RESEARCH CENTER

FIELD
Algorithmics, Programming, Software and Architecture

Activity Report 2016

Section Software

Edition: 2017-08-25

ALGORITHMICS, COMPUTER ALGEBRA AND CRYPTOLOGY

ARCHITECTURE, LANGUAGES AND COMPILATION

EMBEDDED AND REAL-TIME SYSTEMS

PROOFS AND VERIFICATION

SECURITY AND CONFIDENTIALITY

ARIC Project-Team

5. New Software and Platforms

5.1. FPLLL

SCIENTIFIC DESCRIPTION

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.

FUNCTIONAL DESCRIPTION

fplll contains implementations of several lattice algorithms. The implementation relies on floating-point orthogonalization, and the LLL algorithm is central to the code, hence the name. It includes implementations of floating-point LLL reduction algorithms, offering different speed/guarantees ratios. It further includes an implementation of the BKZ reduction algorithm and variants thereof. It includes an implementation of the Kannan-Fincke-Pohst algorithm that finds a shortest non-zero lattice vector. For the same task, the GaussSieve algorithm is also available. Finally, it contains a variant of the enumeration algorithm that computes a lattice vector closest to a given vector belonging to the real span of the lattice.

- Participants: Martin Albrecht, Shi Bai, Guillaume Bonnoron, Léo Ducas, Damien Stehlé and Marc Stevens
- Contact: Damien Stehlé
- URL: https://github.com/fplll/fplll

5.2. HPLLL

hplll is an experimental C++ library companion to fplll.

FUNCTIONAL DESCRIPTION

hplll provides a specific LLL reduction algorithm using Householder orthogonalization, and HPC preliminary solutions especially for integer relation finding.

- Contact: Gilles Villard
- URL: http://perso.ens-lyon.fr/gilles.villard/hplll

5.3. GNU-MPFR

KEYWORDS: Multiple-Precision - Floating-point - Correct Rounding

FUNCTIONAL DESCRIPTION

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.

There have been two new releases in 2016: 3.1.4 and 3.1.5. An MPFR-MPC developers meeting took place on 23 and 24 May 2016.

- Participants: Vincent Lefèvre and Paul Zimmermann
- Contact: Vincent Lefèvre
- URL: http://www.mpfr.org/

5.4. Gfun

A Maple package for solutions of linear differential or recurrence equations
FUNCTIONAL DESCRIPTION

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.

- Contact: Bruno Salvy
- URL: http://perso.ens-lyon.fr/bruno.salvy/software/the-gfun-package/

5.5. Sipe

KEYWORDS: Floating-point - Correct Rounding
FUNCTIONAL DESCRIPTION

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.

- Participant: Vincent Lefèvre
- Contact: Vincent Lefèvre
- URL: https://www.vinc17.net/research/sipe/

5.6. LinBox: a C++ library for exact, high-performance linear algebra computation

LinBox is a C++ template library for exact, high-performance linear algebra computation with dense, sparse, and structured matrices over the integers and over finite fields. LinBox is distributed under the LGPL license. The library is developed by a consortium of researchers in Canada, USA, and France. Clément Pernet is a main contributor, especially with a focus on parallel aspects during the period covered by this report.

- Participants: Clément Pernet, Gilles Villard
- Contact: Clément Pernet
- URL: http://www.linalg.org

AROMATH Project-Team

5. New Software and Platforms

5.1. AXEL

KEYWORDS: CAO - Algebraic geometric modeler

SCIENTIFIC DESCRIPTION

Axel is an algebraic geometric modeler that aims at providing “algebraic modeling” tools for the manipulation and computation with curves, surfaces or volumes described by semi-algebraic representations. These include parametric and implicit representations of geometric objects. Axel also provides algorithms to compute intersection points or curves, singularities of algebraic curves or surfaces, certified topology of curves and surfaces, etc. A plugin mechanism allows to extend easily the data types and functions available in the plateform.

FUNCTIONAL DESCRIPTION

Axel is a cross platform software to visualize, manipulate and compute 3D objects. It is composed of a main application and several plugins. The main application provides atomic geometric data and processes, a viewer based on VTK, a GUI to handle objects, to select data, to apply process on them and to visualize the results. The plugins provides more data with their reader, writer, converter and interactors, more processes on the new or atomic data. It 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. The software is distributed as a source package, as well as binary packages for Linux, MacOSX and Windows.

- Participants: Nicolas Douillet, Anaïs Ducoffe, Valentin Michelet, Bernard Mourrain, Meriadeg Perrinel, Stéphane Chau and Julien Wintz
- Contact: Bernard Mourrain
- URL: http://axel.inria.fr/

Collaboration with Elisa Berrini (MyCFD, Sophia), Tor Dokken (Gotools library, Oslo, Norway), Angelos Mantzaflaris (GISMO library, Linz, Austria), Laura Saini (Post-Doc GALAAD/Missler, TopSolid), Gang Xu (Hangzhou Dianzi University, China).

CARAMBA Project-Team

6. New Software and Platforms

6.1. Belenios

Belenios - Verifiable online voting system
KEYWORD: E-voting
FUNCTIONAL DESCRIPTION

Belenios is an online voting system that provides confidentiality and verifiability. End-to-end verifiability relies on the fact that the ballot box is public (voters can check that their ballots have been taken into account) and on the fact that the tally is publicly verifiable (anyone can recount the votes). Confidentiality relies on the encryption of the votes and the distribution of the decryption key.

Belenios builds upon Helios, a voting protocol used in several elections. The main design enhancement of Belenios vs Helios is that the ballot box can no longer add (fake) ballots, due to the use of credentials.

In 2016 our online platform has been used for several elections, for instance: representatives at the “comité de centre” in several Inria research centers, at the “conseil de laboratoire” at IRISA, and for the head of the “GT Calcul Formel” of the GDR-IM.

- Participants: Pierrick Gaudry, Stéphane Glondu and Véronique Cortier
- Partners: CNRS - Inria
- Contact: Stéphane Glondu
- URL: http://belenios.gforge.inria.fr/

6.2. Kalray-ECM

KEYWORDS: Factorization - Kalray
FUNCTIONAL DESCRIPTION

Implementation of the factorization algorithm based on elliptic curves (ECM) for the MPPA-256 Kalray processor.

- Authors: Jérémie Detrey, Pierrick Gaudry and Masahiro Ishii
- Partner: Nara Institute of Science and Technology, Japan
- Contact: Jérémie Detrey
- URL: https://gforge.inria.fr/projects/kalray-ecm

6.3. TinyGB

- Author: Pierre-Jean Spaenlehauer
- Contact: Pierre-Jean Spaenlehauer
- URL: https://gforge.inria.fr/projects/tinygb/
- Licence: LGPL-3.0+

TinyGB is a software implementing tools for computing Gröbner bases of ideals in polynomial rings over finite fields. It has been released in April 2016.

It is not competitive with state-of-art software for computations over small prime fields. However, for polynomial systems over $\mathbb{Z}/p\mathbb{Z}$, with $p > 2^{31}$, its timings are competitive with the computer algebra system Magma-2.22-2 (although the Magma is much better in terms of memory requirements). This is due to the fact that TinyGB relies on the library MPFQ (developed in the Caramba team) for the efficient arithmetic over large prime fields. For instance, computing the grevlex Gröbner basis of a system of 13 dense homogeneous quadratic equations in 13 variables over the field $\mathbb{Z}/(2^{31}+11)\mathbb{Z}$ can be achieved within 907 seconds with TinyGB, whereas Magma-2.22-2 requires 4459 seconds (on an Intel Core i5-4590@3.30GHz).

The distribution of TinyGB contains the libraries OpenBLAS, FFLAS-FFPACK and MPFQ.

CASCADE Project-Team (section vide)

DATASHAPE Team

6. New Software and Platforms

6.1. GUDHI

Geometric Understanding in Higher Dimensions

SCIENTIFIC DESCRIPTION

The current release of the GUDHI library includes:

- Data structures to represent, construct and manipulate simplicial and cubical complexes.
- Algorithms to compute simplicial complexes from point cloud data.
- Algorithms to compute persistent homology and multi-field persistent homology.
- Simplification methods via implicit representations.

FUNCTIONAL DESCRIPTION

The GUDHI open source library will provide the central data structures and algorithms that underlie 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 transfer of results in applied fields.

- Participants: Jean-Daniel Boissonnat, Marc Glisse, Mariette Yvinec, Clément Maria, David Salinas, Paweł Dłotko, Siargey Kachanovich and Vincent Rouvreau
- Contact: Jean-Daniel Boissonnat
- URL: http://gudhi.gforge.inria.fr/

GRACE Project-Team

6. New Software and Platforms

6.1. ACTIS

FUNCTIONAL DESCRIPTION

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.

- Contact: David Lucas
- https://bil.inria.fr/fr/software/view/2114/tab

During the project, D. Lucas and J. Nielsen proposed a google summer of code project on rank-metric codes under our ACTIS framework. The intern was Arpit Merchant, who visited us for SageDays75.

6.2. muKummer

KEYWORD: Cryptography
FUNCTIONAL DESCRIPTION

A competitive, high-speed, open implementation of the Diffie–Hellman key exchange protocol and a Schnorr-type digital signature scheme, targeting the 128-bit security level on two microcontroller platforms: the classic AVR ATMega 8-bit platform and the more modern ARM Cortex M0 32-bit platform. These downloads contain mixed C and assembly sources for the implementations described in [16].

- Participant: Benjamin Smith
- Contact: Benjamin Smith
- ATMega implementation URL: http://www.cs.ru.nl/~jrenes/software/mukummer-avr.tar.gz
- Cortex M0 implementation URL: http://www.cs.ru.nl/~jrenes/software/mukummer-arm.tar.gz

LFANT Project-Team

5. New Software and Platforms

5.1. APIP

Another Pairing Implementation in PARI
SCIENTIFIC DESCRIPTION

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.
FUNCTIONAL DESCRIPTION

APIP is a library for computing standard and optimised variants of most cryptographic pairings.

- Participant: Jérôme Milan
- Contact: Jérôme Milan
- URL: http://www.lix.polytechnique.fr/~milanj/apip/apip.xhtml

5.2. Arb

FUNCTIONAL DESCRIPTION

Arb is a C library for arbitrary-precision floating-point ball arithmetic. It supports real and complex numbers, polynomials, power series, matrices, and evaluation of many transcendental functions. All is done with automatic, rigorous error bounds. It has been accepted for inclusion in SageMath.

- Participant: Fredrik Johansson
- Contact: Fredrik Johansson
- URL: http://fredrikj.net/arb/

5.3. AVIsogenies

Abelian Varieties and Isogenies
FUNCTIONAL DESCRIPTION

AVIsogenies 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.

- Participants: Gaëtan Bisson, Romain Cosset and Damien Robert
- Contact: Damien Robert
- URL: http://avisogenies.gforge.inria.fr/

5.4. CM

FUNCTIONAL DESCRIPTION

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.

- Participant: Andreas Enge
- Contact: Andreas Enge
- URL: http://www.multiprecision.org/index.php?prog=cm&page=home

5.5. CMH

Computation of Igusa Class Polynomials
KEYWORDS: Mathematics - Cryptography - Number theory
FUNCTIONAL DESCRIPTION

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

- Participants: Emmanuel Thomé, Andreas Enge and Regis Dupont
- Contact: Emmanuel Thomé
- URL: http://cmh.gforge.inria.fr

5.6. CUBIC

FUNCTIONAL DESCRIPTION

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.

- Participant: Karim Belabas
- Contact: Karim Belabas
- URL: http://www.math.u-bordeaux.fr/~belabas/research/software/cubic-1.3.tgz

5.7. Euclid

FUNCTIONAL DESCRIPTION

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.

- Participants: Pierre Lezowski and Jean-Paul Cerri
- Contact: Pierre Lezowski
- URL: http://www.math.u-bordeaux1.fr/~plezowsk/euclid/index.php

5.8. FLINT

FUNCTIONAL DESCRIPTION FLINT is a C library for number theory and basic computer algebra, maintained by William Hart with code by William Hart, Sebastian Pancratz, Andy Novocin, Fredrik Johansson, Tom Bachmann, Mike Hansen, Martin Lee, David Harvey, and a large number of other authors.

FLINT is used as a back end library for polynomial arithmetic and number theory functionality in a large number of applications, including SageMath and Singular.

- Participant: Fredrik Johansson
- Contact: William Hart
- URL: http://flintlib.org/

5.9. GNU MPC

FUNCTIONAL DESCRIPTION

Mpc 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 Mpfr. The library is written by Andreas Enge, Philippe Théveny and Paul Zimmermann.

- Participants: Andreas Enge, Paul Zimmermann, Philippe Theveny and Mickaël Gastineau
- Contact: Andreas Enge
- URL: http://www.multiprecision.org/

5.10. KleinianGroups

FUNCTIONAL DESCRIPTION

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

- Participant: Aurel Page
- Contact: Aurel Page
- URL: http://www.normalesup.org/~page/Recherche/Logiciels/logiciels-en.html

5.11. mpmath

FUNCTIONAL DESCRIPTION mpmath is a Python library for real and complex floating-point arithmetic with arbitrary precision. It has been developed by Fredrik Johansson since 2007, with help from many contributors.

As a dependency of the SymPy computer algebra system as well as SageMath, mpmath is a core component of the Python scientific software ecosystem.

- Participant: Fredrik Johansson
- Contact: Fredrik Johansson
- URL: http://mpmath.org/

5.12. MPFRCX

FUNCTIONAL DESCRIPTION

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.

- Participant: Andreas Enge
- Contact: Andreas Enge
- URL: http://www.multiprecision.org/index.php?prog=mpfrcx

5.13. Nemo

FUNCTIONAL DESCRIPTION Nemo is a computer algebra package for the Julia programming language maintained by William Hart with code by William Hart, Tommy Hofmann, Claus Fieker, Fredrik Johansson, Oleksandr Motsak).

The features of Nemo include multiprecision integers and rationals, integers modulo n, p-adic numbers, finite fields (prime and non-prime order), number field arithmetic, maximal orders of number fields, arithmetic of ideals in maximal orders, arbitrary precision real and complex balls, generic polynomials, power series, fraction fields, residue rings and matrices.

- Participant: Fredrik Johansson
- Contact: William Hart
- URL: http://nemocas.org/

5.14. PARI/GP

FUNCTIONAL DESCRIPTION

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.

- Participants: Karim Belabas, Bill Allombert, Henri Cohen and Andreas Enge
- Contact: Karim Belabas
- URL: http://pari.math.u-bordeaux.fr/

POLSYS Project-Team

5. New Software and Platforms

5.1. Epsilon

FUNCTIONAL DESCRIPTION

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

- Contact: Dongming Wang
- URL: http://wang.cc4cm.org/epsilon/index.html

5.2. FGb

FUNCTIONAL DESCRIPTION

FGb is a powerful software for computing Groebner 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.

- Participant: Jean-Charles Faugère
- Contact: Jean-Charles Faugère
- URL: http://polsys.lip6.fr/~jcf/Software/FGb/index.html

5.3. FGb Light

FUNCTIONAL DESCRIPTION

Gröbner basis computation modulo p (p is a prime integer of 16 bits).

- Participant: Jean-Charles Faugère
- Contact: Jean-Charles Faugère
- URL: http://www-polsys.lip6.fr/~jcf/Software/FGb/

5.4. GBLA

FUNCTIONAL DESCRIPTION

GBLA is an open source C library for linear algebra specialized for eliminating matrices generated during Gröbner basis computations in algorithms like F4 or F5.

- Contact: Jean-Charles Faugère
- URL: http://www-polsys.lip6.fr/~jcf/Software/index.html

5.5. HFEBoost

FUNCTIONAL DESCRIPTION

Public-key cryptography system enabling an authentification of dematerialized data.

- Authors: Jean-Charles Faugère and Ludovic Perret
- Partner: UPMC
- Contact: Jean-Charles Faugère
- URL: http://www-polsys.lip6.fr/Links/hfeboost.html

5.6. RAGlib

Real Algebraic Geometry library
FUNCTIONAL DESCRIPTION

RAGLib is a powerful library, written in Maple, dedicated to solving over the reals polynomial systems. It is based on the FGb library for computing Grobner bases. It provides functionalities for deciding the emptiness and/or computing sample points to real solution sets of polynomial systems of equations and inequalities. This library provides implementations of the state-of-the-art algorithms with the currently best known asymptotic complexity for those problems.

- Contact: Mohab Safey El Din
- URL: http://www-polsys.lip6.fr/~safey/RAGLib/

5.7. SLV

FUNCTIONAL DESCRIPTION

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.

- Contact: Elias Tsigaridas
- URL: http://www-polsys.lip6.fr/~elias/soft

5.8. SPECTRA

Semidefinite Programming solved Exactly with Computational Tools of Real Algebra
FUNCTIONAL DESCRIPTION

SPECTRA is a Maple library devoted to solving exactly Semi-Definite Programs. It can handle rank constraints on the solution. It is based on the FGb library for computing Grobner bases and provides either certified numerical approximations of the solutions or exact representations of them.

- Contact: Mohab Safey El Din
- URL: http://homepages.laas.fr/henrion/software/spectra/

SECRET Project-Team

6. New Software and Platforms

6.1. CFS

FUNCTIONAL DESCRIPTION

Reference implementation of parallel CFS (reinforced version of the digital signature scheme CFS). 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.

- Participants: Nicolas Sendrier and Gregory Landais
- Contact: Nicolas Sendrier
- URL: https://gforge.inria.fr/projects/cfs-signature/

6.2. Collision Decoding

KEYWORDS: Algorithm - Binary linear code

FUNCTIONAL DESCRIPTION

Collision Decoding implements two variants of information set decoding : Stern-Dumer, and MMT. 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.

- Participants: Nicolas Sendrier and Gregory Landais
- Contact: Nicolas Sendrier
- URL: https://gforge.inria.fr/projects/collision-dec/

6.3. ISDF

FUNCTIONAL DESCRIPTION

Implementation of the Stern-Dumer decoding algorithm, and of a varaint of the algorithm due to May, Meurer and Thomae.

- Participants: Nicolas Sendrier and Gregory Landais
- Contact: Anne Canteaut
- URL: https://gforge.inria.fr/projects/collision-dec/

SPECFUN Project-Team

5. New Software and Platforms

5.1. DDMF

Dynamic Dictionary of Mathematical Functions
FUNCTIONAL DESCRIPTION

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.
This year, Maxence Guesdon started to port DDMF to the new DynaMoW. To this end, a special environment has been set up to be able to use the Inria continuous-integration platform.

