pages have been generated using Bibtex2html.

5.9.3. The Coccinelle library for term rewriting

Participant: Évelyne Contejean [contact].

Criteria for Software Self-Assessment: A-2, SO-3, SM-2, EM-2, SLD-2, OC-4.

Coccinelle is a *Coq* library for term rewriting. Besides the usual definitions and theorems of term algebras, term rewriting and term ordering, it also models a number of algorithms implemented in the CiME toolbox, such as matching, matching modulo associativity-commutativity, computation of the one-step reducts of a term, recursive path ordering (RPO) comparison between two terms, etc. The RPO algorithm can effectively be run inside *Coq*, and is used in the Color developement (http://color.inria.fr/) as well as for certifying Spike implicit induction theorems in *Coq* (Sorin Stratulat).

Coccinelle is available at http://www.lri.fr/~contejea/Coccinelle, and is distributed under the Cecill-C license.

5.9.4. OCamlgraph

Participants: Jean-Christophe Filliâtre [contact], Sylvain Conchon.

OCamlgraph is a graph library for *OCaml*. It features many graph data structures, together with many graph algorithms. Data structures and algorithms are provided independently of each other, thanks to *OCaml* module system. OCamlgraph is distributed as open source, under the LGPL license, at http://OCamlgraph.lri.fr/. Latest version is 1.8.5, released in March 2014. It is also distributed as a package in several Linux distributions. OCamlgraph is now widely spread among the community of *OCaml* developers, and available as an OPAM package http://opam.ocaml.org/packages/ocamlgraph/ocamlgraph.1.8.5/.

5.9.5. Mlpost

Participant: Jean-Christophe Filliâtre [contact].

Mlpost is a tool to draw scientific figures to be integrated in LaTeX documents. Contrary to other tools such as TikZ or MetaPost, it does not introduce a new programming language; it is instead designed as a library of an existing programming language, namely *OCaml*. Yet it is based on MetaPost internally and thus provides high-quality PostScript figures and powerful features such as intersection points or clipping. Mlpost is distributed as open source, under the LGPL license, at http://mlpost.lri.fr/. Mlpost was presented at JFLA'09 [52].

Mlpost is available as an OPAM package http://opam.ocaml.org/packages/mlpost/mlpost.0.8.1/.

5.9.6. Functory

Participant: Jean-Christophe Filliâtre [contact].

Functory is a distributed computing library for *OCaml*. The main features of this library include (1) a polymorphic API, (2) several implementations to adapt to different deployment scenarios such as sequential, multi-core or network, and (3) a reliable fault-tolerance mechanism. Functory was presented at JFLA 2011 [91] and at TFP 2011 [90].

Functory is distributed as open source, under the LGPL license, at http://functory.lri.fr/, and in the OPAM packaging system http://opam.ocaml.org/packages/functory/functory.0.5/. Latest version is 0.5, release in March 2013.

5.9.7. The Why Environment

Participants: Claude Marché [contact], Jean-Christophe Filliâtre, Guillaume Melquiond, Andrei Paskevich.

Criteria for Software Self-Assessment: A-3, SO-4, SM-3, EM-2, SDL-5-down, OC-4.

The *Why* platform is a set of tools for deductive verification of Java and C source code. In both cases, the requirements are specified as annotations in the source, in a special style of comments. For Java (and Java Card), these specifications are given in JML and are interpreted by the *Krakatoa* tool. Analysis of C code must be done using the external *Frama-C* environment, and its *Jessie* plugin which is distributed in *Why*.

The platform is distributed as open source, under GPL license, at http://why.lri.fr/.

It also distributed as part of major Linux distributions and in the OPAM packaging system http://opam.ocaml.org/packages/why/why.2.34/. Version 2.34 was released in August 2014, to provide a version compatible with both *Frama-C*Neon and *Why3* 0.83.

The internal VC generator and the translators to external provers are no longer under active development, as superseded by the *Why3*system described above. The *Krakatoa* and *Jessie* front-ends are still maintained, although using now by default the *Why3* VC generator. These front-ends are described in a specific web page http://krakatoa.lri.fr/. They are used for teaching (University of Evry, École Polytechnique, etc.), used by several research groups in the world, e.g at Fraunhofer Institute in Berlin [92], at Universidade do Minho in Portugal [50], at Moscow State University, Russia (http://journal.ub.tu-berlin.de/eceasst/article/view/255).

VERIDIS Project-Team

5. New Software and Platforms

5.1. The veriT Solver

Participants: Haniel Barbosa, David Déharbe, Pablo Federico Dobal, Pascal Fontaine [contact].

