RESEARCH CENTER

FIELD
Algorithmics, Programming, Software and Architecture

Activity Report 2019

Section Software

Edition: 2020-03-21

ARIC Project-Team

6. New Software and Platforms

6.1. FPLLL

KEYWORDS: Euclidean Lattices - Computer algebra system (CAS) - Cryptography

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 LLL is central to the code, hence the name.

It includes implementations of floating-point LLL reduction algorithms, offering different speed/guarantees ratios. It contains a 'wrapper' choosing the estimated best sequence of variants in order to provide a guaranteed output as fast as possible. In the case of the wrapper, the succession of variants is oblivious to the user.

It includes an implementation of the BKZ reduction algorithm, including the BKZ-2.0 improvements (extreme enumeration pruning, pre-processing of blocks, early termination). Additionally, Slide reduction and self dual BKZ are supported.

It also includes a floating-point 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 in fplll. 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.

- Author: Damien Stehlé
- Contact: Damien Stehlé
- URL: https://github.com/fplll/fplll

6.2. Gfun

generating functions package

KEYWORD: Symbolic computation

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/

6.3. GNU-MPFR

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

FUNCTIONAL DESCRIPTION: GNU MPFR is an efficient arbitrary-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. It 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. GNU MPFR is based on the mpn and mpz layers of the GMP library.

- Participants: Guillaume Hanrot, Paul Zimmermann, Philippe Théveny and Vincent Lefèvre
- Contact: Vincent Lefèvre
- Publications: Correctly Rounded Arbitrary-Precision Floating-Point Summation - Optimized Binary64 and Binary128 Arithmetic with GNU MPFR - Évaluation rapide de fonctions hypergéométriques - Arbitrary Precision Error Analysis for computing $\zeta(s)$ with the Cohen-Olivier algorithm: Complete description of the real case and preliminary report on the general case - MPFR: A Multiple-Precision Binary Floating-Point Library with Correct Rounding. - The Generic Multiple-Precision Floating-Point Addition With Exact Rounding (as in the MPFR Library)
- URL: https://www.mpfr.org/

6.4. 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
- Publications: SIPE: Small Integer Plus Exponent - Sipe: a Mini-Library for Very Low Precision Computations with Correct Rounding
- URL: https://www.vinc17.net/research/sipe/

6.5. LinBox

KEYWORD: Exact linear algebra

FUNCTIONAL DESCRIPTION: LinBox is an open-source C++ template library for exact, high-performance linear algebra computations. It is considered as the reference library for numerous computations (such as linear system solving, rank, characteristic polynomial, Smith normal forms,...) over finite fields and integers with dense, sparse, and structured matrices.

- Participants: Clément Pernet and Thierry Gautier
- Contact: Clément Pernet
- URL: http://linalg.org/

6.6. HPLLL

KEYWORDS: Euclidean Lattices - Computer algebra system (CAS)

FUNCTIONAL DESCRIPTION: Software library for linear algebra and Euclidean lattice problems

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

AROMATH Project-Team (section vide)

CARAMBA Project-Team

6. New Software and Platforms

6.1. Belenios

Belenios - Verifiable online voting system

KEYWORD: E-voting

FUNCTIONAL DESCRIPTION: Belenios is an open-source 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 received) 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. Moreover, Belenios includes a practical threshold decryption system that allows splitting the decryption key among several authorities.

NEWS OF THE YEAR: Since 2015, it has been used by CNRS for remote election among its councils (more than 30 elections every year) and since 2016, it has been used by Inria to elect representatives in the “comités de centre” of each Inria center. In 2018, it has been used to organize about 250 elections (not counting test elections). Belenios is typically used for elections in universities as well as in associations. This goes from laboratory councils (e.g. Irisa, Cran), scientific societies (e.g. SMAI) to various associations (e.g. FFBS - Fédération Française de Baseball et Softball, or SRFA - Société du Rat Francophone et de ses Amateurs).

In 2019, a threshold encryption mode has been added that makes the system more robust to the case where (say) one trustee among three loses her part of the decryption key.

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

6.2. CADO-NFS

Crible Algébrique: Distribution, Optimisation - Number Field Sieve

KEYWORDS: Cryptography - Number theory

FUNCTIONAL DESCRIPTION: CADO-NFS is a complete implementation in C/C++ of the Number Field Sieve (NFS) algorithm for factoring integers and computing discrete logarithms in finite fields. It consists in various programs corresponding to all the phases of the algorithm, and a general script that runs them, possibly in parallel over a network of computers.

NEWS OF THE YEAR: The main program for relation collection now supports composite "special-q". The memory footprint of the central step of linear algebra was reduced. Parallelism of many of the Cado-NFS programs was improved considerably (sieving, relation filtering, as well as the central step of linear algebra).

- Participants: Pierrick Gaudry, Emmanuel Thomé and Paul Zimmermann
- Contact: Emmanuel Thomé
- URL: http://cado-nfs.gforge.inria.fr/

6.3. Platforms

6.3.1. Platform: computational resources

Since 2018, the CARAMBA team has been using in particular a computer cluster called `grvingt`, acquired in 2018. This equipment was funded by the CPER «CyberEntreprises» (French Ministry of Research, Région Grand Est, Inria, CNRS) and comprises a 64-node, 2,048-core cluster. This cluster is installed in the Inria facility. Other slightly older hardware (a medium-size cluster called `grcinq` from 2013, funded by ANR, and a special machine funded by the aforementioned CPER grant) is also installed in the same location, to form a coherent platform with about 3,000 cpu cores, 100 TB of storage, and specific machines for RAM-demanding computations. As a whole, this platform provides an excellent support for the computational part of the work done in CARAMBA. This platform is also embedded in the larger Grid'5000/Silecs platform (and accessible as a normal resource within this platform). Technical administration is done by the Grid'5000 staff.

This equipment has played a key role in the record factorization of RSA-240 as well as the computation of discrete logarithms modulo a 240-digit prime, completed in the end of 2019.

CASCADE Project-Team (section vide)

DATASHAPE Project-Team

4. New Software and Platforms

4.1. GUDHI

Geometric Understanding in Higher Dimensions

KEYWORDS: Computational geometry - Topology

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 underly applications in geometry understanding in higher dimensions. It is intended to both help the development of new algorithmic solutions inside and outside the project, and to facilitate the transfer of results in applied fields.

NEWS OF THE YEAR: - Cover complex - Representation of persistence diagrams - Cech complex - weighted periodic 3d alpha-complex - sparse Rips complex - debian / docker / conda-forge packages

- Participants: Clément Maria, François Godi, David Salinas, Jean-Daniel Boissonnat, Marc Glisse, Mariette Yvinec, Pawel Dlotko, Siargey Kachanovich, Vincent Rouvreau, Mathieu Carrière and Bertrand Michel
- Contact: Jean-Daniel Boissonnat
- URL: https://gudhi.inria.fr/

4.2. CGAL module: interval arithmetics

KEYWORD: Arithmetic

FUNCTIONAL DESCRIPTION: This package of CGAL (Computational Geometry Algorithms Library http://www.cgal.org) provides an efficient number type for intervals of double and the corresponding arithmetic operations. It is used in the evaluation of geometric predicates for a first quick computation, which either provides the result with guarantees, or rarely answers that more precision is needed.

RELEASE FUNCTIONAL DESCRIPTION: Partial rewrite to take advantage of SIMD instructions on recent x86 processors.

- Contact: Marc Glisse
- URL: https://www.cgal.org/

4.3. CGAL module: interface to Boost.Multiprecision

KEYWORD: Arithmetic

FUNCTIONAL DESCRIPTION: This package of CGAL (Computational Geometry Algorithms Library http://www.cgal.org) makes it possible to use some number types from Boost.Multiprecision in CGAL.

- Author: Marc Glisse
- Contact: Marc Glisse
- URL: https://www.cgal.org/

4.4. Module CGAL: New dD Geometry Kernel

KEYWORD: Computational geometry

FUNCTIONAL DESCRIPTION: This package of CGAL (Computational Geometry Algorithms Library http://www.cgal.org) provides the basic geometric types (point, vector, etc) and operations (orientation test, etc) used by geometric algorithms in arbitrary dimension. It uses filters for efficient exact predicates.

RELEASE FUNCTIONAL DESCRIPTION: New kernel with lazy exact constructions.

- Author: Marc Glisse
- Contact: Marc Glisse
- URL: http://www.cgal.org/

GAMBLE Project-Team

6. New Software and Platforms

6.1. CGAL Package : 2D periodic hyperbolic triangulations

KEYWORDS: Geometry - Delaunay triangulation - Hyperbolic space

FUNCTIONAL DESCRIPTION: This module implements the computation of Delaunay triangulations of the Bolza surface.

NEWS OF THE YEAR: Integration into CGAL 4.14

- Authors: Iordan Iordanov and Monique Teillaud
- Contact: Monique Teillaud
- Publication: Implementing Delaunay Triangulations of the Bolza Surface
- URL: https://doc.cgal.org/latest/Manual/packages.html#PkgPeriodic4HyperbolicTriangulation2

6.2. CGAL Package : 2D hyperbolic triangulations

KEYWORDS: Geometry - Delaunay triangulation - Hyperbolic space

FUNCTIONAL DESCRIPTION: This package implements the construction of Delaunay triangulations in the Poincaré disk model.

NEWS OF THE YEAR: Integration into CGAL 4.14

- Participants: Mikhail Bogdanov, Olivier Devillers, Iordan Iordanov and Monique Teillaud
- Contact: Monique Teillaud
- Publication: Hyperbolic Delaunay Complexes and Voronoi Diagrams Made Practical
- URL: https://doc.cgal.org/latest/Manual/packages.html#PkgHyperbolicTriangulation2

6.3. clenshaw

KEYWORDS: Numerical solver - Visualization - Polynomial equations

FUNCTIONAL DESCRIPTION: Clenshaw is a mixed C and python library that provides computation and plotting functions for the solutions of polynomial equations in the Taylor or the Chebyshev basis. The library is optimized for machine double precision and for numerically well-conditioned polynomials. In particular, it can find the roots of polynomials with random coefficients of degree one million.

- Contact: Guillaume Moroz
- URL: https://gitlab.inria.fr/gmoro/clenshaw

6.4. voxelize

KEYWORDS: Visualization - Curve plotting - Implicit surface - Polynomial equations

FUNCTIONAL DESCRIPTION: Voxelize is a C++ software to visualize the solutions of polynomial equations and inequalities. The software is optimized for high degree curves and surfaces. Internally, polynomials and sets of boxes are stored in the Compressed Sparse Fiber format. The output is either a mesh or a union of boxes written in the standard 3D file format ply.

RELEASE FUNCTIONAL DESCRIPTION: This is the first published version.

- Contact: Guillaume Moroz
- URL: https://gitlab.inria.fr/gmoro/voxelize

GRACE Project-Team

5. New Software and Platforms

5.1. ACTIS

Algorithmic Coding Theory in Sage

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.

- Partner: Technical University Denmark
- Contact: Daniel Augot

5.2. DECODING

KEYWORD: Algebraic decoding

FUNCTIONAL DESCRIPTION: Decoding is a standalone C library. Its primary goal is to implement Guruswami–Sudan list decoding-related algorithms, as efficiently as possible. Its secondary goal is to give an efficient tool for the implementation of decoding algorithms (not necessarily list decoding algorithms) and their benchmarking.

- Participant: Guillaume Quintin
- Contact: Daniel Augot

5.3. Fast Compact Diffie-Hellman

KEYWORD: Cryptography

FUNCTIONAL DESCRIPTION: A competitive, high-speed, open implementation of the Diffie–Hellman protocol, targeting the 128-bit security level on Intel platforms. This download contains Magma files that demonstrate how to compute scalar multiplications on the x-line of an elliptic curve using endomorphisms. This accompanies the EuroCrypt 2014 paper by Costello, Hisil and Smith, the full version of which can be found here: http://eprint.iacr.org/2013/692. The corresponding SUPERCOP-compatible crypto_dh application can be downloaded from http://hhisil.yasar.edu.tr/files/hisil20140318compact.tar.gz.

- Participant: Ben Smith
- Contact: Ben Smith
- URL: http://research.microsoft.com/en-us/downloads/ef32422a-af38-4c83-a033-a7aafbc1db55/

5.4. CADO-NFS

Crible Algébrique: Distribution, Optimisation - Number Field Sieve

KEYWORDS: Cryptography - Number theory

FUNCTIONAL DESCRIPTION: CADO-NFS is a complete implementation in C/C++ of the Number Field Sieve (NFS) algorithm for factoring integers and computing discrete logarithms in finite fields. It consists in various programs corresponding to all the phases of the algorithm, and a general script that runs them, possibly in parallel over a network of computers.

NEWS OF THE YEAR: The main program for relation collection now supports composite "special-q". The memory footprint of the central step of linear algebra was reduced. Parallelism of many of the Cado-NFS programs was improved considerably (sieving, relation filtering, as well as the central step of linear algebra).

- Participants: Pierrick Gaudry, Emmanuel Thomé and Paul Zimmermann
- Contact: Emmanuel Thomé
- URL: http://cado-nfs.gforge.inria.fr/

LFANT Project-Team

5. New Software and Platforms

5.1. APIP

Another Pairing Implementation in PARI

KEYWORDS: Cryptography - Computational number theory

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: Andreas Enge
- URL: http://www.lix.polytechnique.fr/~milanj/apip/apip.xhtml

5.2. AVIsogenies

Abelian Varieties and Isogenies

KEYWORDS: Computational number theory - Cryptography

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

5.3. CM

KEYWORD: Arithmetic

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.

RELEASE FUNCTIONAL DESCRIPTION: Features - Precisions beyond 300000 bits are now supported by an addition chain of variable length for the -function. Dependencies - The minimal version number of Mpfr has been increased to 3.0.0, that of Mpc to 1.0.0 and that of Pari to 2.7.0.

- Participant: Andreas Enge
- Contact: Andreas Enge
- URL: http://www.multiprecision.org/cm/home.html

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

5.5. CUBIC

KEYWORD: Number theory

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-1.2.tgz

5.6. Euclid

KEYWORD: Number theory

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

5.7. KleinianGroups

KEYWORDS: Computational geometry - Computational number theory

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

5.8. GNU MPC

KEYWORD: Arithmetic

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.

RELEASE FUNCTIONAL DESCRIPTION: Fixed `mpc_pow`, see http://lists.gforge.inria.fr/pipermail/mpc-discuss/2014-October/001315.html - `#18257`: Switched to libtool 2.4.5.

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

5.9. MPFRCX

KEYWORD: Arithmetic

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.

RELEASE FUNCTIONAL DESCRIPTION: - new function `product_and_hecke` - improved memory consumption for unbalanced FFT multiplications

- Participant: Andreas Enge
- Contact: Andreas Enge
- URL: http://www.multiprecision.org/mpfrcx/home.html

5.10. PARI/GP

KEYWORD: Computational number theory

FUNCTIONAL DESCRIPTION: Pari/Gp is a widely used computer algebra system designed for fast computations in number theory (factorisation, algebraic number theory, elliptic curves, modular forms ...), 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: Andreas Enge, Hamish Ivey-Law, Henri Cohen and Karim Belabas
- Partner: CNRS
- Contact: Karim Belabas
- URL: http://pari.math.u-bordeaux.fr/

5.11. Platforms

5.11.1. SageMath

Following the article [19], Xavier Caruso and Thibaut Verron proposed an implementation of Tate algebras and ideals in Tate algebras (including an implementation of Buchberger algorithm) for SageMath; their implementation is now part of the standard distribution.

Xavier Caruso implemented a new unified framework for dealing with ring extensions and field extensions in SageMath. This code will be integrated soon in the standard distribution.

5.11.2. ARB

Fredrik Johansson released a new version, 2.17, of ARB.

OURAGAN Project-Team

6. New Software and Platforms

6.1. ISOTOP

Topology and geometry of planar algebraic curves

KEYWORDS: Topology - Curve plotting - Geometric computing

FUNCTIONAL DESCRIPTION: Isotop is a Maple software for computing the topology of an algebraic plane curve, that is, for computing an arrangement of polylines isotopic to the input curve. This problem is a necessary key step for computing arrangements of algebraic curves and has also applications for curve plotting. This software has been developed since 2007 in collaboration with F. Rouillier from Inria Paris - Rocquencourt.

- Participants: Luis Penaranda, Marc Pouget and Sylvain Lazard
- Contact: Marc Pouget
- Publications: Rational Univariate Representations of Bivariate Systems and Applications - Separating Linear Forms for Bivariate Systems - On The Topology of Planar Algebraic Curves - New bivariate system solver and topology of algebraic curves - Improved algorithm for computing separating linear forms for bivariate systems - Solving bivariate systems using Rational Univariate Representations - On the topology of planar algebraic curves - On the topology of real algebraic plane curves - Bivariate triangular decompositions in the presence of asymptotes - Separating linear forms and Rational Univariate Representations of bivariate systems
- URL: https://isotop.gamble.loria.fr/

6.2. RS

FUNCTIONAL DESCRIPTION: Real Roots isolation for algebraic systems with rational coefficients with a finite number of Complex Roots

- Participant: Fabrice Rouillier
- Contact: Fabrice Rouillier
- URL: https://team.inria.fr/ouragan/software/

6.3. A NewDsc

A New Descartes

KEYWORD: Scientific computing

FUNCTIONAL DESCRIPTION: Computations of the real roots of univariate polynomials with rational coefficients.

- Authors: Fabrice Rouillier, Alexander Kobel and Michael Sagraloff
- Partner: Max Planck Institute for Software Systems
- Contact: Fabrice Rouillier
- URL: https://anewdsc.mpi-inf.mpg.de

6.4. SIROPA

KEYWORDS: Robotics - Kinematics

FUNCTIONAL DESCRIPTION: Library of functions for certified computations of the properties of articulated mechanisms, particularly the study of their singularities

- Authors: Damien Chablat, Fabrice Rouillier, Guillaume Moroz and Philippe Wenger
- Partner: LS2N
- Contact: Guillaume Moroz
- URL: http://siropa.gforge.inria.fr/

6.5. MPFI

KEYWORD: Arithmetic

FUNCTIONAL DESCRIPTION: MPFI is a C library based on MPFR and GMP for multi precision floating point arithmetic.

- Contact: Fabrice Rouillier
- URL: http://mpfi.gforge.inria.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

KEYWORDS: Gröbner bases - Nonlinear system - Computer algebra

FUNCTIONAL DESCRIPTION: FGb is a powerful software for computing Gröbner bases. It includes the new generation of algorihms for computing Gröbner bases polynomial systems (mainly the F4, F5 and FGLM algorithms). It is implemented in C/C++ (approximately 250000 lines), standalone servers are available on demand. Since 2006, FGb is dynamically linked with Maple software (version 11 and higher) and is part of the official distribution of this software.

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

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/GBLA/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. RealCertify

KEYWORDS: Polynomial or analytical systems - Univariate polynomial - Real solving

FUNCTIONAL DESCRIPTION: The package RealCertify aims at providing a full suite of hybrid algorithms for computing certificates of non-negativity based on numerical software for solving linear matrix inequalities. The module univsos handles the univariate case and the module multivsos is designed for the multivariate case.

- Contact: Mohab Safey El Din
- URL: https://gricad-gitlab.univ-grenoble-alpes.fr/magronv/RealCertify

5.8. SLV

KEYWORDS: Univariate polynomial - Real solving

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 approx. 5000 lines.

- Contact: Elias Tsigaridas
- URL: https://who.paris.inria.fr/Elias.Tsigaridas/soft.html

5.9. SPECTRA

Semidefinite Programming solved Exactly with Computational Tools of Real Algebra

KEYWORD: Linear Matrix Inequalities

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 Gröbner bases and provides either certified numerical approximations of the solutions or exact representations thereof.

- 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: Grégory Landais and Nicolas Sendrier
- 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: Grégory Landais and Nicolas Sendrier
- 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 variant of the algorithm due to May, Meurer and Thomae.

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

SPECFUN Project-Team

5. New Software and Platforms

5.1. DynaMoW

Dynamic Mathematics on the Web

FUNCTIONAL DESCRIPTION: Programming tool for controlling the generation of mathematical websites that embed dynamical mathematical contents generated by computer-algebra calculations. Implemented in OCaml.

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

5.2. ECS

Encyclopedia of Combinatorial Structures

FUNCTIONAL DESCRIPTION: On-line 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.

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

5.3. 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.

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

5.4. Mgfun

multivariate generating functions package

FUNCTIONAL DESCRIPTION: The Mgfun Project is a collection of packages for the computer algebra system Maple, and is intended for the symbolic manipulation of a large class of special functions and combinatorial sequences (in one or several variables and indices) that appear in many branches of mathematics, mathematical physics, and engineering sciences. Members of the class satisfy a crucial finiteness property which makes the class amenable to computer algebra methods and enjoy numerous algorithmic closure properties, including algorithmic closures under integration and summation.

- Contact: Frédéric Chyzak
- URL: http://specfun.inria.fr/chyzak/mgfun.html

5.5. Ssreflect

KEYWORD: Proof assistant

SCIENTIFIC DESCRIPTION: Ssreflect is tactic language that helps writing concise and uniform tactic based proof scripts for the Coq system. It was designed during the proofs of the 4 Color Theorem and the Feit-Thompson theorem.

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

NEWS OF THE YEAR: In 2019, we extended the intro pattern functionality of SSreflect and added support for working under binders using the "under" tactical.

- Participants: Assia Mahboubi, Cyril Cohen, Enrico Tassi, Georges Gonthier, Laurence Rideau, Laurent Théry and Yves Bertot
- Contact: Yves Bertot
- URL: http://math-comp.github.io/math-comp/

5.6. Math-Components

Mathematical Components library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: The Mathematical Components library is a set of Coq libraries that cover the prerequiste for the mechanization of the proof of the Odd Order Theorem.

RELEASE FUNCTIONAL DESCRIPTION: This releases is compatible with Coq 8.9 and Coq 8.10 it adds many theorems for finite function, prime numbers, sequences, finite types, bigo operations, natural numbers, cycles in graphs.