- Participants: Alexandre Benoit, Frédéric Chyzak, Alexis Darrasse, Stefan Gerhold, Thomas Grégoire, Maxence Guesdon, Christoph Koutschan, Marc Mezzarobba and Bruno Salvy
- Contact: Frédéric Chyzak
- URL: http://ddmf.msr-inria.inria.fr/1.9.1/ddmf

5.2. DynaMoW

Dynamic Mathematics on the Web
FUNCTIONAL DESCRIPTION

DynaMoW is a programming tool for controlling the generation of mathematical websites that embed dynamical mathematical contents generated by computer-algebra calculations. Implemented in OCaml.
After a complete redesign and rewrite last year, to get more reactiveness and configurability, the implementation of DynaMoW was made more robust this year while porting ECS to this new library. It was next further enhanced, in particular in order have informative and reliable traces of execution, to help with the debugging of asynchronous parallel executions of services.

- Participants: Frédéric Chyzak, Alexis Darrasse and Maxence Guesdon
- Contact: Frédéric Chyzak
- URL: http://ddmf.msr-inria.inria.fr/DynaMoW/

5.3. ECS

Encyclopedia of Combinatorial Structures
FUNCTIONAL DESCRIPTION

ECS is an online mathematical encyclopedia with an emphasis on sequences that arise in the context of decomposable combinatorial structures, with the possibility to search by the first terms in the sequence, keyword, generating function, or closed form.
This year, we finalized the port of ECS to the last evolutions of DynaMoW. A new website was setup, and ECS is now again online, after a few years of discontinuation for technical reasons.

- Participants: Stéphanie Petit, Alexis Darrasse, Frédéric Chyzak and Maxence Guesdon
- Contact: Frédéric Chyzak
- URL: http://ecs.inria.fr/

5.4. Math-Components

Mathematical Components library

FUNCTIONAL DESCRIPTION

The Mathematical Components library is a set of Coq libraries that cover the mechanization of the proof of the Odd Order Theorem.

This year we experimented the maintenance of the library using the public repository stored on the github platform since December 2015. This allowed for merging several contributions from external users and improved significantly the communication with the community of users. A new website has also been set up, which includes pointers to various teaching and documentation resources.

- Participants: Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, Francois Garillot, Georges Gonthier, Stéphane Le Roux, Assia Mahboubi, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi and Russell O'connor
- Contact: Assia Mahboubi
- URL: http://www.msr-inria.fr/projects/mathematical-components-2/

5.5. Ssreflect

FUNCTIONAL DESCRIPTION

Ssreflect is a tactic language extension to the Coq system, developed by the Mathematical Components team.

This year we improved the manual of the language, in order to document new features and to clarify some older parts of the document.

- Participants: Cyril Cohen, Yves Bertot, Laurence Rideau, Enrico Tassi, Laurent Thery, Assia Mahboubi and Georges Gonthier
- Contact: Yves Bertot
- URL: http://ssr.msr-inria.inria.fr/

VEGAS Project-Team

5. New Software and Platforms

5.1. ISOTOP

Topology and geometry of planar algebraic curves
KEYWORDS: Topology - Curve plotting - Geometric computing
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, registered at the APP in June 2011, has been developed since 2007 in collaboration with F. Rouillier from Inria Paris. The distributed version is based on the method described in [3], which presents several improvements over previous methods. In particular, our approach does not require generic position. 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 a new bivariate polynomial solver based on several of our recent results published in [11]. This version is not yet distributed.

Via the Inria ADT FastTrack funding, Eric Biagioli has joined the project in November 2016 for 6 months. He is porting the maple code to C code and enhancing the visualization. This work will prepare for a better diffusion of the software via a webserver and a transfert to Maplesoft with which Inria has signed a contract in April 2016.

- Contact: Sylvain Lazard & Marc Pouget
- URL: http://vegas.loria.fr/isotop/

5.2. SubdivisionSolver

KEYWORDS: Numerical solver - Polynomial or analytical systems
The software SubdivisionSolver solves square systems of analytic equations on a compact subset of a real space of any finite dimension. SubdivisionSolver is a numerical solver and as such it requires that the solutions in the subset are isolated and regular for the input system (i.e. the Jacobian must not vanish). SubdivisionSolver is a subdivision solver using interval arithmetic and multiprecision arithmetic to achieve certified results. If the arithmetic precision required to isolate solutions is known, it can be given as an input parameter of the process, otherwise the precision is increased on the fly. In particular, SubdivisionSolver can be interfaced with the Fast_Polynomial library (https://bil.inria.fr/en/software/view/2423/tab) to solve polynomial systems that are large in terms of degree, number of monomials and bit-size of coefficients.

The software is based on a classic branch and bound algorithm using interval arithmetic: an initial box is subdivided until its sub-boxes are certified to contain either no solution or a unique solution of the input system. Evaluation is performed with a centered evaluation at order two, and existence and uniqueness of solutions is verified thanks to the Krawczyk operator.

SubdivisionSolver uses two implementations of interval arithmetic: the C++ boost library that provides a fast arithmetic when double precision is enough, and otherwise the C mpfi library that allows to work in arbitrary precision. Considering the subdivision process as a breadth first search in a tree, the boost interval arithmetic is used as deeply as possible before a new subdivision process using higher precision arithmetic is performed on the remaining forest.

The software has been improved and a technical report published [28].

- Contact: Rémi Imbach
- URL: https://bil.inria.fr/fr/software/view/2605/tab

CAIRN Project-Team

6. New Software and Platforms

6.1. 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 designed an extensible source-to-source compiler infrastructure that takes advantage of leading edge model-driven object-oriented software engineering principles and technologies.

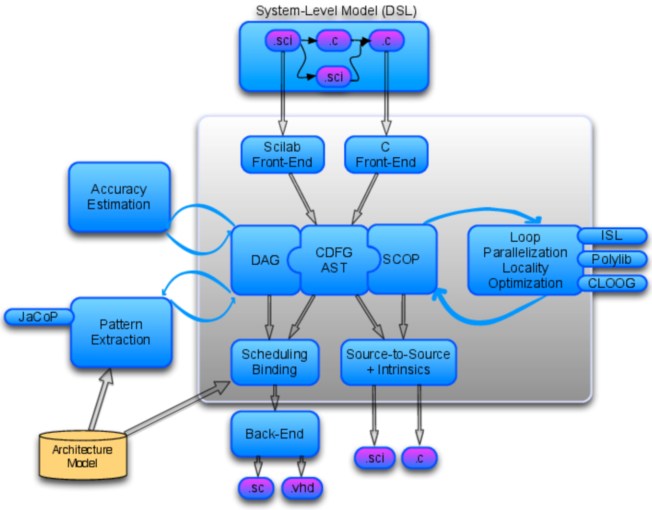


Figure 2. CAIRN's general software development framework.

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 hierarchy of communicating tasks, where each task is expressed using C or Matlab/Scilab, and where the system-level representation and the target platform model are often 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 ASIC/FPGA 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.
- A polyhedral-based loop transformation and parallelization engine (mostly targeted at HLS).
- A custom instruction extraction flow (for ASIP and dynamically reconfigurable architectures). **Durase** is developed for the compilation and the synthesis targeting reconfigurable platforms and the automatic synthesis of application specific processor extensions. It uses advanced technologies, such as graph matching 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).

Gecos, ID.Fix or Durase have been demonstrated during “University Booths” in various conference such as IEEE/ACM DAC or DATE.

6.2. Gecos

KEYWORDS: Source-to-source compiler - Model-driven software engineering - Retargetable compilation

SCIENTIFIC DESCRIPTION

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 between 2012 and 2015 it was used in the context of the FP7 ALMA European project. The Gecos infrastructure will also be used by the EMMTRIX start-up, a spin-off from the ALMA project which aims at commercializing the results of the project and in the context of the H2020 ARGO European project.

FUNCTIONAL DESCRIPTION

Gecos provides a program transformation toolbox facilitating parallelisation of applications for heterogeneous multiprocessor embedded platforms. This includes a polyhedral loop transformation toolbox, efficient SIMD code generation for fixed point arithmetic data-types, coarse-grain parallelization engine targeting the data-flow actor model, and a Matlab/Scilab front-end. In addition to targeting programmable processors, Gecos can regenerate optimized code for High Level Synthesis tools.

- Participants: Steven Derrien, Nicolas Simon, Imen Fassi, and Ali Hassan El-Moussawi
- Partner: Université de Rennes 1
- Contact: Steven Derrien
- URL: http://gecos.gforge.inria.fr/doku/doku.php

6.3. ID-Fix

KEYWORDS: Energy efficiency - Embedded systems - Analytical accuracy evaluation - Fixed-point arithmetic - Accuracy optimization - Dynamic range evaluation - Code optimisation

SCIENTIFIC DESCRIPTION

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 different fixed-point data types. 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).

FUNCTIONAL DESCRIPTION

ID.Fix focuses on computational accuracy and can provide an optimised specification using fixed point arithmetic from a C source code with floating point data types. Fixed point arithmetic is very widely used in embedded systems as it provides better performance and is much more energy efficient. ID.Fix used an analytical model of the software code, which means it can explore more solutions and thereby produce much more efficient code than classical simulation-based tools.

- Participants: Olivier Sentieys, Benjamin Barrois and Nicolas Simon
- Partner: Université de Rennes 1
- Contact: Olivier Sentieys
- URL: http://idfix.gforge.inria.fr/doku.php

6.4. Zyggie

KEYWORDS: Health - Biomechanics - Wireless body sensor networks - Low power - Gesture recognition - Hardware platform - Software platform - Localization

SCIENTIFIC DESCRIPTION

Zyggie is a hardware and software wireless body sensor network platform. Each sensor node, attached to different parts of the human body, contains inertial sensors (IMU) (accelerometer, gyrometer, compass and barometer), an embedded processor and a low-power radio module to communicate data to a coordinator node connected to a computer, tablet or smartphone. One of the system's key innovations is that it collects data from sensors as well as on distances estimated from the power of the radio signal received to make the 3D location of the nodes more precise and thus prevent IMU sensor drift and power consumption overhead. Zyggie can be used to determine posture or gestures and mainly has applications in sport, healthcare and the multimedia industry.

FUNCTIONAL DESCRIPTION

The Zyggie sensor platform was developed 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 a processor, a radio transceiver and different sensors including an Inertial Measurement Unit (IMU) with 3-axis accelerometer, gyrometer, and magnetometer. Zyggie is used for evaluating data fusion algorithms, low power computing algorithms, wireless protocols, and body channel characterization in the BoWI project.

The Zyggie V2 prototype includes new features: a 32-bit microcontroller to manage a custom MAC layer and processe quaternions based on IMU measures, and an UWB radio from DecaWave to measure distances between nodes with Time of Flight (ToF).

- Participants: Arnaud Carer and Olivier Sentieys
- Partners: Lab-STICC - Université de Rennes 1
- Contact: Olivier Sentieys
- URL: http://www.bowi.cominlabs.ueb.eu/fr/zyggie-wbsn-platform

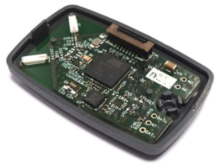


Figure 3. CAIRN*'s Ziggie platform for WBSN*

CAMUS Team

6. New Software and Platforms

6.1. Apollo

Automatic speculative POLyhedral Loop Optimizer
KEYWORD: Automatic parallelization
FUNCTIONAL DESCRIPTION

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. It can apply on-the-fly any kind of polyhedral transformations, including tiling, and can handle nonlinear loops, as while-loops referencing memory through pointers and indirections.

- Participants: Manuel Selva, Juan Manuel Martinez Caamaño, Artiom Baloian, and Philippe Clauss
- Contact: Philippe Clauss
- URL: http://apollo.gforge.inria.fr

6.2. CLooG

Code Generator in the Polyhedral Model
FUNCTIONAL DESCRIPTION

CLooG 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.

- Participant: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://www.cloog.org

6.3. Clan

A Polyhedral Representation Extraction Tool for C-Based High Level Languages
FUNCTIONAL DESCRIPTION

Clan 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.

- Participants: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://icps.u-strasbg.fr/people/bastoul/public_html/development/clan/

6.4. Clay

Chunky Loop Alteration wizardrY

FUNCTIONAL DESCRIPTION

Clay 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.

- Participant: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://icps.u-strasbg.fr/people/bastoul/public_html/development/clay/

6.5. IBB

Iterate-But-Better

FUNCTIONAL DESCRIPTION

IBB is a source-to-source xfor compiler which automatically translates any C source code containing xfor-loops into an equivalent source code where xfor-loops have been transformed into equivalent for-loops.

- Participants: Philippe Clauss and Cédric Bastoul
- Contact: Philippe Clauss
- URL: http://xfor.gforge.inria.fr

6.6. OpenScop

A Specification and a Library for Data Exchange in Polyhedral Compilation Tools

FUNCTIONAL DESCRIPTION

OpenScop 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.

- Participant: Cédric Bastoul
- Contact: Cédric Bastoul
- URL: http://icps.u-strasbg.fr/people/bastoul/public_html/development/openscop/

6.7. PolyLib

The Polyhedral Library

FUNCTIONAL DESCRIPTION

PolyLib 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.

- Participant: Vincent Loechner
- Contact: Vincent Loechner
- URL: http://icps.u-strasbg.fr/PolyLib/

6.8. ORWL and P99

ORWL is a reference implementation of the Ordered Read-Write Lock tools as described in [1]. The macro definitions and tools for programming in C99 that have been implemented for ORWL have been separated out into a toolbox called P99. ORWL is intended to become opensource, once it will be in a publishable state. P99 is available under a QPL at http://p99.gforge.inria.fr/.

- Participants: Jens Gustedt, Mariem Saied, Daniel Salas
- Contact: Jens Gustedt
- http://p99.gforge.inria.fr/, http://orwl.gforge.inria.fr/

6.9. Stdatomic and Musl

We implement the library side of the C11 atomic interface. It needs compiler support for the individual atomic operations and provides library support for the cases where no low-level atomic instruction is available and a lock must be taken.

- This implementation builds entirely on the ABIs of the gcc compiler for atomics.
- It provide all function interfaces that the gcc ABIs and the C standard need.
- For compilers that don't offer the direct language support for atomics it provides a syntactically reduced but fully functional approach to atomic operations.
- At the core of the library is a new and very efficient futex-based lock algorithm that is implemented for the Linux operating system.

A description of the new lock algorithm has been given in [2]. A short version of it has been presented at SAC'16.

The primary target of this library is an integration into musl to which we also contribute. It is a re-implementation of the C library as it is described by the C and POSIX standards. It is *lightweight*, *fast*, *simple*, *free*, and strives to be correct in the sense of standards-conformance and safety. Musl is production quality code that is mainly used in the area of embedded device. It gains more market share also in other area, *e.g.* there are now Linux distributions that are based on musl instead of Gnu LibC.

- Participant: Jens Gustedt
- Contact: Jens Gustedt
- http://stdatomic.gforge.inria.fr/, http://www.musl-libc.org/

COMPSYS Team

6. New Software and Platforms

6.1. Lattifold

Lattice-based Memory Folding
KEYWORDS: Polyhedral compilation - Euclidean Lattices
FUNCTIONAL DESCRIPTION

Implements advanced lattice-based memory folding techniques. The idea is to reduce memory footprint of multidimensional arrays by reducing the size of each dimension. Given a relation denoting conflicting array cells, it produces a new mapping based on affine functions bounded by moduli. The moduli induces memory reuse and bound memory accesses to a tighter area, allowing to reduce the array size without loss of correctness. Status: proof of concept, see related paper [2].

- Partner: ENS Lyon
- Contact: Alexandre Isoard

6.2. PolyOrdo

Polynomial Scheduler
FUNCTIONAL DESCRIPTION

Computes a polynomial schedule for a sequential polyhedral program having no affine schedule, in lieu of multidimensional schedules. Uses algorithms for finding positive polynomials in semi-algebraic sets. Status: proof of concept software, see related paper [14].

- Contact: Paul Feautrier

6.3. OpenOrdo

OpenStream scheduler
FUNCTIONAL DESCRIPTION

Finds polynomial schedules for the streaming language OpenStream. Main use: detecting deadlocks. The scheduler has been extended to bound the size of stream buffers, either directly or as a side-effect of constructing bounded delay schedules. An effort for bounding the number of in-flight tasks is under way.

Status: proof of concept, see related paper [1].

- Contact: Paul Feautrier

6.4. ppcg-paramtiling

Parametric Tiling Extension for PPCG
KEYWORDS: Source-to-source compiler - Polyhedral compilation
FUNCTIONAL DESCRIPTION

PPCG is a source-to-source compiler, based on polyhedral techniques, targeting GPU architectures. It involves automatic parallelization and tiling using polyhedral techniques. This version replaces the static tiling of PPCG by a fully parametric tiling and code generator. It allows to choose tile sizes at run time when the memory size is known. It also provides a symbolic expression of memory usage depending on the problem size and the tile sizes.

Status: proof of concept, unfinished, see Alexandre Isoard's thesis [17].

- Partner: ENS Lyon
- Contact: Alexandre Isoard

CORSE Project-Team

5. New Software and Platforms

5.1. Tirex

TIREX is an extensible, textual intermediate code representation that is intended to be used as an exchange format for compilers and other tools working on low level code. In the scope of the TIREX project we have developed tools for generating TIREX code from higher level languages such as C, as well as a number of static analyses and transformations.

Work on the TIREX project consisted of two main parts, firstly creation of a machine description library for all parts of the TIREX project, secondly, the development of tools for parsing assembly code.

We developed `archinfo`, a LLVM based library that allows programatic access to descriptors for a target CPUs instructions and registers. The focus was to expose information that was not already available from LLVM, such as machine operand types (float or integer, bitwidth, ...) and flags describing the high level behaviour of the instructions.

The, also LLVM based, assembly parser is intended to be used for translating assembly files generated by common compilers to TIREX, but it can also handle a number of idioms usually found in hand written assembly code. It reconstructs some high level information required for the TIREX format, such as the control flow and call graph, from the assembly code. We also started investigating how our existing tools can be extended to directly parse binary code and reconstruct information from them.

5.2. QEMU plugins

We have collaborated with STMicroelectronics on extending the QEMU CPU emulator with a plugin system. These plugins allow users to observe and modify the machine code emitted by QEMUs binary translator.

We have leveraged this to start development on a number of tools for profiling and performance debugging.

- cachesim: A QEMU plugin that feeds memory accesses observed during program execution into the DineroIV cache simulator. This allows estimating the number of a cache misses caused by each instruction of a program. Using this information we can also estimate the amount of memory bandwidth required by a program. This in turn can be used to diagnose if the applications performance is constrained by memory or CPU resources.
- dep-rate: A QEMU plugin that uses a shadow memory to detects data dependencies between instructions and correlates them with cache misses reported by DineroIV to estimate the performance impact of these dependencies.
- cpath: A QEMU plugin that estimates the optimal execution time of a program on an infinitely parallel CPU and compares it to that for a more realistic model of a CPU. This comparison is used to judge the amount of instruction level parallelism existing in a program.