The veriT solver is an SMT (Satisfiability Modulo Theories) solver developed in cooperation with David Déharbe from the Federal University of Rio Grande do Norte in Natal, Brazil. The solver can handle large quantifier-free formulas containing uninterpreted predicates and functions, and arithmetic over integers and reals. It features a very efficient decision procedure for uninterpreted symbols, as well as a simplex-based reasoner for linear arithmetic. It also has some support for user-defined theories, quantifiers, and lambda-expressions. This allows users to easily express properties about concepts involving sets, relations, etc. The prover can produce explicit proof traces when it is used as a decision procedure for quantifier-free formulas with uninterpreted symbols and arithmetic. To support the development of the tool, non-regression tests use Inria's grid infrastructure; it allows us to extensively test the solver on thousands of benchmarks in a few minutes. The veriT solver is available as open source under the BSD license at the veriT Web site.

Efforts in 2014 have been focused on efficiency and stability. The decision procedures for uninterpreted symbols and linear arithmetic have been further improved. There has also been some progress in the integration of the solver Redlog (section 5.4) for non-linear arithmetic in the context of the SMArT project (section 8.2).

The veriT solver participated in the SMT competition SMT-COMP 2014, part of the Vienna Summer Of Logic Olympic Games, and received the gold medal for SMT. The success of the different solvers was measured as a combination of the number of benchmark problems solved in the various categories, the number of erroneous answers, and the time taken.

We target applications where validation of formulas is crucial, such as the validation of TLA$^+$ and B specifications, and work together with the developers of the respective verification platforms to make veriT even more useful in practice. The solver is available as a plugin for the Rodin platform for discharging proof obligations generated in Event-B [50]; on a large repository of industrial and academic cases, this SMT-based plugin decreased by 75% the number of proof obligations requiring human interactions, compared to the original B prover.

5.2. The TLA+ Proof System

Participants: Stephan Merz [contact], Hernán Pablo Vanzetto.

TLAPS, the TLA$^+$ proof system developed at the Joint MSR-Inria Centre, is a platform for developing and mechanically verifying proofs about TLA$^+$ specifications. The TLA$^+$ proof language is hierarchical and explicit, allowing a user to decompose the overall proof into independent proof steps. TLAPS consists of a *proof manager* that interprets the proof language and generates a collection of proof obligations that are sent to *backend verifiers*. The current backends include the tableau-based prover Zenon for first-order logic, Isabelle/TLA$^+$, an encoding of TLA$^+$ as an object logic in the logical framework Isabelle, an SMT backend designed for use with any SMT-lib compatible solver, and an interface to a decision procedure for propositional temporal logic.

The current version 1.3.2 of TLAPS was released in May 2014, it is distributed under a BSD-like license at http://tla.msr-inria.inria.fr/tlaps/content/Home.html. The prover fully handles the non-temporal part of TLA$^+$. The SMT backend, developed in Nancy, has been further improved in 2014, in particular through the development of an appropriate type synthesis procedure, and is now the default backend. A new interface with a decision procedure for propositional temporal logic has been developed in 2014, so that simple temporal proof obligations can now be discharged. It is based on a technique for "coalescing" first-order subformulas of temporal logic, described in section 6.2 . The standard proof library has also been further developed, partly in response to the needs of the ADN4SE project on verifying a real-time micro-kernel system (section 7.2).

TLAPS was presented at tutorials at the TLA$^+$ community event organized during ABZ 2014 in Toulouse in June and at the SPES_XT summer school at the University of Twente (The Netherlands) in September.

5.3. SPASS: An Automated Theorem Prover for First-Order Logic With Equality

Participants: Martin Bromberger, Arnaud Fietzke, Thomas Sturm, Marco Voigt, Uwe Waldmann, Christoph Weidenbach [contact].

SPASS is an automated theorem prover based on superposition that handles first-order logic with equality and several extensions for particular classes of theories. It has been developed since the mid-1990s at the Max-Planck Institut für Informatik in Saarbrücken. Version 3.7 is the current stable release; it is distributed under the FreeBSD license at http://www.spass-prover.org.

The next major release of SPASS will mainly focus on improved theory support: many applications of automated deduction require reasoning in first-order logic modulo background theories, in particular some form of arithmetic. In 2014, we have continued our efforts to improve the superposition calculus as well as to develop dedicated arithmetic decision procedures for various arithmetic theories. Our results are:

- specialized reasoning support for finite subsets,
- specialized decision procedures for linear real arithmetic with one quantifier alternation,
- new efficient and complete procedures for (mixed) linear integer arithmetic,
- decidability results and respective procedures for various combinations of linear arithmetic with first-order logic.

5.4. The Redlog Computer Logic System

Participants: Thomas Sturm [contact], Marek Košta.

Redlog is an integral part of the interactive computer algebra system Reduce. It supplements Reduce's comprehensive collection of powerful methods from symbolic computation by supplying more than 100 functions on first-order formulas. Redlog has been publicly available since 1995 and is constantly being improved. The name Redlog stands for Reduce Logic System. Andreas Dolzmann from Schloss Dagstuhl Leibniz-Zentrum is a co-developer of Redlog.