- Participants: Alexey Solovyev, Andrea Asperti, Assia Mahboubi, Cyril Cohen, Enrico Tassi, François Garillot, Georges Gonthier, Ioana Pasca, Jeremy Avigad, Laurence Rideau, Laurent Théry, Russell O'Connor, Sidi Ould Biha, Stéphane Le Roux and Yves Bertot
- Contact: Assia Mahboubi
- URL: http://math-comp.github.io/math-comp/

CAIRN Project-Team

5. New Software and Platforms

5.1. Gecos

Generic Compiler Suite

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 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.

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 is currently 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 programme transformation toolbox facilitating parallelisation of applications for heterogeneous multiprocessor embedded platforms. In addition to targeting programmable processors, GeCoS can regenerate optimised code for High Level Synthesis tools.

- Participants: Tomofumi Yuki, Thomas Lefeuvre, Imèn Fassi, Mickael Dardaillon, Ali Hassan El Moussawi and Steven Derrien
- Partner: Université de Rennes 1
- Contact: Steven Derrien
- URL: http://gecos.gforge.inria.fr

5.2. ID-Fix

Infrastructure for the Design of Fixed-point systems

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

SCIENTIFIC DESCRIPTION: The different techniques proposed by the team for fixed-point conversion are implemented in the ID.Fix infrastructure. The application is described with a C code using floating-point data types and different pragmas, used to specify parameters (dynamic, input/output word-length, delay operations) for the fixed-point conversion. This tool determines and optimizes the fixed-point specification and then, generates a C code using fixed-point data types (ac_fixed) from Mentor Graphics. The infrastructure is made 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 precision accuracy and can provide an optimized 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 constructs an analytic accuracy model of the program, which means it can explore more solutions and thereby produce a much more efficient code.

- Participant: Olivier Sentieys
- Partner: Université de Rennes 1
- Contact: Olivier Sentieys
- URL: http://idfix.gforge.inria.fr

5.3. SmartSense

KEYWORDS: Wireless Sensor Networks - Smart building - Non-Intrusive Appliance Load Monitoring

FUNCTIONAL DESCRIPTION: To measure energy consumption by equipment in a building, NILM techniques (Non-Intrusive Appliance Load Monitoring) are based on observation of overall variations in electrical voltage. This avoids having to deploy watt-meters on every device and thus reduces the cost. SmartSense goes a step further to improve on these techniques by combining sensors (light, temperature, electromagnetic wave, vibration and sound sensors, etc.) to provide additional information on the activity of equipment and people. Low-cost sensors can be energy-autonomous too.

- Contact: Olivier Sentieys

5.4. Platforms

5.4.1. Zyggie: a Wireless Body Sensor Network Platform

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 (see Figure 2) includes the following features: a 32-bit micro-controller to manage a custom MAC layer and process 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: https://bowi.cominlabs.u-bretagneloire.fr/zyggie-wbsn-platform

5.4.2. E-methodHW: an automatic tool for the evaluation of polynomial and rational function approximations

KEYWORDS: function approximation, FPGA hardware implementation generator

SCIENTIFIC DESCRIPTION: E-methodHW is an open source C/C++ prototype tool written to exemplify what kind of numerical function approximations can be developed using a digit recurrence evaluation scheme for polynomials and rational functions.

Figure 2. CAIRN*'s Zyggie platform for WBSN*

FUNCTIONAL DESCRIPTION: E-methodHW provides a complete design flow from choice of mathematical function operator up to optimised VHDL code that can be readily deployed on an FPGA. The use of the E-method allows the user great flexibility if targeting high throughput applications.

- Participants: Silviu-Ioan Filip, Matei Istoan
- Partners: Université de Rennes 1, Imperial College London
- Contact: Silviu-Ioan Filip
- URL: https://github.com/sfilip/emethod

5.4.3. *Firopt: a tool for the simultaneous design of digital FIR filters along with the dedicated hardware model*

KEYWORDS: FIR filter design, multiplierless hardware implementation generator

SCIENTIFIC DESCRIPTION: the firopt tool is an open source C++ prototype that produces Finite Impulse Response (FIR) filters that have minimal cost in terms of digital adders needed to implement them. This project aims at fusing the filter design problem from a frequency domain specification with the design of the dedicated hardware architecture. The optimality of the results is ensured by solving appropriate mixed integer linear programming (MILP) models developed for the project. It produces results that are generally more efficient than those of other methods found in the literature or from commercial tools (such as MATLAB).

- Participants: Silviu-Ioan Filip, Martin Kumm, Anastasia Volkova
- Partners: Université de Rennes 1, Université de Nantes, Fulda University of Applied Sciences
- Contact: Silviu-Ioan Filip
- URL: https://gitlab.com/filteropt/firopt

5.4.4. *Hybrid-DBT*

KEYWORDS: Dynamic Binary Translation, hardware acceleration, VLIW processor, RISC-V

SCIENTIFIC DESCRIPTION: Hybrid-DBT is a hardware/software Dynamic Binary Translation (DBT) framework capable of translating RISC-V binaries into VLIW binaries. Since the DBT overhead has to be as small as possible, our implementation takes advantage of hardware acceleration for performance critical stages (binary translation, dependency analysis and instruction scheduling) of the flow. Thanks to hardware acceleration, our implementation is two orders of magnitude faster than a pure software implementation and enable an overall performance improvements by 23% on average, compared to a native RISC-V execution.

- Participants: Simon Rokicki, Steven Derrien
- Partners: Université de Rennes 1
- URL: https://github.com/srokicki/HybridDBT

5.4.5. Comet

KEYWORDS: Processor core, RISC-V instruction-set architecture

SCIENTIFIC DESCRIPTION: Comet is a RISC-V pipelined processor with data/instruction caches, fully developed using High-Level Synthesis. The behavior of the core is defined in a small C code which is then fed into a HLS tool to generate the RTL representation. Thanks to this design flow, the C description can be used as a fast and cycle-accurate simulator, which behaves exactly like the final hardware. Moreover, modifications in the core can be done easily at the C level.

- Participants: Simon Rokicki, Steven Derrien, Olivier Sentieys, Davide Pala, Joseph Paturel
- Partners: Université de Rennes 1
- URL: https://gitlab.inria.fr/srokicki/Comet

5.4.6. TypEx

KEYWORDS: Embedded systems, Fixed-point arithmetic, Floating-point, Low power consumption, Energy efficiency, FPGA, ASIC, Accuracy optimization, Automatic floating-point to fixed-point conversion

SCIENTIFIC DESCRIPTION: TypEx is a tool designed to automatically determine custom number representations and word-lengths (i.e., bit-width) for FPGAs and ASIC designs at the C source level. The main goal of TypEx is to explore the design space spanned by possible number formats in the context of High-Level Synthesis. TypEx takes a C code written using floating-point datatypes specifying the application to be explored. The tool also takes as inputs a cost model as well as some user constraints and generates a C code where the floating-point datatypes are replaced by the wordlengths found after exploration. The best set of word-lengths is the one found by the tool that respects the given accuracy constraint and that minimizes a parametrized cost function. Figure 3 presents an overview of the TypEx design flow.

- Participants: Olivier Sentieys, Tomofumi Yuki, Van-Phu Ha
- Partners: Université de Rennes 1
- URL: https://gitlab.inria.fr/gecos/gecos-float2fix

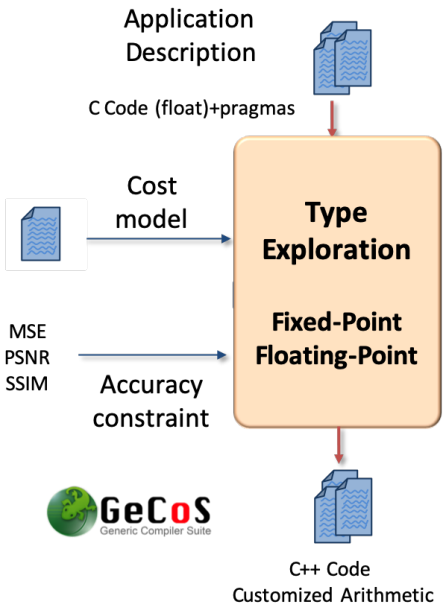

Figure 3. TypEx: a tool for type exploration and automatic floating-point to fixed-point conversion

CAMUS Project-Team

6. New Software and Platforms

6.1. CLooG

Code Generator in the Polyhedral Model

KEYWORDS: Polyhedral compilation - Optimizing compiler - Code generator

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.

RELEASE FUNCTIONAL DESCRIPTION: It mostly solves building and offers a better OpenScop support.

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

6.2. OpenScop

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

KEYWORDS: Polyhedral compilation - Optimizing compiler

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/

6.3. ORWL

Ordered Read-Write Lock

KEYWORDS: Task scheduling - Deadlock detection

FUNCTIONAL DESCRIPTION: ORWL is a reference implementation of the Ordered Read-Write Lock tools. The macro definitions and tools for programming in C99 that have been implemented for ORWL have been separated out into a toolbox called P99.

- Participants: Jens Gustedt, Mariem Saied and Stéphane Vialle
- Contact: Jens Gustedt
- Publications: Iterative Computations with Ordered Read-Write Locks - Automatic, Abstracted and Portable Topology-Aware Thread Placement - Resource-Centered Distributed Processing of Large Histopathology Images - Automatic Code Generation for Iterative Multi-dimensional Stencil Computations

6.4. musl

KEYWORDS: Standards - Library

SCIENTIFIC DESCRIPTION: musl provides consistent quality and implementation behavior from tiny embedded systems to full-fledged servers. Minimal machine-specific code means less chance of breakage on minority architectures and better success with “write once run everywhere” C development.

musl’s efficiency is unparalleled in Linux libc implementations. Designed from the ground up for static linking, musl carefully avoids pulling in large amounts of code or data that the application will not use. Dynamic linking is also efficient, by integrating the entire standard library implementation, including threads, math, and even the dynamic linker itself into a single shared object, most of the startup time and memory overhead of dynamic linking have been eliminated.

FUNCTIONAL DESCRIPTION: We participate in the development of musl, 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 devices. It gains more market share also in other areas, e.g. there are now Linux distributions that are based on musl instead of Gnu LibC.

- Participant: Jens Gustedt
- Contact: Jens Gustedt
- URL: http://www.musl-libc.org/

6.5. Modular C

KEYWORDS: Programming language - Modularity

FUNCTIONAL DESCRIPTION: The change to the C language is minimal since we only add one feature, composed identifiers, to the core language. Our modules can import other modules as long as the import relation remains acyclic and a module can refer to its own identifiers and those of the imported modules through freely chosen abbreviations. Other than traditional C include, our import directive ensures complete encapsulation between modules. The abbreviation scheme allows to seamlessly replace an imported module by another one with an equivalent interface. In addition to the export of symbols, we provide parameterized code injection through the import of “snippets”. This implements a mechanism that allows for code reuse, similar to X macros or templates. Additional features of our proposal are a simple dynamic module initialization scheme, a structured approach to the C library and a migration path for existing software projects.

- Author: Jens Gustedt
- Contact: Jens Gustedt
- Publications: Modular C - Arbogast: Higher order automatic differentiation for special functions with Modular C - Futex based locks for C11’s generic atomics
- URL: http://cmod.gforge.inria.fr/

6.6. arbogast

KEYWORD: Automatic differentiation

SCIENTIFIC DESCRIPTION: This high-level toolbox for the calculus with Taylor polynomials is named after L.F.A. Arbogast (1759-1803), a French mathematician from Strasbourg (Alsace), for his pioneering work in derivation calculus. Its modular structure ensures unmatched efficiency for computing higher order Taylor polynomials. In particular it permits compilers to apply sophisticated vector parallelization to the derivation of nearly unmodified application code.

FUNCTIONAL DESCRIPTION: Arbogast is based on a well-defined extension of the C programming language, Modular C, and places itself between tools that proceed by operator overloading on one side and by rewriting, on the other. The approach is best described as contextualization of C code because it permits the programmer to place his code in different contexts – usual math or AD – to reinterpret it as a usual C function or as a differential operator. Because of the type generic features of modern C, all specializations can be delegated to the compiler.

- Author: Jens Gustedt
- Contact: Jens Gustedt
- Publications: Arbogast: Higher order automatic differentiation for special functions with Modular C - Arbogast – Origine d'un outil de dérivation automatique
- URL: https://gforge.inria.fr/projects/arbo

6.7. 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 notations and tactics for manipulating characteristic formulae interactively in Coq.

- Participants: Arthur Charguéraud, Armaël Guéneau and François Pottier
- Contact: Arthur Charguéraud
- URL: http://www.chargueraud.org/softs/cfml/

6.8. SPETABARU

SPEculative TAsk-BAsed RUntime system

KEYWORDS: HPC - Parallel computing - Task-based algorithm

FUNCTIONAL DESCRIPTION: SPETABARU is a task-based runtime system for multi-core architectures that includes speculative execution models. It is a pure C++11 product without external dependency. It uses advanced meta-programming and allows for an easy customization of the scheduler. It is also capable to generate execution traces in SVG to better understand the behavior of the applications.

- Contact: Bérenger Bramas
- URL: https://gitlab.inria.fr/bramas/spetabaru

6.9. APAC

KEYWORDS: Source-to-source compiler - Automatic parallelization - Parallelisation - Parallel programming

SCIENTIFIC DESCRIPTION: APAC is a compiler for automatic parallelization that transforms C++ source code to make it parallel by inserting tasks. It uses the tasks+dependencies paradigm and relies on OpenMP or SPETABARU as runtime system. Internally, it is based on Clang-LLVM.

FUNCTIONAL DESCRIPTION: Automatic task-based parallelization compiler

- Participants: Bérenger Bramas, Stéphane Genaud and Garip Kusoglu
- Contact: Bérenger Bramas
- URL: https://gitlab.inria.fr/bramas/apac

6.10. Dagpar

KEYWORDS: Graph algorithmics - Clustering - Partitioning

SCIENTIFIC DESCRIPTION: This library is a clustering algorithm to create macro-tasks in a DAG of tasks. It extends a clustering/partitioning strategy proposed by Rossignon et al. to speed up the parallel execution of a task-based application. In this package, we provide two additional heuristics to this algorithm, which have been validated on a large graph set. The objective of clustering the nodes of task graphs is to increase the granularity of the tasks and thus obtain faster execution by mitigating the overhead from the management of the dependencies. An important asset of this approach is that working at the graph level allows us to create a generic method independent of the application and of what is done at the user level, but also independent of the task-based runtime system that could be used underneath.

FUNCTIONAL DESCRIPTION: Acyclic Dag Partitioning.

- Participants: Bérenger Bramas and Alain Ketterlin
- Contact: Bérenger Bramas
- URL: https://gitlab.inria.fr/bramas/dagpar

6.11. LetItBench

Lenient to Errors, Transformations, Irregularities and Turbulence Benchmarks

KEYWORDS: Approximate computing - Benchmarking

FUNCTIONAL DESCRIPTION: LetItBench is a benchmark set to help evaluating works on approximate compilation techniques. We propose a set of meaningful applications with an iterative kernel, that is not too complex for automatic analysis and can be analyzed by polyhedral tools. The benchmark set called LetItBench (Lenient to Errors, Transformations, Irregularities and Turbulence Benchmarks) is composed of standalone applications written in C, and a benchmark runner based on CMake. The benchmark set includes fluid simulation, FDTD, heat equations, game of life or K-means clustering. It spans various kind of applications that are resilient to approximation.

- Contact: Cédric Bastoul
- URL: https://github.com/Syllo/LetItBench

6.12. ACR

Adaptive Code Refinement

KEYWORDS: Approximate computing - Optimizing compiler

FUNCTIONAL DESCRIPTION: ACR is to approximate programming what OpenMP is to parallel programming. It is an API including a set of language extensions to provide the compiler with pertinent information about how to approximate a code block, a high-level compiler to automatically generate the approximated code, and a runtime library to exploit the approximation information at runtime according to the dataset properties. ACR is designed to provide approximate computing to non experts. The programmer may write a trivial code without approximation, provide approximation information thanks to pragmas, and let the compiler generate an optimized code based on approximation.

- Contact: Cédric Bastoul
- URL: https://github.com/Syllo/acr

6.13. 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: Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño, Manuel Selva and Philippe Clauss
- Contact: Philippe Clauss
- URL: http://apollo.gforge.inria.fr

CASH Project-Team

5. New Software and Platforms

5.1. DCC

DPN C Compiler

KEYWORDS: Polyhedral compilation - Automatic parallelization - High-level synthesis

FUNCTIONAL DESCRIPTION: Dcc (Data-aware process network C compiler) analyzes a sequential regular program written in C and generates an equivalent architecture of parallel computer as a communicating process network (Data-aware Process Network, DPN). Internal communications (channels) and external communications (external memory) are automatically handled while fitting optimally the characteristics of the global memory (latency and throughput). The parallelism can be tuned. Dcc has been registered at the APP ("Agence de protection des programmes") and transferred to the XtremLogic start-up under an Inria license.

- Participants: Alexandru Plesco and Christophe Alias
- Contact: Christophe Alias

5.2. PoCo

Polyhedral Compilation Library

KEYWORDS: Polyhedral compilation - Automatic parallelization

FUNCTIONAL DESCRIPTION: PoCo (Polyhedral Compilation Library) is a compilation framework allowing to develop parallelizing compilers for regular programs. PoCo features many state-of-the-art polyhedral program analysis and a symbolic calculator on execution traces (represented as convex polyhedra). PoCo has been registered at the APP (“agence de protection des programmes”) and transferred to the XtremLogic start-up under an Inria licence.

- Participant: Christophe Alias
- Contact: Christophe Alias

5.3. MPPcodegen

Source-to-source loop tiling based on MPP

KEYWORDS: Source-to-source compiler - Polyhedral compilation

FUNCTIONAL DESCRIPTION: MPPcodegen applies a monoparametric tiling to a C program enriched with pragmas specifying the tiling and the scheduling function. The tiling can be generated by any convex polyhedron and translation functions, it is not necessarily a partition. The result is a C program depending on a scaling factor (the parameter). MPPcodegen relies on the MPP mathematical library to tile the iteration sets.

- Partner: Colorado State University
- Contact: Christophe Alias
- URL: http://foobar.ens-lyon.fr/mppcodegen/

CORSE Project-Team

5. New Software and Platforms

5.1. Verde

KEYWORDS: Debug - Verification

FUNCTIONAL DESCRIPTION: Interactive Debugging with a traditional debugger can be tedious. One has to manually run a program step by step and set breakpoints to track a bug.

i-RV is an approach to bug fixing that aims to help developpers during their Interactive Debbugging sessions using Runtime Verification.

Verde is the reference implementation of i-RV.

- Participants: Kevin Pouget, Ylies Falcone, Raphael Jakse and Jean-François Méhaut
- Contact: Raphael Jakse
- Publication: Interactive Runtime Verification - When Interactive Debugging meets Runtime Verification
- URL: https://gitlab.inria.fr/monitoring/verde

5.2. Mickey

KEYWORDS: Dynamic Analysis - Performance analysis - Profiling - Polyhedral compilation

FUNCTIONAL DESCRIPTION: Mickey is a set of tools for profiling based performance debugging for compiled binaries. It uses a dynamic binary translator to instrument arbitrary programs as they are being run to reconstruct the control flow and track data dependencies. This information is then fed to a polyhedral optimizer that proposes structured transformations for the original code.

Mickey can handle both inter- and intra-procedural control and data flow in a unified way, thus enabling inter-procedural structured transformations. It is based on QEMU to allow for portability, both in terms of targeted CPU architectures, but also in terms of programming environment and the use of third-party libraries for which no source code is available.

- Partner: STMicroelectronics
- Contact: Fabrice Rastello
- Publications: hal-02060796v1 - hal-01967828v2

5.3. GUS

KEYWORDS: CPU - Microarchitecture simulation - Performance analysis - Dynamic Analysis

FUNCTIONAL DESCRIPTION: GUS' goal is to detect performance bottlenecks at the very low level on monothread applications by the use of sensitivity analysis. It is coded as a QEMU plug-in in order to collect runtime information that are later treated by the generic CPU model.

- Contact: Nicolas Derumigny
- URL: https://gitlab.inria.fr/nderumig/gus

5.4. Pipedream

KEYWORDS: Performance analysis - CPU - Reverse engineering

SCIENTIFIC DESCRIPTION: Pipedream reverse engineers the following performance characteristics: - Instruction latency – The number of cycles an instruction requires to execute. - Peak micro-op retirement rate – How many fused micro-ops the CPU can retire per cycle. - Micro-fusion – The number of fused micro-ops an instruction decomposes into. - Micro-op decomposition and micro-op port usage – The list of unfused micro-ops every instruction decomposes into and the list of execution ports every one of these micro-ops can execute on.

The first step of the reverse engineering process consists of generating a number of microbenchmarks. Pipedream then runs these benchmark, measuring their performance using hardware counters. The latency, throughput, and micro-fusion of different instructions can then be read directly from these measurements.

The process of finding port mappings, i.e. micro-op decompositions and micro-op port usage, however, is more involved. For this purpose, we have defined a variation of the maximum flow problem which we call the "instruction flow problem". We have developed a linear program (LP) formulation of the instruction flow problem which can be used to calculate the peak IPC and micro-operations per cycle (MPC) a benchmark kernel can theoretically achieve with a given port mapping. The actual port mapping of the underlying hardware is then determined by finding the mapping for which the throughput predicted by instruction flow best matches the actual measured IPC and MPC.

FUNCTIONAL DESCRIPTION: Pipedream is a tool for measuring specific performance characteristics of CPUs It is used to build the performance model of another tool called Gus (https://gitlab.inria.fr/nderumig/gus). Pipedream finds measured performance characteristics such as the throughput and latency of instructions by running a large set of automatically generated microbenchmarks. The tool can also find port mappings, a model of part of the CPU instruction scheduler, by analysing performance measurements of specially crafted microkernels using a LP solver. We have used it to produce a port mapping for the Intel Skylake CPU architecture. Pipedream is able to find the port mappings for some instructions for which existing approaches fall back to manual analysis.