5.3. Givy

Givy is a runtime 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 executes dynamic data-flow task graphs, annotated with memory dependencies. It automatically handles 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. An important part of the work corresponds on implementing and testing a memory allocator with specific properties that is a building block of the whole run time. 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 multith readed calls.

5.4. Dynamic Dependence Graph (DDG)

By instrumenting the memory accesses, at the LLVM IR level, of a hand selected region of a program, the DDG tool builds a graph with all dynamic instructions. Each instruction, i.e. a node in the graph, is identified by a statement identifier, mapping the dynamic instruction to a static statement, and an induction vector, containing the trip counters of loops surrounding the related statement. Edges connecting these nodes represent either data dependence, reuse or anti-dependence among the instructions, obtained by using the shadow memory technique, that labels ownership to a given written memory position to a dynamic instruction, and creating relationship to it to instructions that read the exact same memory position. Instructions that have a statically known formula (SCEVs) are not tracked, allowing our technique to remove, for example, obvious dependencies from a loop iteration to the next, and still track integer instructions. As the number of dynamic instructions, even in very simple applications, grows extremely fast, the generated graph does not to fit in main memory just after a few hundred loop iterations, our tool allows limiting the number of loop iterations that are tracked. Dependencies between iterations outside the observed iteration space can either be ignored or clamped as being generated by a single instruction. The generated graph can be used to guide loop optimizers, that could not extract precise dependencies. It can also be used by performance debugging tools, in order to determine if it is possible to obtain a new instruction schedule that would improve locality.

5.5. Integer polynomial Fourier-Motzkin elimination

Quantifier elimination is the process of removing existential variables of a given formula, obtaining one that is simpler in the number of variables, and that is implied by the original formula. A very well known algorithm is the Fourier-Motzkin elimination process, that given a system (or formula) of inequalities removes variables by combining all upper and lower bounds of such variables. At each step a variable is selected and eliminated. The very first limitation of this algorithm is the fact that it is designed for linear systems, where all coefficients of the variable being eliminated are numeric values, and the inequality can be classified as either a upper or lower bound. When dealing with polynomials, all possible values, positive, negative, or zero, for an coefficient, that is, a symbolic expression, must be explored. To avoid this requirement we use the positiveness algorithm, proposed by Mark Schweighofer, to retrieve symbolic coefficient signs. In fact, this algorithm is of major importance when resolving system over integer variables, instead of reals, as it is used in many other techniques required to preserve the precision of the simplified formula, such as symbolic normalization, convex hull detection, redundancy removing. Our C++ implementation uses GiNaC for symbolic expressions manipulation.

5.6. BOAST: Metaprogramming of Computing Kernels

BOAST aims at providing a framework to metaprogram, benchmark and validate computing kernels. BOAST is a programming framework dedicated to code generation and autotuning. This software allows the transformation from code written in the BOAST DSL to classical HPC targets like FORTRAN, C, OpenMP, OpenCL or CUDA. It also enables the meta-programming of optimization that can be (de)activated when needed. BOAST can also benchmark and do non regression tests on the generated kernels. This approach gives, both, performance gains and improved performance portability.

BOAST can be dowloaded at this address https://forge.imag.fr/projects/boast/.

BOAST was already used to generate and optimize the computing kernels of three scientific applications:

- BigDFT: A massively parallel electronic structure code using wavelet basis set.
- SPECFEM: Computational Infrastructure for Geodynamics.
- Gysela: Fusion plasma simulations.

BOAST is currently used in the context of the European H2020/HPC4E project. The computing kernels of two scientific applications are currently studied with BOAST:

- Alya: Large Scale Computational Mechanics.
- Hou10ni: Solutions to accoustics wave propagation problems. This code is developed by the Magique3D Inria team (Pau, Julien Diaz).

Frédéric Desprez presented BOAST at the CSCD workshop http://www.netlib.org/utk/people/JackDongarra/CCDSC-2016/ in October 2016. After this workshop, a paper was submited at the Internationaj Journal on High Performance Computing Applications (IJHPCA).

BOAST was also used in the Bulldog project during the last CERMACS summer school http://smai.emath.fr/cemracs/cemracs16/ in July 2016. A joint paper with CEA researchers from Cadarache and Maison de la Simulation was also submitted to present the results of the Bulldog project.

5.7. mcGDB: Interactive debugging of OpenMP programs

MCGDB introduced the concept of *programming-model centric* source-level interactive debugging as an extension of the traditional language-level interactive debugging. The idea was to integrate into debuggers the notion of *programming models*, as abstract machines running over the physical ones. These abstract machines, implemented by runtime libraries and programming frameworks, provide the high-level primitives required for the implementation of today's parallel applications.

We developed a proof-of-concept, mcGDB, as a Python extension of GDB, the debugger of the GNU project. mcGDB was initially developed by Kevin Pouget during his thesis with STMicroelectronics. mcGDB is currently extended with the Nano2017/DEMA project.

We proposed the new support of mcGDB for OpenMP task-based programming. This support consists of task-based execution representation and control improvements, in cooperation with Temanejo graphical debugger. We also studied import implementation details of mcGDB, related to the support of multiple OpenMP environments and CPU architectures; the separation of cross-cutting concerns (user interaction and execution representing) through aspect-oriented programming, and the first steps of mcGDB micro-benchmarking.

mcGDB [30] was presented at the second OpenMPCon developpers conference in Nara.

DREAMPAL Project-Team

5. New Software and Platforms

5.1. HoMade

KEYWORDS: SoC - Multicore - Softcore
FUNCTIONAL DESCRIPTION

HoMade is a softcore processor. 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.

- Participant: Jean Luc Dekeyser
- Partner: LIFL
- Contact: Jean Luc Dekeyser
- URL: https://sites.google.com/site/homadeguideen/home

5.2. JHomade

FUNCTIONAL DESCRIPTION

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.

- Contact: Vlad Rusu
- URL: https://gforge.inria.fr/frs/?group_id=3646

PACAP Project-Team

6. New Software and Platforms

6.1. ATMI

KEYWORDS: Analytic model - Chip design - Temperature
SCIENTIFIC DESCRIPTION

Research on temperature-aware computer architecture requires a chip temperature model. General purpose models based on classical numerical methods like finite differences or finite elements are not appropriate for such research, because they are generally too slow for modeling the time-varying thermal behavior of a processing chip.

We have developed an ad hoc temperature model, ATMI (Analytical model of Temperature in MIcroprocessors), for studying thermal behaviors over a time scale ranging from microseconds to several minutes. ATMI is based on an explicit solution to the heat equation and on the principle of superposition. ATMI can model any power density map that can be described as a superposition of rectangle sources, which is appropriate for modeling the microarchitectural units of a microprocessor.
FUNCTIONAL DESCRIPTION

ATMI is a library for modelling steady-state and time-varying temperature in microprocessors. ATMI uses a simplified representation of microprocessor packaging.

- Participant: Pierre Michaud
- Contact: Pierre Michaud
- URL: https://team.inria.fr/pacap/software/atmi/

6.2. Heptane

KEYWORDS: Static analysis - Real time - Performance - WCET - IPET - Worst Case Execution Time
SCIENTIFIC DESCRIPTION

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.
FUNCTIONAL DESCRIPTION

In a hard real-time system, it is essential to comply with timing constraints, and Worst Case Execution Time (WCET) in particular. Timing analysis is performed at two levels: analysis of the WCET for each task in isolation taking account of the hardware architecture, and schedulability analysis of all the tasks in the system. Heptane is a static WCET analyser designed to address the first issue.

- Participants: Isabelle Puaut, Damien Hardy, Loïc Besnard
- Partner: Université de Rennes 1
- Contact: Isabelle Puaut
- URL: https://team.inria.fr/pacap/software/heptane/

6.3. Tiptop

KEYWORDS: HPC - Performance - CPU - Cache - Cycles - Instructions - Branch predictor
SCIENTIFIC DESCRIPTION

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

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 2.3, released July 2015.

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
FUNCTIONAL DESCRIPTION

Today's microprocessors have become extremely complex. To better understand the multitude of internal events, manufacturers have integrated many monitoring counters. Tiptop can be used to collect and display the values from these performance counters very easily. Tiptop may be of interest to anyone who wants to optimise the performance of their HPC applications.

- Participant: Erven Rohou
- Contact: Erven Rohou
- URL: http://tiptop.gforge.inria.fr

6.4. ATC

Address Trace Compression
KEYWORDS: Compressing - Decompressing - Address traces
FUNCTIONAL DESCRIPTION

ATC is a utility and a C library for compressing/decompressing address traces. It implements a new lossless transformation, Bytesort, that exploits spatial locality in address traces. ATC leverages existing general-purpose compressors such as gzip and bzip2. ATC also provides a lossy compression mode that yields higher compression ratios while preserving certain important characteristics of the original trace.

- Participant: Pierre Michaud
- Contact: Pierre Michaud
- URL: https://team.inria.fr/pacap/software/atc/

6.5. Barra

Modelisation of a GPU architecture
KEYWORDS: Simulator - GPU - Computer architecture
SCIENTIFIC DESCRIPTION

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 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.

FUNCTIONAL DESCRIPTION

Barra simulates CUDA programs at the assembly language level (Tesla ISA). Its ultimate goal is to provide a 100 % bit-accurate simulation, offering bug-for-bug compatibility with NVIDIA G80-based GPUs. It works directly with CUDA executables, neither source modification nor recompilation is required.

Barra is primarily intended as a tool for research in computer architecture, although it can also be used to debug, profile and optimize CUDA programs at the lowest level.

- Participants: Sylvain Collange, David Defour, Alexandre Kouyoumdjian and Fabrice Mouhartem
- Contact: Sylvain Collange
- URL: http://barra.gforge.inria.fr/

6.6. If-memo

KEYWORD: Performance, function memoization, dynamic optimization
Status: Ongoing development, early prototype. Registered with APP (Agence de Protection des Programmes) under number IDDN.FR.001.250013.000.S.P.2015.000.10800.

SCIENTIFIC DESCRIPTION

Memoization is the technique of saving result of executions so that future executions can be omitted when the inputs repeat. Memoization has been proposed in previous literature at the instruction level, basic block level and function level using hardware as well as pure software level approaches including changes to programming language.

We proposed software memoization of pure functions for procedural languages. We rely on the operating system loader, taking advantage of the LD_PRELOAD feature of UNIX systems. By setting this variable to the path of a shared library, we instruct the loader to first look to missing symbols in that library. Our library redefines the functions we wish to intercept. The interception code is very straightforward: it receives the same parameter as the target function and checks in a table (a software cache) if this value is readily available. In the favorable case, the result value is immediately returned. Otherwise, we invoke the original function, and store the result in the cache before returning it.

Our technique does not require the availability of source code and thus can be applied even to commercial applications as well as applications with legacy codes. As far as users are concerned, enabling memoization is as simple as setting an environment variable. We validated If-memo with x86-64 platform using both GCC and icc compiler tool-chains, and ARM cortex-A9 platform using GCC.

- Participants: Erven Rohou and Arjun Suresh
- Contact: Erven Rohou

6.7. Padrone

KEYWORDS: Legacy code - Optimization - Performance analysis - Dynamic Optimization
Status: Registered with APP (Agence de Protection des Programmes) under number IDDN.FR.001.250013.000.S.P.2015.000.10800.

FUNCTIONAL DESCRIPTION

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 especially 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.

- Participants: Erven Rohou and Emmanuel Riou
- Contact: Erven Rohou
- https://team.inria.fr/pacap/software/Padrone/

6.8. STiMuL

Steady temperature in Multi-Layers components
FUNCTIONAL DESCRIPTION

STiMuL is a C library for modeling steady-state heat conduction in microprocessors. It can be used to obtain temperature from power density or power density from temperature. It can also be used to model stacked dies. STiMuL does not model time-varying temperature. For time-varying temperature, other models must be used, such as ATMI.

- Participant: Pierre Michaud
- Contact: Pierre Michaud
- URL: https://team.inria.fr/pacap/software/stimul/

6.9. TPCalc

Throughput calculator
KEYWORDS: Architecture - Performance analysis
FUNCTIONAL DESCRIPTION

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 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.

- Participant: Pierre Michaud
- Partner: Ghent University
- Contact: Pierre Michaud
- URL: https://team.inria.fr/pacap/software/tpcalc/

6.10. Parasuite

Participants: Sylvain Collange, Imane Lasri, Erven Rohou, André Seznec.

Parasuite: parallel benchmarks for multi-core CPUs, clusters and accelerators

Despite the ubiquity of parallel architectures in all computing segments, the research community often lacks benchmarks representative of parallel applications. The Inria Parallel Benchmark Suite (Parasuite) seeks to address this need by providing a set of representative parallel benchmarks for the architecture, compiler and system research communities. Parasuite targets the main contemporary parallel programming technologies: shared-memory multi-thread parallelism for multi-core, message-passing parallelism for clusters and fine-grained data-level parallelism for GPU architectures and SIMD extensions.

All benchmarks come with input datasets of various sizes, to accommodate use cases ranging from microarchitecture simulation to large-scale performance evaluation. Correctness checks on the computed results enable automated regression testing. In order to support computer arithmetic optimization and approximate computing research scenarios, the correctness checks favor accuracy metrics evaluating domain-specific relevance rather than bit-exact comparisons against an arbitrary reference output.

Visit: http://parasuite.inria.fr/

6.11. Simty

Participant: Sylvain Collange.

Simty: A Synthesizable General-Purpose SIMT Processor.

Simty is a massively multi-threaded processor core that dynamically assembles SIMD instructions from scalar multi-thread code. It runs the RISC-V (RV32-I) instruction set. Unlike existing SIMD or SIMT processors like GPUs, Simty takes binaries compiled for general-purpose processors without any instruction set extension or compiler changes. Simty is described in synthesizable VHDL.

Visit: http://team.inria.fr/pacap/simty

TASC Project-Team

6. New Software and Platforms

6.1. AIUR

(Artificial Intelligence Using Randomness

FUNCTIONAL DESCRIPTION

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.

- Contact: Florian Richoux
- URL: https://github.com/AIUR-group/AIUR

6.2. CHOCO

KEYWORDS: Constraint Programming - Scheduling - Optimisation - Operational research - Financial analysis - Planning

SCIENTIFIC DESCRIPTION

or second consecutive year, CHOCO has participated at the MiniZinc Challenge , an annual competition of constraint programming solvers. In concurrency 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.

Within the context of the PhD thesis of Charles Prud’homme, a domain specific language that allows prototyping propagation engines was integrated within CHOCO, A paper appears at Constraints.

Within the context of the PhD thesis of Charles Prud’homme, a generic strategy based on explanations for large neighborhood search was designed and integrated within CHOCO. A corresponding paper appears at Constraints.

Within the context of the PhD thesis of Jean-Guillaume Fages, a documented package for graph variables was designed and integrated within CHOCO .

FUNCTIONAL DESCRIPTION

CHOCO is a Java discrete constraints library for describing hard combinatorial problems in the form of Constraint Satisfaction Problems and solving them with Constraint Programming techniques. Choco can be used to solve a broad range of real combinatorial problems. It is easy to use and offers excellent performance. This technique enables non-specialists to tackle strategic or operational problems, for instance, problems related to planning, scheduling, logistics, financial analysis and bio-informatics.

- Participants: Charles Prud’homme, Nicolas Beldiceanu, Jean-Guillaume Fages, Xavier Lorca, Thierry Petit and Rémi Douence
- Partner: Ecole des Mines de Nantes
- Contact: Julien Prud’homme
- URL: http://www.choco-solver.org/

6.3. GCCat

Global Constraint Catalog
KEYWORDS: Constraint Programming - Graph - Global constraint
FUNCTIONAL DESCRIPTION

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

- Participants: Nicolas Beldiceanu and Sophie Demassey
- Contact: Nicolas Beldiceanu
- URL: http://sofdem.github.io/gccat/gccat/index.html

6.4. GCCat on time series

Global Constraint Catalog, Volume II, time-series constraints
KEYWORDS: Constraint Programming - Sequence - Transducer - Global constraint
FUNCTIONAL DESCRIPTION

The second volume of the Global Constraint Catalogue is devoted to time-series constraints. Within the context of Constraint Programming, time-series constraints go back to the work of Goldin and Kanellakis. This volume contains 626 constraints, which are explicitly described in terms of automata with accumulators. Checkers and propagators for all these constraints were synthesised from 22 transducers.

As in the first volume, the global constraints described in this second volume are not only accessible to humans, who can read the catalogue when searching for some information. It is also available to machines, which can read and interpret it. This is why there also exists an electronic version of this catalogue where one can get, for all time-series constraints, a complete description in terms of meta-data used in the first volume. In fact, unlike the first volume, *all the meta-data* of the electronic version as well as *all text and figures* of this second volume were automatically generated. While this second volume is by no means supposed to contain all possible time-series constraints, it contributes in the context of time-series constraints to the *systematic reconstruction* of the Global Constraint Catalogue that we have previously advocated. This reconstruction is based on the following methodology:

- First reuse, adapt or come up with abstractions, which allow to concisely represent structures and properties of time series as abstract combinatorial objects. In our context these abstractions essentially correspond to:
  1. Transducers where letters of the output alphabet are interpreted as semantic letters indicating how to recognise pattern occurrences.
  2. Transducers glue matrices expressing the relationship between the prefix, the suffix and the full sequence passed to a transducer.
  3. Properties associated to regular expressions corresponding to fragments of the input language of our transducers.
- Second, create from these abstract combinatorial objects a data base of concrete combinatorial objects.
- Third, synthesise concrete code for various technologies, languages, tasks from this data base of concrete combinatorial objects. In this context, correctness and efficiency of the synthesised code are essentially side product of:
  - The correctness of the formulae of our data base which is itself based on the wellformedness of our abstractions.
  - The generality behind our abstract combinatorial objects.