Reduce and Redlog are open-source and freely available under a modified BSD license at http://reduce-algebra.sourceforge.net/. The Redlog homepage is located at http://www.redlog.eu/. Redlog generally works with interpreted first-order logic in contrast to free first-order logic. Each first-order formula in Redlog must exclusively contain atoms from one particular Redlog-supported theory, which corresponds to a choice of admissible functions and relations with fixed semantics. Redlog-supported theories include Nonlinear Real Arithmetic (Real Closed Fields), Presburger Arithmetic, Parametric QSAT, and many more.

Effective quantifier elimination procedures for the various supported theories establish an important class of methods available in Redlog. For the theories supported by Redlog, quantifier elimination procedures immediately yield decision procedures. Besides these quantifier elimination-based decision methods there are specialized, and partly incomplete, decision methods, which are tailored to input from particular fields of application.

In 2014, Redlog made two important steps into distinct but equally important future directions. On the one hand, it integrated for the first time learning strategies, as they are known from CDCL-based SMT solving, into a classical real quantifier elimination procedure, viz. virtual substitution for linear formulas [28]. On the other hand, there was important progress concerning incomplete decision procedures for the reals. A journal submission currently under review describes identification of a Hopf bifurcation for the important MAPK model within less than a minute. The corresponding polynomial relevant for root-finding has dimension 10, total degree 100, and contains more than 850,000 monomials.

Redlog is a widely accepted tool and highly visible in mathematics, informatics, engineering and the sciences. The seminal article on Redlog [4] has received more than 300 citations in the scientific literature so far.

CARTE Project-Team

5. New Software and Platforms

5.1. Morphus/MMDEX

MMDEX is a virus detector based on morphological analysis. It is composed of our own disassembler tool, on a graph transformer and a specific tree-automaton implementation. The tool is used in the EU-Fiware project and by some other partners (e.g., DAVFI project).
Written in C, 20k lines.
APP License, IDDN.FR.001.300033.000.R.P.2009.000.10000, 2009.

5.2. DynamicTracer

DynamicTracer is a new tool with a public web interface which provides run trace of executable files. The trace is obtained by recording a dynamic execution in a safe environment. The trace contain instruction addresses, instruction opcodes and other optional informations.
Written in C++, 2.5k lines.
http://www.lhs.loria.fr/wp/?page_id=96

5.3. CoDisasm

Codisasm is a new disassembly program which support self-modifying code and code overlapping. Up to our knowledge, this is the first to cope both aspects of program obfuscation. The tool is based on the notion of wave developed in the group.
Written in C, 3k lines.

CASSIS Project-Team

5. New Software and Platforms

5.1. Protocol Verification Tools

Participants: Véronique Cortier, Stéphane Glondu, Pierre-Cyrille Héam, Olga Kouchnarenko, Steve Kremer, Michaël Rusinowitch, Mathieu Turuani, Laurent Vigneron.

5.1.1. CL-AtSe

We develop *CL-AtSe*, a Constraint Logic based Attack Searcher for cryptographic protocols, initiated and continued by the European projects *AVISPA*, AVANTSSAR (for web-services) and Nessos respectively. The *CL-AtSe* approach to verification consists in a symbolic state exploration of the protocol execution for a bounded number of sessions, thus is both correct and complete. *CL-AtSe* includes a proper handling of sets, lists, choice points, specification of any attack states through a language for expressing e.g., secrecy, authentication, fairness, or non-abuse freeness, advanced protocol simplifications and optimizations to reduce the problem complexity, and protocol analysis modulo the algebraic properties of cryptographic operators such as XOR (exclusive or) and Exp (modular exponentiation).

CL-AtSe has been successfully used to analyse protocols from e.g., France Telecom R&D, Siemens AG, IETF, Gemalto, Electrum in funded projects. It is also employed by external users, e.g., from the AVISPA's community. Moreover, *CL-AtSe* achieves good analysis times, comparable and sometimes better than other state-of-the art tools.

CL-AtSe has been enhanced in various ways. It fully supports the Aslan semantics designed in the context of the AVANTSSAR project, including Horn clauses (for intruder-independent deductions, e.g., for credential management), and a large fragment of LTL-based security properties. A Bugzilla server collects bug reports, and online analysis and orchestration are available on our team server (https://cassis.loria.fr). Large models can be analysed on the TALC Cluster in Nancy with parallel processing. *CL-AtSe* also supports negative constraints on the intruder's knowledge, which reduces drastically the orchestrator's processing times and allows separation of duties and non-disclosure policies, as well as conditional security properties, like: i) an authentication to be verified iff some session key is safe; ii) relying on a leaking condition on some private data instead of an honesty predicate to trigger or block some agent's property. This was crucial for e.g., the Electrum's wallet where all clients can be dishonest but security guarantees must be preserved anyway.

5.1.2. Akiss

Akiss (Active Knowledge in Security Protocols) is a tool for verifying indistinguishability properties in cryptographic protocols, modelled as trace equivalence in a process calculus. Indistinguishability is used to model a variety of properties including anonymity properties, strong versions of confidentiality and resistance against offline guessing attacks, etc. *Akiss* implements a procedure to verify equivalence properties for a bounded number of sessions based on a fully abstract modelling of the traces of a bounded number of sessions of the protocols into first-order Horn clauses and a dedicated resolution procedure. The procedure can handle a large set of cryptographic primitives, namely those that can be modeled by an optimally reducing convergent rewrite system.