- Contact: Nicolas Derumigny
- URL: https://gitlab.inria.fr/fgruber/pipedream

5.5. Platforms

5.5.1. Grid'5000

Grid'5000 [0] is a large-scale and versatile testbed for experiment-driven research in all areas of computer science, with a focus on parallel and distributed computing including Cloud, HPC and Big Data. It provides access to a large amount of resources: 14828 cores, 829 compute-nodes grouped in homogeneous clusters located in 8 sites in France connected through a dedicated network (Renater), and featuring various technologies (GPU, SSD, NVMe, 10G and 25G Ethernet, Infiniband, Omni-Path) and advanced monitoring and measurement features for traces collection of networking and power consumption, providing a deep understanding of experiments. It is highly reconfigurable and controllable. Researchers can experiment with a fully customized software stack thanks to bare-metal deployment features, and can isolate their experiment at the networking layer advanced monitoring and measurement features for traces collection of networking and power consumption, providing a deep understanding of experiments designed to support Open Science and reproducible research, with full traceability of infrastructure and software changes on the testbed. Frédéric Desprez is director of the GRID5000 GIS.

5.5.2. SILECS/SLICES

Frédéric Desprez is co-PI with Serge Fdida (Université Sorbonne) of the SILECS [0] infrastructure (IR ministère) which goal is to provide an experimental platform for experimental computer Science (Internet of things, clouds, HPC, big data, IA, wireless technologies, ...). This new infrastructure is based on two existing infrastructures, Grid'5000 and FIT. A European infrastructure (SLICES) is also currently designed with other european partners (Spain, Cyprus, Greece, Netherland, Switzerland, Poland, ...).

[0] https://www.grid5000.fr/
[0] https://www.silecs.net/

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.

ATMI (Analytical model of Temperature in MIcroprocessors) is an ad hoc temperature model 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: IPET - WCET - Performance - Real time - Static analysis - Worst Case Execution Time

SCIENTIFIC DESCRIPTION: WCET estimation

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.

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: Benjamin Lesage, Loïc Besnard, Damien Hardy, François Joulaud, Isabelle Puaut and Thomas Piquet
- Partner: Université de Rennes 1
- Contact: Isabelle Puaut
- URL: https://team.inria.fr/pacap/software/heptane/

6.3. tiptop

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

SCIENTIFIC DESCRIPTION: Tiptop is a new simple and flexible user-level tool that collects hardware counter data on Linux platforms (version 2.6.31+) and displays them in a way simple to the Linux "top" utility. The goal is to make the collection of performance and bottleneck data as simple as possible, including simple installation and usage. No privilege is required, any user can run tiptop.

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

Current version is 2.3.1, released October 2017. Tiptop has been integrated in major Linux distributions, such as Fedora, Debian, Ubuntu, CentOS.

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. PADRONE

KEYWORDS: Legacy code - Optimization - Performance analysis - Dynamic Optimization

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

6.5. If-memo

KEYWORD: Performance

SCIENTIFIC DESCRIPTION: We propose a linker based technique for enabling software memorizing of any dynamically linked pure function by function interception and we illustrate our framework using a set of computationally expensive pure functions - the transcendental functions.

FUNCTIONAL DESCRIPTION: If-memo is a linker-based technique for enabling software memorizing of any dynamically linked pure function by function interception. Typically, this framework is useful to intercept the computationally expensive pure functions - the transcendental functions from the math library. Our technique does not need the availability of source code and thus can even be applied 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. Our framework does not make any specific assumptions about the underlying architecture or compiler too-chains, and can work with a variety of current architectures.

- Participants: Arjun Suresh and Erven Rohou
- Contact: Erven Rohou
- URL: https://team.inria.fr/pacap/software/if-memo/

6.6. Simty

KEYWORDS: GPU - Softcore - FPGA - SIMT - Multi-threading - RISC-V

FUNCTIONAL DESCRIPTION: 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.

- Author: Caroline Collange
- Contact: Caroline Collange
- URL: https://gforge.inria.fr/projects/simty

6.7. Barra

KEYWORDS: GPU - GPGPU - Tesla ISA - Debug - Computer architecture - Performance - Profiling - Simulator - HPC - CUDA

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 is a Graphics Processing Unit (GPU) architecture simulator. It simulates NVIDIA CUDA programs at the assembly language level. Barra is a tool for research on computer architecture, and can also be used to debug, profile and optimize CUDA programs at the lowest level.

RELEASE FUNCTIONAL DESCRIPTION: Version 0.5.10 introduces: Timing model, Tesla-like architecture model, Fermi-like architecture model, New per-PC control-flow divergence management, Support for Simultaneous branch and warp interweaving, Support for Affine vector cache.

- Participants: Alexandre Kouyoumdjian, David Defour, Fabrice Mouhartem and Caroline Collange
- Partners: ENS Lyon - UPVD
- Contact: Caroline Collange
- URL: http://barra.gforge.inria.fr/

6.8. Memoization

KEYWORDS: Optimization - Pure function - Memoization

FUNCTIONAL DESCRIPTION: Memoization is a technique used at runtime that consists in caching results of pure functions and retrieving them instead of computing them when the arguments repeat. It can be applied to C and C++ programs. To be memoized, the interface of a pure function (or a method) must verify the following properties: (1) the function/method has at most four arguments of same type T, (2) the function/method returns a data of type T, (3) T is either 'double', 'float', or 'int'.

The memoization operation of a function/method is controlled by several parameters: the size of the internal table (number of entries), the replacement policy to be used in case of index conflict (whether the value of the table must be replaced or not), an approximation threshold that allows to not distinguish very close values). It is also possible to initialize the table with the content of a file, and to save the content of the table to a file at the end of the execution (the data may be used as input for a future execution).

- Participants: Loïc Besnard, Imane Lasri and Erven Rohou
- Contact: Loïc Besnard

6.9. FiPlib

KEYWORDS: Compilation - Approximate computing - Fixed-point representation

FUNCTIONAL DESCRIPTION: FiPlib is a C++ library that provides type definition and conversion operations for computations in fixed-point representation. Basic arithmetic as well as logical operations are transparently supported thanks to operator overloading. FiPlib also provides optimized implementations of the transcendental math functions of libm. For convenient integration, FiPlib is released as C++ header files only. Optionally, FiPlib can detect overflows and compute errors compared to floating point representation.

- Participants: Pierre Le Meur, Imane Lasri and Erven Rohou
- Contact: Erven Rohou

6.10. sigmask

KEYWORDS: Compilation - Side-channel - Masking - Security - Embedded systems

SCIENTIFIC DESCRIPTION: Sigmask is a compiler plugin based on the LLVM infrastructure that automatically protects secret information in programs, such as encryption keys, against side-channel attacks. The programmer annotates their source code to highlight variables containing sensitive data. The compiler automatically analyzes the program and computes all memory locations potentially derived from the secret. It then applies a masking scheme to avoid information leakage. Sigmask provides several schemes: OSDM (Orthogonal Direct Sum Masking), IP (Inner Product) Masking, and simple random bit masking. The programmer may also provide their own masking scheme through a well-defined API.

FUNCTIONAL DESCRIPTION: Sigmask is a compiler plugin based on the LLVM infrastructure that automatically protects secret information in programs, such as encryption keys, against side-channel attacks. The programmer annotates their source code to highlight variables containing sensitive data. The compiler automatically analyzes the program and computes all memory locations potentially derived from the secret. It then applies a masking scheme to avoid information leakage. Sigmask provides several schemes: ODSM (Orthogonal Direct Sum Masking), IP (Inner Product) Masking, and simple random bit masking. The programmer may also provide their own masking scheme through a well-defined API.

- Participants: Nicolas Kiss, Damien Hardy and Erven Rohou
- Contact: Erven Rohou

HYCOMES Project-Team

5. New Software and Platforms

5.1. Demodocos

Demodocos (Examples to Generic Scenario Models Generator)

KEYWORDS: Surgical process modelling - Net synthesis - Process mining

SCIENTIFIC DESCRIPTION: Demodocos is used to construct a Test and Flip net (Petri net variant) from a collection of instances of a given procedure. 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 result is a Test and Flip net and its marking graph. The tool can also build a #SEVEN scenario for integration into a virtual reality environment. The scenario obtained corresponds to the generalization of the input instances, namely the instances synthesis enriched with new behaviors respecting the relations of causality, conflicts and competition observed.

Demodocos is a 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.

FUNCTIONAL DESCRIPTION: The tool Demodocos allows to build a generic model for a given procedure from some examples of instances of this procedure. The generated model can take the form of a graph, a Test 'n Flip net or a SEVEN scenario (intended for integration into a virtual reality environment).

The classic use of the tool is to apply the summary operation to a set of files describing instances of the target procedure. Several file formats are supported, including the standard XES format for log events. As output, several files are generated. These files represent the generic procedure in different forms, responding to varied uses.

This application is of limited interest in the case of an isolated use, out of context and without a specific objective when using the model generated. It was developed as part of a research project focusing in particular on surgical procedures, and requiring the generation of a generic model for integration into a virtual reality training environment. It is also quite possible to apply the same method in another context.

- Participants: Aurélien Lamercerie and Benoît Caillaud
- Contact: Benoît Caillaud
- Publication: Surgical Process Mining with Test and Flip Net Synthesis

5.2. 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: Benoît Caillaud
- Contact: Benoît Caillaud
- URL: http://www.irisa.fr/s4/tools/mica/

5.3. IsamDAE

Implicit Structural Analysis of Multimode DAE systems

KEYWORDS: Structural analysis - Differential algebraic equations - Multimode - Scheduling

SCIENTIFIC DESCRIPTION: Modeling languages and tools based on Differential Algebraic Equations (DAE) bring several specific issues that do not exist with modeling languages based on Ordinary Differential Equations. The main problem is the determination of the differentiation index and latent equations. Prior to generating simulation code and calling solvers, the compilation of a model requires a structural analysis step, which reduces the differentiation index to a level acceptable by numerical solvers.

The Modelica language, among others, allows hybrid models with multiple modes, mode-dependent dynamics and state-dependent mode switching. These Multimode DAE (mDAE) systems are much harder to deal with. The main difficulties are (i) the combinatorial explosion of the number of modes, and (ii) the correct handling of mode switchings.

The aim of the software is on the first issue, namely: How can one perform a structural analysis of an mDAE in all possible modes, without enumerating these modes? A structural analysis algorithm for mDAE systems has been designed and implemented, based on an implicit representation of the varying structure of an mDAE. It generalizes J. Pryce's Sigma-method to the multimode case and uses Binary Decision Diagrams (BDD) to represent the mode-dependent structure of an mDAE. The algorithm determines, as a function of the mode, the set of latent equations, the leading variables and the state vector. This is then used to compute a mode-dependent block-triangular decomposition of the system, that can be used to generate simulation code with a mode-dependent scheduling of the blocks of equations.

FUNCTIONAL DESCRIPTION: IsamDAE (Implicit Structural Analysis of Multimode DAE systems) is a software library for testing new structural analysis algorithms for multimode DAE systems, based on an implicit representation of incidence graphs, matchings between equations and variables, and block decompositions. The input of the software is a variable dimension multimode DAE system consisting in a set of guarded equations and guarded variable declarations. It computes a mode-dependent structural index-reduction of the multimode system and produces a mode-dependent graph for the scheduling of blocks of equations. Evaluation functions make it possible to return the lists of leading equations and leading variables, as well as the actual scheduling of blocks, in a specified mode.

IsamDAE is coded in OCaml, and uses the following packages: * MLBDD by Arlen Cox, * Menhir by François Pottier and Yann Régis-Gianas, * Pprint by François Pottier, * XML-Light by Nicolas Cannasse and Jacques Garrigue.

RELEASE FUNCTIONAL DESCRIPTION: Version 0.2: * MEL: ad hoc language for the declaration of variable dimension multi-mode DAE systems * automatic parsing, model checking and model allocation * XML output for the list of evaluation blocks (parameters, equations, unknowns to be computed) * new algorithms for the mode-dependent scheduling and the evaluation of the scheduling in a given mode

NEWS OF THE YEAR: It has been possible to perform the structural analysis of systems with more than 750 equations and 10 to the power 23 modes, therefore demonstrating the scalability of the method.

- Authors: Benoît Caillaud and Mathias Malandain
- Contact: Benoît Caillaud

Kairos Project-Team

6. New Software and Platforms

6.1. VerCors

VERification of models for distributed communicating COmponants, with safety and Security

KEYWORDS: Software Verification - Specification language - Model Checking

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.

RELEASE FUNCTIONAL DESCRIPTION: It includes integrated graphical editors for GCM component architecture descriptions, UML classes, interfaces, and state-machines. The user diagrams can be checked using the recently published validation rules from, then the corresponding GCM components can be executed using an automatic generation of the application ADL, and skeletons of Java files.

The experimental version (2019) also includes algorithms for computing the symbolic semantics of Open Systems, using symbolic methods based on the Z3 SMT engine.

NEWS OF THE YEAR: The experimental version (2019) also includes: - algorithms for computing the symbolic semantics of Open Systems, using symbolic methods based on the Z3 SMT engine. - a stand alone textual editor for (open) pNet systems, that generates API code to construct their internal representation in the platform.

- Participants: Antonio Cansado, Bartlomiej Szejna, Eric Madelaine, Ludovic Henrio, Marcela Rivera, Nassim Jibai, Oleksandra Kulankhina, Siqi Li, Xudong Qin and Zechen Hou
- Partner: East China Normal University Shanghai (ECNU)
- Contact: Eric Madelaine
- URL: https://team.inria.fr/scale/software/vercors/

6.2. 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: Benoît Ferrero, Charles André, Frédéric Mallet, Julien DeAntoni and Nicolas Chleq
- Contact: Julien DeAntoni
- URL: http://timesquare.inria.fr

6.3. GEMOC Studio

KEYWORDS: DSL - Language workbench - Model debugging

SCIENTIFIC DESCRIPTION: The language workbench put together the following tools seamlessly integrated to the Eclipse Modeling Framework (EMF):

- Melange, a tool-supported meta-language to modularly define executable modeling languages with execution functions and data, and to extend (EMF-based) existing modeling languages.
- MoCCML, a tool-supported meta-language dedicated to the specification of a Model of Concurrency and Communication (MoCC) and its mapping to a specific abstract syntax and associated execution functions of a modeling language.
- GEL, a tool-supported meta-language dedicated to the specification of the protocol between the execution functions and the MoCC to support the feedback of the data as well as the callback of other expected execution functions.
- BCOoL, a tool-supported meta-language dedicated to the specification of language coordination patterns to automatically coordinates the execution of, possibly heterogeneous, models.
- Sirius Animator, an extension to the model editor designer Sirius to create graphical animators for executable modeling languages.

FUNCTIONAL DESCRIPTION: The GEMOC Studio is an eclipse package that contains components supporting the GEMOC methodology for building and composing executable Domain-Specific Modeling Languages (DSMLs). It includes the two workbenches: The GEMOC Language Workbench: intended to be used by language designers (aka domain experts), it allows to build and compose new executable DSMLs. The GEMOC Modeling Workbench: intended to be used by domain designersto create, execute and coordinate models conforming to executable DSMLs. The different concerns of a DSML, as defined with the tools of the language workbench, are automatically deployed into the modeling workbench. They parametrize a generic execution framework that provide various generic services such as graphical animation, debugging tools, trace and event managers, timeline, etc.

- Participants: Didier Vojtisek, Dorian Leroy, Erwan Bousse, Fabien Coulon and Julien DeAntoni
- Partners: IRIT - ENSTA - I3S - OBEO - Thales TRT
- Contact: Benoît Combemale
- URL: http://gemoc.org/studio.html

6.4. BCOol

BCOoL

KEYWORDS: DSL - Language workbench - Behavior modeling - Model debugging - Model animation

FUNCTIONAL DESCRIPTION: BCOoL is a tool-supported meta-language dedicated to the specification of language coordination patterns to automatically coordinates the execution of, possibly heterogeneous, models.

- Participants: Julien DeAntoni, Matias Vara Larsen, Benoît Combemale and Didier Vojtisek
- Contact: Julien DeAntoni
- URL: http://www.gemoc.org

6.5. myMed

A geo-localised Framework for building Publish-Subscribe applications in a fixed and mobile environment

KEYWORDS: Framework - Peer-to-peer. - NoSQL - Mobile application - Social network - Publish-subscribe - Iot - Peer-to-peer

SCIENTIFIC DESCRIPTION: myMed : an ad-hoc framework to design, develop, host, and execute Publish-Subscribe based fully distributed applications running in a static or mobile network. Application examples can be found in Online Social Networks or in Resource Discovery for the IoT. In a nutshell myMed is composed by:

• A myMed Software Development Kit (SDK) to develop fixed and mobile web applications, but also to build native applications on Smartphones equipped with Android or iOS. Every module can be freely used without interfering with other applications, in a true Lego fashion.

• A myMed cloud to execute the mobile applications: the cloud is composed of a backbone of 50PCs, distributed through the "AlpMed" EuroRegion and following some precise network criteria (4G, optical Fiber, ..). The operating system running on those PC is a customised and partitioned version of Ubuntu to allow to use the PC as a myMed server as well as a ordinary desktops. As in Peer-to-Peer technology, we do not require that all machines belonging to the backbone are constantly running.

• A myMed backbone, based on a well-tested noSQL database, Cassandra, which can accommodate any number of users without any code changes. Machines can be classically concentrated on a data-center or – more interestingly – fully decentralized (modulo a decent internet connection). Failures of one or many machines do not affect the running of the system, thanks to replication of the data on several servers. A little collection of proof of concept applications to validate, experiment, and testing the development kit and the execution cloud have been implemented.

FUNCTIONAL DESCRIPTION: myMed is an experimental framework for implementing, hosting and deploying, on the top of a built-in cloud platform, many applications using intensively the Publish-Subscribe (PUB/SUB) paradigm, like e.g. Open Social Networks or Resource Discovery in a distributed data-base. Those applications could take advantage of sharing common software modules, hardware resources, making inter-communication and inter-interaction simpler and improving rapid development and deployement.

- Participants: Luigi Liquori, Claudio Casetti, Mariangiola Dezani and Mino Anglano
- Partners: Politecnico di Torino - Université de Nice Sophia Antipolis (UNS) - Università di Torino - Università del Piemonte Orientale
- Contact: Luigi Liquori
- URL: http://www.mymed.fr

6.6. JMaxGraph

KEYWORDS: Java - HPC - Graph algorithmics

FUNCTIONAL DESCRIPTION: JMaxGraph is a collection of techniques for the computation of large graphs on one single computer. The motivation for such a centralized computing platform originates in the constantly increasing efficiency of computers which now come with hundred gigabytes of RAM, tens of cores and fast drives. JMaxGraph implements a compact adjacency-table for the representation of the graph in memory. This data structure is designed to 1) be fed page by page, à-la GraphChi, 2) enable fast iteration, avoiding memory jumps as much as possible in order to benefit from hardware caches, 3) be tackled in parallel by multiple-threads. Also, JMaxGraph comes with a flexible and resilient batch-oriented middleware, which is suited to executing long computations on shared clusters. The first use-case of JMaxGraph allowed F. Giroire, T. Trolliet and S. Pérennes to count K2,2s, and various types of directed triangles in the Twitter graph of users (23G arcs, 400M vertices). The computation campaign took 4 days, using up to 400 cores in the NEF Inria cluster.

- Contact: Luc Hogie
- URL: http://www.i3s.unice.fr/~hogie/software/?name=jmaxgraph

6.7. Lopht

Logical to Physical Time Compiler

KEYWORDS: Real time - Compilation

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

FUNCTIONAL DESCRIPTION: Compilation of high-level embedded systems specifications into executable code for IMA/ARINC 653 avionics platforms. It ensures the functional and non-functional correctness of the generated code.

- Participants: Dumitru Potop-Butucaru, Manel Djemal, Thomas Carle and Zhen Zhang
- Contact: Dumitru Potop-Butucaru

6.8. LoPhT-manycore

Logical to Physical Time compiler for many cores

KEYWORDS: Real time - Compilation - Task scheduling - Automatic parallelization

SCIENTIFIC DESCRIPTION: Lopht is a system-level compiler for embedded systems, whose objective is to fully automate the implementation process for certain classes of embedded systems. Like in a classical compiler (e.g. gcc), its input is formed of two objects. The first is a program providing a platform-indepedent description of the functionality to implement and of the non-functional requirements it must satisfy (e.g. real-time, partitioning). This is provided under the form of a data-flow synchronous program annotated with non-functional requirements. The second is a description of the implementation platform, defining the topology of the platform, the capacity of its elements, and possibly platform-dependent requirements (e.g. allocation).

From these inputs, Lopht produces all the C code and configuration information needed to allow compilation and execution on the physical target platform. Implementations are correct by construction Resulting implementations are functionally correct and satisfy the non-functional requirements. Lopht-manycore is a version of Lopht targeting shared-memory many-core architectures.

The algorithmic core of Lopht-manycore is formed of timing analysis, allocation, scheduling, and code generation heuristics which rely on four fundamental choices. 1) A static (off-line) real-time scheduling approach where allocation and scheduling are represented using time tables (also known as scheduling or reservation tables). 2) Scalability, attained through the use of low-complexity heuristics for all synthesis and associated analysis steps. 3) Efficiency (of generated implementations) is attained through the use of precise representations of both functionality and the platform, which allow for fine-grain allocation of resources such as CPU, memory, and communication devices such as network-on-chip multiplexers. 4) Full automation, including that of the timing analysis phase.

The last point is characteristic to Lopht-manycore. Existing methods for schedulability analysis and real-time software synthesis assume the existence of a high-level timing characterization that hides much of the hardware complexity. For instance, a common hypothesis is that synchronization and interference costs are accounted for in the duration of computations. However, the high-level timing characterization is seldom (if ever) soundly derived from the properties of the platform and the program. In practice, large margins (e.g. 100%) with little formal justification are added to computation durations to account for hidden hardware complexity. Lopht-manycore overcomes this limitation. Starting from the worst-case execution time (WCET) estimations of

computation operations and from a precise and safe timing model of the platform, it maintains a precise timing accounting throughout the mapping process. To do this, timing accounting must take into account all details of allocation, scheduling, and code generation, which in turn must satisfy specific hypotheses.

FUNCTIONAL DESCRIPTION: Accepted input languages for functional specifications include dialects of Lustre such as Heptagon and Scade v4. To ensure the respect of real-time requirements, Lopht-manycore pilots the use of the worst-case execution time (WCET) analysis tool (ait from AbsInt). By doing this, and by using a precise timing model for the platform, Lopht-manycore eliminates the need to adjust the WCET values through the addition of margins to the WCET values that are usually both large and without formal safety guarantees. The output of Lopht-manycore is formed of all the multi-threaded C code and configuration information needed to allow compilation, linking/loading, and real-time execution on the target platform.