The time-series catalogue is done in the following way:

- All time-series constraints are now defined in a *compositional way* from a few basic constituents, i.e., patterns, features, aggregators, and predicates, which completely define the meaning of a constraint, where patterns are defined using regular expressions.
- Constraint names are now constructed in a systematic way as the *concatenation* of pattern name, feature name, and aggregation or predicate name.
- Given a pattern p, checkers and constraints are now *systematically synthesised* from a transducer that, given an input sequence over the input alphabet $\{<,=,>\}$, compares two adjacent values of a time-series and determines an output sequence over a output semantic alphabet describing how to recognise the occurrences of p.
- For each time-series constraint associated with a pattern p, the generation of an automaton with accumulators is completely driven by the transducer associated with pattern p as well as by *decoration tables* describing for each semantic letter of the output alphabet of the transducers how to generate accumulator updates. Code optimisation is ensured by using decoration tables that depend on properties of the pattern, of the feature, and of the aggregator associated with the time-series constraint.
- Lower and upper bounds of characteristics of time-series that appear in the restriction slot of a time-series constraint are synthesised from a *few parameterised formulae* that only depend on a restricted set of characteristics of the regular expression associated with the pattern.
- Parametrised glue matrices are provided for each transducer that corresponds to reversible time-series constraints. A concrete glue matrix is given for each reversible time-series constraint.
- Linear invariants are systematically obtained by applying the Farkas Lemma to the automata with accumulators that were synthesised. They consist of *linear constraints typically linking consecutive accumulator values*, e.g., see the legend of the second automaton of the constraints, which are generated even with non-linear accumulator updates. Missing linear invariants will be completed later on.
- Last but not least, time-series constraints were used for generating time-series verifying a conjunction of constraints both in the context of Constraint Programming and in the context of Linear Programming.
- In the context of sequential pattern mining, time-series constraint checkers can be used to identify and extract patterns from fixed sequences. While the time-series catalogue may need to be extended in order to capture more patterns, having a possibly large set of fixed time-series constraints is a natural safeguard to prevent overfitting when dealing with few sequences, at a price of not finding patterns that are not covered by the catalogue.
- Finally, both SICStus and MiniZinc code are synthesised. The later allows using time series constraints on many plate forms such as Choco, Gecode, ORtools, Cplex or Gurobi and is available Electronic Constraint Catalogue.
- Participants: Ekaterina Arafailova, Nicolas Beldiceanu, Rémi Douence, Mats Carlsson, Pierre Flener, Maria Andreina Francisco Rodriguez, Justin Pearson, Helmut Simonis
- Contact: Nicolas Beldiceanu
- URL: https://arxiv.org/abs/1609.08925

6.5. GHOST

General meta-Heuristic Optimization Solving Tool

FUNCTIONAL DESCRIPTION

GHOST is a template C++ library designed for StarCraft:BroodWartm. GHOST implements a meta-heuristic solver aiming to solve any kind of combinatorial and optimization RTS-related problems represented by a csp /cop. The solver handles dedicated geometric and assignment constraints in a way that is compatible with very strong real time requirements.

- Contact: Florian Richoux
- URL: http://github.com/richoux/GHOST

6.6. TorchCraft

Machine learning framework for games
FUNCTIONAL DESCRIPTION

TorchCraft is a library that enables deep learning research on Real-Time Strategy (RTS) games such as StarCraft: Brood War, by making it easier to control these games from a machine learning framework, here Torch. This white paper argues for using RTS games as a benchmark for AI research, and describes the design and components of TorchCraft.

- Participants: Gabriel Synnaeve, Nantas Nardelli, Alex Auvolat, Soumith Chintala, Timothe´e Lacroix, Zeming Lin, Florian Richoux, Nicolas Usunier
- Contact: Florian Richoux
- URL: https://arxiv.org/abs/1611.00625

AOSTE Project-Team

5. New Software and Platforms

5.1. EVT Kopernic

Extreme Value Theory for Keeping Worst Reasoning Appropriate for Different Criticalities
FUNCTIONAL DESCRIPTION This software provides a probabilistic bound on the worst case execution time of a program. Its third version, released in March 2016, covers the case of statistically dependent execution times. Currently integrated in Rapitime (Rapita tool chain), a lighter version is under preparation for integration in FUI Waruna framework as well as in the preparation of hybrid versions to be released in 2017 as output of Capacites project.

- Participants: Liliana Cucu and Adriana Gogonel
- Contact: Liliana Cucu
- URL: https://who.rocq.inria.fr/Liliana.Cucu/Software.html

5.2. KPASSA

K-Periodic Asap Static Schedule Analyser
FUNCTIONAL DESCRIPTION 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.

- Participants: Jean Vivien Millo and Robert De Simone
- Contact: Robert de Simone
- URL: http://www-sop.inria.fr/members/Jean-Vivien.Millo/kpassa/index.php

5.3. Lopht

Logical to Physical Time Compiler
SCIENTIFIC DESCRIPTION Lopht is a system-level compiler for embedded systems. Its input is formed of three objects:

- A functional specification in a high-level synchronous language.
- A description of the implementation platform, defining the topology of the parallel execution platform, and the capacity of its elements.
- A set of non-functional requirements, provided under the form of annotations on both functional specification and platform description.

The algorithmic core of Lopht is formed of allocation and scheduling heuristics which rely on two fundamental choices: the use of table-based static scheduling and the use of low-complexity heuristics based on list scheduling. The output of Lopht is formed of all the C code and configuration information needed to allow real deployment on the physical target platform.

FUNCTIONAL DESCRIPTION Accepted input languages for functional specifications include Heptagon and Scade v4. Lopht uses as f ront-end a modified version of the Heptagon compiler developed at Inria. The use of this front-end also allows the use of legacy/business C code satisfying the Heptagon calling convention.

Regarding scheduling, the originality of Lopht resides in a strong focus on classical compiler optimizations e.g. software pipelining), on novel architectural targets (many-core chips and time-triggered embedded systems), and the possibility to handle multiple, complex non-functional requirements covering real-time (release dates a nd deadlines possibly larger than the period, end-to-end flow constraints), ARINC 653 partitioning, the possibility to preempt or not each task, and allocation.
The output of Lopht is formed of all the C code and configuration information needed to allow compilation, linking/loading, and real-time execution on the target platform. Lopht fully automates the creation of tasks, partition, the full synthesis of C code compliant with the target API (e.g. C/APEX for ARINC 653 platforms), including communication code, and OS configuration for each computer), as well as the synthesis of communication schedules for the system

Two Lopht back-ends provide distinct input languages for platform description:

- One for distributed time-triggered architectures using ARINC 653-based processing nodes (SBCs) and Time-Triggered Ethernet networks
- One for many-core processors with support with timing predictability.

An ongoing research effort aims at providing a unified, formal platform description language allowing the unification of these back-ends.

- Participants: Dumitru Potop Butucaru, Raul Gorcitz, and Keryan Didier
- Contact: Dumitru Potop Butucaru

5.4. SAS

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

- Participants: Daniel De Rauglaudre and Yves Sorel
- Contact: Yves Sorel
- URL: http://pauillac.inria.fr/~ddr/sas-dameid/

5.5. SynDEx

KEYWORDS: Embedded systems - Real time - Optimization - Distributed - Scheduling analyses
SCIENTIFIC DESCRIPTION SynDEx is a system level CAD software implementing the AAA methodology for rapid prototyping and for optimizing distributed real-time embedded applications. It is developed in OCaML.

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.
FUNCTIONAL DESCRIPTION Software for optimising the implementation of embedded distributed real-time applications and generating efficient and correct by construction code

- Participants: Yves Sorel and Meriem Zidouni
- URL: http://www.syndex.org

5.6. TimeSquare

KEYWORDS: Profil MARTE - Embedded systems - UML - IDM
SCIENTIFIC DESCRIPTION TimeSquare offers six main functionalities:

- graphical and/or textual interactive specification of logical clocks and relative constraints between them,
- definition and handling of user-defined clock constraint libraries,
- automated simulation of concurrent behavior traces respecting such constraints, using a Boolean solver for consistent trace extraction,
- call-back mechanisms for the traceability of results (animation of models, display and interaction with waveform representations, generation of sequence diagrams...).
- 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.
- a generation of the whole state space of a specification (if finite of course) in order to enable model checking of temporal properties on it

FUNCTIONAL DESCRIPTION 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, and more accurately on the associated Clock Constraint Specification Language (CCSL) for the expression of timing constraints.

- Participants: Frederic Mallet, and Julien Deantoni
- Contact: Frederic Mallet
- URL: http://timesquare.inria.fr

5.7. Vercors

KEYWORD:

- Participants: Eric Madelaine, Oleksandra Kulankhina, Jimmy Awk, Xudong Qin
- Contact: Eric Madelaine
- URL: http://www-sop.inria.fr/oasis/Vercors

FUNCTIONAL DESCRIPTION The Vercors tools include front-ends for specifying the architecture and behaviour of components in the form of UML diagrams. We translate these high-level specifications, into behavioural models in various formats, and we also transform these models using abstractions. In a final step, abstract models are translated into the input format for various verification toolsets. Currently we mainly use the various analysis modules of the CADP toolset.

We have achieved this year a major version of the platform frontend, named VCE-v4, that is now distributed on our website, and used by some of our partners. This version features a full chain of tools from the design of systems in the graphical component editors (VCE), the checking of static semantics correcteness, the generation of a semantic model suitable for model-checking, and finally the generation of executable code for the Proactive/GCM platform. These new features, and the tool architecture, have been described in [29] and [18].

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 [34] 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 [40], [39] 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 [35] 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 [42], CTL [30], ACTL [32], etc.) to be defined in XTL.

- PBG (*Partitioned BCG Graph*) is a file format implementing the theoretical concept of *Partitioned LTS* [38] 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 [50] 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 [36] 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, see § 5.2), 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 [37], being used in all recent compilers developed by CONVECS.

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

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

HYCOMES Project-Team

6. New Software and Platforms

6.1. SunDAE

Structural analysis tool for multimode DAE systems

FUNCTIONAL DESCRIPTION

SunDAE is a multimode DAE (mDAE) structural analysis tool. Structural differentiation index is determined, impulsion analysis is performed and a BTF scheduling of the equations is performed, for each mode of a mDAE system. The input language consists in guarded equations. The output is a state-machine where states define continuous-time dynamics and transitions define resets. Both are defined by scheduled blocks of equations. SunDAE has been developed since 2016 by the Hycomes team and is distributed as an open-source software, under the CeCCIL Free Software Licensing Agreement.

- Contact: Benoit Caillaud

6.2. Flipflop

Test & Flip Net Synthesis Tool for the Inference of Technical Procedure Models

FUNCTIONAL DESCRIPTION

Flipflop is a Test and Flip net synthesis tool implementing a linear algebraic polynomial time algorithm. Computations are done in the Z/2Z 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.

- Contact: Benoit Caillaud
- URL: http://tinyurl.com/oql6f3y

6.3. MICA

Model Interface Compositional Analysis Library

KEYWORDS: Modal interfaces - Contract-based desing

SCIENTIFIC DESCRIPTION

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.

FUNCTIONAL DESCRIPTION

Mica is an Ocaml library implementing the Modal Interface algebra. 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.

- Participant: Benoit Caillaud
- Contact: Benoit Caillaud
- URL: http://www.irisa.fr/s4/tools/mica/

MUTANT Project-Team

5. New Software and Platforms

5.1. Antescofo

Anticipatory Score Following and Real-time Language

FUNCTIONAL DESCRIPTION. Antescofo is a modular polyphonic Score Following system as well as a Synchronous Programming language for musical composition. The first 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 (DSL) within Antescofo allows flexible writing of time and interaction in computer music.

- Participants: Arshia Cont, Jean-Louis Giavitto, Florent Jacquemard and José Echeveste
- Contact: Arshia Cont
- URL: http://forumnet.ircam.fr/product/antescofo/

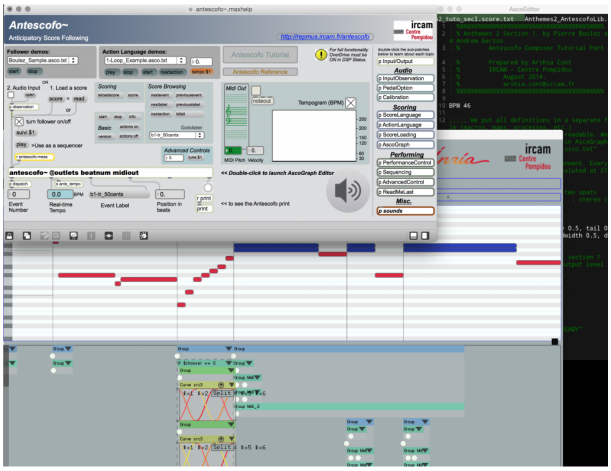

Antescofo~
Anticipatory Score Following
ircam
Centre Pompidou
Antescofo Tutorial
Antescofo Reference
Tempogram (BPM)
<< Double-click to launch AscoGraph Editor
<< to see the Antescofo print

Figure 2. Antescofo and AscoGraph Screenshots

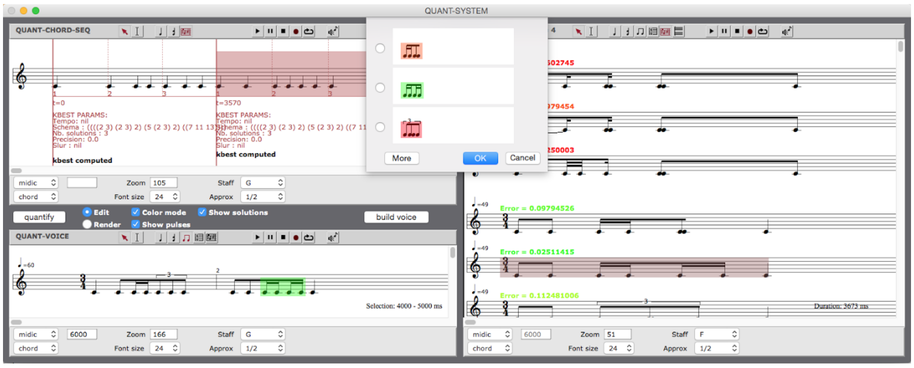
The design of the Antescofo DSL clearly benefits of a strong and continuous involvement in the production of world-class composer pieces and their continuous recreation throughout the world. These interactions motivate new developments, challenge the state of the art and in return, opens new creative dimensions for composers and musicians. The maturity of the system is assessed by the generalization of its use in a large proportion of Ircam new productions, and its use outside Ircam all around the world (Brasil, Chile, Cuba, Italy, China, US, etc.). Antescofo enjoys an active community of 150 active users: http://forumnet.ircam.fr/user-groups/antescofo/

5.2. OMRQ

Library for rhythm transcription integrated in the assisted composition environment OpenMusic.

FUNCTIONAL DESCRIPTION. Rhythm transcription is the conversion of sequence of timed events into the structured representations of conventional Western music notation. Available as a graphical component of OpenMusisc, the library OMRQ privileges user interactions in order to search for an appropriate balance between different criteria, in particular the precision of the transcription and the readability of the musical scores produced.

This system follows a uniform approach, using hierarchical representations of timing notations in the form of rhythm trees, and efficient parsing algorithms for the lazy enumeration of solutions of transcription.



t=0
KBEST PARAMS:
Tempo: nil
Schema : ((((2 3) (2 3) 2) (5 (2 3) 2) ((7 11 13
Nb. solutions : 3
Precision: 0.0
Slur : nil
kbest computed

t=3570
KBEST PARAMS:
Tempo: nil
Schema : ((((2 3) (2 3) 2) (5 (2 3) 2) ((7 11
Nb. solutions : 3
Precision: 0.0
Slur : nil
kbest computed

Selection: 4000 - 5000 ms

Error = 0.09794526

Error = 0.02511415

Error = 0.112481006

Duration: 3673 ms

Figure 3. Screenshot of the Open Music Rhythm Quantization library

Its implementation is carried out via a dedicated interface allowing interactive exploration of the solutions space, their visualization and local editing, with particular attention to the processing of grace notes and rests.

- Participants: Florent Jacquemard and Adrien Ycart
- Contact: Florent Jacquemard
- URL: http://repmus.ircam.fr/cao/rq, https://bil.inria.fr/fr/software/view/2904/tab

5.3. Antescofo Timed Test Platform

Timed testing plateform for Antescofo.

FUNCTIONAL DESCRIPTION. 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 is based on state of the art techniques and tools for *model-based testing* of embedded systems [50], and makes it possible to automate the following main tasks:

1. offline and on-the-fly generation of relevant input data for testing (i.e. fake performances of musicians, including timing values), with the sake of exhaustiveness,
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. black-box execution of the input test data on the System Under Test,
4. comparison of expected and real output and production of a test verdict.

The input and output data are timed traces (sequences of discrete events together with inter-event durations). Our method is based on formal models (specifications) in an ad hoc medium-level intermediate representation (IR). We have developed a compiler for producing automatically such IR models from Antescofo high level mixed scores.

Then, in the offline approach, the IR is passed, after conversion to Timed Automata, to the model-checker Uppaal, to which is delegated the above task (1), following coverage criteria, and the task (2), by simulation. In the online approach, tasks (1) and (2) are realized during the execution of the IR by a Virtual Machine developed on purpose. 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.

- Participants: Clément Poncelet, Florent Jacquemard, Pierre Donat-Bouillud
- Contact: Clément Poncelet

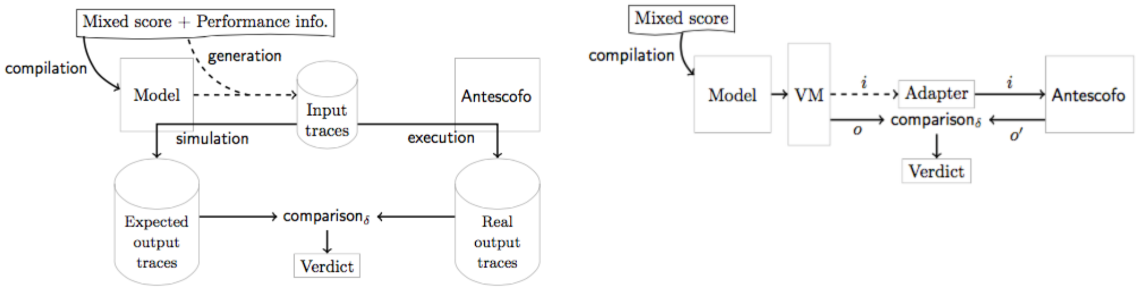


Figure 4. Offline and Online workflows for Antescofo Model Based Testing

These implementations have been conducted as a part of Clément Poncelet's PhD Thesis.

5.4. Ascograph

The Antescofo graphical score editor.

FUNCTIONAL DESCRIPTION. AscoGraph, 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 (Figure 5). 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.