Recent developments include the possibility for checking everlasting indistinguishability properties [72]. This feature was added when analyzing everlasting privacy properties in electronic voting protocols. The tool is still under active development, including optimisations to improve efficiency, but also the addition of new features, such as the possibility to model protocols using weak secrets.

The *Akiss* tool is freely available at https://github.com/glondu/akiss.

5.1.3. Belenios

In collaboration with the Caramel project-team, we develop an open-source private and verifiable electronic voting protocol, named *Belenios*. Our system is an evolution and a new implementation of an existing system, Helios, developed by Ben Adida, and used e.g., by UCL and the IACR association in real elections. The main differences with Helios are a cryptographic protection against ballot stuffing and a practical threshold decryption system that allows to split the decryption key among several authorities, k out of n authorities being sufficient to decrypt. We will continue to add new cryptographic and protocol improvements to offer a secure, proved, and practical electronic voting system.

Belenios has been implemented (cf. http://belenios.gforge.inria.fr) by Stéphane Glondu and has been tested in December 2014 "in real conditions", in a test election involving the members of Inria Nancy-Grand Est center and of the Loria lab (more than 500 potential voters) that had to elect the best pictures of the Loria.

5.1.4. SAPIC

SAPIC is a tool that translates protocols from a high-level protocol description language akin to the applied pi calculus into multiset rewrite rules, that can then be be analysed using the Tamarin Prover.

Its aim is the analysis of protocols that include states, for example Hardware Security Tokens communicating with a possibly malicious user, or protocols that rely on databases. It has been succesfully applied on several case studies including the Yubikey authentication protocol.

A recent extension, *SAPIC** extends SAPIC by a Kleene star operator (*) which allows to iterate a process a finite but arbitrary number of times. This construction is useful to specify for instance stream authentication protocols. We used it to analyse a simple version of the TESLA protocol.

The *SAPIC* tool is freely available at http://sapic.gforge.inria.fr/.

5.2. Testing Tools

Participants: Fabrice Bouquet, Frédéric Dadeau, Kalou Cabrera, Ivan Enderlin.

5.2.1. Hydra

Hydra is an Eclipse-like platform, based on Plug-ins architecture. Plug-ins can be of five kinds: *parser* is used to analyze source files and build an intermediate format representation of the source; *translator* is used to translate from a format to another or to a specific file; *service* denotes the application itself, i.e., the interface with the user; *library* denotes an internal service that can be used by a service, or by other libraries; *tool* encapsulates an external tool. The following services have been developed so far:

- BZPAnimator: performs the animation of a BZP model (a B-like intermediate format);
- Angluin: makes it possible to perform a machine learning algorithm (à la Angluin) in order to extract an abstraction of a system behavior;
- UML2SMT: aims at extracting first order logic formulas from the UML Diagrams and OCL code of a UML/OCL model to check them with a SMT solver.

These services involve various libraries (sometimes reusing each other), and rely on several *tool* plug-ins that are: SMTProver (encapsulating the Z3 solver), PrologTools (encapsulating the CLPS-B solver), Grappa (encapsulating a graph library). We are currently working on transferringthe existing work on test generation from B abstract machines, JML, and statecharts using constraint solving techniques.

5.2.2. jMuHLPSL

jMuHLPSL [6] is a mutant generator tool that takes as input a verified HLPSL protocol, and computes mutants of this protocol by applying systematic mutation operators on its contents. The mutated protocol then has to be analyzed by a dedicated protocol analysis tool (here, the AVISPA tool-set). Three verdicts may then arise. The protocol can still be *safe*, after the mutation, this means that the protocol is not sensitive to the realistic "fault" represented by the considered mutation. This information can be used to inform the protocol designers

of the robustness of the protocol w.r.t. potential implementation choices, etc. The protocol can also become *incoherent*, meaning that the mutation introduced a functional failure that prevents the protocol from being executed entirely (one of the participants remains blocked in a given non-final state). The protocol can finally become *unsafe* when the mutation introduces a security flaw that can be exploited by an attacker. In this case, the AVISPA tool-set is able to compute an attack-trace, that represents a test case for the implementation of the protocol. If the attack can be replayed entirely, then the protocol is not safe. If the attack can not be replayed then the implementation does not contain the error introduced in the original protocol.

The tool is written in Java, and it is freely available at: http://members.femto-st.fr/sites/femto-st.fr.frederic-dadeau/files/content/pub/jMuHLPSL.jar.

5.2.3. Praspel

Praspel is both a specification language, a test data generator and test execution driver for PHP programs. These latter are annotated to describe class (resp. method) contracts using invariants (resp. pre- and postconditions). Praspel contracts allow to describe data typing informations, by means of *realistic domains*. According to the contract-driven testing principles, the tool uses the contracts to both generate test data, using dedicated test generators (random for integer variables, grammar-based for strings, constraint-based for arrays), and establish the test verdict by checking the contract assertions at run-time.