NEWS OF THE YEAR: In the framework of the ITEA3 ASSUME project we have extended the Lopht-manycore to allow multiple cores to access the same memory bank at the same time. To do this, the timing accounting of Lopht has been extended to take into account memory access interferences during the allocation and scheduling process. Lopht now also pilots the aiT static WCET analysis tool from AbsInt by generating the analysis scripts, thus ensuring the consistency between the hypotheses made by Lopht and the way timing analysis is performed by aiT. As a result, we are now able to synthesize code for the computing clusters of the Kalray MPPA256 platform. Lopht-manycore is evaluated on avionics case studies in the perspective of increasing its technology readiness level for this application class.

- Participants: Dumitru Potop-Butucaru and Keryan Didier
- Contact: Dumitru Potop-Butucaru

KOPERNIC Team

6. New Software and Platforms

6.1. SynDEx

KEYWORDS: Distributed - Optimization - Real time - Embedded systems - 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

NEWS OF THE YEAR: We improved the distribution and scheduling heuristics to take into account the needs of co-simulation.

- Participant: Yves Sorel
- Contact: Yves Sorel
- URL: http://www.syndex.org

6.2. EVT Kopernic

KEYWORDS: Embedded systems - Worst Case Execution Time - Real-time application - Statistics

SCIENTIFIC DESCRIPTION: The EVT-Kopernic tool is an implementation of the Extreme Value Theory (EVT) for the problem of the statistical estimation of worst-case bounds for the execution time of a program on a processor. Our implementation uses the two versions of EVT - GEV and GPD - to propose two independent methods of estimation. Their results are compared and only results that are sufficiently close allow to validate an estimation. Our tool is proved predictable by its unique choice of block (GEV) and threshold (GPD) while proposant reproducible estimations.

FUNCTIONAL DESCRIPTION: EVT-Kopernic is tool proposing a statistical estimation for bounds on worst-case execution time of a program on a processor. The estimator takes into account dependences between execution times by learning from the history of execution, while dealing also with cases of small variability of the execution times.

NEWS OF THE YEAR: Any statistical estimator should come with an representative measurement protocole based on the processus of composition, proved correct. We propose the first such principle of composition while using a Bayesien modeling taking into account iteratively different measurement models. The composition model has been described in a patent submitted this year with a scientific publication under preparation.

- Participants: Adriana Gogonel and Liliana Cucu
- Contact: Adriana Gogonel
- URL: http://inria-rscript.serveftp.com/

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

5.2. GCC

KEYWORDS: Compilation - Polyhedral compilation

FUNCTIONAL DESCRIPTION: The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgcj,...). GCC was originally written as the compiler for the GNU operating system. The GNU system was developed to be 100% free software, free in the sense that it respects the user's freedom.

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

5.3. Heptagon

KEYWORDS: Compilers - Synchronous Language - Controller synthesis

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

The current version of the compiler includes the following features: - Inclusion of discrete controller synthesis within the compilation: the language is equipped with a behavioral contract mechanisms, where assumptions can be described, as well as an "enforce" property part. The semantics of this latter is that the property should be enforced by controlling the behaviour of the node equipped with the contract. This property will be enforced by an automatically built controller, which will act on free controllable variables given by the programmer. This extension has been named BZR in previous works. - Expression and compilation of array values with modular memory optimization. The language allows the expression and operations on arrays (access, modification, iterators). With the use of location annotations, the programmer can avoid unnecessary array copies.

- Participants: Adrien Guatto, Brice Gelineau, Cédric Pasteur, Eric Rutten, Gwenaël Delaval, Léonard Gérard and Marc Pouzet
- Partners: UGA - ENS Paris - Inria - LIG
- Contact: Gwenaël Delaval
- URL: http://heptagon.gforge.inria.fr

5.4. isl

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

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

5.5. Lem

lightweight executable mathematics

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

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

5.6. Lucid Synchrone

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

RELEASE FUNCTIONAL DESCRIPTION: The language is still used for teaching and in our research but we do not develop it anymore. Nonetheless, we have integrated several features from Lucid Synchrone in new research prototypes described below. The Heptagon language and compiler are a direct descendent of it. The new language Zélus for hybrid systems modeling borrows many features originaly introduced in Lucid Synchrone.

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

5.7. Lucy-n

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

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

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

5.8. Ott

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

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

a Coq version of the definition,

an Isabelle version of the definition, and

a HOL version of the definition.

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

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

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

5.9. PPCG

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

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

5.10. ReactiveML

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

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

- Participants: Cédric Pasteur, Guillaume Baudart and Louis Mandel
- Contact: Guillaume Baudart

5.11. SundialsML

Sundials/ML

KEYWORDS: Simulation - Mathematics - Numerical simulations

SCIENTIFIC DESCRIPTION: Sundials/ML is a comprehensive OCaml interface to the Sundials suite of numerical solvers (CVODE, CVODES, IDA, IDAS, KINSOL). Its structure mostly follows that of the Sundials library, both for ease of reading the existing documentation and for adapting existing source code, but several changes have been made for programming convenience and to increase safety, namely:

solver sessions are mostly configured via algebraic data types rather than multiple function calls,

errors are signalled by exceptions not return codes (also from user-supplied callback routines),

user data is shared between callback routines via closures (partial applications of functions),

vectors are checked for compatibility (using a combination of static and dynamic checks), and

explicit free commands are not necessary since OCaml is a garbage-collected language.

FUNCTIONAL DESCRIPTION: Sundials/ML is a comprehensive OCaml interface to the Sundials suite of numerical solvers (CVODE, CVODES, IDA, IDAS, KINSOL, ARKODE).

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

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

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

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

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

5.12. Zelus

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

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

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

5.13. Telamon

KEYWORDS: Compilation - Monte-Carlo methods - Constraint Programming - GPU - Dense linear algebra

FUNCTIONAL DESCRIPTION: Telamon is a framework for the optimization of computational kernels for GPUs through efficient search in a well-behaved optimization space in which optimization decisions commute and satisfaction of constraints restricting legal optimizations.

- Contact: Ulysse Beaugnon
- URL: https://github.com/ulysseB/telamon/

5.14. Vélus

Verified Lustre Compiler

KEYWORDS: Synchronous Language - Compilation - Software Verification - Coq - Ocaml

FUNCTIONAL DESCRIPTION: Vélus is a prototype compiler from a subset of Lustre to assembly code. It is written in a mix of Coq and OCaml and incorporates the CompCert verified C compiler. The compiler includes formal specifications of the semantics and type systems of Lustre, as well as the semantics of intermediate languages, and a proof of correctness that relates the high-level dataflow model to the values produced by iterating the generated assembly code.

RELEASE FUNCTIONAL DESCRIPTION: First source-code release. Treatment of primitive reset construct. Clocks allowed for node arguments.

- Contact: Timothy Bourke
- URL: https://velus.inria.fr

5.15. Tensor Comprehensions

KEYWORDS: Machine learning - Matrix calculation - Polyhedral compilation - GPU - CUDA

FUNCTIONAL DESCRIPTION: Tensor Comprehensions (TC) is a notation based on generalized Einstein notation for computing on multi-dimensional arrays. TC greatly simplifies ML framework implementations by providing a concise and powerful syntax which can be efficiently translated to high-performance computation kernels, automatically.

RELEASE FUNCTIONAL DESCRIPTION: Integration of the loop tactics matching framework for identifying linear algebra operations and optimizing them.

- Partner: Eindhoven University of Technology
- Contact: Albert Cohen

5.16. Aftermath

KEYWORDS: Performance analysis - Runtime system - Parallel programming - High-performance calculation - Execution trace

FUNCTIONAL DESCRIPTION: Aftermath is a toolkit for building custom graphical tools for trace-based performance analysis of parallel programs, run-time systems and compilers.

- Partner: The University of Manchester
- Contact: Andi Drebes

5.17. MPPcodegen

Source-to-source loop tiling based on MPP

KEYWORDS: Source-to-source compiler - Polyhedral compilation

FUNCTIONAL DESCRIPTION: MPPcodegen applies a monoparametric tiling to a C program enriched with pragmas specifying the tiling and the scheduling function. The tiling can be generated by any convex polyhedron and translation functions, it is not necessarily a partition. The result is a C program depending on a scaling factor (the parameter). MPPcodegen relies on the MPP mathematical library to tile the iteration sets.

- Partner: Colorado State University
- Contact: Christophe Alias
- URL: http://foobar.ens-lyon.fr/mppcodegen/

5.18. MPP

MonoParametric Partitionning transformation

KEYWORDS: Compilation - Polyhedral compilation

FUNCTIONAL DESCRIPTION: This library applies a monoparametric partitioning transformation to polyhedra and affine functions. This transformation is a subset of the parametric sized tiling transformation, specialized for the case where shapes depend only on a single parameter. Unlike in the general case, the resulting sets and functions remain in the polyhedral model.

- Contact: Guillaume Iooss
- URL: https://github.com/guillaumeiooss/MPP

5.19. Obelisk

KEYWORDS: LaTeX - HTML - Ocaml

FUNCTIONAL DESCRIPTION: Obelisk is a simple tool which produces pretty-printed output from a Menhir parser file (.mly).

It is inspired from yacc2latex and it is also written in OCaml, but it is aimed at supporting features from Menhir instead of only those of ocamlyacc.

- Contact: Lelio Brun
- URL: https://github.com/Lelio-Brun/Obelisk

SPADES Project-Team

5. New Software and Platforms

5.1. pyCPA_TWCA

Analysis tool for weakly-hard real-time systems

KEYWORDS: Real time - Scheduling analyses

FUNCTIONAL DESCRIPTION: pyCPA_TWCA is a pyCPA plugin for Typical Worst-Case Analysis. 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 that tool that is co-developed by Sophie Quinton and Zain Hammadeh at TU Braunschweig. It allows in particular the computation of weakly-hard guarantees for real-time tasks, i.e. number of deadline misses out of a sequence of executions. So far, pyCPA_TWCA is restricted to uniprocessor systems of independent tasks. pyCPA_TWCA can handle the following scheduling policies: Fixed Priority Preemptive, Fixed Priority Non-Preemptive, Weighted Round-Robin, Earliest Deadline First.

- Contact: Sophie Quinton

5.2. CertiCAN

Certifier of CAN bus analysis results

KEYWORDS: Certification - CAN bus - Real time - Static analysis

FUNCTIONAL DESCRIPTION: CertiCAN is a tool, produced using the Coq proof assistant, allowing the formal certification of the correctness of CAN bus analysis results. Result certification is a process that is lightweight and flexible compared to tool certification, which makes it a practical choice for industrial purposes. The analysis underlying CertiCAN, which is based on a combined use of two well-known CAN analysis techniques, is computationally efficient. Experiments demonstrate that CertiCAN is able to certify the results of RTaW-Pegase, an industrial CAN analysis tool, even for large systems. Furthermore, CertiCAN can certify the results of any other RTA tool for the same analysis and system model (periodic tasks with offsets in transactions).

- Contact: Xiaojie Guo

TEA Project-Team

6. New Software and Platforms

6.1. ADFG

Affine data-flow graphs schedule synthesizer

KEYWORDS: Code generation - Scheduling - Static program analysis

FUNCTIONAL DESCRIPTION: ADFG is a synthesis tool of real-time system scheduling parameters: ADFG computes task periods and buffer sizes of systems resulting in a trade-off between throughput maximization and buffer size minimization. ADFG synthesizes systems modeled by ultimately cyclo-static dataflow (UCSDF) graphs, an extension of the standard CSDF model.

Knowing the WCET (Worst Case Execute Time) of the actors and their exchanges on the channels, ADFG tries to synthezise the scheduler of the application. ADFG offers several scheduling policies and can detect unschedulable systems. It ensures that the real scheduling does not cause overflows or underflows and tries to maximize the throughput (the processors utilization) while minimizing the storage space needed between the actors (i.e. the buffer sizes).

Abstract affine scheduling is first applied on the dataflow graph, 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 schedulability policies analysis (i.e., synthesis of timing and scheduling parameters of actors) is applied to produce the scheduller for the actors.

ADFG, initially defined to synthesize real-time schedulers for SCJ/L1 applications, may be used for scheduling analysis of AADL programs.

- Authors: Thierry Gautier, Jean-Pierre Talpin, Adnan Bouakaz, Alexandre Honorat and Loïc Besnard
- Contact: Loïc Besnard

6.2. POLYCHRONY

KEYWORDS: Code generation - AADL - Proof - Optimization - Multi-clock - GALS - Architecture - Cosimulation - Real time - Synchronous Language

FUNCTIONAL DESCRIPTION: Polychrony 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. 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 POP Eclipse platform, a front-end to the Signal toolbox in the Eclipse environment. It is distributed under EPL license.

- Participants: Loïc Besnard, Paul Le Guernic and Thierry Gautier
- Partners: CNRS - Inria
- Contact: Loïc Besnard
- URL: https://www.polarsys.org/projects/polarsys.pop

6.3. Polychrony AADL2SIGNAL

KEYWORDS: Real-time application - Polychrone - Synchronous model - Polarsys - Polychrony - Signal - AADL - Eclipse - Meta model

FUNCTIONAL DESCRIPTION: This polychronous MoC has been used previously as semantic model for systems described in the core AADL standard. The core AADL is extended with annexes, such as the Behavior Annex, which allows to specify more precisely architectural behaviors. The translation from AADL specifications into the polychronous model should take into account these behavior specifications, which are based on description of automata.

For that purpose, the AADL state transition systems are translated as Signal automata (a slight extension of the Signal language has been defined to support the model of polychronous automata).

Once the AADL model of a system transformed into a Signal program, one can analyze the program using the Polychrony framework in order to check if timing, scheduling and logical requirements over the whole system are met.

We have implemented the translation and experimented it using a concrete case study, which is the AADL modeling of an Adaptive Cruise Control (ACC) system, a highly safety-critical system embedded in recent cars.

- Participants: Huafeng Yu, Loïc Besnard, Paul Le Guernic, Thierry Gautier and Yue Ma
- Partner: CNRS
- Contact: Loïc Besnard
- URL: http://www.inria.fr/equipes/tea

6.4. POP

Polychrony on Polarsys

KEYWORDS: Synchronous model - Model-driven engineering

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

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 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.

- Participants: Jean-Pierre Talpin, Loïc Besnard, Paul Le Guernic and Thierry Gautier
- Contact: Loïc Besnard
- URL: https://www.polarsys.org/projects/polarsys.pop

6.5. Sigali

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

- Contact: Hervé Marchand

0

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

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

The AstréeA Static Analyzer of Asynchronous Software

KEYWORDS: Static analysis - Static program analysis - Program verification - Software Verification - Abstraction

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 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.

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: Antoine Miné, Jérôme Feret, Laurent Mauborgne, Patrick Cousot, Radhia Cousot and Xavier Rival
- Partners: CNRS - ENS Paris - AbsInt Angewandte Informatik GmbH
- Contact: Patrick Cousot
- URL: http://www.astree.ens.fr/

6.3. AstréeA

The AstréeA Static Analyzer of Asynchronous Software

KEYWORDS: Static analysis - Static program analysis

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: Antoine Miné, Jérôme Feret, Patrick Cousot, Radhia Cousot and Xavier Rival
- Partners: CNRS - ENS Paris - AbsInt Angewandte Informatik GmbH
- Contact: Patrick Cousot
- URL: http://www.astreea.ens.fr/

6.4. ClangML

KEYWORD: Compilation

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: Devin Mccoughlin, François Berenger and Pippijn Van Steenhoven
- Contact: Xavier Rival
- 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: Antoine Miné and Caterina Urban
- Contact: Caterina Urban
- URL: http://www.di.ens.fr/~urban/FuncTion.html

6.6. HOO

Heap Abstraction for Open Objects

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

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

- Participant: Arlen Cox
- Contact: Arlen Cox

6.7. MemCAD

The MemCAD static analyzer

KEYWORDS: Static analysis - Abstraction

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 300 small size test cases that are used as regression tests.

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

6.8. KAPPA

A rule-based language for modeling interaction networks

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.

FUNCTIONAL DESCRIPTION: Kappa is provided with the following tools: - a compiler - a stochastic simulator - a static analyzer - a trace compression algorithm - an ODE generator.

RELEASE FUNCTIONAL DESCRIPTION: On line UI, Simulation is based on a new data-structure (see ESOP 2017), New abstract domains are available in the static analyzer (see SASB 2016), Local traces (see TCBB 2018), Reasoning on polymers (see SASB 2018).

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

6.9. 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

6.10. LCertify

KEYWORD: Compilation

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 guarantees semantic equivalence. 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
- Partners: CNRS - ENS Paris
- Contact: Xavier Rival
- URL: http://www.di.ens.fr/~rival/lcertify.html

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

6.12. PYPPAI

Pyro Probabilistic Program Analyzer

KEYWORDS: Probability - Static analysis - Program verification - Abstraction

FUNCTIONAL DESCRIPTION: PYPPAI is a program analyzer to verify the correct semantic definition of probabilistic programs written in Pyro. At the moment, PYPPAI verifies consistency conditions between models and guides used in probabilistic inference programs.

PYPPAI is written in OCaml and uses the pyml Python in OCaml library. It features a numerical abstract domain based on Apron, an abstract domain to represent zones in tensors, and dedicated abstract domains to describe distributions and states in probabilistic programs.

- Contact: Xavier Rival
- URL: https://github.com/wonyeol/static-analysis-for-support-match

CAMBIUM Project-Team

5. New Software and Platforms

5.1. OCaml

KEYWORDS: Functional programming - Static typing - Compilation

FUNCTIONAL DESCRIPTION: 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 System Z), a debugger, a documentation generator, a compilation manager, a package manager, and many libraries contributed by the user community.

- Participants: Damien Doligez, Xavier Leroy, Fabrice Le Fessant, Luc Maranget, Gabriel Scherer, Alain Frisch, Jacques Garrigue, Marc Shinwell, Jeremy Yallop and Leo White
- Contact: Damien Doligez
- URL: https://ocaml.org/

5.2. Compcert

The CompCert formally-verified C compiler

KEYWORDS: Compilers - Formal methods - Deductive program verification - C - Coq

FUNCTIONAL DESCRIPTION: CompCert is a compiler for the C programming language. Its intended use is the compilation of life-critical and mission-critical software written in C and meeting high levels of assurance. It accepts most of the ISO C 99 language, with some exceptions and a few extensions. It produces machine code for the ARM, PowerPC, RISC-V, and x86 architectures. What sets CompCert C apart from any other production compiler, is that it is formally verified to be exempt from miscompilation issues, using machine-assisted mathematical proofs (the Coq proof assistant). In other words, the executable code it produces is proved to behave exactly as specified by the semantics of the source C program. This level of confidence in the correctness of the compilation process is unprecedented and contributes to meeting the highest levels of software assurance. In particular, using the CompCert C compiler is a natural complement to applying formal verification techniques (static analysis, program proof, model checking) at the source code level: the correctness proof of CompCert C guarantees that all safety properties verified on the source code automatically hold as well for the generated executable.

RELEASE FUNCTIONAL DESCRIPTION: Novelties include a formally-verified type checker for CompCert C, a more careful modeling of pointer comparisons against the null pointer, algorithmic improvements in the handling of deeply nested struct and union types, much better ABI compatibility for passing composite values, support for GCC-style extended inline asm, and more complete generation of DWARF debugging information (contributed by AbsInt).

- Participants: Xavier Leroy, Sandrine Blazy, Jacques-Henri Jourdan, Sylvie Boldo and Guillaume Melquiond
- Partner: AbsInt Angewandte Informatik GmbH
- Contact: Xavier Leroy
- URL: http://compcert.inria.fr/

5.3. Diy

Do It Yourself

KEYWORD: Parallelism

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

- Participants: Jade Alglave and Luc Maranget
- Partner: University College London UK
- Contact: Luc Maranget
- URL: http://diy.inria.fr/

5.4. Menhir

KEYWORDS: Compilation - Context-free grammars - Parsing

FUNCTIONAL DESCRIPTION: Menhir is a LR(1) parser generator for the OCaml programming language. That is, Menhir compiles LR(1) grammar specifications down to OCaml code. Menhir was designed and implemented by François Pottier and Yann Régis-Gianas.

- Contact: François Pottier
- Publications: A Simple, Possibly Correct LR Parser for C11 - Reachability and Error Diagnosis in LR(1) Parsers

5.5. 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 notations and tactics for manipulating characteristic formulae interactively in Coq.

- Participants: Arthur Charguéraud, Armaël Guéneau and François Pottier
- Contact: Arthur Charguéraud
- URL: http://www.chargueraud.org/softs/cfml/

5.6. TLAPS

TLA+ proof system

KEYWORD: Proof assistant

SCIENTIFIC DESCRIPTION: 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 proof steps that can be checked independently. 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+ set theory 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.

FUNCTIONAL DESCRIPTION: TLAPS is a proof assistant for the TLA+ specification language.

NEWS OF THE YEAR: Work in 2019 focused on providing support for reasoning about TLA+'s ENABLED and action composition constructs. We also prepared a minor release, fixing some issues and switching to Z3 as the default SMT back-end solver.

- Participants: Damien Doligez, Stephan Merz and Ioannis Filippidis
- Contact: Stephan Merz
- URL: https://tla.msr-inria.inria.fr/tlaps/content/Home.html

5.7. ZENON

KEYWORD: Automated theorem proving

FUNCTIONAL DESCRIPTION: 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 retargeted to output scripts for different frameworks (for example, Isabelle and Dedukti).

- Author: Damien Doligez
- Contact: Damien Doligez
- URL: http://zenon-prover.org/

5.8. hevea

hevea is a fast latex to html translator.

KEYWORDS: LaTeX - Web

FUNCTIONAL DESCRIPTION: HEVEA is a LATEX to html translator. The input language is a fairly complete subset of LATEX 2 (old LATEX style is also accepted) and the output language is html that is (hopefully) correct with respect to version 5. HEVEA understands LATEX macro definitions. Simple user style files are understood with little or no modifications. Furthermore, HEVEA customisation is done by writing LATEX code.