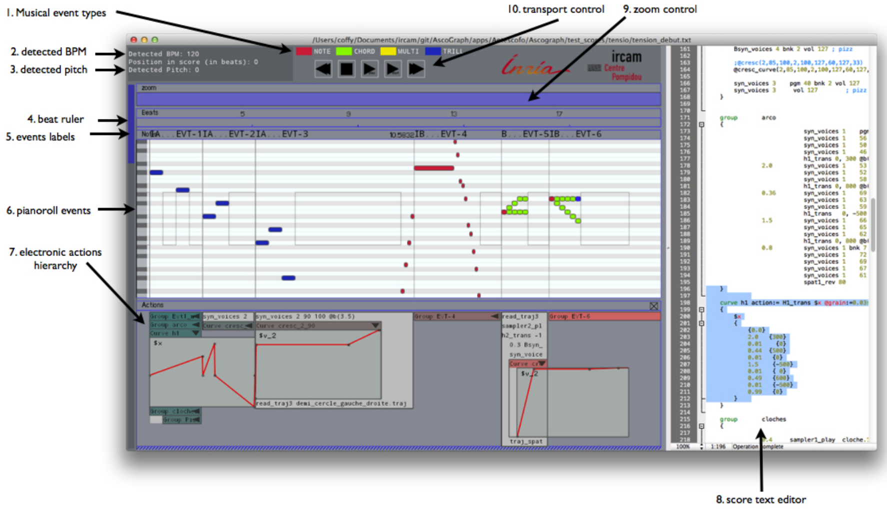


Figure 5. Screenshot of Ascograph, the Antescofo graphical score editor

PARKAS Project-Team

5. New Software and Platforms

5.1. Cmmtest

FUNCTIONAL DESCRIPTION

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

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

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

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

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

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

5.2. GCC

KEYWORDS: Compilation - Polyhedral compilation
FUNCTIONAL DESCRIPTION

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

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

5.3. Heptagon

FUNCTIONAL DESCRIPTION

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

- Participants: Adrien Guatto, Marc Pouzet, Cédric Pasteur, Léonard Gerard, Brice Gelineau, Gwenael Delaval and Eric Rutten
- Contact: Marc Pouzet

5.4. Lem

lightweight executable mathematics
FUNCTIONAL DESCRIPTION

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

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

5.5. Lucid Synchrone

FUNCTIONAL DESCRIPTION

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

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

5.6. Lucy-n

Lucy-n: an n-synchronous data-flow programming language
FUNCTIONAL DESCRIPTION

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

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

5.7. Ott

FUNCTIONAL DESCRIPTION

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

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

a Coq version of the definition,

an Isabelle version of the definition, and

a HOL version of the definition.

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

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

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

5.8. PPCG

Functional Description

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

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

5.9. ReactiveML

Functional Description

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

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

- Participants: Guillaume Baudart, Louis Mandel and Cédric Pasteur
- Contact: Guillaume Baudart
- URL: http://rml.lri.fr

5.10. SundialsML

Sundials/ML
Keywords: Simulation - Mathematics - Numerical simulations
Scientific Description

Sundials/ML is an OCaml interface to the Sundials suite of numerical solvers (CVODE, CVODES, IDA, IDAS, KINSOL, ARKODE). It supports all features except for the Hypre and PETSC nvectors (which require additional libraries). Its structure mostly follows that of the Sundials library, both for ease of reading the existing documentation and for adapting existing source code, but several changes have been made for programming convenience and to increase safety, namely:

- solver sessions are mostly configured via algebraic data types rather than multiple function calls,
- errors are signalled by exceptions not return codes (also from user-supplied callback routines),
- user data is shared between callback routines via closures (partial applications of functions),
- vectors are checked for compatibility (using a combination of static and dynamic checks), and
- explicit free commands are not necessary since OCaml is a garbage-collected language.

FUNCTIONAL DESCRIPTION

Sundials/ML is an OCaml interface to the Sundials suite of numerical solvers (CVODE, CVODES, IDA, IDAS, KINSOL, ARKODE).

NEW PROGRESS

This year we updated our interface to work with versions 2.6.0 and 2.7.0 of the Sundials library. This included significant work to support the new ARKODE solver, sparse matrices and the KLU and SuperLU/MT linear solvers, OpenMP and Pthreads nvectors, and various new functions and linear solvers in existing solvers. The source files were completely reorganized. The OCaml types for nvectors were adapted to support multiple nvectors. Memory leaks were eliminated and the performance problems investigated. This work was presented at the ACM Workshop on ML [28].

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

5.11. Zélus

SCIENTIFIC DESCRIPTION

The Zélus implementation has two main parts: a compiler that transforms Zélus programs into OCaml programs and a runtime library that orchestrates compiled programs and numeric solvers. The runtime can use the Sundials numeric solver, or custom implementations of well-known algorithms for numerically approximating continuous dynamics.

FUNCTIONAL DESCRIPTION

Zélus is a new programming language for hybrid system modeling. It is based on a synchronous language but extends it with Ordinary Differential Equations (ODEs) to model continuous-time behaviors. It allows for combining arbitrarily data-flow equations, hierarchical automata and ODEs. The language keeps all the fundamental features of synchronous languages: the compiler statically ensure the absence of deadlocks and critical races, it is able to generate statically scheduled code running in bounded time and space and a type-system is used to distinguish discrete and logical-time signals from continuous-time ones. The ability to combines those features with ODEs made the language usable both for programming discrete controllers and their physical environment.

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

5.12. isl

FUNCTIONAL DESCRIPTION

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

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

POSET Team

6. New Software and Platforms

6.1. T-calculus

Sketched in [10], the *T-calculus* is a Domain Specific Language [0] to provide simple and robust high-level description mechanisms of reactive systems. It will offer a programmatic view of the tile modeling paradigm [3], [9]. Its definition has been refined a number of times (see e.g. [10], [8]). A prototype implementation of its reactive kernel has eventually been achieved in Haskell on top of the Euterpea libraries during the spring 2016 [15], [21], [19], [23]

[0] See [30] for an early note by Hudak about the notion of Domain Specific Language, and see [29], [32] for application of this notion is computer music.

SPADES Project-Team

5. New Software and Platforms

5.1. pyCPA_TWCA: A pyCPA plugin for computing deadline miss models

FUNCTIONAL DESCRIPTION

We are developing pyCPA_TWCA, a pyCPA plugin for Typical Worst-Case Analysis as described in Section 6.2.5 . pyCPA is an open-source Python implementation of Compositional Performance Analysis developed at TU Braunschweig, which allows in particular response-time analysis. pyCPA_TWCA is an extension of this tool that is co-developed by Sophie Quinton, Zain Hammadeh (TU Braunschweig) and Leonie Ahrendts (TU Braunschweig). It allows in particular the computation of weakly-hard guarantees for real-time tasks, *i.e.*, the number of deadline misses out of a sequence of executions. This year, pyCPA_TWCA has been extended to task chains but remains limited to uniprocessor systems, scheduled according to static priority scheduling. A public release is planned but has not yet taken place.

- Authors: Zain Hammadeh and Leonie Ahrendts and Sophie Quinton.
- Contact: Sophie Quinton.

TEA Project-Team

6. New Software and Platforms

6.1. ADFG: Affine data-flow graphs scheduler synthesis

Participants: Alexandre Honorat, Jean-Pierre Talpin, Thierry Gautier, Loïc Besnard.

We proposed [2], and implemented [0], a new data-flow design model: ADFG, initially to synthesize schedulers for SCJ/L1 applications. The principle of ADFG is to perform a linear abstraction of complex cyclo-static scheduling problems followed by the exploration of a concrete solution extracted from the abstract solution space, hence the name: abstract affine data-flow scheduling. ADFG guarantees schedules that ordinary (e.g. RTJ, SCJ) task-sets do not cause overflows or underflows. ADFG objectives are to maximize the throughput (the processors utilization) while minimizing buffering storage space needed between actors. ADFG supports EDF and fixed-priority scheduling policies for uni-, multi-processors and distributed systems.

The data-flow design model of ADFG 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. Abstract affine scheduling is first applied on the data-flow 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.

In the case of safety-critical Java, and through a model-to-text transformations using Acceleo, 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.

6.2. The Eclipse project POP

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

The distribution of project POP is a major achievement of the ESPRESSO (and now TEA) project-team. The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. It was finalized 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. Project-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.

Project POP is composed of the Polychrony tool set, under GPL license, and its Eclipse framework, under EPL license. SSME (Syntactic Signal-Meta under Eclipse), is the meta-model of the Signal language implemented with Eclipse/Ecore. It describes all syntactic elements specified in Signal Reference Manual [0]: all Signal operators (e.g. arithmetic, clock synchronization), model (e.g. process frame, module), and construction (e.g. iteration, type declaration). 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, P) within an Eclipse-based development tool-chain. Polychrony now comprises the

[0] *The ADFG tool*, Adnan Bouakaz, http://people.irisa.fr/Adnan.Bouakaz/software.htm

[0] *SIGNAL V4-Inria version: Reference Manual*. Besnard, L., Gautier, T. and Le Guernic, P. http://www.irisa.fr/espresso/Polychrony, 2010

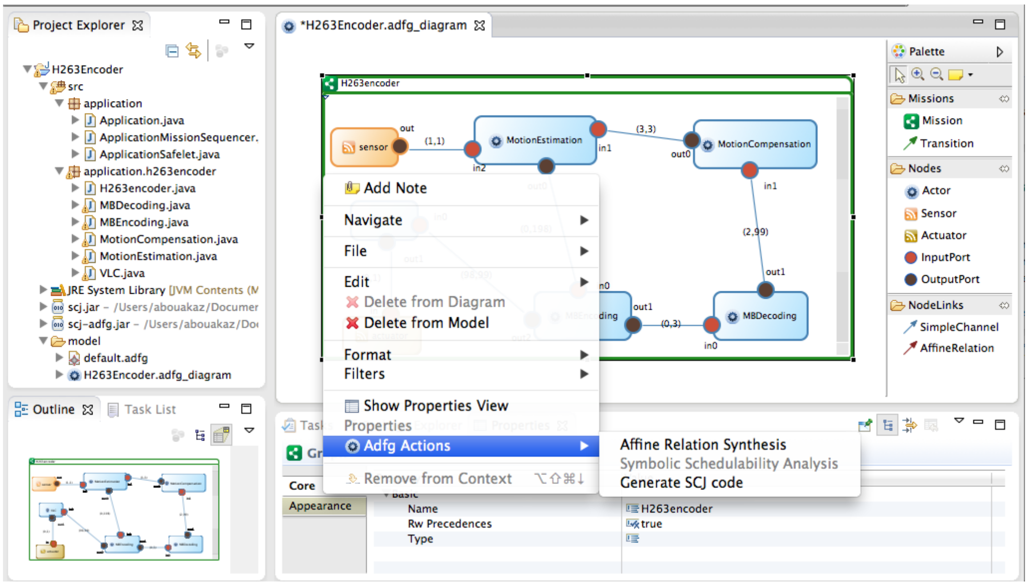


Figure 1. The ADFG tool

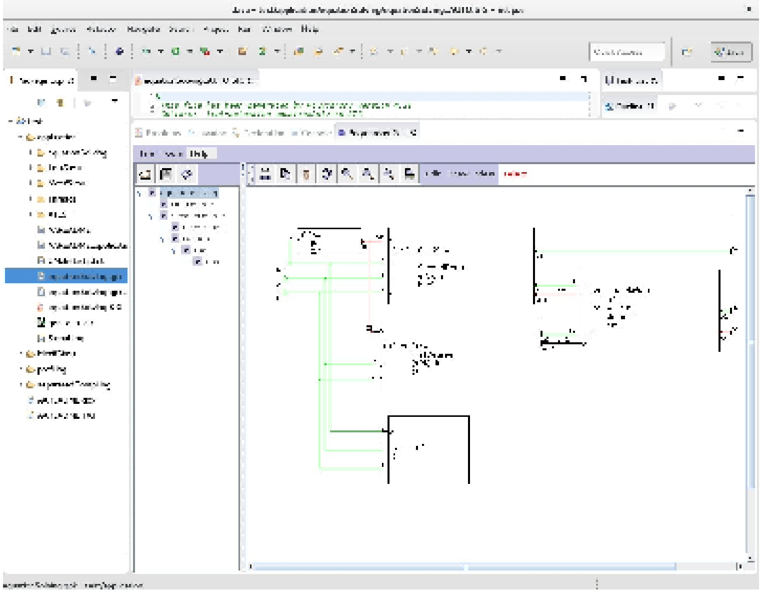


Figure 2. The Eclipse POP Environment

capability to directly import and export Ecore models instead of textual Signal programs, in order to facilitate interaction between components within such a tool-chain. The download site for project POP has opened in 2015 at https://www.polarsys.org/projects/polarsys.pop. 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.

6.3. The Polychrony toolset

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

The Polychrony toolset is an Open Source development environment for critical/embedded systems. It is based on Signal, a real-time polychronous data-flow 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 tool-set provides a formal framework to: validate a design at different levels, by the way of formal verification and/or simulation; refine descriptions in a top-down approach; abstract properties needed for black-box composition; compose heterogeneous components (bottom-up with COTS); generate executable code for various architectures. The Polychrony tool-set 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. Itcan be installed without other components and is distributed under GPL V2 license.
- The Signal GUI, a Graphical User Interface to the Signal toolbox (editor + interactive access to compiling functionalities). It can be used either as a specific tool or as a graphical view under Eclipse. In 2015, it has been transformed and restructured, in order to get a more up-to-date interface allowing multi-window manipulation of programs. It is distributed under GPL V2 license.
- The SSME platform, a front-end to the Signal toolbox in the Eclipse environment. It is distributed under EPL license.

As part of its open-source release, the Polychrony tool-set 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 tool-set. The Polychrony tool-set also provides a large library of Signal programs and examples, user documentations and developer-oriented implementation documents, and facilities to generate new versions. The Polychrony tool-set can be freely downloaded from http://polychrony.inria.fr/. This site, intended for users and for developers, contains executable and source versions of the software for different platforms, user documentation, examples, libraries, scientific publications and implementation documentation. In particular, this is the site for the open-source distribution of Polychrony. The Inria GForge https://gforge.inria.fr contains the whole source of the environment and its documentation. It is intended for developers.

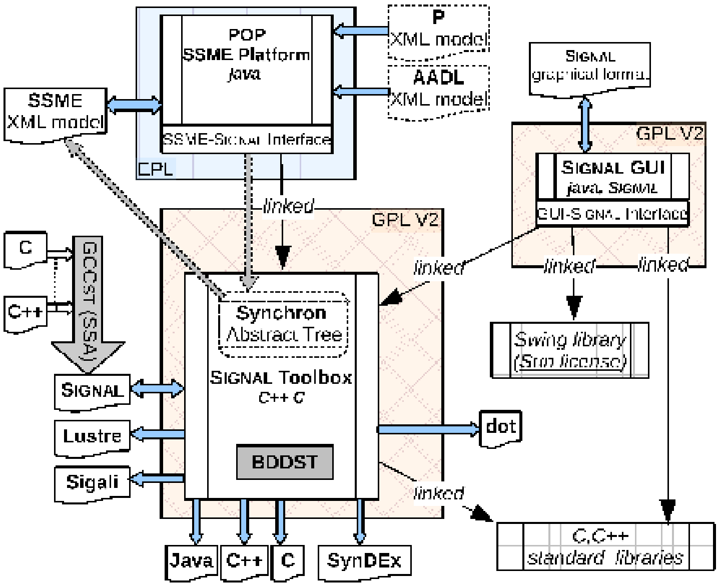


Figure 3. The Polychrony toolset high-level architecture

ANTIQUE Project-Team

6. New Software and Platforms

6.1. APRON

SCIENTIFIC DESCRIPTION

The APRON library is intended to be a common interface to various underlying libraries/abstract domains and to provide additional services that can be implemented independently from the underlying library/abstract domain, as shown by the poster on the right (presented at the SAS 2007 conference. You may also look at:

FUNCTIONAL DESCRIPTION

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.

- Participants: Antoine Miné and Bertrand Jeannet
- Contact: Antoine Miné
- URL: http://apron.cri.ensmp.fr/library/

6.2. Astrée

SCIENTIFIC DESCRIPTION

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.

FUNCTIONAL DESCRIPTION

Astrée is a static analyzer for sequential programs based on abstract interpretation. The Astrée static analyzer aims at proving the absence of runtime errors in programs written in the C programming language.

- Participants: Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné and Xavier Rival
- Partner: CNRS
- Contact: Patrick Cousot
- URL: http://www.astree.ens.fr/

6.3. AstréeA

The AstréeA Static Analyzer of Asynchronous Software

SCIENTIFIC DESCRIPTION

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.

FUNCTIONAL DESCRIPTION

AstréeA is a static analyzer prototype for parallel software based on abstract interpretation. The AstréeA prototype is a fork of the Astrée static analyzer that adds support for analyzing parallel embedded C software.

- Participants: Patrick Cousot, Radhia Cousot, Jérôme Feret, Antoine Miné and Xavier Rival est toujours membre de Inria. logiciels Inria): https://bil.inria.fr/
- Contact: Patrick Cousot
- URL: http://www.astreea.ens.fr/

6.4. ClangML

FUNCTIONAL DESCRIPTION

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.

- Participants: François Berenger, Pippijn Van Steenhoven and Devin Mccoughlin toujours membre de Inria. Inria): https://bil.inria.fr/
- Contact: François Berenger
- URL: https://github.com/Antique-team/clangml/tree/master/clang

6.5. FuncTion

SCIENTIFIC DESCRIPTION

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. 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).

FUNCTIONAL DESCRIPTION

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.

- Participants: Caterina Urban and Antoine Miné
- Contact: Caterina Urban
- URL: http://www.di.ens.fr/~urban/FuncTion.html

6.6. MemCAD

The MemCAD static analyzer
FUNCTIONAL DESCRIPTION

MemCAD is 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 350 small size test cases that are used as regression tests.

- Participants: Antoine Toubhans, Huisong Li, François Berenger and Xavier Rival
- Contact: Xavier Rival
- URL: http://www.di.ens.fr/~rival/memcad.html

6.7. OPENKAPPA

La platte-forme de modélisation OpenKappa
KEYWORDS: Systems Biology - Modeling - Static analysis - Simulation - Model reduction
SCIENTIFIC DESCRIPTION

OpenKappa is a collection of tools to build, debug and run models of biological pathways. It contains a compiler for the Kappa Language, a static analyzer (for debugging models), a simulator, a compression tool for causal traces, and a model reduction tool.

- Participants: Pierre Boutillier, Vincent Danos, Jérôme Feret, Walter Fontana, Russ Harmer, Jean Krivine and Kim Quyen Ly
- Partners: ENS Lyon - Université Paris-Diderot - Harvard Medical School
- Contact: Jérôme Feret
- URL: http://www.kappalanguage.org/

6.8. QUICr

FUNCTIONAL DESCRIPTION

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.

- Participant: Arlen Cox
- Contact: Arlen Cox

6.9. Translation Validation

SCIENTIFIC DESCRIPTION

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.
FUNCTIONAL DESCRIPTION

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 guara 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.ntees that no compiler bug did cause incorrect code to be generated.

- Participant: Xavier Rival
- Contact: Xavier Rival

6.10. Zarith

FUNCTIONAL DESCRIPTION

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 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.

- Participants: Antoine Miné, Xavier Leroy and Pascal Cuoq
- Contact: Antoine Miné
- URL: http://forge.ocamlcore.org/projects/zarith

6.11. CELIA

The MemCAD static analyzer

FUNCTIONAL DESCRIPTION

CELIA is a tool for the static analysis and verification of C programs manipulating dynamic lists. The static analyzer computes for each control point of a C program the assertions which are true (i.e., invariant) at this control point. The specification language is a combination of Separation Logic with a first order logic over sequences of integers. The inferred properties describe the shape of the lists, their size, the relations between the data (or the sum, or the multiset of data) in list cells. The analysis is inter-procedural, i.e., the assertions computed relate the procedure local heap on entry to the corresponding local heap on exit of the procedure. The results of the analysis can provide insights about equivalence of procedures on lists or null pointer dereferencing. The analysis is currently extended to programs manipulating concurrent data structures.