The tool is open source and freely available at: http://hoa-project.net. It has been integrated into a PHP framework named Hoa, and coupled with the atoum tool (https://github.com/atoum/atoum) that can be used to execute the tests and report on their code coverage.

5.3. Other Tools

Several software tools described in previous sections are using tools that we have developed in the past. For instance BZ-TT uses the set constraints solver CLPS. Note that the development of the SMT prover haRVey has been stopped. The successor of haRVey is called veriT and is developed by David Déharbe (UFRN Natal, Brasil) and Pascal Fontaine (Veridis team). We have also developed, as a second back-end of *AVISPA*, TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols), an automata based tool dedicated to the validation of security protocols for an unbounded number of sessions.

We have also designed tools to manage collaborative works on shared documents using flexible access control models. These tools have been developed in order to validate and evaluate our approach on combining collaborative edition with optimistic access control.

COMETE Project-Team

5. New Software and Platforms

5.1. Location Guard

Participants: Konstantinos Chatzikokolakis [correspondant], Marco Stronati.

https://github.com/chatziko/location-guard

The purpose of *Location Guard* is to implement obfuscation techniques for achieving location privacy, in a an easy and intuitive way that makes them available to the general public. Various modern applications, running either on smartphones or on the web, allow third parties to obtain the user's location. A smartphone application can obtain this information from the operating system using a system call, while web application obtain it from the browser using a JavaScript call.

Although both mobile operating systems and browsers require the user's permission to disclose location information, the user faces an "all-or-nothing" choice: either disclose his exact location and give up his privacy, or stop using the application. This forces many users to disclose their location, although ideally they would like to enjoy some privacy.

The API level of a browser or an operating system is an ideal place for integrating a location obfuscation technique, in a way that is easy to understand for the average user, and readily available to all applications. When an application asks for the user's location, the browser or operating system can ask the user's permission, but including the option to provide an obfuscated location instead of the real one! Different levels of obfuscation can be also offered, so that the user can chose to provide more accurate location to applications that really need it, and more noisy location to those that don't.

Location Guard was created as a prototype for Google Chrome at the end for 2013. In 2014, Location Guard matured from a prototype to a high quality software, supporting both desktop and mobile browsers:

- Google Chrome / Chromium
- Mozilla Firefox and Firefox for Android
- Opera

After only a short period online, the extension has more than 8500 daily users, and it was presented in an article by the popular technology news site Ghacks. Our experience so far shows that end users do care about location privacy, and geo-indistinguishability is a practical approach for providing it.

In the future we plan to make Location Guard more widely available on smartphones, supporting more mobile browsers as well as providing direct integration into the operating system, primarily on Android.

5.2. libqif - A Quantitative Information Flow C++ Toolkit Library

Participants: Konstantinos Chatzikokolakis [correspondant], Marco Stronati.

https://github.com/chatziko/libqif

The goal of libqif is to provide an efficient C++ toolkit implementing a variety of techniques and algorithms from the area of quantitative information flow and differential privacy. We plan to implement all techniques produced by Comète in recent years, as well as several ones produced outside the group, giving the ability to privacy researchers to reproduce our results and compare different techniques in a uniform and efficient framework.

Some of these techniques were previously implemented in an ad-hoc fashion, in small, incompatible with each-other, non-maintained and usually inefficient tools, used only for the purposes of a single paper and then abandoned. We aim at reimplementing those – as well as adding several new ones not previously implemented – in a structured, efficient and maintainable manner, providing a tool of great value for future research. Of particular interest is the ability to easily re-run evaluations, experiments and case-studies from all our papers, which will be of great value for comparing new research results in the future.

The library is still in under heavy development but substantial progress has been made in 2014. Some of the techniques already implemented are:

- Standard leakage measures: Shannon, min-entropy, guessing entropy
- Measures from the g-leakage framework [32]
- Channel factorization
- Standard differential privacy mechanisms from the literature
- The planar Laplace mechanism of [33]
- The standard Kantorovich metric as well as the multiplicative variant from [19]
- All operations are supported for both doubles (for precision) and floats (for memory efficiency)
- All operations involving only rational quantities are supported using arbitrary precision rational arithmetic, allowing to obtain exact results
- Native linear programing for rationals

Many more are scheduled to be added in the near future.

5.3. LeakWatch: Estimating Information Leakage from Java Programs

Participant: Yusuke Kawamoto.

http://www.cs.bham.ac.uk/research/projects/infotools/leakwatch/

Comète contributed to the development of LeakWatch, a quantitative information leakage analysis tool for the Java programming language, created by several people at the University of Birmingham.