HEVEA is written in Objective Caml, as many lexers. It is quite fast and flexible. Using HEVEA it is possible to translate large documents such as manuals, books, etc. very quickly. All documents are translated as one single html file. Then, the output file can be cut into smaller files, using the companion program HACHA. HEVEA can also be instructed to output plain text or info files.

Information on HEVEA is available at http://hevea.inria.fr/.

- Author: Luc Maranget
- Contact: Luc Maranget
- URL: http://hevea.inria.fr/

CELTIQUE Project-Team

3. New Software and Platforms

3.1. CompcertSSA

KEYWORDS: Optimizing compiler - Formal methods - Proof assistant - SSA

FUNCTIONAL DESCRIPTION: CompcertSSA is built on top of the Compcert verified C compiler, by adding a middle-end based on the SSA form (Static Single Assignment) : conversion to SSA, SSA-based optimizations, and destruction of SSA.

- Participants: Sandrine Blazy, Delphine Demange, Yon Fernandez de Retana, David Pichardie and Leo Stefanesco
- Contact: Delphine Demange
- Publications: Mechanizing conventional SSA for a verified destruction with coalescing - Validating Dominator Trees for a Fast, Verified Dominance Test - A Formally Verified SSA-based Middle-end : Static Single Assignment meets CompCert - Formal Verification of an SSA-based Middle-end for CompCert - Verifying Fast and Sparse SSA-based Optimizations in Coq.
- URL: http://compcertssa.gforge.inria.fr/

3.2. Jacal

JAvaCard AnaLyseur

KEYWORDS: JavaCard - Certification - Static program analysis - AFSCM

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

- Participants: David Pichardie, Delphine Demange, Frédéric Besson and Thomas Jensen
- Contact: Thomas Jensen

3.3. Javalib

KEYWORDS: Library - Java - Ocaml

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

3.4. JSCert

Certified JavaScript

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

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

3.5. 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: David Pichardie, Frédéric Besson and Laurent Guillo
- Partners: CNRS - ENS Cachan
- Contact: David Pichardie
- URL: http://sawja.inria.fr/

3.6. Timbuk

KEYWORDS: Automated deduction - Ocaml - Program verification - Tree Automata - Term Rewriting Systems

FUNCTIONAL DESCRIPTION: Timbuk is a tool designed to compute or over-approximate sets of terms reachable by a given term rewriting system. The libray also provides an OCaml toplevel with all usual functions on Bottom-up Nondeterministic Tree Automata.

- Participant: Thomas Genet
- Contact: Thomas Genet
- URL: http://people.irisa.fr/Thomas.Genet/timbuk/index.html

3.7. jsexplain

JSExplain

KEYWORDS: JavaScript - Compilation - Standards - Debug - Interpreter

FUNCTIONAL DESCRIPTION: JSExplain is a reference interpreter for JavaScript that closely follows the specification and that produces execution traces. These traces may be interactively investigated in a browser, with an interface that displays not only the code and the state of the interpreter, but also the code and the state of the interpreted program. Conditional breakpoints may be expressed with respect to both the interpreter and the interpreted program. In that respect, JSExplain is a double-debugger for the specification of JavaScript.

- Partner: Imperial College London
- Contact: Alan Schmitt
- Publication: JSExplain: A Double Debugger for JavaScript
- URL: https://gitlab.inria.fr/star-explain/jsexplain

CONVECS Project-Team

6. New Software and Platforms

6.1. CADP

Construction and Analysis of Distributed Processes

KEYWORDS: Formal methods - Verification

FUNCTIONAL DESCRIPTION: CADP (*Construction and Analysis of Distributed Processes* – formerly known as *CAESAR/ALDEBARAN Development Package*) [5] is a toolbox for protocols and distributed systems engineering.

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,
  - 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 [43], CTL [32], ACTL [33], 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 [45] 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 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.

- Participants: Hubert Garavel, Frédéric Lang, Radu Mateescu and Wendelin Serwe
- Contact: Hubert Garavel
- URL: http://cadp.inria.fr/

6.2. TRAIAN

KEYWORDS: Compilation - LOTOS NT

FUNCTIONAL DESCRIPTION: TRAIAN is a compiler 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.

- Participants: Hubert Garavel, Frédéric Lang and Wendelin Serwe
- Contact: Hubert Garavel
- URL: http://convecs.inria.fr/software/traian/

DEDUCTEAM Project-Team

5. New Software and Platforms

5.1. Autotheo

KEYWORD: Automated deduction

SCIENTIFIC DESCRIPTION: Transformation of axiomatic theories into rewriting systems that can be used by iProverModulo.

FUNCTIONAL DESCRIPTION: Autotheo is a tool that transforms axiomatic theories into polarized rewriting systems, thus making them usable in iProverModulo. It supports several strategies to orient the axioms, some of them being proved to be complete, in the sense that ordered polarized resolution modulo the resulting systems is refutationally complete, some others being merely heuristics. In practice, Autotheo takes a TPTP input file and produces an input file for iProverModulo.

NEWS OF THE YEAR: Maintenance.

- Participant: Guillaume Burel
- Partner: ENSIIE
- Contact: Guillaume Burel
- Publication: Consistency Implies Cut Admissibility
- URL: http://www.ensiie.fr/~guillaume.burel/blackandwhite_autotheo.html.en

5.2. CoLoR

Coq Library on Rewriting and termination

KEYWORDS: Coq - Formalisation

FUNCTIONAL DESCRIPTION: CoLoR is a Coq library on rewriting theory and termination. It provides many definitions and theorems on various mathematical structures (quasi-ordered sets, relations, ordered semi-rings, etc.), data structures (lists, vectors, matrices, polynomials, finite graphs), term structures (strings, first-order terms, lambda-terms, etc.), transformation techniques (dependency pairs, semantic labeling, etc.) and (non-)termination criteria (polynomial and matrix interpretations, recursive path ordering, computability closure, etc.).

- Authors: Frédéric Blanqui and Sébastien Hinderer
- Contact: Frédéric Blanqui
- Publications: CoLoR: a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates - Automated Verification of Termination Certificates - CoLoR: a Coq library on rewriting and termination
- URL: http://color.inria.fr/

5.3. Coqine

Coq In dEdukti

KEYWORDS: Higher-order logic - Formal methods - Proof

FUNCTIONAL DESCRIPTION: CoqInE is a plugin for the Coq software translating Coq proofs into Dedukti terms. It provides a Dedukti signature file faithfully encoding the underlying theory of Coq (or a sufficiently large subset of it). Current development is mostly focused on implementing support for Coq universe polymorphism. The generated ouput is meant to be type-checkable using the latest version of Dedukti.

- Contact: Guillaume Burel
- URL: http://www.ensiie.fr/~guillaume.burel/blackandwhite_coqInE.html.en

5.4. Dedukti

KEYWORD: Logical Framework

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

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

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

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

NEWS OF THE YEAR: There has been a new release 2.6 in 2018. This release provides a better control on module loading, and a better log of rewrite steps.

- Participants: François Thiré, Gaspard Ferey, Guillaume Genestier and Rodolphe Lepigre
- Contact: François Thiré
- Publications: Dedukti:un vérificateur de preuves universel - Rewriting Modulo β in the $\lambda\,\Pi$-Calculus Modulo - Expressing theories in the $\lambda\Pi$-calculus modulo theory and in the Dedukti system
- URL: https://deducteam.github.io/

5.5. Holide

KEYWORD: Proof

FUNCTIONAL DESCRIPTION: Holide translates HOL proofs to Dedukti[OT] proofs, using the OpenTheory standard (common to HOL Light and HOL4). Dedukti[OT] being the encoding of OpenTheory in Dedukti.

- Contact: Guillaume Burel
- URL: http://deducteam.gforge.inria.fr/holide/

5.6. HOT

Higher-Order Termination

FUNCTIONAL DESCRIPTION: HOT is an automated termination prover for higher-order rewriting, based on the notion of computability closure.

- Contact: Frédéric Blanqui
- URL: http://rewriting.gforge.inria.fr/hot.html

5.7. iProver Modulo

KEYWORDS: Automated deduction - Automated theorem proving

SCIENTIFIC DESCRIPTION: Integration of ordered polarized resolution modulo theory into the prover iProver.

FUNCTIONAL DESCRIPTION: iProver Modulo is an extension of the automated theorem prover iProver originally developed by Konstantin Korovin at the University of Manchester. It implements ordered polarized resolution modulo theory, a refinement of the resolution method based on deduction modulo theory. It takes as input a proposition in predicate logic and a clausal rewriting system defining the theory in which the formula has to be proved. Normalization with respect to the term rewriting rules is performed very efficiently through translation into OCaml code, compilation and dynamic linking. Experiments have shown that ordered polarized resolution modulo dramatically improves proof search compared to using raw axioms.

NEWS OF THE YEAR: Maintenance of Dedukti output

- Participant: Guillaume Burel
- Partner: ENSIIE
- Contact: Guillaume Burel
- Publications: A Shallow Embedding of Resolution and Superposition Proofs into the ??-Calculus Modulo - Experimenting with deduction modulo
- URL: https://github.com/gburel/iProverModulo

5.8. mSAT

KEYWORD: Propositional logic

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

- Contact: Guillaume Bury
- Publication: mSAT:An OCaml SAT Solver
- URL: https://github.com/Gbury/mSAT

5.9. Rainbow

Termination certificate verifier

KEYWORDS: Demonstration - Code generation - Verification

FUNCTIONAL DESCRIPTION: Rainbow is a set of tools for automatically verifying the correctness of termination certificates expressed in the CPF format used in the annual international competition of termination tools. It contains: a tool xsd2coq for generating Coq data types for representing XML files valid with respect to some XML Schema, a tool xsd2ml for generating OCaml data types and functions for parsing XML files valid with respect to some XML Schema, a tool for translating a CPF file into a Coq script, and a standalone Coq certified tool for verifying the correctness of a CPF file.

- Author: Frédéric Blanqui
- Contact: Frédéric Blanqui
- Publications: Automated verification of termination certificates - Automated verification of termination certificates
- URL: http://color.inria.fr/rainbow.html

5.10. Krajono

KEYWORD: Proof

FUNCTIONAL DESCRIPTION: Krajono translates Matita proofs into Dedukti[CiC] (encoding of CiC in Dedukti) terms.

- Contact: François Thiré

5.11. archsat

KEYWORDS: Automated theorem proving - First-order logic - Propositional logic

FUNCTIONAL DESCRIPTION: Archsat is an automated theorem prover aimed at studying the integration of first-order theorem prover technologies, such as rewriting, into SMT solvers.

- Contact: Guillaume Bury
- URL: https://gforge.inria.fr/projects/archsat

5.12. lrat2dk

KEYWORDS: Automated theorem proving - Proof

FUNCTIONAL DESCRIPTION: Take as input a SAT proof trace in LRAT format, which can be obtained from the de facto standard format DRAT using drat-trim. Output a proof checkable by Dedukti, in a shallow encoding of propositional logic.

- Participant: Guillaume Burel
- Partner: ENSIIE
- Contact: Guillaume Burel
- URL: https://github.com/gburel/lrat2dk

5.13. ekstrakto

KEYWORDS: TPTP - TSTP - Proof assistant - Dedukti

FUNCTIONAL DESCRIPTION: Extracting TPTP problems from a TSTP trace. Proof reconstruction in Dedukti from TSTP trace.

- Contact: Mohamed Yacine El Haddad
- URL: https://github.com/elhaddadyacine/ekstrakto

5.14. SizeChangeTool

KEYWORDS: Rewriting systems - Proof assistant - Termination

FUNCTIONAL DESCRIPTION: A termination-checker for higher-order rewriting with dependent types. Took part in the Termination Competition 2018 (http://termination-portal.org/wiki/Termination_Competition_2018) in the "Higher-Order Rewriting (union Beta)" category.

- Partner: Mines ParisTech
- Contact: Guillaume Genestier
- URL: https://github.com/Deducteam/SizeChangeTool

5.15. Logipedia

KEYWORDS: Formal methods - Web Services - Logical Framework

FUNCTIONAL DESCRIPTION: Logipedia is composed of two distinct parts: 1) A back-end that translates proofs expressed in a theory encoded in Dedukti to other systems such as Coq, Lean or HOL 2) A front-end that prints these proofs in a "nice way" via a website. Using the website, the user can search for a definition or a theorem then, download the whole proof into the wanted system.

Currently, the available systems are: Coq, Matita, Lean, PVS and OpenTheory. The proofs comes from a logic called STTForall.

In the long run, more systems and more logic should be added.

RELEASE FUNCTIONAL DESCRIPTION: This is the beta version of Logipedia. It implements the functionalities mentioned above.

- Contact: François Thiré
- URL: http://www.logipedia.science

GALLINETTE Project-Team

5. New Software and Platforms

5.1. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

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, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes, - a critical bug fix related to template polymorphism, - native 63-bit machine integers, - a new sort of definitionally proof-irrelevant propositions: SProp, - private universes for opaque polymorphic constants, - string notations and numeral notations, - a new simplex-based proof engine for the tactics lia, nia, lra and nra, - new introduction patterns for SSReflect, - a tactic to rewrite under binders: under, - easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

5.2. Math-Components

Mathematical Components library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: The Mathematical Components library is a set of Coq libraries that cover the prerequiste for the mechanization of the proof of the Odd Order Theorem.

RELEASE FUNCTIONAL DESCRIPTION: The library includes 16 more theory files, covering in particular field and Galois theory, advanced character theory, and a construction of algebraic numbers.

- Participants: Alexey Solovyev, Andrea Asperti, Assia Mahboubi, Cyril Cohen, Enrico Tassi, François Garillot, Georges Gonthier, Ioana Pasca, Jeremy Avigad, Laurence Rideau, Laurent Théry, Russell O'Connor, Sidi Ould Biha, Stéphane Le Roux and Yves Bertot
- Contact: Assia Mahboubi
- URL: http://math-comp.github.io/math-comp/

5.3. Ssreflect

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

- Participants: Assia Mahboubi, Cyril Cohen, Enrico Tassi, Georges Gonthier, Laurence Rideau, Laurent Théry and Yves Bertot
- Contact: Yves Bertot
- URL: http://math-comp.github.io/math-comp/

5.4. Ltac2

KEYWORDS: Coq - Proof assistant

FUNCTIONAL DESCRIPTION: A replacement for Ltac, the tactic language of Coq.

- Contact: Pierre-Marie Pédrot

MEXICO Project-Team

6. New Software and Platforms

6.1. COSMOS

KEYWORD: Model Checker

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.

It is easy to generate and use a C code for discrete Simulink models (using only discrete blocks, which are sampled at fixed intervals) using MathWorks tools. However, it limits the expressivity of the models. In order to use more diverse Simulink models and control the flow of a multi-model simulation (with Discrete Event Stochastic Processes) we developed a Simulink Simulation Engine embedded into Cosmos.

COSMOS is written in C++

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

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

6.3. 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/

MOCQUA Team

6. New Software and Platforms

6.1. FiatLux

KEYWORDS: Cellular automaton - Multi-agent - Distributed systems

SCIENTIFIC DESCRIPTION: FiatLux is a discrete dynamical systems simulator that allows the user to experiment with various models and to perturb them. It includes 1D and 2D cellular automata, moving agents, interacting particle systems, etc. Its main feature is to allow users to change the type of updating, for example from a deterministic parallel updating to an asynchronous random updating. FiatLux has a Graphical User Interface and can also be launched in a batch mode for the experiments that require statistics.

FUNCTIONAL DESCRIPTION: FiatLux is a cellular automata simulator in Java specially designed for the study of the robustness of the models. Its main distinctive features is to allow to perturb the updating of the system (synchrony rate) and to perturb the topology of the grid.

- Participants: Nazim Fatès and Olivier Boure
- Partners: ENS Lyon - Université de Lorraine
- Contact: Nazim Fatès
- URL: http://fiatlux.loria.fr/

6.2. ComplexityParser

KEYWORDS: Complexity - Static typing - Parsing

FUNCTIONAL DESCRIPTION: ComplexityParser is a static complexity analyzer of Java programs written in Java (approximatively 5000 lines of code). The program consists in a type inference and checking program based on the data tiering principle. It allows the program to certify that the typed program has a polynomial time complexity.

- Participants: Olivier Zeyen, Emmanuel Hainry, Romain Péchoux and Emmanuel Jeandel
- Contact: Emmanuel Hainry

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, Gopalan Nadathur, Kaustuv Chaudhuri, Mary Southern, Matteo Cimini, Olivier Savary-Bélanger and Yuting Wang
- 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

KEYWORD: Model Checker

FUNCTIONAL DESCRIPTION: Bedwyr is a generalization of logic programming that allows model checking directly on syntactic expressions 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: Dale Miller, Quentin Heath and Roberto Blanco Martinez
- Contact: Dale Miller
- 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: Giselle Machado Nogueira Reis, Marco Volpe and Tomer Libal
- Contact: Tomer Libal
- URL: https://github.com/proofcert/checkers

6.4. Psyche

Proof-Search factorY for Collaborative HEuristics

KEYWORD: Automated theorem proving

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.

RELEASE FUNCTIONAL DESCRIPTION: It is now equipped with the machinery to handle quantifiers and quantifier-handling techniques. Concretely, it uses meta-variables to delay the instantiation of existential variables, and constraints on meta-variables are propagated through the various branches of the search-space, in a way that allows local backtracking. The kernel, of about 800 l.o.c., is purely functional.

- Participants: Assia Mahboubi, Jean-Marc Notin and Stéphane Graham-Lengrand
- Contact: Stéphane Graham-Lengrand
- URL: http://www.csl.sri.com/users/sgl/

6.5. Maetning

FUNCTIONAL DESCRIPTION: Mætning is an automated theorem prover for intuitionistic predicate logic that is designed to disprove non-theorems.

- Contact: Kaustuv Chaudhuri
- URL: https://github.com/chaudhuri/maetning/

6.6. OCaml

KEYWORDS: Functional programming - Static typing - Compilation

FUNCTIONAL DESCRIPTION: 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 System Z), a debugger, a documentation generator, a compilation manager, a package manager, and many libraries contributed by the user community.

- Participants: Damien Doligez, Xavier Leroy, Fabrice Le Fessant, Luc Maranget, Gabriel Scherer, Alain Frisch, Jacques Garrigue, Marc Shinwell, Jeremy Yallop and Leo White
- Contact: Damien Doligez
- URL: https://ocaml.org/

PI.R2 Project-Team

5. New Software and Platforms

5.1. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

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, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes, - a critical bug fix related to template polymorphism, - native 63-bit machine integers, - a new sort of definitionally proof-irrelevant propositions: SProp, - private universes for opaque polymorphic constants, - string notations and numeral notations, - a new simplex-based proof engine for the tactics lia, nia, lra and nra, - new introduction patterns for SSReflect, - a tactic to rewrite under binders: under, - easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

5.2. Equations

KEYWORDS: Coq - Dependent Pattern-Matching - Proof assistant - Functional programming

SCIENTIFIC DESCRIPTION: Equations is a tool designed to help with the definition of programs in the setting of dependent type theory, as implemented in the Coq proof assistant. Equations provides a syntax for defining programs by dependent pattern-matching and well-founded recursion and compiles them down to the core type theory of Coq, using the primitive eliminators for inductive types, accessibility and equality. In addition to the definitions of programs, it also automatically derives useful reasoning principles in the form of propositional equations describing the functions, and an elimination principle for calls to this function. It realizes this using a purely definitional translation of high-level definitions to core terms, without changing the core calculus in any way, or using axioms.

FUNCTIONAL DESCRIPTION: Equations is a function definition plugin for Coq (supporting Coq 8.8 to 8.10, with special support for the Coq-HoTT library), that allows the definition of functions by dependent pattern-matching and well-founded, mutual or nested structural recursion and compiles them into core terms. It automatically derives the clauses equations, the graph of the function and its associated elimination principle.

Equations is based on a simplification engine for the dependent equalities appearing in dependent eliminations that is also usable as a separate tactic, providing an axiom-free variant of dependent destruction. The main features of Equations include:

Dependent pattern-matching in the style of Agda/Epigram, with inaccessible patterns, with and where clauses. The use of the K axiom or a proof of K is configurable, and it is able to solve unification problems without resorting to the K rule if not necessary.

Support for well-founded and mutual recursion using measure/well-foundedness annotations, even on indexed inductive types, using an automatic derivation of the subterm relation for inductive families.

Support for mutual and nested structural recursion using with and where auxilliary definitions, allowing to factor multiple uses of the same nested fixpoint definition. It proves the expected elimination principles for mutual and nested definitions.

Automatic generation of the defining equations as rewrite rules for every definition.

Automatic generation of the unfolding lemma for well-founded definitions (requiring only functional extensionality).

Automatic derivation of the graph of the function and its elimination principle. In case the automation fails to prove these principles, the user is asked to provide a proof.

A new dependent elimination tactic based on the same splitting tree compilation scheme that can advantageously replace dependent destruction and sometimes inversion as well. The as clause of dependent elimination allows to specify exactly the patterns and naming of new variables needed for an elimination.

A set of Derive commands for automatic derivation of constructions from an inductive type: its signature, no-confusion property, well-founded subterm relation and decidable equality proof, if applicable.

RELEASE FUNCTIONAL DESCRIPTION: This version of Equations is based on an improved simplification engine for the dependent equalities appearing during dependent eliminations that is also usable as a separate dependent elimination tactic, providing an axiom-free variant of dependent destruction and a more powerful form of inversion. See http://mattam82.github.io/Coq-Equations/equations/2019/01/28/1.2beta.html and the following release notes for more information.

NEWS OF THE YEAR: Equations 1.2 was first released in may this year, after 3 years of development. It provides a refined simplification engine based on the work published at ICFP'19 (see the "Equations Reloaded" paper for details). The system has been improved to also work in the setting of Homotopy Type Theory and provides a more expressive source language and robust dependent elimination tactics.

- Participants: Matthieu Sozeau and Cyprien Mangin
- Contact: Matthieu Sozeau

- Publications: Equations reloaded - Equations for Hereditary Substitution in Leivant's Predicative System F: A Case Study - Equations: A Dependent Pattern-Matching Compiler
- URL: http://mattam82.github.io/Coq-Equations/

5.3. Rewr

Rewriting methods in algebra

KEYWORDS: Computer algebra system (CAS) - Rewriting systems - Algebra

FUNCTIONAL DESCRIPTION: Rewr is a prototype of computer algebra system, using rewriting methods to compute resolutions and homotopical invariants of monoids. The library implements various classical constructions of rewriting theory (such as completion), improved by experimental features coming from Garside theory, and allows homotopical algebra computations based on Squier theory. Specific functionalities have been developed for usual classes of monoids, such as Artin monoids and plactic monoids.