- Participants: Ahmed Bouajjani, Cezara Drăgoi, Constantin Enea, Mihaela Sighireanu
- Contact: Cezara Drăgoi
- URL: http://www.liafa.jussieu.fr/celia/

6.12. DAFT

DAFT

FUNCTIONAL DESCRIPTION

DAFT is a distributed file management system in user-space, with a command-line interface. DAFT is intended at computational scientists, involved in data-intensive, distributed experiments and when no distributed filesystem is available on computing nodes. DAFT is secure; all messages are cryptographically signed and encrypted by default.

- Participants: Francois Berenger and Camille Coti.
- Contact: Francois Berenger and Camille Coti.
- URL: https://github.com/UnixJunkie/daft.

CELTIQUE Project-Team

3. New Software and Platforms

3.1. JSCert

Certified JavaScript
FUNCTIONAL DESCRIPTION

The JSCert project aims to develop a formal understanding of the JavaScript programming language. 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.

- Participants: Martin Bodin and Alan Schmitt
- Partner: Imperial College London
- Contact: Alan Schmitt
- URL: http://jscert.org/

3.2. Javalib

FUNCTIONAL DESCRIPTION

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.

- Participants: Frederic Besson, David Pichardie, Pierre Vittet, Laurent Guillo, Laurent Hubert, Tiphaine Turpin and Nicolas Barre
- Contact: Frederic Besson
- URL: http://sawja.inria.fr/

3.3. SAWJA

Static Analysis Workshop for Java
KEYWORDS: Security - Software - Code review - Smart card
SCIENTIFIC DESCRIPTION

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
FUNCTIONAL DESCRIPTION

Sawja is a toolbox for developing static analysis of Java code in bytecode format. Sawja provides advanced algorithms for reconstructing high-level programme representations. The SawjaCard tool dedicated to JavaCard is based on the Sawja infrastructure and automatically validates the security guidelines issued by AFSCM (http://www.afscm.org/). SawjaCard can automate the code audit process and automatic verification of functional properties.

- Participants: Frederic Besson, David Pichardie and Laurent Guillo
- Partners: CNRS - ENS Rennes
- Contact: Frederic Besson
- URL: http://sawja.inria.fr/

3.4. Timbuk

KEYWORDS: Proof - Ocaml - Program verification - Tree Automata
FUNCTIONAL DESCRIPTION

Timbuk is a collection of tools for achieving proofs of reachability over Term Rewriting Systems and for manipulating Tree Automata (bottom-up non-deterministic finite tree automata)

- Participant: Thomas Genet
- Contact: Thomas Genet
- URL: http://www.irisa.fr/celtique/genet/timbuk/

3.5. CompCertSSA

KEYWORDS: Verified compilation - Single Static Assignment form - Optimization - Coq - OCaml
FUNCTIONAL DESCRIPTION

CompCertSSA is built on top of the C CompCert verified compiler, by adding a SSA-based middle-end (conversion to SSA, SSA-based optimizations, destruction of SSA).

Notably, the middle-end features:

- new important optimizations (Sparse Conditional Constant Propagation, and a coalescing phase on Conventional SSA)
- a generic dominance-based proof framework that rationalizes the proof process
- improved performance regarding compilation time

It is verified in the Coq proof assistant.

- Participant: Delphine Demange, David Pichardie, Yon Fernandez de Retana, Leo Stefanesco
- Contact: Delphine Demange
- URL: http://compcertssa.gforge.inria.fr/

DEDUCTEAM Team

5. New Software and Platforms

5.1. Software of the team

Deducteam develops several kinds of tools or libraries:

- Proof checkers:
  - Dedukti: proof checker for the $\lambda\Pi$-calculus modulo rewriting
  - Sukerujo: extension of Dedukti with syntactic constructions for records, strings, lists, etc.
  - Rainbow: CPF termination certificate verifier
- Tools for translating into Dedukti's proof format proofs coming from various other provers:
  - Coqine translates Coq proofs
  - Focalide translates Focalize proofs
  - Holide translates OpenTheory proofs (HOL-Light, HOL4, ProofPower)
  - Krajono translates Matita proofs
  - Sigmaid translates ς-calculus
- Automated theorem provers:
  - iProverModulo: theorem prover based on polarized resolution modulo
  - SuperZenon: extension of Zenon using superdeduction
  - ZenonArith: extension of Zenon using the simplex algorithm for arithmetic
  - ZenonModulo: extension of Zenon using deduction modulo and producing Dedukti proofs
  - Zipperposition: superposition prover featuring arithmetic and induction
  - HOT: automated termination prover for higher-order rewrite systems
  - Archsat: theorem prover using tableaux-like rules with a SAT core
- Libraries or generation tools:
  - CoLoR: Coq library on rewriting theory and termination
  - Logtk: library for first-order automated reasoning
  - mSat: modular SAT/SMT solver with proof output
  - Moca: generator of construction functions for types with relations on constructors

5.2. Novelties of the year

The main novelties this year are:

- CoLoR has been ported to Coq 8.5.
- F. Blanqui started to develop a prototype for developing Dedukti proofs interactively.

GALLIUM Project-Team

6. New Software and Platforms

6.1. CompCert

Participants: Xavier Leroy [**contact**], Sandrine Blazy [team Celtique], Jacques-Henri Jourdan, Bernhard Schommer [AbsInt GmbH].

The CompCert project investigates the formal verification of realistic compilers usable for critical embedded software. Such verified compilers come with a mathematical, machine-checked proof that the generated executable code behaves exactly as prescribed by the semantics of the source program. By ruling out the possibility of compiler-introduced bugs, verified compilers strengthen the guarantees that can be obtained by applying formal methods to source programs. AbsInt Angewandte Informatik GmbH sells a commercial version of CompCert with long-term maintenance.

- URL: http://compcert.inria.fr/ (academic), http://www.absint.com/compcert/ (commercial).

6.2. Diy

Participants: Luc Maranget [**contact**], Jade Alglave [Microsoft Research, Cambridge].

The **diy** suite (for “Do It Yourself”) 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, ARM or generic (LISA) assembler that can thus be generated from concise specifications, run on hardware, or simulated on top of memory models. Test results can be handled and compared using additional tools. Recent versions also take a subset of the C language as input, so as to test and simulate the C11 model. Recent releases (“Seven”) provide a new license (Cecill-B), a simplified build process and numerous features, including a simple macro system that connects the C input language and LISA annotations.

- URL: http://diy.inria.fr/

6.3. Menhir

Participants: François Pottier [**contact**], Yann Régis-Gianas [Université Paris Diderot].

Menhir is a LR(1) parser generator for the OCaml programming language. That is, Menhir compiles LR(1) grammar specifications down to OCaml code.

- URL: http://gallium.inria.fr/~fpottier/menhir/

6.4. OCaml

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

The OCaml language is a functional programming language that combines safety with expressiveness through the use of a precise and flexible type system with automatic type inference. The OCaml system is a comprehensive implementation of this language, featuring two compilers (a bytecode compiler, for fast prototyping and interactive use, and a native-code compiler producing efficient machine code for x86, ARM, PowerPC and SPARC), a debugger, a documentation generator, a compilation manager, a package manager, and many libraries contributed by the user community.

- URL: http://ocaml.org/

6.5. OPAM Builder

Participant: Fabrice Le Fessant.

OPAM Builder checks in real time the installability on a computer of all packages after any modification of the OPAM repository. To achieve this result, it uses smart mechanisms to compute incremental differences between package updates, to be able to reuse cached compilations, and go down from quadratic complexity to linear complexity.

- URL: http://github.com/OCamlPro/opam-builder

6.6. PASL

Participants: Michael Rainey [**contact**], Arthur Charguéraud, Umut Acar.

PASL is a C++ library for writing parallel programs targeting the broadly available multicore computers. The library provides a high level interface and can still guarantee very good efficiency and performance, primarily due to its scheduling and automatic granularity control mechanisms.

- URL: http://deepsea.inria.fr/pasl/

6.7. TLAPS

Participants: Damien Doligez [**contact**], Stefan Merz [team Veridis], Martin Riener [team Veridis].

TLAPS 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.

- URL: https://tla.msr-inria.inria.fr/tlaps/content/Home.html

6.8. Zenon

Participants: Damien Doligez [**contact**], Guillaume Bury [CNAM], David Delahaye [CNAM], Pierre Halmagrand [team Deducteam], Olivier Hermant [MINES ParisTech].

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).

- URL: http://zenon-prover.org/

MARELLE Project-Team

4. New Software and Platforms

4.1. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

FUNCTIONAL DESCRIPTION

Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

- Participants: Benjamin Gregoire, Enrico Tassi, Bruno Barras, Yves Bertot, Pierre Boutillier, Xavier Clerc, Pierre Courtieu, Maxime Dénès, Stephane Glondu, Vincent Gross, Hugo Herbelin, Pierre Letouzey, Assia Mahboubi, Julien Narboux, Jean-Marc Notin, Christine Paulin-Mohring, Pierre-Marie Pedrot, Loic Pottier, Matthias Puech, Yann Regis-Gianas, François Ripault, Matthieu Sozeau, Arnaud Spiwack, Pierre-Yves Strub, Benjamin Werner, Guillaume Melquiond and Jean-Christophe Filliatre
- Partners: CNRS - ENS Lyon - Université Paris-Diderot - Université Paris-Sud
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

The Marelle team, in collaboration with the pi.r2 team, plays an important role in the development of Coq. During this year, we contributed to the 8.6 version of Coq, released in December. As the *release manager*, Maxime Dénès led the implementation of a time-based release process, aiming at shorter and more predicitible release cycles. We successfully transitioned to 10-month cycles and hope to soon move to 6-month cycles, making it easier for users to benefit from the latest improvements.

At a more detailed level, members of the Marelle team attended the Coq developer meetings (organized in Paris by Maxime Dénès and Matthieu Sozeau) and contributed to the development of Coq concerning bug fixes for virtual machine execution (Benjamin Grégoire and Maxime Dénès), cleaning up the API for plug-in developers (Matej Košík), improving the State Transaction Machine (Enrico Tassi), setting up a package index based on OPAM (Enrico Tassi), introducing a system to discuss Coq Enhancement Proposals (Enrico Tassi), and implementing a new configurable system of warnings (Maxime Dénès).

We supervise of an engineer working at MIT on questions related to efficient proof construction and proof development environments, in cooperation with researchers from the pi.r2 team. The collaboration with MIT was also an occasion to reflect on the licence framework governing collaborations around the Coq system.

We also prepared the set-up of a consortium to gather intensive users and contributors to the development of Coq. This was an occasion to work with the promotors of the InriaSoft structure which is expected to host the consortium in the long run.

4.2. Easycrypt

FUNCTIONAL DESCRIPTION

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.

- Participants: Gilles Barthe, Benjamin Gregoire and Pierre-Yves Strub
- Contact: Benjamin Grégoire
- URL: https://www.easycrypt.info/trac/

This year, development on this software system concerned the development of new logical settings to work on differential privay problems: a Hoare logic based on union bound and a logic based on probabilistic couplings.

4.3. Math-Components

Mathematical Components library
FUNCTIONAL DESCRIPTION

The Mathematical Components library is a set of Coq libraries that cover the mechanization of the proof of the Odd Order Theorem.

- Participants: Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, Francois Garillot, Georges Gonthier, Stéphane Le Roux, Assia Mahboubi, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, Laurent Théry and Russell O'Connor
- Contact: Assia Mahboubi
- URL: http://www.msr-inria.fr/projects/mathematical-components-2/

This year we contributed to the library by adding a new module to cover finite sets within potentially infinite finite types, organizing tutorials and schools to teach its usage:

- in January in Sophia Antipolis (one-week format) https://team.inria.fr/marelle/en/advanced-coq-winter-school-2016/ (organized by Enrico Tassi, with contributions by Cyril Cohen, Laurence Rideau, Laurent Théry)
- in August in Nancy (one-day tutorial format, colocated with the ITP conference, organized by Assia Mahboubi and Enrico Tassi, with contributions by Yves Bertot, Cyril Cohen, and Laurent Théry) https://github.com/math-comp/wiki/wiki/tutorial-itp2016
- in November in Sophia Antipols https://team.inria.fr/marelle/en/advanced-coq-winter-school-2016-2017/ (organized by Enrico Tassi, with contributions by Yves Bertot, Cyril Cohen, Laurence Rideau).

4.4. Ssreflect

FUNCTIONAL DESCRIPTION

Ssreflect is a tactic language extension to the Coq system, developed by the Mathematical Components team.

- Participants: Cyril Cohen, Yves Bertot, Laurence Rideau, Enrico Tassi, Laurent Thery, Assia Mahboubi and Georges Gonthier
- Contact: Yves Bertot
- URL: http://ssr.msr-inria.inria.fr/

This year we mainly performed maintenance operations on this software extension to the Coq system (Enrico Tassi).

4.5. Zoocrypt

FUNCTIONAL DESCRIPTION

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 year, we extended the tool to be able to deal with schemes based on cyclic groups and bilinear maps.

- Participants: Benjamin Gregoire, Gilles Barthe and Pierre-Yves Strub
- Contact: Benjamin Grégoire
- URL: https://www.easycrypt.info/zoocrypt/

MEXICO Project-Team

6. New Software and Platforms

6.1. DarkSider

FUNCTIONAL DESCRIPTION

DarkSider computes anti-alignments between a Petri net model and a log of observed traces, as described in [15], [25].

- Participant: Thomas Chatain
- Contact: Thomas Chatain
- URL: http://www.lsv.ens-cachan.fr/~chatain/darksider/

6.2. COSMOS

FUNCTIONAL DESCRIPTION

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++

- Participants: Benoît Barbot, Hilal Djafri, Paolo Ballarini, Marie Duflot-Kremer and Serge Haddad
- Contact: Hilal Djafri
- URL: http://www.lsv.ens-cachan.fr/~barbot/cosmos/

6.3. CosyVerif

FUNCTIONAL DESCRIPTION

CosyVerif 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.

- Participants: Serge Haddad, Fabrice Kordon, Laure Petrucci and Alban Linard
- Partners: LIP6 - LIPN (Laboratoire d'Informatique de l'Université Paris Nord) - LSV
- Contact: Serge Haddad
- URL: http://www.cosyverif.org/

6.4. Mole

FUNCTIONAL DESCRIPTION

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.

- Participant: Stefan Schwoon
- Contact: Stefan Schwoon
- URL: http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/

PARSIFAL Project-Team

6. New Software and Platforms

6.1. Abella

FUNCTIONAL DESCRIPTION

Abella is an interactive theorem prover for reasoning about computations given as relational specifications. Abella is particuarly well suited for reasoning about binding constructs.

- Participants: Dale Miller, Olivier Savary-Bélanger, Mary Southern, Yuting Wang, Kaustuv Chaudhuri, Matteo Cimini and Gopalan Nadathur
- Partner: Department of Computer Science and Engineering, University of Minnesota
- Contact: Kaustuv Chaudhuri
- URL: http://abella-prover.org/

6.2. Bedwyr

Bedwyr - A proof search approach to model checking
FUNCTIONAL DESCRIPTION

Bedwyr is a generalization of logic programming that allows model checking directly on syntactic expression that possibly contain bindings. This system, written in OCaml, is a direct implementation of two recent advances in the theory of proof search.

It is possible to capture both finite success and finite failure in a sequent calculus. Proof search in such a proof system can capture both may and must behavior in operational semantics. Higher-order abstract syntax is directly supported using term-level lambda-binders, the nabla quantifier, higher-order pattern unification, and explicit substitutions. These features allow reasoning directly on expressions containing bound variables.

The distributed system comes with several example applications, including the finite pi-calculus (operational semantics, bisimulation, trace analyses, and modal logics), the spi-calculus (operational semantics), value-passing CCS, the lambda-calculus, winning strategies for games, and various other model checking problems.

- Participants: Quentin Heath, Roberto Blanco, and Dale Miller
- Contact: Quentin Heath
- URL: http://slimmer.gforge.inria.fr/bedwyr/

6.3. Checkers

Checkers - A proof verifier
KEYWORDS: Proof - Certification - Verification
FUNCTIONAL DESCRIPTION

Checkers is a tool in Lambda-prolog for the certification of proofs. Checkers consists of a kernel which is based on LKF and is based on the notion of ProofCert.

- Participants: Tomer Libal, Giselle Machado Nogueira Reis and Marco Volpe
- Contact: Tomer Libal
- URL: https://github.com/proofcert/checkers

6.4. Psyche

Proof-Search factorY for Collaborative HEuristics

Functional Description

Psyche is a modular platform for automated or interactive theorem proving, programmed in OCaml and built on an architecture (similar to LCF) where a trusted kernel interacts with plugins. The kernel offers an API of proof-search primitives, and plugins are programmed on top of the API to implement search strategies. This architecture is set up for pure logical reasoning as well as for theory-specific reasoning, for various theories.

The major effort in 2016 was the release of version 2.1 that allows the combination of theories, integrating and subsuming both the Nelson-Oppen methodology [79] and the *model constructing satisfiability* (MCSAT) methodology recently proposed by De Moura and Jovanovic [89], [65].

- Participants: Assia Mahboubi, Jean-Marc Notin and Stéphane Graham-Lengrand
- Contact: Stéphane Graham-Lengrand
- URL: http://www.lix.polytechnique.fr/~lengrand/Psyche/

PI.R2 Project-Team

4. New Software and Platforms

4.1. Coq

KEYWORDS: Proof - Certification - Formalisation

FUNCTIONAL DESCRIPTION

Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

- Closest participants: Benjamin Grégoire, Enrico Tassi, Bruno Barras, Yves Bertot, Pierre Courtieu, Maxime Dénès, Hugo Herbelin, Matej Košík, Pierre Letouzey, Assia Mahboubi, Cyprien Mangin, Guillaume Melquiond, Jean-Marc Notin, Pierre-Marie Pédrot, Yann Régis-Gianas, Matthieu Sozeau, Arnaud Spiwack, Théo Zimmermann.
- Partners: CNRS - ENS Lyon - Université Paris-Diderot - Université Paris-Sud
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

4.1.1. Coq 8.6

The 8.6 version of Coq was released in December 2016. It initiates a time-based release cycle and concentrates on a smaller set of features than Coq 8.5 for which compatibility and testing were done more intensively. In the πr^2 team, Hugo Herbelin, Cyprien Mangin, Théo Zimmermann and Matthieu Sozeau contributed to the coordination of the development, to the discussion of the roadmap, to the implementation of the features, and to many bugfixes.