LeakWatch is based on a flexible "point-to-point" information leakage model, where secret and publicly-observable data may occur at any time during a program's execution. LeakWatch repeatedly executes a Java program containing both secret and publicly-observable data and uses robust statistical techniques to provide estimates, with confidence intervals, for min-entropy leakage (using a new theoretical result from [23]) and mutual information.

DICE Team

4. New Software and Platforms

4.1. GPeer: a peer-to-peer javascript communication library

Our software development has been oriented towards systems working in browsers, with the support of an **Inria ADT project** in cooperation with the ASAP team. To answer our technological objectives, we are working on decentralized architectures, browser to browser, developed in javascript/HTML5. We rely on the WebRTC JavaScript protocol proposed by Google to develop a communication layer between peers. Many peer-to-peer protocols share common elements, that we group in a generic library for developing peer-to-peer systems. The joint library developed with the ASAP team handles any gossip based communication overlay. We design peer messages, tracker management and resilient behavior. The library is a standard bridge between complex browser to browser applications and low level networking layers such as WebRTC. With the use of our library, we can reproduce systems such as BitTorrent, but also provide new applications without the need of either native applications or identified servers.

4.2. Fluxion: a software plugin for flows in AngularJS

The **joint project with Worldline** aims at managing mobile code in complex Web architectures. Load variation in data-centers is currently poorly resolved. Most of the time, systems overestimate resource consumption in order to absorb burst usage. These consumption overestimation has a cost both in terms of the SLA negotiated with the client and the non-availability of reserved resources. With Wordline we focus on code mobility for high performance Web architectures and design a fast and reactive framework, transparently moving functions between running systems. The Fluxion model is our approach to design mobile application modules that are a mix of functional programming and flow based reactive systems.

4.3. BitBallot: a decentralized voting protocol

The BitBallot voting protocol is designed to target large scale communities. The protocol allows users to share only restricted amounts of their data and computation with central platforms as well as other peers. Convinced by the need of new election mechanisms, to support emerging forms of more continuous democracy, we are developing BitBallot, to allow elections to be organized independently of any central authority. The protocol guarantees the following properties, anonymity of the data sources, non interruptible run-time, global access to results, and non predictability of results through partial communication spying.

4.4. Odin: an intermediation platform

Odin is a middleware framework for building intermediation platforms. It is build over a kernel that stores users data and activities on a noSQL database and a full client/server JavaScript communication stack. The kernel is used to build intermediation platforms for any kind of project management systems, and where projects peculiarities are handled through a plugin architecture. Plugins are used to define dedicated crawlers over the Web that gather information and push recommendation toward users. The framework maintains an internal currency used to trigger a subset of agents used for recommendation. These recommendations must be mapped to the project keywords and user profile. Each user project is associated to a specific amount of money in our currency, and project users may use this currency to drive their virtual agents. If agents are correctly driven, projects may gain more money used to obtain better recommendations or used on other projects. Our goal is to gather a huge amount of users in order to study system scalability in a real life application. We use odin to test and validate search engines, recommendation engines, external resource crawling, and social network user experiences.

4.5. C3PO: Collaborative Creation of Contents and Publishing using Opportunistic Networks

Social networks put together individuals with common interests and/or existing real-life relationships so that they can produce and share information. There is a strong interest of individuals towards those networks. They rely on a stable, centralized network infrastructure and a user will always be provided with the same services no matter what their current context is. By contrast, the C3PO project aims at promoting "spontaneous and ephemeral social networks" (SESN), built on top of a peer-to-peer distributed architecture leveraging ad-hoc mobile networks and the resources and services offered by mobile devices. As with traditional social networks, SESN can put together nomad individuals based on their affinities and common interests so that they can collaboratively work on tasks as part of a SESN. In C3PO, we strive for incitation in collaborating through a SESN. Several application domains have been anticipated for SESN, especially those involving gathering information and producing content as part of cultural or sport events. In such types of SESN, photo sharing, collaborative document edition and sport results spreading services can be used for building structured digital contents that relate the events of sports gatherings. Generated contents can be consulted through the multiple production sources. They can then be replicated on dedicated servers or published to traditional, centralized social networks and made available to Internet users beyond the lifespan of the SESN where they were initially produced. The C3PO project aims at investigating the problems posed by SESN, and especially those induced by the dynamic and unreliable nature of the ad-hoc mobile networks. It will offer innovative scientific and software solutions for services provision with intermittent connectivity, the definition of an infrastructure for the collaborative management of services in the context of SESN, and an analysis of the value adapted to this context. C3PO is a 3 years ANR industrial research project involving 4 academic research groups and an industrial partner. The proposed contributions will be validated by experimentation in real-world conditions.

PRIVATICS Project-Team

4. New Software and Platforms

4.1. Mobilitics

Mobilitics is a joint project, started in 2012 between Inria and CNIL, which targets privacy issues on smartphones. The goal is to analyze the behavior of smartphones applications and their operating system regarding users private data, that is, the time they are accessed or sent to third party companies usually neither with user's awareness nor consent.