NEWS OF THE YEAR: Rewr has been extended with the experimental KGB completion algorithm, based on Knuth-Bendix completion procedure improved by techniques coming from Garside theory.

- Participants: Yves Guiraud and Samuel Mimram
- Contact: Yves Guiraud
- Publications: Higher-dimensional categories with finite derivation type - Higher-dimensional normalisation strategies for acyclicity - Coherent presentations of Artin monoids - A Homotopical Completion Procedure with Applications to Coherence of Monoids - Polygraphs of finite derivation type - Quadratic normalisation in monoids
- URL: http://www.lix.polytechnique.fr/Labo/Samuel.Mimram/rewr

5.4. Catex

KEYWORDS: LaTeX - String diagram - Algebra

FUNCTIONAL DESCRIPTION: Catex is a Latex package and an external tool to typeset string diagrams easily from their algebraic expression. Catex works similarly to Bibtex.

NEWS OF THE YEAR: It is now possible to add labels to objects and morphisms

- Participant: Yves Guiraud
- Contact: Yves Guiraud
- URL: https://www.irif.fr/~guiraud/catex/catex.zip

5.5. Cox

KEYWORDS: Computer algebra system (CAS) - Rewriting systems - Algebra

FUNCTIONAL DESCRIPTION: Cox is a Python library for the computation of coherent presentations of Artin monoids, with experimental features to compute the lower dimensions of the Salvetti complex.

- Participant: Yves Guiraud
- Contact: Yves Guiraud
- Publications: Coherent presentations of Artin monoids - A Homotopical Completion Procedure with Applications to Coherence of Monoids
- URL: https://www.irif.fr/~guiraud/cox/cox.zip

5.6. jsCoq

KEYWORDS: Coq - Program verification - Interactive - Formal concept analysis - Proof assistant - Ocaml - Education - JavaScript

FUNCTIONAL DESCRIPTION: jsCoq is an Online Integrated Development Environment for the Coq proof assistant and runs in your browser! It aims to enable new UI/interaction possibilities and to improve the accessibility of the Coq platform itself.

RELEASE FUNCTIONAL DESCRIPTION: - Coq 8.10 support - Much improved interaction and general experience - Open / Save dialogs - AST and full serialization of Coq's datatypes - NPM packaging - Timeout support

- Participant: Emilio Jesus Gallego Arias
- Partners: Mines ParisTech - Technion, Israel Institute of Technology
- Contact: Emilio Jesus Gallego Arias
- Publication: jsCoq: Towards Hybrid Theorem Proving Interfaces
- URL: https://github.com/ejgallego/jscoq

5.7. coq-serapi

KEYWORDS: Interaction - Coq - Ocaml - Data centric - User Interfaces - GUI (Graphical User Interface) - Toolkit

FUNCTIONAL DESCRIPTION: SerAPI is a library for machine-to-machine interaction with the Coq proof assistant, with particular emphasis on applications in IDEs, code analysis tools, and machine learning. SerAPI provides automatic serialization of Coq's internal OCaml datatypes from/to JSON or S-expressions (sexps).

RELEASE FUNCTIONAL DESCRIPTION: - Support Coq 8.10 - Serialization of extensive AST - Serialization of kernel structures - Support for kernel traces [dumping and replay] - Tokenization of Coq documents - Serialization to JSON - Improved protocol and printing - Bug fixes

- Participant: Karl Palmskog
- Partner: KTH Royal Institute of Technology
- Contact: Emilio Jesus Gallego Arias
- Publication: SerAPI: Machine-Friendly, Data-Centric Serialization for COQ : Technical Report
- URL: https://github.com/ejgallego/coq-serapi

STAMP Project-Team

5. New Software and Platforms

5.1. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

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, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes,
- a critical bug fix related to template polymorphism,
- native 63-bit machine integers,
- a new sort of definitionally proof-irrelevant propositions: SProp,
- private universes for opaque polymorphic constants,
- string notations and numeral notations,
- a new simplex-based proof engine for the tactics lia, nia, lra and nra,
- new introduction patterns for SSReflect,
- a tactic to rewrite under binders: under,
- easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

5.2. Math-Components

Mathematical Components library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: The Mathematical Components library is a set of Coq libraries that cover the prerequiste for the mechanization of the proof of the Odd Order Theorem.

RELEASE FUNCTIONAL DESCRIPTION: This releases is compatible with Coq 8.9 and Coq 8.10 it adds many theorems for finite function, prime numbers, sequences, finite types, bigo operations, natural numbers, cycles in graphs.

- Participants: Alexey Solovyev, Andrea Asperti, Assia Mahboubi, Cyril Cohen, Enrico Tassi, François Garillot, Georges Gonthier, Ioana Pasca, Jeremy Avigad, Laurence Rideau, Laurent Théry, Russell O'Connor, Sidi Ould Biha, Stéphane Le Roux and Yves Bertot
- Contact: Assia Mahboubi
- URL: http://math-comp.github.io/math-comp/

5.3. Semantics

KEYWORDS: Semantic - Programming language - Coq

FUNCTIONAL DESCRIPTION: A didactical Coq development to introduce various semantics styles. Shows how to derive an interpreter, a verifier, or a program analyser from formal descriptions, and how to prove their consistency.

This is a library for the Coq system, where the description of a toy programming language is presented. The value of this library is that it can be re-used in classrooms to teach programming language semantics or the Coq system. The topics covered include introductory notions to domain theory, pre and post-conditions, abstract interpretation, and the proofs of consistency between all these point of views on the same programming language. Standalone tools for the object programming language can be derived from this development.

- Participants: Christine Paulin and Yves Bertot
- Contact: Yves Bertot
- URL: http://www-sop.inria.fr/members/Yves.Bertot/proofs/semantics_survey.tgz

5.4. Easycrypt

KEYWORDS: Proof assistant - Cryptography

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

5.5. ELPI

Embeddable Lambda Prolog Interpreter

KEYWORDS: Constraint Programming - Programming language - Higher-order logic

SCIENTIFIC DESCRIPTION: The programming language has the following features

- Native support for variable binding and substitution, via an Higher Order Abstract Syntax (HOAS) embedding of the object language. The programmer needs not to care about De Bruijn indexes.

- Native support for hypothetical context. When moving under a binder one can attach to the bound variable extra information that is collected when the variable gets out of scope. For example when writing a type-checker the programmer needs not to care about managing the typing context.

- Native support for higher order unification variables, again via HOAS. Unification variables of the meta-language (lambdaProlog) can be reused to represent the unification variables of the object language. The programmer does not need to care about the unification-variable assignment map and cannot assign to a unification variable a term containing variables out of scope, or build a circular assignment.

- Native support for syntactic constraints and their meta-level handling rules. The generative semantics of Prolog can be disabled by turning a goal into a syntactic constraint (suspended goal). A syntactic constraint is resumed as soon as relevant variables gets assigned. Syntactic constraints can be manipulated by constraint handling rules (CHR).

- Native support for backtracking. To ease implementation of search.

- The constraint store is extensible. The host application can declare non-syntactic constraints and use custom constraint solvers to check their consistency.

- Clauses are graftable. The user is free to extend an existing program by inserting/removing clauses, both at runtime (using implication) and at "compilation" time by accumulating files.

Most of these feature come with lambdaProlog. Constraints and propagation rules are novel in ELPI.

FUNCTIONAL DESCRIPTION: ELPI implements a variant of lambdaProlog enriched with Constraint Handling Rules, a programming language well suited to manipulate syntax trees with binders and unification variables.

ELPI is a research project aimed at providing a programming platform for the so called elaborator component of an interactive theorem prover.

ELPI is designed to be embedded into larger applications written in OCaml as an extension language. It comes with an API to drive the interpreter and with an FFI for defining built-in predicates and data types, as well as quotations and similar goodies that come in handy to adapt the language to the host application.

RELEASE FUNCTIONAL DESCRIPTION: improvement to the parser (parsing negative numbers) improvement to the foreign function interface (accepting ternary comparison, instead of equality) adds ternary comparisons to the standard library provides a builtin comparison cmp_term provides a builtin to check whether a term is ground

NEWS OF THE YEAR: There were 7 releases in 2019. Work done mostly in these areas:

- consolidation (documentation, bug fixes, test suits)

- API and FFI (making it easier to export host applications to ELPI)

- standard library

- Participant: Claudio Sacerdoti Coen
- Contact: Enrico Tassi
- Publications: ELPI: fast, Embeddable, λProlog Interpreter - Implementing Type Theory in Higher Order Constraint Logic Programming - Deriving proved equality tests in Coq-elpi: Stronger induction principles for containers in Coq
- URL: https://github.com/lpcic/elpi/

5.6. Coq-elpi

KEYWORDS: Metaprogramming - Extension

SCIENTIFIC DESCRIPTION: Coq-elpi provides a Coq plugin that embeds ELPI. It also provides a way to embed Coq's terms into lambdaProlog using the Higher-Order Abstract Syntax approach (HOAS) and a way to read terms back. In addition to that it exports to ELPI a set of Coq's primitives, e.g. printing a message, accessing the environment of theorems and data types, defining a new constant and so on. For convenience it also provides a quotation and anti-quotation for Coq's syntax in lambdaProlog. E.g. {{nat}} is expanded to the type name of natural numbers, or {{A -> B}} to the representation of a product by unfolding the -> notation. Finally it provides a way to define new vernacular commands and new tactics.

FUNCTIONAL DESCRIPTION: Coq plugin embedding ELPI

RELEASE FUNCTIONAL DESCRIPTION: Minor relase for extra API for global reference data types

NEWS OF THE YEAR: Releases 1.0, 1.1, and 1.2 were made in 2019, they constitute the first public release with tutorials and examples.

Work done in 2019 is mostly in these areas:

- expose a complete set of API to script Coq's vernacular language

- take advantage or recent ELPI API and FFI to convert back and forth terms containing existential variables (Evars)

- Contact: Enrico Tassi
- Publications: Deriving proved equality tests in Coq-elpi: Stronger induction principles for containers in Coq - Elpi: an extension language for Coq (Metaprogramming Coq in the Elpi λProlog dialect)

5.7. AutoGnP

KEYWORDS: Formal methods - Security - Cryptography

FUNCTIONAL DESCRIPTION: autoGnP is an automated tool for analyzing the security of padding-based public-key encryption schemes (i.e. schemes built from trapdoor permutations and hash functions). This years we extended the tool to be able to deal with schemes based on cyclic groups and bilinear maps.

- Participants: Benjamin Grégoire, Gilles Barthe and Pierre-Yves Strub
- Contact: Gilles Barthe
- URL: https://github.com/ZooCrypt/AutoGnP

5.8. MaskComp

KEYWORD: Masking

FUNCTIONAL DESCRIPTION: MaskComp is a compiler generating masked implémentation protected against side channel attack based on differential power analysis. It take a unmasked program in a syntaxe close to C and generate a new C protected program. We did not claim that the generate C program will be secure after compilation (C compiler can break protection), but it provide a good support for generating masked implementation.

- Contact: Benjamin Grégoire
- URL: https://sites.google.com/site/maskingcompiler/home

5.9. Jasmin

Jasmin compiler and analyser

KEYWORDS: Cryptography - Static analysis - Compilers

FUNCTIONAL DESCRIPTION: Analysing the execution time of a cryptographic code can be a way to discover the secret protected by this code. To avoid this pitfall, Jasmin proposes a high-level language and an analyzer for this language that makes it possible to predict when the execution of this code will happen in constant time and thus does not unveil the secret (for instance, the cryptographic key). Once the Jasmin code is valid with respect to the analyzer, the compiler produces assembly code that still preserves this property of constant time.

- Contact: Benjamin Grégoire

5.10. MaskVerif

KEYWORDS: Masking - Hardware and Software Platform

FUNCTIONAL DESCRIPTION: MaskVerif is a tool to verify the security of implementations protected against side channel attacks, in particular differential power analysis. It allows to check different security notions in the probing model: - Probing security - Non Interference - Strong Non Interference. The tool is able to analyse software implementations and hardware implementations (written in Verilog). It can prove the different security notions in presence of glitch or transition.

- Contact: Benjamin Grégoire
- URL: https://sites.google.com/view/maskverif/home

5.11. CoqEAL

The Coq Effective Algebra Library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: This library contains formal developments in algebra and optimized algorithms on mathcomp data structures and a framework to ease change of data representation during a proof.

RELEASE FUNCTIONAL DESCRIPTION: First release

- Contact: Cyril Cohen

5.12. math-comp-analysis

Mathematical Components Analysis

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: This library adds definitions and theorems for real numbers and their mathematical structures

RELEASE FUNCTIONAL DESCRIPTION: Compatible with mathcomp 1.8.0, 1.9.0, and 1.10.0

NEWS OF THE YEAR: In 2019, there were 3 releases.

- Partners: Ecole Polytechnique - AIST Tsukuba
- Contact: Cyril Cohen
- Publication: Formalization Techniques for Asymptotic Reasoning in Classical Analysis
- URL: https://github.com/math-comp/analysis

5.13. math-comp-finmap

Finite maps and ordered types library

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: Support for reasoning about finite maps and ordered types

RELEASE FUNCTIONAL DESCRIPTION: This release is solely an update of order.v and set.v in order to integrate the changes in math-comp/math-comp#270

- Contact: Cyril Cohen

5.14. math-comp-real-closed

Real Closed Fields

KEYWORD: Proof assistant

FUNCTIONAL DESCRIPTION: Theorems for real closed fields

RELEASE FUNCTIONAL DESCRIPTION: First release

- Contact: Cyril Cohen
- URL: https://github.com/math-comp/real-closed

SUMO Project-Team

6. New Software and Platforms

6.1. Active Workspaces

KEYWORDS: Active workspace - Collaborative systems - 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.

- Authors: Éric Badouel and Robert Nsaibirni
- Contact: Éric Badouel
- URL: http://people.rennes.inria.fr/Eric.Badouel/Research/ActiveWorkspaces.html

6.2. SIMSTORS

Simulator for stochastic regulated systems

KEYWORDS: Simulation - Public transport - Stochastic models - Distributed systems

FUNCTIONAL DESCRIPTION: SIMSTORS is a software for the simulation of stochastic concurrent timed systems. The heart of the software is a variant of stochastic and timed Petri nets, whose execution is controlled by a regulation policy (a controller), or a predetermined theoretical schedule. The role of the regulation policy is to control the system to realize objectives or a schedule when it exists with the best possible precision. SIMSTORS is well adapted to represent systems with randomness, parallelism, tasks scheduling, and resources. From 2015 to 2018, it was used for the P22 collaboration with Asltom Transport, to model metro traffic and evaluate performance of regulation solutions. It is now (2019) at the heart of a collaboration on multi-modal networks with Alstom transport Madrid. This software allows for step by step simulation, but also for efficient performance analysis of systems such as production cells or train systems. The initial implementation was released in 2015, and the software is protected by the APP.

Since then, SIMSTORS has been extended along two main axes: on one hand, SIMSTORS models were extended to handle situations where shared resources can be occupied by more than one object (this is of paramount importance to represent conveyors, roads occupied by cars, or train tracks with smoothed scheduling allowing shared sections among trains) with priorities, constraint on their ordering and individual characteristics. This allows for instance to model vehicles with different speeds on a road, while handling safety distance constraints. On the other hand, SIMSTORS models were extended to allow control of stochastic nets based on decision rules that follow optimization schemes. In 2019, it was extended to include planning-based regulation techniques during a collaboration with Roma 3 University.

RELEASE FUNCTIONAL DESCRIPTION: modeling of continuous vehicles movements

- Participants: Abd El Karim Kecir and Loïc Hélouët
- 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.

RELEASE FUNCTIONAL DESCRIPTION: the "SAT solving" part can now be delegated to an external plugin, new experimental SAT solver based on mini-SAT, provided as a plugin. This solver is, in general, more efficient on ground problems, heuristics simplification in the default SAT solver and in the matching (instantiation) module, re-implementation of internal literals representation, improvement of theories combination architecture, rewriting some parts of the formulas module, bugfixes in records and numbers modules, new option "-no-Ematching" to perform matching without equality reasoning (i.e. without considering "equivalence classes"). This option is very useful for benchmarks coming from Atelier-B, two new experimental options: "-save-used-context" and "-replay-used-context". When the goal is proved valid, the first option allows to save the names of useful axioms into a ".used" file. The second one is used to replay the proof using only the axioms listed in the corresponding ".used" file. Note that the replay may fail because of the absence of necessary ground terms generated by useless axioms (that are not included in .used file) during the initial run.

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

6.2. CoqInterval

Interval package for Coq

KEYWORDS: Interval arithmetic - Coq

FUNCTIONAL DESCRIPTION: CoqInterval is a library for the proof assistant Coq.

It provides several tactics for proving 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 in Coq. In 2014, this library has been included in CoqInterval.

- Participants: Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Jean-Michel Muller, Laurence Rideau, Laurent Théry, Micaela Mayero, Mioara Joldes, Nicolas Brisebarre and Thomas Sibut-Pinote
- Contact: Guillaume Melquiond
- Publications: Proving bounds on real-valued functions with computations - Floating-point arithmetic in the Coq system - Proving Tight Bounds on Univariate Expressions with Elementary Functions in Coq - Formally Verified Approximations of Definite Integrals - Formally Verified Approximations of Definite Integrals
- URL: http://coq-interval.gforge.inria.fr/

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

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

6.5. 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 verify numerical applications.

Flocq is currently used by the CompCert verified compiler to support floating-point computations.

- Participants: Guillaume Melquiond, Pierre Roux and Sylvie Boldo
- Contact: Sylvie Boldo
- Publications: Flocq: A Unified Library for Proving Floating-point Algorithms in Coq - A Formally-Verified C Compiler Supporting Floating-Point Arithmetic - Verified Compilation of Floating-Point Computations - Innocuous Double Rounding of Basic Arithmetic Operations - Formal Proofs of Rounding Error Bounds : With application to an automatic positive definiteness check - Computer Arithmetic and Formal Proofs : Verifying Floating-point Algorithms with the Coq System
- URL: http://flocq.gforge.inria.fr/

6.6. 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 formally verifying 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 verify 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.

- Participant: Guillaume Melquiond
- Contact: Guillaume Melquiond
- Publications: Generating formally certified bounds on values and round-off errors - Formal certification of arithmetic filters for geometric predicates - Assisted verification of elementary functions - From interval arithmetic to program verification - Formally Certified Floating-Point Filters For Homogeneous Geometric Predicates - Combining Coq and Gappa for Certifying Floating-Point Programs - Handbook of Floating-Point Arithmetic - Certifying the floating-point implementation of an elementary function using Gappa - Automations for verifying floating-point algorithms - Automating the verification of floating-point algorithms - Computer Arithmetic and Formal Proofs : Verifying Floating-point Algorithms with the Coq System
- URL: http://gappa.gforge.inria.fr/

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

6.8. Coq

The Coq Proof Assistant

KEYWORDS: Proof - Certification - Formalisation

SCIENTIFIC DESCRIPTION: Coq is an interactive proof assistant based on the Calculus of (Co-)Inductive Constructions, extended with universe polymorphism. This type theory features inductive and co-inductive families, an impredicative sort and a hierarchy of predicative universes, making it a very expressive logic. The calculus allows to formalize both general mathematics and computer programs, ranging from theories of finite structures to abstract algebra and categories to programming language metatheory and compiler verification. Coq is organised as a (relatively small) kernel including efficient conversion tests on which are built a set of higher-level layers: a powerful proof engine and unification algorithm, various tactics/decision procedures, a transactional document model and, at the very top an IDE.

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, ...

RELEASE FUNCTIONAL DESCRIPTION: Coq version 8.10 contains two major new features: support for a native fixed-precision integer type and a new sort SProp of strict propositions. It is also the result of refinements and stabilization of previous features, deprecations or removals of deprecated features, cleanups of the internals of the system and API, and many documentation improvements. This release includes many user-visible changes, including deprecations that are documented in the next subsection, and new features that are documented in the reference manual.

Version 8.10 is the fifth release of Coq developed on a time-based development cycle. Its development spanned 6 months from the release of Coq 8.9. Vincent Laporte is the release manager and maintainer of this release. This release is the result of 2500 commits and 650 PRs merged, closing 150+ issues.

See the Zenodo citation for more information on this release: https://zenodo.org/record/3476303#.Xe54f5NKjOQ

NEWS OF THE YEAR: Coq 8.10.0 contains:

- some quality-of-life bug fixes, - a critical bug fix related to template polymorphism, - native 63-bit machine integers, - a new sort of definitionally proof-irrelevant propositions: SProp, - private universes for opaque polymorphic constants, - string notations and numeral notations, - a new simplex-based proof engine for the tactics lia, nia, lra and nra, - new introduction patterns for SSReflect, - a tactic to rewrite under binders: under, - easy input of non-ASCII symbols in CoqIDE, which now uses GTK3.

All details can be found in the user manual.

- Participants: Yves Bertot, Frédéric Besson, Maxime Denes, Emilio Jesús Gallego Arias, Gaëtan Gilbert, Jason Gross, Hugo Herbelin, Assia Mahboubi, Érik Martin-Dorel, Guillaume Melquiond, Pierre-Marie Pédrot, Michael Soegtrop, Matthieu Sozeau, Enrico Tassi, Laurent Théry, Théo Zimmermann, Theo Winterhalter, Vincent Laporte, Arthur Charguéraud, Cyril Cohen, Christian Doczkal and Chantal Keller
- Partners: CNRS - Université Paris-Sud - ENS Lyon - Université Paris-Diderot
- Contact: Matthieu Sozeau
- URL: http://coq.inria.fr/

VERIDIS Project-Team

6. New Software and Platforms

6.1. Redlog

Reduce Logic System

KEYWORDS: Computer algebra system (CAS) - First-order logic - Constraint solving

SCIENTIFIC 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.