Matthieu Sozeau followed up his work on universe polymorphism making the explicit annotation system more accessible and resolving issues in the minimization algorithm used during refinement, resulting in a more predictable system. These improvements were used in the Coq/HoTT library for Homotopy Type Theory, which is described in an upcoming article [26].

Matthieu Sozeau implemented a new variant of the proof-search tactic for typeclasses that is set to replace the existing auto and eauto tactics in the following version. The new variant fully benefits from the features of the underlying proof engine, and allows much more control on proof-search (patterns used consistently, modes for triggering hints, ...). It is at the basis of the work of Théo Zimmermann described below.

4.1.2. The Equations plugin

Cyprien Mangin and Matthieu Sozeau continued work on the Equations plugin, modularizing it so that the use of axioms can be minimized, and making it compatible with developments in Homotopy Type Theory. To achieve this, it has moved to a simplification engine in ML based on telescopes and is able to produce axiom-free proofs of the examples that were previously implicitly using them. This work will be presented at the POPL workshop Type-Theoretic Tools (TTT), next January 2017.

4.1.3. Maintenance

Among other contributions, Hugo Herbelin, Pierre Letouzey, Matej Košík and Matthieu Sozeau worked at the maintenance of the system.

In particular, Pierre Letouzey vastly reworked the build mechanism of Coq, taking advantage of code evolutions driven by Pierre-Marie Pédrot. Pierre Letouzey also administrated (and improved) several machines or systems that are critical for the Coq community (web server, build test server, git repositories ...), in coordination with Inria's SIC support team.

Matej Košík developed a new benchmarking infrastructure based on Jenkins and continuous integration (http://ci.inria.fr). It allows easily testing any developer branch on the benchmark suite prior to integration to the main archive.

4.1.4. Coordination and animation

After 10 years coordinating the Coq development team, Hugo Herbelin handed over the coordination to Matthieu Sozeau.

A Coq working group is organised every two months (5 times a year). Discussions about the development happen, in particular, on `coq-dev@inria.fr`, Coq's GitHub http://github.com/coq and http://coq.inria.fr/bugs. This year, a week-long working group organized in Sophia-Antipolis was devoted to the 8.6 roadmap discussion.

4.1.5. Documentation and stabilization of Coq's programming interface

Matej Košík worked on the programming interfaces of Coq, starting to isolate a subset of key functions to be used by Coq plugin developers.

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 develops the "Hacking Dojo", a web platform to automatically grade programming exercises. The platform is now used in several courses of the University Paris Diderot. Yann Régis-Gianas develops a reference implementation of a syntactic analyzer for the POSIX shell programming language. This analyzer is used by the Colis project to analyze the scripts embedded in the packages of the Debian GNU/Linux distribution. In collaboration with Beta Ziliani (LIIS, Cordoba, Argentine), Yann Régis-Gianas, Béatrice Carré and Jacques-Pascal Deplaix develop MetaCoq, an extension of Coq to use Coq as a metalanguage for itself.

Yves Guiraud has updated the Catex tool for Latex, whose purpose is to automatise the production of string diagrams from algebraic expressions http://www.irif.fr/~guiraud/catex/catex.zip. Yves Guiraud collaborates with Samuel Mimram (LIX) to develop the prototype Rewr that implements several algorithms developed in the "Effective higher-dimensional algebra" research direction, including the homotopical completion-reduction procedure of [10]. An online version is available at http://www.lix.polytechnique.fr/Labo/Samuel.Mimram/rewr.

SUMO Project-Team

6. New Software and Platforms

6.1. Active Workspaces

KEYWORDS: Guarded attribute grammar - Active workspace - Artifact centric workflow system

SCIENTIFIC DESCRIPTION

Tool for computer supported cooperative work where a user's workspace is given by an active structured repository containing the pending tasks together with information needed to perform the tasks. Communication between active workspaces is asynchronous using message passing. The tool is based on the model of guarded attribute grammars [44]. Late in 2015 Éric Badouel produced in Haskell a software prototype implementing active workspaces based on Guarded Attribute Grammars (GAGs).

Concurrently, Christophe Morvan was beginning a startup project consisting in making on-line collective decision making tools: *Open Agora*. This project included collaboration workspaces for people participating in constructing possible decisions. There was a natural connection between the prototype, and the startup project.

In order to make industrial use of the GAG prototype, Olivier Bache (already involved in the Open agora project) applied to a 6 month InriaHub program (between April and September 2016). During these 6 months he bundeled the prototype into an API (also programmed in Haskell) and developped a web infrastructure, based on the PHP framework, to allow the interaction with Active Workspaces in a browser. This developpement will be licenced to Open Agora SAS after its creation expected in January 2017.

FUNCTIONAL DESCRIPTION

Prototype in Haskell of user's active workspaces based on Guarded Attribute Grammars.

- Author: Eric Badouel
- Contact: Eric Badouel
- URL: http://people.rennes.inria.fr/Eric.Badouel/Research/ActiveWorkspaces.html

6.2. SIMSTORS

SIMSTORS is a simulator for regulated stochastic timed Petri nets. These Petri nets are a variant of stochastic and timed nets, whose execution is controlled by a regulation policy an a predetermined theoretical schedule. The role of the regulation policy is to control the system to realize the schedule with the best possible precision. This software allows not only for step by step simulation, but also for performance analysis of systems such as production cells or train systems.

SIMSTORS was used successfully during a collaboration with Alstom transport to model existing urban railway systems and their regulation schemes. Alstom transport is willing to transfer this software and use it during early design phase of regulation algorithms in their metro lines. This year, the software has been extended to consider headway management.

- Participants: Loïc Hélouët and Karim Kecir
- Contact: Loïc Hélouët
- URL: http://www.irisa.fr/sumo/Software/SIMSTORS/

TOCCATA Project-Team

6. New Software and Platforms

6.1. Alt-Ergo

Automated theorem prover for software verification
KEYWORDS: Software Verification - Automated theorem proving
FUNCTIONAL DESCRIPTION

Alt-Ergo is an automatic solver of formulas based on SMT technology. It is especially designed to prove mathematical formulas generated by program verification tools, such as Frama-C for C programs, or SPARK for Ada code. Initially developed in Toccata research team, Alt-Ergo's distribution and support are provided by OCamlPro since September 2013.

- Participants: Sylvain Conchon, Evelyne Contejean, Mohamed Iguernelala, Stephane Lescuyer and Alain Mebsout
- Partner: OCamlPro
- Contact: Sylvain Conchon
- URL: http://alt-ergo.lri.fr

6.2. CFML

Interactive program verification using characteristic formulae
KEYWORDS: Coq - Software Verification - Deductive program verification - Separation Logic
FUNCTIONAL DESCRIPTION

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.

- Contact: Arthur Chargueraud
- URL: http://www.chargueraud.org/softs/cfml/

6.3. Coq

KEYWORDS: Proof - Certification - Formalisation
FUNCTIONAL DESCRIPTION

Coq provides both a dependently-typed functional programming language and a logical formalism, which, altogether, support the formalisation of mathematical theories and the specification and certification of properties of programs. Coq also provides a large and extensible set of automatic or semi-automatic proof methods. Coq's programs are extractible to OCaml, Haskell, Scheme, ...

- Participants: Benjamin Gregoire, Enrico Tassi, Bruno Barras, Yves Bertot, Pierre Boutillier, Xavier Clerc, Pierre Courtieu, Maxime Denes, Stephane Glondu, Vincent Gross, Hugo Herbelin, Pierre Letouzey, Assia Mahboubi, Julien Narboux, Jean-Marc Notin, Christine Paulin-Mohring, Pierre-Marie Pedrot, Loic Pottier, Matthias Puech, Yann Regis-Gianas, François Ripault, Matthieu Sozeau, Arnaud Spiwack, Pierre-Yves Strub, Benjamin Werner, Guillaume Melquiond and Jean-Christophe Filliatre
- Partners: CNRS - ENS Lyon - Université Paris-Diderot - Université Paris-Sud
- Contact: Hugo Herbelin
- URL: http://coq.inria.fr/

6.4. CoqInterval

Interval package for Coq
KEYWORDS: Interval arithmetic - Coq
FUNCTIONAL DESCRIPTION

CoqInterval is a library for the proof assistant Coq. CoqInterval provides a method for proving automatically the inequality of two expression of real values.

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.

The Marelle team developed 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.

- Participants: Guillaume Melquiond, Erik Martin Dorel, Nicolas Brisebarre, Miora Maria Joldes, Micaela Mayero, Jean Michel Muller, Laurence Rideau and Laurent Thery
- Contact: Guillaume Melquiond
- URL: http://coq-interval.gforge.inria.fr/

6.5. Coquelicot

The Coquelicot library for real analysis in Coq
KEYWORDS: Coq - Real analysis
FUNCTIONAL DESCRIPTION

Coquelicot is library aimed for supporting real analysis in the Coq proof assistant. It 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.

- Participants: Sylvie Boldo, Catherine Lelay and Guillaume Melquiond
- Contact: Sylvie Boldo
- URL: http://coquelicot.saclay.inria.fr/

6.6. Cubicle

The Cubicle model checker modulo theories
KEYWORDS: Model Checking - Software Verification
FUNCTIONAL DESCRIPTION

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.

- Participants: Sylvain Conchon and Alain Mebsout
- Contact: Sylvain Conchon
- URL: http://cubicle.lri.fr/

6.7. Flocq

The Flocq library for formalizing floating-point arithmetic in Coq
KEYWORDS: Floating-point - Arithmetic code - Coq
FUNCTIONAL DESCRIPTION

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. 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.

- Participants: Guillaume Melquiond and Sylvie Boldo
- Contact: Sylvie Boldo
- URL: http://flocq.gforge.inria.fr/

6.8. Gappa

The Gappa tool for automated proofs of arithmetic properties
KEYWORDS: Floating-point - Arithmetic code - Software Verification - Constraint solving
FUNCTIONAL DESCRIPTION

Gappa is a tool intended to help verifying and formally proving properties on numerical programs dealing with floating-point or fixed-point arithmetic. It has been used to write robust floating-point filters for CGAL and it is used to certify elementary functions in CRlibm. While Gappa is intended to be used directly, it can also act as a backend prover for the Why3 software verification plateform or as an automatic tactic for the Coq proof assistant.

- Contact: Guillaume Melquiond
- URL: http://gappa.gforge.inria.fr/

6.9. Why3

The Why3 environment for deductive verification
KEYWORDS: Formal methods - Trusted software - Software Verification - Deductive program verification
FUNCTIONAL DESCRIPTION

Why3 is an environment for deductive program verification. It provides a rich language for specification and programming, called WhyML, and relies on external theorem provers, both automated and interactive, to discharge verification conditions. Why3 comes with a standard library of logical theories (integer and real arithmetic, Boolean operations, sets and maps, etc.) and basic programming data structures (arrays, queues, hash tables, etc.). A user can write WhyML programs directly and get correct-by-construction OCaml programs through an automated extraction mechanism. WhyML is also used as an intermediate language for the verification of C, Java, or Ada programs.

- Participants: Jean-Christophe Filliatre, Claude Marche, Guillaume Melquiond, Andriy Paskevych, François Bobot, Martin Clochard and Levs Gondelmans
- Partners: CNRS - Université Paris-Sud
- Contact: Claude Marche
- URL: http://why3.lri.fr/

VERIDIS Project-Team

6. New Software and Platforms

6.1. The Nunchaku Higher-Order Model Finder

FUNCTIONAL DESCRIPTION

Nunchaku is a model finder for higher-order logic, with dedicated support for various definitional principles. It is designed to work as a backend for various proof assistants and to use state-of-the-art model finders and other solvers as backends.

In 2016, the first three versions of the tools were released (0.1 through 0.3). The Isabelle2016-1 release includes Nunchaku as well as the frontend that bridges the gap between the proof assistant and the model finder. Work has commenced on a Coq frontend [28] and a TLA$^+$ frontend. Currently, the backends CVC4, Kodkod, and Paradox are supported.

- Participants: Jasmin Blanchette and Simon Cruanes
- Contact: Jasmin Blanchette
- URL: https://github.com/nunchaku-inria

6.2. The Redlog Computer Logic System

FUNCTIONAL DESCRIPTION

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 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.

In 2016 there was significant progress with the generation of models for real satisfiability problems [15]. When obtained via quantifier elimination by virtual substitutions, such models contain in general non-standard numbers like positive infinitesimal and infinite values. In an efficient post-processing step Redlog now generates standard models.

- Participants: Thomas Sturm and Marek Kosta
- Contact: Thomas Sturm
- URL: http://www.redlog.eu/

6.3. The SPASS automated theorem prover

FUNCTIONAL DESCRIPTION

The classic SPASS is an automated theorem prover based on superposition that handles first-order logic with equality and several extensions for particular classes of theories. With version SPASS 3.9 we have stopped the development of the classic prover and have started the bottom-up development of SPASS 4.0 that will actually be a workbench of automated reasoning tools.

In 2016 we have made available for the first time our LIA solver SPASS-IQ. Furthermore, we have developed a state-of-the-art SAT solver SPASS-SATT that will be available in 2017.

- Contact: Christoph Weidenbach
- URL: http://www.spass-prover.org/

6.4. TLAPS, the TLA+ Proof System

FUNCTIONAL DESCRIPTION

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 1.4.3 of TLAPS was released in June 2015, it is distributed under a BSD-like license. The prover fully handles the non-temporal part of TLA$^+$. Basic temporal logic reasoning is supported through an interface with a decision procedure for propositional temporal logic that performs on-the-fly abstraction of first-order subformulas. Symmetrically, subformulas whose main operator is a connective of temporal logic are abstracted before being sent to backends for first-order logic.

A complete rewrite of the proof manager is ongoing. Its objectives are a cleaner interaction with the standard TLA$^+$ front-ends, in particular SANY, the standard parser and semantic analyzer. This is necessary for extending the scope of the fragment of TLA$^+$ that is handled by TLAPS, such as full temporal logic and module instantiation.

TLAPS has been used in several case studies, including the proof of determinacy of PharOS [21] and the verification of the Pastry routing protocol [22]. These case studies feed back into the development of the proof system and of its standard library.

- Contact: Stephan Merz
- URL: https://tla.msr-inria.inria.fr/tlaps/content/Home.html

6.5. The veriT Solver

SCIENTIFIC DESCRIPTION

veriT comprises a SAT solver, a congruence closure based decision procedure for uninterpreted symbols, a simplex-based decision procedure for linear arithmetic, and instantiation-based quantifier handling.

FUNCTIONAL DESCRIPTION

VeriT is an open, trustable and efficient SMT (Satisfiability Modulo Theories) solver, featuring efficient decision procedure for uninterpreted symbols and linear arithmetic, and quantifier reasoning.

Efforts in 2016 have been focused on non-linear arithmetic reasoning and quantifier handling. The reasoning capabilities of veriT have been significantly improved along those two axes, but non-linear arithmetic reasoning is not yet stable.

The veriT solver participated in the SMT competition SMT-COMP 2016 with good results.

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, it is integrated within the Atelier B.

- Participants: Pascal Fontaine, David Déharbe and Haniel Barbosa
- Partners: Université de Lorraine - Federal University of Rio Grande do Norte
- Contact: Pascal Fontaine
- URL: http://www.veriT-solver.org

CARTE Team (section vide)

COMETE Project-Team

6. New Software and Platforms

6.1. libqif - A Quantitative Information Flow C++ Toolkit Library

Participants: Konstantinos Chatzikokolakis [correspondant], Susheel Suresh, Tymofii Prokopenko.

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 was under constant development in 2016 with several new features added this year. The project's git repository shows for this year 77 commits by 2 contributors, containing 5697 line additions and 4067 line removals. Some of the techniques already implemented are:

- Standard leakage measures: Shannon, min-entropy, guessing entropy
- Measures from the g-leakage framework [26]
- Channel factorization
- Hyper distribution produced by a channel run under a prior
- Standard differential privacy mechanisms from the literature
- The planar Laplace mechanism of [27]
- The planar Geometric mechanism
- The tight-constraints mechanism of [29] (also with equality constraints)
- Optimal mechanism construction under DP
- The standard Kantorovich metric as well as the multiplicative variant from [28]
- Additive capacity for specific prior over all gain functions [2]
- 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
- Simple installation in OSX via Homebrew

Many more are scheduled to be added in the near future.

6.2. D-SPACES - constraint systems with space and extrusion operators

Participants: Frank Valencia, Yamil Salim Perchy.

http://www.lix.polytechnique.fr/~perchy/d-spaces/

D-SPACES is an implementation of constraint systems with space and extrusion operators. Constraint systems are algebraic models that allow for a semantic language-like representation of information in systems where the concept of space is a primary structural feature. We give this information mainly an epistemic interpretation and consider various agents as entities acting upon it. D-SPACES is coded as a c++11 library providing implementations for constraint systems, space functions and extrusion functions. The interfaces to access each implementation are minimal and thoroughly documented. D-SPACES also provides property-checking methods as well as an implementation of a specific type of constraint systems (a boolean algebra). This last implementation serves as an entry point for quick access and proof of concept when using these models. In [22] an illustrative example of using the library is given, in the form of a small social network where users post their beliefs and utter their opinions.

6.3. Trace Slicer for Timed Concurrent Constraint Programming

Participants: Catuscia Palamidessi, Carlos Olarte.

http://subsell.logic.at/slicer/

Concurrent Constraint Programming (CCP) is a declarative model for concurrency aimed at specifying reactive systems, i.e. systems that continuously interact with the environment. Some previous works have developed (approximated) declarative debuggers for CCP languages. However, the task of debugging concurrent programs remains difficult. This tool is a companion for the existing debugging techniques. Slicing in our proposal consists of considering partial computations, which show the presence of bugs. Often, the quantity of information in a trace is overwhelming, and the user gets easily lost, since she cannot focus on the sources of the bugs. Our slicer allows for marking part of the state of the computation and assists the user to eliminate most of the redundant information in order to highlight the errors. See [19] for further details.

DICE Team

5. New Software and Platforms

5.1. BitBallot

FUNCTIONAL DESCRIPTION

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.

- Contact: Stéphane Grumbach, Ste´phane Frénot, Damien Reimert, Robert Riemann

5.2. C3PO

FUNCTIONAL DESCRIPTION

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.

- Contact: Ste´phane Frénot, Damien Reimert

5.3. Jumplyn

Jumplyn is a student project delivery platform. It offers a service based on three features: the ongoing management of the project, resources recommendation, and enhancement of the activity. Like any intermediation platform, it speaks directly to its users, students, and puts them in relation to relevant information.

FUNCTIONAL DESCRIPTION

Jumplyn is a student project delivery platform. It offers a service based on three features: the ongoing management of the project, resources recommendation, and enhancement of the activity. Like any intermediation platform, it speaks directly to its users, students, and puts them in relation to relevant information.