In the presence of a wide range of different smartphones available in terms of operating systems and hardware architecture, Mobilitics project focuses actually its study on the two mostly used mobile platforms, IOS (Iphone) and Android. Both versions of the Mobilitics software: (1) capture any access to private data, any modification (e.g., ciphering or hashing of private data), or transmission of data to remote locations on the Internet; (2) store these events in a local database on the phone for offline analysis; and (3) provide the ability to perform an in depth database analysis in order to identify personnal information leakage.

A Mobilitics prototype for iOS has been developed since early 2012. A Mobilitics prototype for Android has been developped since mid-2013, running on Galaxy Nexus smartphones. In parallel an analysis tool has been developped, capable of analyzing the databases containing the raw data of both Mobile Operating Systems.

A first live experiment has been conducted by CNIL with the Mobilitics sofwtare for IOS with the help of volunteers equipped with iphones in September 2012-January 2013. As a result, some visualization tools have been developed for the data collected in order to showcase private data leakage by the apps which the participants of the experiment have used. A press conference has been held by CNIL and Inria in Paris in April 2013 and several Mobilitics results have been published in French newspapers (see Section 8.3).

A second live experiment has been conducted by CNIL with the Mobilitics software for Android, with the help of volunteers equipped with Galaxy Nexus smartphones, in June-September 2014. A press conference has been held by CNIL and Inria in December 2014, and several results have been published in French newspapers (see Section 8.3).

4.2. Omen+

Omen+ is a password cracker following our previous work. It is used to guess possible passwords based on specific information about the target. It can also be used to check the strength of user password by effectively looking at the similarity of that password with both usual structures and information relative to the user, such as his name, birth date...

It is based on a Markov analysis of known passwords to build guesses. The previous work Omen needs to be cleaned in order to be scaled to real problems and to be distributed or transfered to the security community (maintainability): eventually it will become an open source software. The main challenge of Omen+ is to optimize the memory consumption.

The actual efficiency of that implementation in the cracking of passwords will be tested in the coming days. The processing of the personal information will be implemented before the end of January. The hardest part of that side of Omen+ will be the collection and classification of the information for a particular target.

4.3. OpenFEC

OpenFEC (http://openfec.org) is an open-source C-language implementation of several Application-Level Forward Erasure Correction (AL-FEC) codecs, namely: 2D-parity, Reed-Solomon (RFC 5510, http://tools.ietf.org/html/rfc5510) and LDPC-Staircase (RFC 5170, http://tools.ietf.org/html/rfc5170) codes. The OpenFEC project also provides a complete performance evaluation tool-set, capable of automatically assessing the performance of various codecs, both in terms of erasure recovery and encoding/decoding speed or memory consumption.

A commercial, highly optimized version of OpenFEC is available, along with an implementation of the FLUTE (RFC 6726, http://tools.ietf.org/html/rfc6726) large scale content delivery protocol, and both softwares are currently commercialized by the Expway (http://expway.com) French SME. These softwares have been deployed in many places throughout the world (for instance there were more than 1.5 millions of terminals in Japan implementing the ISDB-Tmm standard, powered by our FLUTE/LDPC-Staircase softwares, in Q3-2013).

Thanks to the success of the industrial transfer of the OpenFEC and FLUTE softwares to Expway, Vincent Roca has been awarded the third FIEEC (Federation des Industries Electriques, Electroniques et Communications) applied research prize in October 2014.

PROSECCO Project-Team

5. New Software and Platforms

5.1. ProVerif

Participants: Bruno Blanchet [correspondant], Xavier Allamigeon [April–July 2004], Vincent Cheval [Sept. 2011–], Benjamin Smyth [Sept. 2009–Feb. 2010].

PROVERIF (proverif.inria.fr) is an automatic security protocol verifier in the symbolic model (so called Dolev-Yao model). In this model, cryptographic primitives are considered as black boxes. This protocol verifier is based on an abstract representation of the protocol by Horn clauses. Its main features are:

- It can handle many different cryptographic primitives, specified as rewrite rules or as equations.
- It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space.

The PROVERIF verifier can prove the following properties:

- secrecy (the adversary cannot obtain the secret);
- authentication and more generally correspondence properties, of the form "if an event has been executed, then other events have been executed as well";
- strong secrecy (the adversary does not see the difference when the value of the secret changes);
- equivalences between processes that differ only by terms.

PROVERIF is widely used by the research community on the verification of security protocols (see http://proverif.inria.fr/proverif-users.html for references).

PROVERIF is freely available on the web, at proverif.inria.fr, under the GPL license.

5.2. CryptoVerif

Participants: Bruno Blanchet [correspondant], David Cadé [Sept. 2009–].

CRYPTOVERIF(cryptoverif.inria.fr) is an automatic protocol prover sound in the computational model. In this model, messages are bitstrings and the adversary is a polynomial-time probabilistic Turing machine. CRYPTOVERIF can prove secrecy and correspondences, which include in particular authentication. It provides a generic mechanism for specifying the security assumptions on cryptographic primitives, which can handle in particular symmetric encryption, message authentication codes, public-key encryption, signatures, hash functions, and Diffie-Hellman key agreements.