NEWS OF THE YEAR: Parts of the Redlog code are 25 years old now. Version 1 of the underlying computer algebra system Reduce has been published even 50 years ago. In 2018 we therefore started to go for major revisions and improvements of Redlog's software architecture, which are still under way.

Redlog, as well as the underlying Reduce, depends on a quite minimalistic Lisp 1 dialect called Standard Lisp. Today, there are two independent implementations of Standard Lisp left, which are supported only on the basis of private commitment of essentially one individual per Lisp. With the large code base of Redlog plus the necessary algebraic algorithms from Reduce, a migration to a different language or computer algebra system is not feasible. We are therefore experimenting with the realization of a Standard Lisp on the basis of ANSI Common Lisp.

Scientifically we are currently improving on Parametric Gaussian Elimination in Reduce/Redlog, which has various applications in our bilateral interdisciplinary ANR/DFG project SYMBIONT (Symbolic Methods for Biological Networks), e.g., classification of real singularities of systems of implicit ordinary differential equations.

- Participant: Thomas Sturm
- Contact: Thomas Sturm
- URL: http://www.redlog.eu/

6.2. SPASS

KEYWORD: First-order logic

SCIENTIFIC 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. Furthermore, we use SPASS 3.9 as a test bed for the development of new calculi.

SPASS 3.9 has been used as the basis for SPASS-AR, a new approximation refinement theorem proving approach.

FUNCTIONAL DESCRIPTION: SPASS is an automated theorem prover based on superposition that handles first-order logic with equality and several extensions for particular classes of theories.

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

6.3. SPASS-SATT

KEYWORDS: Automated deduction - Decision

SCIENTIFIC DESCRIPTION: SPASS -SATT is an SMT solver for the theories of linear integer arithmetic, linear rational arithmetic and mixed linear arithmetic. It features new tests for the satisfiability of unbounded systems, as well as new algorithms for the detection of integer solutions.

We further investigated the use of redundancy elimination in SAT solving and underlying implementation techniques. Our aim is a new approach to SAT solving that needs fewer conflicts (on average) *and* is faster than the current state-of-the art solvers. Furthermore, we have developed a new calculus and first prototypical implementation of a SAT solver with mixed OR/XOR clauses.

FUNCTIONAL DESCRIPTION: SPASS-SATT is an SMT solver for linear integer arithmetic, mixed linear arithmetic and rational linear arithmetic.

NEWS OF THE YEAR: SPASS-SATT participated in the SMT competition 2019 in the quantifier free integer and rational linear arithmetic categories. It scored first on rational linear arithmetic and second on integer linear arithmetic. (The winner of the latter category was a portfolio solver that includes SPASS-SATT.) The main improvements are due to an advanced translation to clause normal form, a close interaction between the theory and the SAT solvers, and a new transformation turning unbounded integer problems into bounded integer problems.

- Participants: Martin Bromberger, Mathias Fleury and Christoph Weidenbach
- Contact: Martin Bromberger
- URL: https://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/spass-satt/

6.4. veriT

KEYWORDS: Automated deduction - Formula solving - Verification

SCIENTIFIC DESCRIPTION: veriT comprises a SAT solver, a decision procedure for uninterpreted symbols based on congruence closure, 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.

NEWS OF THE YEAR: Efforts in 2019 have been focused on quantifier handling, higher logic, and proof production.

The veriT solver participated in the SMT competition SMT-COMP 2019 with good results. In particular, it took the bronze medal in the QF_UF division, solving as many problems as the two leading solvers but taking somewhat more time.

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, and it is integrated within *Atelier B*.

veriT is also a prototype platform for ideas developed within the Matryoshka project, aiming at greater availability of automated reasoning for proof assistants.

- Participants: Haniel Barbosa, Daniel El Ouraoui, Pascal Fontaine and Hans-JÖrg Schurr
- Partner: Université de Lorraine
- Contact: Pascal Fontaine
- URL: http://www.veriT-solver.org

6.5. SPIKE

KEYWORDS: Proof - Automated deduction - Automated theorem proving - Term Rewriting Systems - Formal methods

SCIENTIFIC DESCRIPTION: SPIKE, an automatic induction-based theorem prover built to reason on conditional theories with equality, is one of the few formal tools able to perform automatically mutual and lazy induction. Designed in the 1990s, it has been successfully used in many non-trivial applications and served as a prototype for different proof experiments and extensions.

FUNCTIONAL DESCRIPTION: Automated induction-based theorem prover

RELEASE FUNCTIONAL DESCRIPTION: Proof certification with Coq, cyclic induction, decision procedures

- Participant: Sorin Stratulat
- Contact: Sorin Stratulat
- URL: https://github.com/sorinica/spike-prover/wiki

6.6. TLAPS

TLA+ proof system

KEYWORD: Proof assistant

SCIENTIFIC DESCRIPTION: 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 proof steps that can be checked independently. 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+ set theory 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.

FUNCTIONAL DESCRIPTION: TLAPS is a proof assistant for the TLA+ specification language.

NEWS OF THE YEAR: Work in 2019 focused on providing support for reasoning about TLA+'s ENABLED and action composition constructs. We also prepared a minor release, fixing some issues and switching to Z3 as the default SMT back-end solver.

- Participants: Damien Doligez, Stephan Merz and Ioannis Filippidis
- Contact: Stephan Merz
- URL: https://tla.msr-inria.inria.fr/tlaps/content/Home.html

6.7. Apalache

Abstraction-based Parameterized TLA+ Checker

KEYWORD: Model Checker

SCIENTIFIC DESCRIPTION: Apalache is a symbolic model checker that works under the following assumptions:

(1) As in TLC, all specification parameters are fixed and finite, e.g., the system is initialized integers, finite sets, and functions of finite domains and co-domains. (2) As in TLC, all data structures evaluated during an execution are finite, e.g., a system specification cannot operate on the set of all integers. (3) Only finite executions up to a given bound are analysed.

Apalache translates bounded executions of a TLA+ specifications into a set of quantifier-free SMT constraints. By querying the SMT solver, the model checker either finds a counterexample to an invariant, or proves that there is no counterexample up to given computation length.

FUNCTIONAL DESCRIPTION: The first version implements a symbolic bounded model checker for TLA$^+$ that runs under the same assumptions as the explicit-state model checker TLC. It checks whether a TLA$^+$ specification satisfies an invariant candidate by checking satisfiability of an SMT formula that encodes: (1) an execution of bounded length, and (2) preservation of the invariant candidate in every state of the execution. Our tool is still in the experimental phase, due to a number of challenges posed by the semantics of TLA$^+$ to SMT solvers.

NEWS OF THE YEAR: In 2019, we have simplified the set of rewriting rules, which are used in the translation from TLA+ to SMT. We have shown that the rules are sound, that is, that the translator produces a set of SMT constraints that are equisatisfiable to the given TLA+ formula. We have conducted the experiments on 10 TLA+ specifications of distributed algorithms. When running bounded model checking, Apalache outperforms TLC in some cases. When checking inductive invariants, Apalache runs significantly faster than TLC. These results were reported at ACM OOPSLA 2019.

- Partner: Technische Universität Wien
- Contact: Igor Konnov
- Publications: hal-01899719v1 - hal-01871131v1 - hal-02280888v1
- URL: https://forsyte.at/research/apalache/

6.8. IMITATOR

KEYWORDS: Verification - Parametric model - Parameter synthesis - Model Checking - Model Checker - Timed automata

FUNCTIONAL DESCRIPTION: IMITATOR is a software tool for parametric verification and robustness analysis of real-time systems with parameters. It relies on the formalism of networks of parametric timed automata, augmented with integer variables and stopwatches.

- Participants: Etienne Andre and Jaime Eduardo Arias Almeida
- Partner: Loria
- Contact: Etienne Andre
- Publications: The Inverse Method - Formalizing Time4sys using parametric timed automata - Minimal-Time Synthesis for Parametric Timed Automata - A benchmark library for parametric timed model checking
- URL: https://www.imitator.fr/

6.9. ByMC

Byzantine Model Checker

KEYWORDS: Model Checker - Distributed computing - Verification

SCIENTIFIC DESCRIPTION: In recent work, we have introduced a series of techniques for automatic verification of threshold-guarded distributed algorithms that have the following features: (1) up to t of n processes may exhibit crash or Byzantine failures, (2) the correct processes count messages and progress when they receive sufficiently many messages, e.g., at least $t + 1$, (3) the number n of processes in the system is a parameter, as well as t, (4) and the parameters are restricted by a resilience condition, e.g., $n > 3t$.

ByMC supports a parallel mode, which allows one to run verification experiments in an MPI cluster such as Grid5000 and Vienna Scientific Cluster.

FUNCTIONAL DESCRIPTION: ByMC implements several techniques for the parameterized verification of threshold-guarded distributed algorithms such as reliable broadcast, one-step Byzantine consensus, non-blocking atomic commit, condition-based consensus, and randomized consensus. The tool accepts two kinds of inputs: (i) threshold automata (the framework of our verification techniques) and (ii) Parametric Promela (which is similar to the way in which the distributed algorithms are presented in the distributed computing literature). Internally, the tool analyzes representative executions by querying an SMT solver. Apart from verification, ByMC also implements a technique for the automatic synthesis of threshold guards.

The tool can run on a single computer as well as in an MPI cluster, e.g., Grid5000 or Vienna Scientific Cluster.

NEWS OF THE YEAR: In 2019, we have shown how to apply ByMC to randomized fault-tolerant consensus algorithms such as randomized consensus by Ben-Or and RS-BOSCO. This result was presented at CONCUR 2019.

- Partner: Technische Universität Wien
- Contact: Igor Konnov
- Publications: ByMC: Byzantine Model Checker - Reachability in Parameterized Systems: All Flavors of Threshold Automata - Model Checking of Fault-Tolerant Distributed Algorithms: from Classics towards Contemporary - Verification of Randomized Distributed Algorithms under Round-Rigid Adversaries
- URL: https://forsyte.at/software/bymc/

CIDRE Project-Team

5. New Software and Platforms

5.1. Blare

To detect intrusion using information flows

KEYWORDS: Cybersecurity - Intrusion Detection Systems (IDS) - Data Leakage Protection

SCIENTIFIC DESCRIPTION: Blare implements our approach of illegal information flow detection for a single node (Android and Linux kernel, JVM) and a set of nodes (monitoring of flows between linux machines).

FUNCTIONAL DESCRIPTION: Blare IDS is a set of tools that implements our approach to illegal information flow detection for a single node and a set of nodes.

NEWS OF THE YEAR: During this year, Laurent Georget has modified the implementation of Blare in order to correctly monitor the kernel system calls with LSM hooks. He add also ported this new version of Blare to the Lollipop Android emulator.

- Partner: CentraleSupélec
- Contact: Frédéric Tronel
- Publications: Information Flow Tracking for Linux Handling Concurrent System Calls and Shared Memory - Verifying the Reliability of Operating System-Level Information Flow Control Systems in Linux - Monitoring both OS and program level information flows to detect intrusions against network servers - Experimenting a Policy-Based HIDS Based on an Information Flow Control Model - Introducing reference flow control for intrusion detection at the OS level - Blare Tools: A Policy-Based Intrusion Detection System Automatically Set by the Security Policy - Diagnosing intrusions in Android operating system using system flow graph - Intrusion detection in distributed systems, an approach based on taint marking - BSPL: A Language to Specify and Compose Fine-grained Information Flow Policies - Information Flow Policies vs Malware - A taint marking approach to confidentiality violation detection - Designing information flow policies for Android's operating system - Information Flow Control for Intrusion Detection derived from MAC Policy - Flow based interpretation of access control: Detection of illegal information flows - A taint marking approach to confidentiality violation detection
- URL: http://www.blare-ids.org

5.2. GroddDroid

KEYWORDS: Android - Detection - Malware

SCIENTIFIC DESCRIPTION: GroddDroid automates the dynamic analysis of a malware. When a piece of suspicious code is detected, groddDroid interacts with the user interface and eventually forces the execution of the identified code. Using Blare (Information Flow Monitor), GroddDroid monitors how an execution contaminates the operating system. The output of GroddDroid can be visualized in an web browser. GroddDroid is used by the Kharon software.

FUNCTIONAL DESCRIPTION: GroddDroid 1 - locates suspicious code in Android application 2 - computes execution paths towards suspicious code 3 - forces executions of suspicious code 4 - automate the execution of a malware or a regular Android application

NEWS OF THE YEAR: In 2017, GroddDroid has integrated the work of Mourad Leslous, who have implemented GFinder. GPFinder improves the computation of control flow paths by taking into account the Android framework. The end of the year has been used to clean the code and to improves the graphical interface.

- Authors: Mourad Leslous, Adrien Abraham, Pierre Graux, Jean François Lalande, Valérie Viet Triem Tong and Pierre Wilke
- Partners: CentraleSupélec - Insa Centre Val-de-Loire
- Contact: Valérie Viet Triem Tong
- Publications: Kharon dataset: Android malware under a microscope - GroddDroid: a Gorilla for Triggering Malicious Behaviors - GPFinder: Tracking the Invisible in Android Malware - Information flows at OS level unmask sophisticated Android malware
- URL: http://kharon.gforge.inria.fr/grodddroid.html

5.3. HardBlare

KEYWORDS: Intrusion Detection Systems (IDS) - FPGA - Static analysis

FUNCTIONAL DESCRIPTION: HardBlare is a hardware/software framework to implement hardware DIFC on Xilinx Zynq Platform. HardBlare consists of three components : 1) the VHDL code of the coprocessor, 2) a modified LLVM compiler to compute the static analysis, and 3) a dedicated Linux kernel. This last component is a specific version of the Blare monitor.

- Partners: CentraleSupélec - Lab-STICC
- Contact: Guillaume Hiet
- Publications: ARMHEx: A hardware extension for DIFT on ARM-based SoCs - ARMHEx: a framework for efficient DIFT in real-world SoCs - ARMHEx: embedded security through hardware-enhanced information flow tracking - HardBlare: a Hardware-Assisted Approach for Dynamic Information Flow Tracking - A portable approach for SoC-based Dynamic Information Flow Tracking implementations - Towards a hardware-assisted information flow tracking ecosystem for ARM processors - HardBlare: an efficient hardware-assisted DIFC for non-modified embedded processors

5.4. GroddViewer

KEYWORDS: Android - Detection - Malware

FUNCTIONAL DESCRIPTION: To visualise data from GroddDroid

- Authors: Jean-François Lalande, Valérie Viet Triem Tong, Sébastien Campion, Mathieu Simon and Pierre Wilke
- Contact: Valérie Viet Triem Tong

5.5. Survivor

KEYWORDS: Intrusion Response - Intrusion Recovery - Survivability - Resiliency - Linux - Checkpoint/Restore - Threat Mitigation

FUNCTIONAL DESCRIPTION: Survivor is a set of low-level components to design a Linux-based operating system able to withstand ongoing intrusions and to allow business continuity despite the presence of an active adversary. Survivor provides an Intrusion Response System (IRS) with the low-level components and interfaces needed to orchestrate a per-service checkpoint, recovery, and mitigation actions. It recovers infected services (i.e., their processes and their associated files) to a previous safe state and it protects their state by applying a set of mitigations (e.g., privilege restrictions and resource quotas) aimed at withstanding further reinfections.

- Participants: Ronny Chevalier, Guillaume Hiet, David Plaquin and Chris Dalton
- Partners: CentraleSupélec - HP Labs
- Contact: Ronny Chevalier

5.6. PyMaO

Python Malware Orchestrator

KEYWORDS: Android - Malware

FUNCTIONAL DESCRIPTION: PyMaO chains several analyses that are part of an experiment. An analysis is most of the time, a call to an external tool that returns a result, for example apktool, grep, Androguard, Apkid. An experiment is a collection of analyses that are run one by one, chained, if some conditions hold. For example, if the unpacking of an application with Apktool succeeds, then you can grep the code for searching a string.

PyMaO has a nice old-fashion graphical interface (ncurses).

RELEASE FUNCTIONAL DESCRIPTION: Initial release corresponding to the demo presented at MASCOTS 2019.

NEWS OF THE YEAR: A demo has been presented at the MASCOTS 2019 conference: https://hal-centralesupelec.archives-ouvertes.fr/hal-02305473

- Authors: Jean-François Lalande, Pierre Graux and Tomas Javier Concepcion Miranda
- Contact: Jean-François Lalande
- URL: https://gitlab.inria.fr/cidre-public/pymao

5.7. OATs'inside

KEYWORDS: Android - Malware - Reverse engineering - Code analysis

FUNCTIONAL DESCRIPTION: OATs'inside is a Android reverse engineering tool that handles all native obfuscation techniques. This tool uses a hybrid approach based on dynamic monitoring and trace-based symbolic execution to output control flow graphs (CFGs) for each method of the analyzed application. These CFGs spare users the need to dive into low-level instructions, which are difficult to reverse engineer.

- Participants: Pierre Graux, Jean-François Lalande, Valérie Viet Triem Tong and Pierre Wilke
- Contact: Pierre Graux

COMETE Project-Team

6. New Software and Platforms

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

KEYWORDS: Information leakage - Privacy - C++ - Linear optimization

FUNCTIONAL DESCRIPTION: 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's development continued in 2018 with several new added features. 82 new commits were pushed to the project's git repository during this year. The new functionality was directly applied to the exeperimental results of several publications of the team (QEST'18, Entropy'18, POST'18, CSF'18).

- Contact: Konstantinos Chatzikokolakis
- URL: https://github.com/chatziko/libqif

6.2. F-BLEAU

KEYWORDS: Information leakage - Machine learning - Privacy

FUNCTIONAL DESCRIPTION: F-BLEAU is a tool for estimating the leakage of a system about its secrets in a black-box manner (i.e., by only looking at examples of secret inputs and respective outputs). It considers a generic system as a black-box, taking secret inputs and returning outputs accordingly, and it measures how much the outputs "leak" about the inputs.

F-BLEAU is based on the equivalence between estimating the error of a Machine Learning model of a specific class and the estimation of information leakage.

This code was also used for the experiments of a paper under submission, on the following evaluations: Gowalla, e-passport, and side channel attack to finite field exponentiation.

RELEASE FUNCTIONAL DESCRIPTION: First F-BLEAU release. Supports frequentist and k-NN estimates with several parameters, and it allows stopping according to delta-convergence criteria.

- Contact: Konstantinos Chatzikokolakis
- URL: https://github.com/gchers/fbleau

6.3. Location Guard

KEYWORDS: Privacy - Geolocation - Browser Extensions

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

FUNCTIONAL DESCRIPTION: Websites can ask the browser for your location (via JavaScript). When they do so, the browser first asks your permission, and if you accept, it detects your location (typically by transmitting a list of available wifi access points to a geolocation provider such as Google Location Services, or via GPS if available) and gives it to the website.

Location Guard is a browser extension that intercepts this procedure. The permission dialog appears as usual, and you can still choose to deny. If you give permission, then Location Guard obtains your location and adds "random noise" to it, creating a fake location. Only the fake location is then given to the website.

Location Guard is by now a stable tool with a large user base. No new features were added in 2018, however the tool is still actively maintained, and several issues have beed fixed during this year (new geocoder API, manual installation method for Opera users, etc).

- Participants: Catuscia Palamidessi, Konstantinos Chatzikokolakis, Marco Stronati, Miguel Andrés and Nicolas Bordenabe
- Contact: Konstantinos Chatzikokolakis
- URL: https://github.com/chatziko/location-guard

6.4. dspacenet

Distributed-Spaces Network.

KEYWORDS: Social networks - Distributed programming

FUNCTIONAL DESCRIPTION: DSpaceNet is a tool for social networking based on multi-agent spatial and timed concurrent constraint language.

I - The fundamental structure of DSPaceNet is that of *space*: A space may contain

(1) spatial-mobile-reactive tcc programs, and (2) other spaces.

Furthermore, (3) each space belongs to a given agent. Thus, a space of an agent j within the space of agent i means that agent i allows agent j to use a computation sub-space within its space.

II - The fundamental operation of DSPaceNet is that of *program posting*: In each time unit, agents can post spatial-mobile-reactive tcc programs in the spaces they are allowed to do so (ordinary message posting corresponds to the posting of tell processes). Thus, an agent can for example post a watchdog tcc process to react to messages in their space, e.g. whenever (*happy b*frank*) do tell("thank you!"). More complex mobile programs are also allowed (see below).

The language of programs is a spatial mobile extension of tcc programs:

$P, Q... := tell(c)|whencdoP||nextP|P||Q|unlesscnextP|[P]_i|\uparrow_i P|recX.P$

Computation of timed processes proceeds as in tcc. The spatial construct $[P]_i$ runs P in the space of agent i and the mobile process $\uparrow_i P$, extrudes P from the space of i. By combining space and mobility, arbitrary processes can be moved from one a space into another. For example, one could send a trojan watchdog to another space for spying for a given message and report back to one's space.

III- Constraint systems can be used to specify advance text message deduction, arithmetic deductions, scheduling, etc.

IV - Epistemic Interpretation of spaces can be used to derive whether they are users with conflicting/inconsistent information, or whether a group of agents may be able to deduce certain message.

V - The scheduling of agent requests for program posts, privacy settings, friendship lists are handled by an external interface. For example, one could use type systems to check whether a program complies with privacy settings (for example checking that the a program does not move other program into a space it is not allowed into).

- Partner: Pontificia Universidad Javeriana Cali
- Contact: Frank Valencia
- URL: http://www.dspacenet.com

DATASPHERE Team

5. New Software and Platforms

5.1. DNS data analysis

Data analytics tools for DNS data analysis were developed in a cooperation with ICT, Chinese Academy of Sciences in the frame of the thesis of Jingxiu SU.

5.1.1. BGP Monitoring platform

An observatory of global BGP connectivity has been developed that is used to monitor and detect in real time BGP level attacks. In addition, a set of tools were developed to analyse the structure of information propagation over social networks.

5.1.2. Atlas of Data

A platform to visualize data flows over the planet is under construction. It can be accessed online at https://theatlasofdata.earth/.

5.1.3. Observatory of foreign influence on social media

This observatory is monitoring on twitter and facebook the evolution of foreign influence. It is based on a twitter collection platform that is using an extensive database of foreign actors to detect and monitor foreign interference.