- Contact: Ste´phane Frénot, Ste´phane Grumbach
- URL: http://www.jumplyn.com

PESTO Project-Team

6. New Software and Platforms

6.1. 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. The tool also includes the possibility for checking everlasting indistinguishability properties [32].

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, and the addition of support for exclusive or.

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

6.2. 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.

6.3. Belenios

In collaboration with the Caramba 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 us 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 (SED Team). Since 2015, it is used by CNRS for remote election among its councils and since 2016, it used by Inria to elect representatives in the "comités de centre" of each Inria center. It has also been used to elect the leader of the GdR-IM working groups C2 and Calcul Formel. It has also been used in smaller elections (e.g., to choose an invited speaker).

6.4. Tamarin

The *TAMARIN* prover is a security protocol verification tool that supports both falsification and unbounded verification of security protocols specified as multiset rewriting systems with respect to (temporal) first-order properties and a message theory that models Diffie-Hellman exponentiation combined with a user-defined subterm-convergent rewriting theory.

Its main advantages are its ability to handle stateful protocols and its interactive proof mode. Moreover, it has recently been extended to verify equivalence properties.

The tool is developed jointly by the PESTO team, the Institute of Information Security at ETH Zurich, and the University of Oxford.

TAMARIN is freely available at http://tamarin-prover.github.io/. In a joint effort, the partners wrote and published a user manual in 2016, available from the same website.

6.5. 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. *TAMARIN* has also bee extended with dedicated heuristics that exploit the form of translated rules and favour termination.

SAPIC offers support for 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 also allows us to verify liveness properties and a recent extension adds a notion of location and reporting used for modelling trusted execution environments. It has been successfully applied on several case studies including the Yubikey authentication protocol, extensions of the PKCS#11 standard and fair exchange protocols.

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

PRIVATICS Project-Team

5. New Software and Platforms

5.1. FECFRAME

FEC Framework following RFC 6363 specifications (https://datatracker.ietf.org/doc/rfc6363/)
KEYWORDS: Error Correction Code - Content delivery protocol - Robust transmission
FUNCTIONAL DESCRIPTION

This sofware implements the FECFRAME IETF standard (RFC 6363) co-authored by V. ROCA, and is compliant with 3GPP specifications for mobile terminals. It enables the simultaneous transmission of multimedia flows to one or several destinations, while being robust to packet erasures that happen on wireless networks (e.g., 4G or Wifi). This software relies on the OpenFEC library (the open-source http://openfec.org version or the commercial version) that provides the erasure correction codes (or FEC) and thereby offer robustness in front of packet erasures.

- Author: Vincent Roca
- Contact: Vincent Roca
- URL: http://openfec.org/

5.2. Mobilitics

FUNCTIONAL DESCRIPTION

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.

- Authors: Jagdish Achara, James-Douglass Lefruit, Claude Castelluccia, Vincent Roca, Gwendal Le Grand, Geoffrey Delcroix, Franck Baudot and Stéphane Petitcolas
- Contact: Claude Castelluccia
- URL: https://team.inria.fr/privatics/mobilitics/

5.3. MyTrackingChoices

KEYWORDS: Privacy - User control
FUNCTIONAL DESCRIPTION

This extension lets you control how you are being tracked on the Internet. It allows you to choose the categories (e.g., health, adult) of the websites where you don't want to be tracked on. When you browse the web, your visited webpages will be categorized on the fly and, depending on your choices, the extension will block the trackers (webpage by webpage) or not.

Existing anti-tracking (Ghostery, Disconnect etc.) and ad-blocking (AdBlock Plus etc.) tools block almost ALL trackers and as a result, ads. This has a negative impact on the Internet economy because free services/content on the Internet are fuelled by ads. As a result, websites are starting to block access to their content if they detect use of Ad-blockers or they ask users to move to a subscription-based model (where users have to pay to get access to the website).

This extension is testing another approach: It is trying to find a trade-off between privacy and economy, that would allow users to protect their privacy while still accessing to free content.

It is based on the assumption that most people are not against advertisements, but want to keep control over their data. We believe that some sites are more sensitive than others. In fact, most people don't want to be tracked on "sensitive" websites (for example related to religion, health,. . .), but don't see any problem to be tracked on less sensitive ones (such as news, sport,. . .). This extension allows you to take control and specify which on which categories of sites you don't want to be tracked on! Furthermore, the extension also gives you the option to block the trackers on specific websites.

- Contact: Claude Castelluccia
- URL: https://addons.mozilla.org/FR/firefox/addon/mytrackingchoices/

5.4. OMEN+

Functional Description

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.

- Participants: Pierre Rouveyrol and Claude Castelluccia
- Contact: Claude Castelluccia

5.5. OPENFEC

Functional Description

OpenFEC is an open-source C-language implementation of several Application-Level Forward Erasure Correction (AL-FEC) codecs, namely: 2D-parity, Reed-Solomon (RFC 5510) and LDPC-Staircase (RFC 5170) 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.

- Participants: Mathieu Cunche, Jonathan Detchart, Julien Laboure, Christophe Neumann, Vincent Roca, Jérome Lacan and Kevin Chaumont
- Contact: Vincent Roca
- URL: http://openfec.org/

PROSECCO Project-Team

6. New Software and Platforms

6.1. ProVerif

Participants: Bruno Blanchet [correspondant], Xavier Allamigeon [April–July 2004], Vincent Cheval [Sept. 2011–Dec. 2014], Benjamin Smyth [Sept. 2009–Feb. 2010], Marc Sylvestre [Oct. 2016–].

PROVERIF (http://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 http://proverif.inria.fr, under the GPL license.

6.2. CryptoVerif

Participants: Bruno Blanchet [correspondant], David Cadé [Sept. 2009–Dec. 2013].

CRYPTOVERIF(http://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 http://cryptoverif.inria.fr, under the CeCILL license.

6.3. miTLS

Participants: Karthikeyan Bhargavan [correspondant], Cedric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Antoine Delignat-Lavaud [Microsoft Research], Nikhil Swamy [Microsoft Research], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoué, Benjamin Beurdouche, Alfredo Pironti.

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.

Papers describing the miTLS library was published at IEEE S&P 2013, CRYPTO 2014, and IEEE S&P Journal 2016. miTLS is now being developed on GitHub with dozens of contributors and regular updates. The miTLS team was awarded the Levchin prize for contributions to Real-World Cryptography in 2016. The software and associated research materials are available from http://mitls.org.

6.4. F*

Participants: Alejandro Aguirre, Danel Ahman [University of Edinburgh], Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud [Microsoft Research], Cédric Fournet [Microsoft Research], Catalin Hritcu, Chantal Keller [Université Paris-Sud], Kenji Maillard, Guido Martínez, Gordon Plotkin, Samin Ishtiaq [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Jonathan Protzenko [Microsoft Research], Tahina Ramananandro [Microsoft Research], Aseem Rastogi [Microsoft Research], Nikhil Swamy [Microsoft Research], Peng Wang [MIT], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoué.

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 [44]. 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.

6.5. HACL*

Participants: Karthikeyan Bhargavan, Jean Karim Zinzindohoué, Marina Polubelova, Benjamin Beurdouche, Jonathan Protzenko [Microsoft Research].

HACL* is a verified cryptographic library written in F*. It implements modern primitives, including elliptic curves like Curve25519, symmetric ciphers like Chacha20, and MAC algorithms like Poly1305. These primitives are then composed into higher-level constructions like Authenticated Encryption with Additional Data (AEAD) and the NaCl API. All the code in HACL* is verified for memory safety, side channel resistance, and where applicable, also for functional correctness and absence of integer overflow. HACL* code is used as the basis for cryptographic proofs for security in the miTLS project.

HACL* code can be compiled to OCaml using the standard F* compiler, or to C, using the Kremlin backend of F*. The generated C code is as fast as state-of-the-art cryptographic libraries written in C. HACL* is being actively developed on Github; see https://github.com/mitls/hacl-star

6.6. ProScript

Participants: Nadim Kobeissi [correspondant], Karthikeyan Bhargavan, Bruno Blanchet.

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. ProScript is a typed subset of JavaScript, inspired by DJS, that is focused on writing verifiable cryptographic protocol implementations. In addition to DJS typing, ProScript imposes a functional style that results in more readable and easily verifiable ProVerif models. ProScript has been used to write and verify a full implementation of the Signal and TLS 1.3 protocols in JavaScript.

The ProScript compiler and various libraries written in ProScript are being developed on Github and will be publicly released in 2017.

6.7. QuickChick

Participants: Maxime Dénès [Inria Sophia-Antipolis], Catalin Hritcu, John Hughes [Chalmers University], Leonidas Lampropoulos [University of Pennsylvania], Zoe Paraskevopoulou [Princeton University], Benjamin Pierce [University of Pennsylvania].

QuickChick is a verified plugin that integrates property-based testing and proving in the Coq proof assistant. This integration is aimed at reducing the cost of formal verification and at providing stronger, formal foundations to property-based testing. https://github.com/QuickChick/QuickChick

6.8. Luck

Participants: Leonidas Lampropoulos [University of Pennsylvania], Diane Gallois-Wong [ENS and Inria Paris], Catalin Hritcu, John Hughes [Chalmers University], Benjamin Pierce [University of Pennsylvania], Li-Yao Xia [ENS and Inria Paris].

Property-based random testing a la QuickCheck requires building efficient generators for well-distributed random data satisfying complex logical predicates, but writing these generators can be difficult and error prone. We propose a domain-specific language in which generators are conveniently expressed by decorating predicates with lightweight annotations to control both the distribution of generated values and the amount of constraint solving that happens before each variable is instantiated. This language, called Luck, makes generators easier to write, read, and maintain. https://github.com/QuickChick/Luck

6.9. Privacy-preserving federated identity

Participants: Harry Halpin, George Danezis [University College London].

security protocols, secure messaging, decentralization, blockchain

Working with the partners in the NEXTLEAP project, we helped design a protocol for decentralized and privacy-preserving identity and key management, creating both a fix privacy vulnerabilities in OAuth (UnlimitID) and on generic versions of blockchain for usages such as key management (ClaimChain). While this software has been prototyped in 2016 using Python working with NEXTLEAP project partner University College London (George Danezis), we expected either formal analysis using ProVerif or verified implementations using Fstar in 2017, as well and integrate this work with formal verification work done in Prosecco on end-to-end secure messaging.

TAMIS Team

6. New Software and Platforms

6.1. MHD

GNU libmicrohttpd
KEYWORDS: Embedded - Web 2.0
SCIENTIFIC DESCRIPTION

We are providing a standards compliant and complete implementation of the HTTP server protocol that allows developers to easily write correct HTTP servers. Key challenges include code size minimization (for IoT devices), performance (zero copy, scalability to 100k concurrent connections), portability and security. MHD is already widely used in production by both academic and industrial users. Ongoing research challenges include formal verification.

FUNCTIONAL DESCRIPTION

GNU libmicrohttpd is a small C library that is supposed to make it easy to run an HTTP server as part of another application.

- Participants: Evgeny Grin, Christian Grothoff and Sree Hrsha Totakura
- Partner: The GNU Project
- Contact: Christian Grothoff
- URL: http://www.gnu.org/software/libmicrohttpd/

6.2. PLASMA Lab

KEYWORDS: Model Checking - Statistical - Model Checker - Runtime Analysis - Security - Code analysis - Statistics - Energy - Aeronautics
SCIENTIFIC DESCRIPTION

Statistical model checking (SMC) is a fast emerging technology for industrial scale verification and optimisation problems. Plasma was conceived to have high performance and be extensible, using a proprietary virtual machine. Since SMC requires only an executable semantics and is not constrained by decidability, we can easily implement different modelling languages and logics.

FUNCTIONAL DESCRIPTION

PLASMA Lab is a compact, efficient and flexible platform for statistical model checking of stochastic models. PLASMA Lab demonstrates the following advances: -Use your own simulator and checker via our plugin system. -Build your software around Plasma Lab using our API. -Prism (Reactives Modules Language-RML) and Biological languages supported. -Matlab and SytemC plugins. -Distributed architecture. Whether you plan to use several computers on a local area network or a grid, you can run PLASMA Lab in an easy way. -Fast algorithms. -Efficient data structure, low memory consumption. -Developed with Java for compatibility.

- Participants: Axel Legay, Sean Sedwards, Benoit Boyer, Louis-Marie Traonouez, Kevin Corre and Matthieu Simonin
- Contact: Axel Legay
- URL: http://plasma-lab.gforge.inria.fr/plasma_lab_doc/1.4.0/html/introduction.html#

6.3. Quail

FUNCTIONAL DESCRIPTION

Privacy is a central concern 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.

- Participants: Fabrizio Biondi, Axel Legay, Louis-Marie Traonouez and Andrzej Wasowski
- Contact: Axel Legay
- URL: https://project.inria.fr/quail/

6.4. GNUnet

Scientific Description

The GNUnet project seeks to answer the question what a modern Internet architecture should look like for a society that care about security and privacy. We are considering all layers of the existing well-known Internet, but are also providing new and higher-level abstractions (such as voting protocols, Byzantine consensus, etc.) that are today solved in application-specific ways. Research questions include the desired functionality of the overall stack, protocol design for the various layers as well as implementation considerations, i.e. how to implement the design securely.

Functional Description

GNUnet is a framework for secure peer-to-peer networking that does not use any centralized or otherwise trusted services. Our high-level goal is to provide a strong free software foundation for a global network that provides security and in particular respects privacy.

GNUnet started with an idea for anonymous censorship-resistant file-sharing, but has grown to incorporate other applications as well as many generic building blocks for secure networking applications. In particular, GNUnet now includes the GNU Name System, a privacy-preserving, decentralized public key infrastructure.

- Participants: Christian Grothoff, Florian Dold, Jeffrey Paul Burdges, Gabor Toth, Sree Hrsha Totakura and Alvaro Garcia Recuero
- Partner: The GNU Project
- Contact: Christian Grothoff
- URL: https://gnunet.org/

6.5. Taler

GNU Taler
Keyword: Privacy
Scientific Description

Taler is a Chaum-style digital payment system that enables anonymous payments while ensuring that entities that receive payments are auditable. In Taler, customers can never defraud anyone, merchants can only fail to deliver the merchandise to the customer, and payment service providers can be fully audited. All parties receive cryptographic evidence for all transactions, still, each party only receives the minimum information required to execute transactions. Enforcement of honest behavior is timely, and is at least as strict as with legacy credit card payment systems that do not provide for privacy.

The key technical contribution underpinning Taler is a new refresh protocol which allows fractional payments and refunds while maintaining untraceability of the customer and unlinkability of transactions. The refresh protocol combines an efficient cut-and-choose mechanism with a link step to ensure that refreshing is not abused for transactional payments.

We argue that Taler provides a secure digital currency for modern liberal societies as it is a flexible, *libre* and efficient protocol and adequately balances the state's need for monetary control with the citizen's needs for private economic activity.

FUNCTIONAL DESCRIPTION

Taler is a new electronic payment system. It includes an electronic wallet for customers, a payment backend for merchants and the main payment service provider logic called the exchange. Taler offers Chaum-style anonymous payments for citizens, and income-transparency for taxability.

- Participants: Jeffrey Paul Burdges, Marcello Stanisci, Florian Dold, Gabor Toth and Christian Grothoff
- Partner: The GNU Project
- Contact: Christian Grothoff
- URL: http://taler.net/

6.6. VITRAIL - Visualisation Tool

Real-Time, Advanced, Immersive Visualization of Software / Visualizer
KEYWORD: Visualization of software
SCIENTIFIC DESCRIPTION

It is difficult for developers to explore and understand the source code of large programs, for example in objet-oriented languages programs featuring thousands of classes. Visualization methods based on daily life metaphors have thus been proposed. The VITRAIL Visualization tool (or VITRAIL Vizualizer) makes it possible to display, visualize and explore Java programs in a metaphorical way, using the city metaphor. An execution trace of the Java (byte)code provided by VITRAIL JBInstrace tool, is provided as input to VITRAIL Visualizer which displays a city-like metaphorical world showing the static structure of the code as well as some dynamic elements (calls).
This program may be used in Tamis as a basis for tools for the visualization of security events in programs.

FUNCTIONAL DESCRIPTION

This program makes it possible to displays, visualizes and explores Java programs in a metaphorical way (using the city metaphore). Useful for complex application developers/architects.

- Participants: Damien Bodenes, Olivier Zendra and Olivier Demengeon
- Contact: Olivier Zendra

6.7. VITRAIL 6 JBInsTrace

Real-Time, Advanced, Immersive Visualization of Software / Java Bytecode Instrumenter and Tracer
KEYWORDS: Java - Bytecode - Instrumentation - Profiling - Execution trace - Basic block
SCIENTIFIC DESCRIPTION

VITRAIL JBInsTrace is a program to instrument Java bytecode to trace its execution. The trace contains both static and dynamic information (calls). It is produced by intercepting the JVM class loader and replacing it by ours. Thus Java bytecode file are not modified, since instrumentation is performed on the fly, in memory. This makes it possible to instrument the whole program code, including libraries. Java source code is not needed. The trace which is then fed into our program VITRAIL Visualizer is an XML-like file.
This program may be used in Tamis as a basis for tools to instrument Java bytecode for security.

FUNCTIONAL DESCRIPTION

VITRAIL JBInsTrace is a program to instrument Java bytecode files to trace their execution. The trace is then fed into our VITRAIL Visualizer tool.

- Participants: Pierre Caserta and Olivier Zendra
- Contact: Olivier Zendra

6.8. Platforms

6.8.1. Malware'o'Matic

This LHS platform is dedicated to the collect, the categorization and the analyze of malware. We are currently interested in a specific kind of malware the ransomware. The platform grabs periodically samples of public data bases, executes the ransomware without virtualization on a victim PC and evaluate the implemented detection mechanisms. Once a ransomware has been executed the image of the OS is automatically restored and a new sample is evaluated. The platform is fully automatic and target Windows platforms (seven, W10) in both 32 bits and 64 bits versions.

6.8.2. Faustine

This LHS platform is dedicated to the EM fault injection experiments. It is composed of a motion table (XY), a pulse generator, an amplifier and a control PC. It injects EM pulses in a controlled way on a targeted device using an EM probe. It controls with a high precision the timing and the edges of the pulse. A recent development consists in adding a FPGA board to control the trigger in a more convenient and precise way. Then, the pulse can be triggered while a specific information is sent to the board under attack.

6.8.3. EMA

This last LHS platform is dedicated to side channel analysis (SCA) for evaluating the capabilities of dynamic countermeasure developed in the ANR Cogito. This platform uses a low cost oscilloscope, an EM probe and a set of software developed to perform the analysis. An additional oscilloscope, more performant has been added to the platform to target faster devices. We use the Julia language platform and custom developments to control the target and to analyze the EM traces.