The generated proofs are proofs by sequences of games, as used by cryptographers. These proofs are valid for a number of sessions polynomial in the security parameter, in the presence of an active adversary. CRYPTOVERIF can also evaluate the probability of success of an attack against the protocol as a function of the probability of breaking each cryptographic primitive and of the number of sessions (exact security).

CRYPTOVERIF has been used in particular for a study of Kerberos in the computational model, and as a back-end for verifying implementations of protocols in F# and C.

CRYPTOVERIF is freely available on the web, at cryptoverif.inria.fr, under the CeCILL license.

5.3. Cryptosense Analyzer

Participants: Graham Steel [correspondant], Romain Bardou.

See also the web page http://cryptosense.com.

Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards, security tokens and Hardware Security Modules that support the most widely-used industry standard interface, RSA PKCS#11. Each device implements PKCS#11 in a slightly different way since the standard is quite open, but finding a subset of the standard that results in a secure device, i.e. one where cryptographic keys cannot be revealed in clear, is actually rather tricky. Cryptosense Analyzer analyses a device by first reverse engineering the exact implementation of PKCS#11 in use, then building a logical model of this implementation for a model checker, calling a model checker to search for attacks, and in the case where an attack is found, executing it directly on the device. It has been used to find at least a dozen previously unknown flaws in commercially available devices.

In June 2013 we submitted a patent application (13 55374) on the reverse engineering process. We also concluded a license agreement between Inria PROSECCO and the nascent spin-off company Cryptosense to commercialize the tool.

5.4. miTLS

Participants: Karthikeyan Bhargavan [correspondant], Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Alfredo Pironti, Pierre-Yves Strub [IMDEA], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoue.

miTLS is a verified reference implementation of the TLS security protocol in F#, a dialect of OCaml for the .NET platform. It supports SSL version 3.0 and TLS versions 1.0-1.2 and interoperates with mainstream web browsers and servers. miTLS has been verified for functional correctness and cryptographic security using the refinement typechecker F7.

A paper describing the miTLS library was published at IEEE S&P 2013, and two updates to the software were released in 2013. The software and associated research materials are available from http://mitls.rocq.inria.fr.

5.5. WebSpi

Participants: Karthikeyan Bhargavan [correspondant], Chetan Bansal [Microsoft], Antoine Delignat-Lavaud, Sergio Maffeis [Imperial College London].

WebSpi is a library that aims to make it easy to develop models of web security mechanisms and protocols and verify them using ProVerif. It captures common modeling idioms (such as principals and dynamic compromise) and defines a customizable attacker model using a set of flags. It defines an attacker API that is designed to make it easy to extract concrete attacks from ProVerif counterexamples.

WebSpi has been used to analyze social sign-on and social sharing services offered by prominent social networks, such as Facebook, Twitter, and Google, on the basis of new open standards such as the OAuth 2.0 authorization protocol.

WebSpi has also been used to investigate the security of a number of cryptographi web applications, including password managers, cloud storage providers, an e-voting website and a conference management system.

WebSpi is under development and released as an open source library at http://prosecco.inria.fr/webspi/

5.6. Defensive JavaScript

Participants: Antoine Delignat-Lavaud [correspondant], Karthikeyan Bhargavan, Sergio Maffeis [Imperial College London].

Defensive JavaScript (DJS) is a subset of the JavaScript language that guarantees the behaviour of trusted scripts when loaded in an untrusted web page. Code in this subset runs independently of the rest of the JavaScript environment. When propertly wrapped, DJS code can run safely on untrusted pages and keep secrets such as decryption keys. DJS is especially useful to write security APIs that can be loaded in untrusted pages, for instance an OAuth library such as the one used by "Login with Facebook". It is also useful to write secure host-proof web applications, and more generally for cryptography that happens on the browser.

The DJS type checker and various libraries written in DJS are available from http://www.defensivejs.com.

5.7. F*

Participants: Nikhil Swamy [Microsoft Research], Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Catalin Hritcu, Chantal Keller, Aseem Rastogi, Pierre-Yves Strub.

F* is a new higher order, effectful programming language (like ML) designed with program verification in mind. Its type system is based on a core that resembles System Fω (hence the name), but is extended with dependent types, refined monadic effects, refinement types, and higher kinds. Together, these features allow expressing precise and compact specifications for programs, including functional correctness properties. The F* type-checker aims to prove that programs meet their specifications using an automated theorem prover (usually Z3) behind the scenes to discharge proof obligations. Programs written in F* can be translated to OCaml, F#, or JavaScript for execution.

A detailed description of F* (circa 2011) appeared in the Journal of Functional Programming [88]. F* has evolved substantially since then. The latest version of F* is written entirely in F*, and bootstraps in OCaml and F#. It is under active development at GitHub: https://github.com/FStarLang and the official webpage is at http://fstar-lang.org.