PESTO Project-Team

6. New Software and Platforms

6.1. Akiss

AKISS - Active Knowledge in Security Protocols

KEYWORDS: Security - Verification

FUNCTIONAL DESCRIPTION: 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, as well as the exclusive or (xor) operator.

- Contact: Steve Kremer
- URL: https://github.com/akiss

6.2. Belenios

Belenios - Verifiable online voting system

KEYWORD: E-voting

FUNCTIONAL DESCRIPTION: Belenios is an open-source 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 received) 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. Moreover, Belenios includes a practical threshold decryption system that allows splitting the decryption key among several authorities.

NEWS OF THE YEAR: Since 2015, it has been used by CNRS for remote election among its councils (more than 30 elections every year) and since 2016, it has been used by Inria to elect representatives in the “comités de centre” of each Inria center. In 2018, it has been used to organize about 250 elections (not counting test elections). Belenios is typically used for elections in universities as well as in associations. This goes from laboratory councils (e.g. Irisa, Cran), scientific societies (e.g. SMAI) to various associations (e.g. FFBS - Fédération Française de Baseball et Softball, or SRFA - Société du Rat Francophone et de ses Amateurs).

In 2019, a threshold encryption mode has been added that makes the system more robust to the case where (say) one trustee among three loses her part of the decryption key.

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

6.3. Deepsec

DEEPSEC - DEciding Equivalence Properties in SECurity protocols

KEYWORDS: Security - Verification

FUNCTIONAL DESCRIPTION: DEEPSEC (DEciding Equivalence Properties 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. DEEPSEC implements a decision procedure to verify trace equivalence for a bounded number of sessions and cryptographic primitives modeled by a subterm convergent destructor rewrite system. The procedure is based on constraint solving techniques. The tool also implements state-of-the-art partial order reductions and allows to distribute the computation on multiple cores and multiple machines.

NEWS OF THE YEAR: In 2019, to improve efficiency for non-determinate processes, we developed new optimisation techniques. This is achieved through a new, stronger equivalence for which partial-order reductions are sound even for non-determinate processes, as well as new symmetry reductions. We demonstrated that these techniques provide a significant (several orders of magnitude) speed-up in practice, thus increasing the size of the protocols that can be analysed fully automatically. Even though the new equivalence is stronger, it is nevertheless coarse enough to avoid false attacks on most practical examples.

- Participants: Steve Kremer, Itsaka Rakotonirina and Vincent Cheval
- Contact: Vincent Cheval
- Publications: Exploiting Symmetries When Proving Equivalence Properties for Security Protocols - Exploiting symmetries when proving equivalence properties for security protocols (Technical report) - DEEPSEC: Deciding Equivalence Properties in Security Protocols Theory and Practice - DEEPSEC: Deciding Equivalence Properties in Security Protocols - Theory and Practice - The DEEPSEC prover
- URL: https://deepsec-prover.github.io/

6.4. Tamarin

TAMARIN prover

KEYWORDS: Security - Verification

FUNCTIONAL DESCRIPTION: 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, bilinear pairing, multisets, and exclusive-or (XOR), combined with a user-defined convergent rewriting theory. Its main advantages are its ability to handle stateful protocols and its interactive proof mode. Moreover, it has 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. In a joint effort, the partners wrote and published a user manual in 2016, available from the Tamarin website.

- Contact: Jannik Dreier
- URL: http://tamarin-prover.github.io/

6.5. SAPIC

SAPIC: Stateful Applied Pi Calculus

KEYWORDS: Security - Verification

FUNCTIONAL DESCRIPTION: SAPIC is a plugin of the TAMARIN 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 analysed by the TAMARIN prover. TAMARIN has also been extended with dedicated heuristics that exploit the form of translated rules and favor 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 notion of location and reporting used for modelling trusted execution environments. It has been successfully applied to several case studies including the Yubikey authentication protocol, and extensions of the PKCS#11 standard. SAPIC also includes support for verifying liveness properties, which are for instance important in fair exchange and contract signing protocols, as well as support for constructions useful when modelling isolated execution environments.

- Contact: Steve Kremer
- URL: http://sapic.gforge.inria.fr/

6.6. TypeEquiv

A type checker for privacy properties

KEYWORDS: Security - Cryptographic protocol - Privacy

FUNCTIONAL DESCRIPTION: TypeEquiv provides a (sound) type system for proving equivalence of protocols (to analyse privacy properties such as vote privacy, anonymity, unlinkability), for both a bounded or an unbounded number of sessions and for the standard cryptographic primitives. TypeEquiv takes as input the specification of a pair of security protocols, written in a dialect of the applied-pi calculus, together with some type annotations. It checks whether the two protocols are in equivalence or not. The tool provides a significant speed-up compared with tools that decide equivalence of security protocols for a bounded number of sessions.

- Partner: Technische Universität Wien
- Contact: Véronique Cortier

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.

- Participant: Vincent Roca
- Contact: Vincent Roca

5.2. Wombat

Wi-Fi tracking system for testing and demonstrational purpose

KEYWORDS: Wi-Fi - Privacy - Multimodal tracking of human activity - Wireless network

FUNCTIONAL DESCRIPTION: Wombat is a fully functional Wi-Fi tracking platform supporting three main features: collection, storage/processing, query/output. These three features are implemented through a distributed infrastructure composed of:

Sensor nodes: small devices with wireless monitoring capabilities. They collect information sent on wireless channels and forward it to the server. Central server: the central entity of the system. It receives data sent by sensor nodes and then stores it in an internal data structure. It is also in charge of answering queries related to the stored data.

To ensure communication between the sensor nodes and the server, the Wombat system relies on a wired network (Ethernet). In addition, Wombat can be enriched with a user interface and an opt-out node:

User interface: a device in charge of displaying detailed information about one or several tracked devices (see figure below). The device to display can be specified manually by its MAC address or through proximity detection. Opt-out node: an element in charge of implementing an opt-out mechanism for users refusing to be tracked by the system.

The system is made to work on a dedicated network (the server includes a DHCP server). Nodes can be switched off at any time (they function in read-only mode to be crash-proof).

- Partner: Insa de Lyon
- Contact: Mathieu Cunche
- URL: https://github.com/Perdu/wombat

5.3. Cookie glasses

KEYWORDS: GDPR - Cookie - Consent

SCIENTIFIC DESCRIPTION: In the paper Do Cookie Banners Respect my Choice? Measuring Legal Compliance of Banners from IAB Europe's Transparency and Consent Framework, we show that Consent Management Providers (CMPs) of IAB Europe's Transparency & Consent Framework (TCF) do not always respect user's choice. This extension allows users to verify that their consent is stored appropriately by themselves.

This extension for Firefox and Chrome queries CMPs of IAB Europe's TCF in the same position as a third-party advertiser, making it possible to see consent set by CMPs in real time. In other words, you can see whether consent registered by cookie banners is actually the consent you gave. Will only work with cookie banners of IAB Europe's TCF.

We also added a functionality to manually decode a so-called "consent string" of the framework.

- Participants: Célestin Matte and Nataliia Bielova
- Contact: Alain Prette

5.4. BELL

Browser fingerprinting via Extensions and Login-Leaks

KEYWORDS: Browser Extensions - Security and Privacy in Web Services - Social Networks Security and Privacy

FUNCTIONAL DESCRIPTION: Recent studies show that users can be tracked based on their web browser properties. This software is designed to conduct an experiment on such kinds of user tracking. In this experiment, we demonstrate that a Web user can also be tracked by

- her browser extensions (such as AdBlock, Pinterest, or Ghostery), and

- the websites she has logged in (such as Facebook, Gmail, or Twitter).

In the experiment, we collect user's browser fingerprint, together with the browser extensions installed and a list of websites she has logged in. We only collect anonymous data during the experiment (more details in our Privacy Policy [0]), we will securely store the data on an Inria server, use it only for research purposes and not share it with anyone outside of Inria.

- Contact: Gabor Gulyas
- URL: https://extensions.inrialpes.fr/

5.5. SWIF-codec

An open-source sliding window FEC codec

KEYWORD: Error Correction Code

FUNCTIONAL DESCRIPTION: This development is done in the context of the "Coding for Efficient Network Communications" IRTF Research Group (NWCRG, [https://datatracker.ietf.org/rg/nwcrg]) and IETF hackathon.

This work has strong relationships with the Generic API I-D [https://datatracker.ietf.org/doc/draft-roca-nwcrg-generic-fec-api/] and RFC 8681 on RLC codes [https://www.rfc-editor.org/rfc/rfc8681] as examples of sliding window codes.

- Authors: Vincent Roca, Cédric Adjih, Oumaima Attia and François Michel
- Contact: Vincent Roca
- URL: https://github.com/irtf-nwcrg/swif-codec

[0] https://extensions.inrialpes.fr/privacy.php

PROSECCO Project-Team

6. New Software and Platforms

6.1. Cryptosense Analyzer

SCIENTIFIC DESCRIPTION: Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards, security tokens and Hardware Security Modules that support the most widely-used industry standard interface, RSA PKCS#11. Each device implements PKCS#11 in a slightly different way since the standard is quite open, but finding a subset of the standard that results in a secure device, i.e. one where cryptographic keys cannot be revealed in clear, is actually rather tricky. Cryptosense Analyzer analyses a device by first reverse engineering the exact implementation of PKCS#11 in use, then building a logical model of this implementation for a model checker, calling a model checker to search for attacks, and in the case where an attack is found, executing it directly on the device. It has been used to find at least a dozen previously unknown flaws in commercially available devices.

FUNCTIONAL DESCRIPTION: Cryptosense Analyzer (formerly known as Tookan) is a security analysis tool for cryptographic devices such as smartcards,

- Participants: Graham Steel and Romain Bardou
- Contact: Graham Steel
- URL: https://cryptosense.com/

6.2. CryptoVerif

Cryptographic protocol verifier in the computational model

KEYWORDS: Security - Verification - Cryptographic protocol

FUNCTIONAL DESCRIPTION: CryptoVerif 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. It also provides an explicit formula that gives the probability of breaking the protocol as a function of the probability of breaking each primitives, this is the exact security framework.

NEWS OF THE YEAR: We implemented the following features in CryptoVerif:

1) We added to the library of cryptographic primitives several variants of the PRF-ODH (pseudo-random function oracle Diffie-Hellman) assumption, pre-image resistant and second-preimage resistant hash functions, IND-CPA encryption with a nonce, IND-CPA and INT-CTXT encryption with a nonce, encryption schemes that satisfy IND$-CPA instead of IND-CPA.

2) To facilitate modular proofs, we allow querying indistinguishability properties with exactly the same syntax as the one used to specify indistinguishability assumptions on primitives.

3) To simplify declarations of assumptions on primitives, replications (which model any number of copies of processes or oracles) can be omitted at the root of indistinguishability assumptions. CryptoVerif adds them internally, thus inferring the assumption for N independent copies from the assumption for one copy. For instance, it infers the assumption for encryption with N keys from the assumption for encryption with a single key.

4) When we delay random number generations, we allow the user to specify expressions for which it is not necessary to generate the random value, so that the generation of the moved random value can be delayed further. In particular, we used this extension to prove that the OAEP scheme is IND-CCA2 assuming the underlying permutation is partial-domain one-way (a famous cryptographic result).

5) CryptoVerif can now remove parts of the code cannot be executed in case the adversary wins the game, by replacing them with event "AdversaryLoses". That is specially helpful in order to deal with complex cases of key compromise, e.g. for forward secrecy, by proving authentication by ignoring the compromise, showing that authentication is preserved in case the key is compromised (because the adversary never wins against the considered authentication property in case of compromise), and using the authentication to prove secrecy even in case of compromise. For instance, that allows us to show that the PSK-DHE handshake of TLS 1.3 preserves forward secrecy in case of compromise of the PSK.

6) After a cryptographic transformation, CryptoVerif expands terms into processes, which leads to duplicating code until the end of the protocol for each test that is expanded. The cryptographic transformation and the expansion were initially considered as a single transformation. There are now considered as separate transformations, so that other transformations can be performed in between, in particular to cut some branches of the code and reduce the code duplication.

These changes are included in CryptoVerif version 2.02 available at https://cryptoverif.inria.fr.

- Participants: Bruno Blanchet and David Cadé
- Contact: Bruno Blanchet
- Publications: Composition Theorems for CryptoVerif and Application to TLS 1.3 - Composition Theorems for CryptoVerif and Application to TLS 1.3 - A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol - A Mechanised Cryptographic Proof of the WireGuard Virtual Private Network Protocol - Proved Implementations of Cryptographic Protocols in the Computational Model - Proved Generation of Implementations from Computationally Secure Protocol Specifications - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols - Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach
- URL: http://cryptoverif.inria.fr/

6.3. F*

FStar

KEYWORDS: Programming language - Software Verification

FUNCTIONAL DESCRIPTION: 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 Fw (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.

- Participants: Antoine Delignat-Lavaud, Catalin Hritcu, Cedric Fournet, Chantal Keller, Karthikeyan Bhargavan and Pierre-Yves Strub
- Contact: Catalin Hritcu
- URL: https://www.fstar-lang.org/

6.4. miTLS

KEYWORDS: Cryptographic protocol - Software Verification

FUNCTIONAL DESCRIPTION: miTLS is a verified reference implementation of the TLS protocol. Our code fully supports its wire formats, ciphersuites, sessions and connections, re-handshakes and resumptions, alerts and errors, and data fragmentation, as prescribed in the RFCs, it interoperates with mainstream web browsers and servers. At the same time, our code is carefully structured to enable its modular, automated verification, from its main API down to computational assumptions on its cryptographic algorithms.

- Participants: Alfredo Pironti, Antoine Delignat-Lavaud, Cedric Fournet, Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Pierre-Yves Strub and Santiago Zanella
- Contact: Karthikeyan Bhargavan
- URL: https://github.com/mitls/mitls-fstar

6.5. ProVerif

KEYWORDS: Security - Verification - Cryptographic protocol

FUNCTIONAL DESCRIPTION: ProVerif 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 verify various security properties (secrecy, authentication, process equivalences).

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.

NEWS OF THE YEAR: Vincent Cheval and Bruno Blanchet worked on several extensions of ProVerif: 1) support for integer counters, with incrementation and inequality tests, 2) lemmas and axioms to give intermediate results to ProVerif, which it exploits to help proving subsequent queries, by deriving additional information in the Horn clauses that it uses to perform the proofs, 3) proofs by induction on the length of the trace, by giving as lemma the property to prove, but obviously for strictly shorter traces. Detailed soundness proofs for these extensions are in progress. These features are not released yet.

- Participants: Bruno Blanchet, Marc Sylvestre and Vincent Cheval
- Contact: Bruno Blanchet
- Publications: Automated reasoning for equivalences in the applied pi calculus with barriers - Automated Reasoning for Equivalences in the Applied Pi Calculus with Barriers - Automated reasoning for equivalences in the applied pi calculus with barriers - Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif - Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate - Automated Verification for Secure Messaging Protocols and Their Implementations: A Symbolic and Computational Approach - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols - Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols
- URL: http://proverif.inria.fr/

6.6. HACL*

High Assurance Cryptography Library

KEYWORDS: Cryptography - Software Verification

FUNCTIONAL DESCRIPTION: HACL* is a formally verified cryptographic library in F*, developed by the Prosecco team at Inria Paris in collaboration with Microsoft Research, as part of Project Everest.

HACL stands for High-Assurance Cryptographic Library and its design is inspired by discussions at the HACS series of workshops. The goal of this library is to develop verified C reference implementations for popular cryptographic primitives and to verify them for memory safety, functional correctness, and secret independence.

- Contact: Karthikeyan Bhargavan
- URL: https://github.com/mitls/hacl-star

TAMIS Project-Team

5. New Software and Platforms

5.1. MASSE

Modular Automated Syntactic Signature Extraction

KEYWORDS: Malware - Syntactic analysis

FUNCTIONAL DESCRIPTION: The Modular Automated Syntactic Signature Extraction (MASSE) architecture is a new integrated open source client-server architecture for syntactic malware detection and analysis based on the YARA, developed with Teclib'. MASSE includes highly effective automated syntactic malware detection rule generation for the clients based on a server-side modular malware detection system. Multiple techniques are used to make MASSE effective at detecting malware while keeping it from disrupting users and hindering reverse-engineering of its malware analysis by malware creators. MASSE integrates YARA in a distributed system able to detect malware on endpoint systems using YARA, analyze malware with multiple analysis techniques, automatically generate syntactic malware detection rules, and deploy the new rules to the endpoints. The MASSE architecture is freely available to companies and institutions as a complete, modular, self-maintained antivirus solution. Using MASSE, a security department can immediately update the rule database of the whole company, stopping an infection on its tracks and preventing future ones.

- Participants: Bruno Lebon, Olivier Zendra, Alexander Zhdanov and Fabrizio Biondi
- Contact: Bruno Lebon

5.2. IoTMLT

IoT Modeling Language and tool

KEYWORDS: Internet of things - Modeling language - Cyber attack

SCIENTIFIC DESCRIPTION: We propose a framework to analyze security in IoT systems consisting of a formal languages for modeling IoT systems and of attack trees for modeling the possible attacks on the system. In our approach a malicious entity is present in the system, called the Attacker. The other IoT entities can inadvertently help the Attacker, by leaking their sensitive data. Equipped with the acquired knowledge the Attacker can then communicate with the IoT entities undetected. The attack tree provided with the model acts as a monitor: It observes the interactions the Attacker has with the system and detects when an attack is successful.

An IoT system is then analyzed using statistical model checking (SMC). The first method we use is Monte Carlo, which consists of sampling the executions of an IoT system and computing the probability of a successful attack based on the number of executions for which the attack was successful. However, the evaluation may be difficult if a successful attack is rare. We therefore propose a second SMC method, developed for rare events, called importance splitting. Both methods are proposed by Plasma, the SMC tool we use.

FUNCTIONAL DESCRIPTION: The IoT modeling language is a formal language and tool for specifying and enforcing security in IoT systems.

- Participants: Delphine Beaulaton, Ioana-Domnina Cristescu and Najah Ben Said
- Partner: Vérimag
- Contact: Delphine Beaulaton
- URL: http://iot-modeling.gforge.inria.fr

5.3. SimFI

Tool for Simulation Fault injection

KEYWORDS: Fault injection - Fault-tolerance

FUNCTIONAL DESCRIPTION: Fault injections are used to test the robust and security of systems. We have developed SimFI, a tool that can be used to simulate fault injection attacks against binary files. SimFI is lightweight utility designed to be integrated into larger environments as part of robustness testing and fault injection vulnerability detection.

- Contact: Nisrine Jafri
- URL: https://github.com/nisrine/Fault-Injection-Tool

5.4. AHMA

Automatic Malware Hardware Analysis

KEYWORDS: Side-channel - Deep learning - Malware

FUNCTIONAL DESCRIPTION: This framework is composed of several parts, each one of them taking in charge the generation and the processing of the data at different levels. Drivers have been developed to automatically control the different oscilloscopes we are working with (picoscope 6407 et infiniium keysight). We use signal processing tools on the raw data to feed a deep neural network which is in charge of classifying the observed malwares. We are using two different approaches to manage the infection of the system. The first one is to reinitialize it each time we make a measurement to ensure its integrity. We have proposed a method allowing to speed the procedure up a lot. Besides, we developed several malwares, to make our experiments in a controlled environment, to avoid the necessity of cleaning the system up after each measurement.

- Contact: Annelie Heuser

5.5. SABR

Semantic-driven Analysis of BinaRies

KEYWORDS: Malware - Semantic - Binary analysis - Unsupervised graph clustering SCDG - Machine learning

FUNCTIONAL DESCRIPTION: Toolchain for binary analysis based on different techniques for capturing binaries' semantics and performing machine learning-assisted analysis. The primary use is malware analysis for malware detection and classification, either based on supervised and unsupervised learning.

This toolchain includes modules of the former BMA toolchain, specifically the SCDG extraction.

Our approach is based on artificial intelligence. We use concolic analysis to extract behavioral signatures from binaries in a form of system call dependency graphs (SCDGs). Our software can do both supervised and unsupervised learning. The former learns the distinctive features of different malware families on a large training set in order to classify the new binaries as malware or cleanware according to their behavioural signatures. In the unsupervised learning the binaries are clustered according to their graph similarity. The toolchain is orchestrated by an experiment manager that allows to easily setup, launch and view results of all modules of the toolchain.

- Contact: Olivier Zendra

5.6. ORQAL

ORQchestration of ALgorithms

KEYWORDS: Docker - Orchestration

FUNCTIONAL DESCRIPTION: ORQAL is a simple batch scheduler for docker cluster which can be used to remotely and without overhead in scientific experiment.

- Contact: Olivier Zendra

5.7. Side-channel deep learning evaluation platform

KEYWORDS: Deep learning - Evaluation

FUNCTIONAL DESCRIPTION: Our platform is based on several software. The first software permits to train a deep neural network and evaluate it for side-channel analysis, we evaluate our neural network with guessing entropy metrics. The second software is used for programming and communicating with the target devices, but we also develop a software to communicate with the equipment and made some measurement for side-channel analysis. The last software is used to make some attack and analysis of side-channel (e.g. made Correlation Power Analysis)

- Contact: Annelie Heuser

5.8. E-PAC

Evolving-PAcker Classifier

KEYWORDS: Packer classification - Incremental learning - Clustering - Malware - Obfuscation

FUNCTIONAL DESCRIPTION: E-PAC is an Evolving packer classifier that identifies the class of the packer used in a batch of packed binaries given in input. The software has the ability to identify both known packer classes and new unseen packer classes. After each update, the evolving classifier self-updates itself with the predicted packer classes.

The software is based on a semi-supervised machine learning system composed of an offline phase and an online phase. In the offline phase, a set of features is extracted from a collection of packed binaries provided with their ground truth labels, then a density-based clustering algorithm (DBSCAN) is used to group similar packers together with respect to a distance measure. In this step, the similarity threshold is tuned in order to form the clusters that fit the best with the the set of labels provided.

In the online phase, the software reproduces the same operations of features extraction and distances calculation with the incoming packed samples, then uses a customized version of the incremental clustering algorithm DBSCAN in order to classify them, either in knowns packer classes or fom new packer classes, or provisoirely leave them unclassified (notion of noise with DBSCAN).

The clusters formed after each update serve as a baseline for the application to self-evolve.

- Contact: Lamine Noureddine