RESEARCH CENTER
Paris - Rocquencourt

FIELD

Activity Report 2014

Section Software

Edition: 2015-03-24

ALPAGE Project-Team

5. New Software and Platforms

5.1. Syntax

Participants: Pierre Boullier [correspondant], Benoît Sagot.

See also the web page http://syntax.gforge.inria.fr/.

The (currently beta) version 6.0 of the SYNTAX system (freely available on Inria GForge) includes various deterministic and non-deterministic CFG parser generators. It includes in particular an efficient implementation of the Earley algorithm, with many original optimizations, that is used in several of Alpage's NLP tools, including the pre-processing chain SXPipe and the LFG deep parser SXLFG. This implementation of the Earley algorithm has been recently extended to handle probabilistic CFG (PCFG), by taking into account probabilities both during parsing (beam) and after parsing (n-best computation). SYNTAX 6.0 also includes parsers for various contextual formalisms, including a parser for Range Concatenation Grammars (RCG) that can be used among others for TAG and MC-TAG parsing.

In 2014, an in-depth rewriting of the RCG parser has started, in order for RCG parsers produced by SYNTAX to handle input DAGs while remaining efficient [60], although parsing time complexity might, on such inputs, become exponential w.r.t. their length, whereas RCGs exactly cover the set of languages that are parsable in polynomial time (if the input is a string).

Direct NLP users of SYNTAX for NLP, outside Alpage, include Alexis Nasr (Marseilles) and other members of the (now closed) SEQUOIA ANR project, Owen Rambow and co-workers at Columbia University (New York), as well as (indirectly) all SXPipe and/or SXLFG users. The project-team VASY (Inria Rhône-Alpes) is one of SYNTAX' user for non-NLP applications.

5.2. DyALog

Participant: Éric Villemonte de La Clergerie [maintainer].

DYALOG *on Inria GForge: http://dyalog.gforge.inria.fr/*

DYALOG provides an environment to compile and execute grammars and logic programs. It is essentially based on the notion of tabulation, i.e. of sharing computations by tabulating traces of them. DYALOG is mainly used to build parsers for Natural Language Processing (NLP). It may nevertheless be used as a replacement for traditional PROLOG systems in the context of highly ambiguous applications where sub-computations can be shared.

The current release of DYALOG (version 1.14.0) is freely available by FTP under an open source license and runs on Linux platforms for x86 and architectures and on Mac OS intel (both 32 and 64bits architectures). In particular, it has been ported for the CLANG/LLVM compiler used in recent Mac OS systems (Mavericks).

The current release handles logic programs, DCGs (Definite Clause Grammars), FTAGs (Feature Tree Adjoining Grammars), FTIGs (Feature Tree Insertion Grammars) and XRCGs (Range Concatenation Grammars with logic arguments). Several extensions have been added to most of these formalisms such as intersection, Kleene star, and interleave operators. Typed Feature Structures (TFS) as well as finite domains may be used for writing more compact and declarative grammars [135]. Version 1.14.0 now includes an efficient handler for feature-based statistical models, derived from the work on DYALOG-SR and now used in FRMG parser.

C libraries can be used from within DYALOG to import APIs (mysql, libxml, SQLite, ...).

DYALOG is largely used within ALPAGE to build parsers but also derivative softwares, such as a compiler of Meta-Grammars (cf. 5.3). It has also been used for building FRMG, a parser from a large coverage French TIG/TAG grammar derived from a Meta-Grammar. This parser has been used for the Parsing Evaluation campaign EASy, the two Passage campaigns (Dec. 2007 and Nov. 2009) [130], [134], and very large amount of data (700 millions of words) in the SCRIBO project. New results concerning FRMG are described in 6.5 .

DYALOG is also used to run DYALOG-SR, a transition-based dependency parser (see new results in 6.5)

DYALOG and other companion modules (including DYALOG-SR) are available on Inria GForge.

5.3. Tools and resources for Meta-Grammars

Participant: Éric Villemonte de La Clergerie [maintainer].

`mgcomp`, MGTOOLS, and FRMG on Inria GForge: http://mgkit.gforge.inria.fr/

DYALOG (cf. 5.2) has been used to implement `mgcomp`, Meta-Grammar compiler. Starting from an XML representation of a MG, `mgcomp` produces an XML representation of its TAG expansion.

The current version **1.5.0** is freely available by FTP under an open source license. It is used within ALPAGE and (occasionally) at LORIA (Nancy) and at University of Pennsylvania.

The current version adds the notion of namespace, to get more compact and less error-prone meta-grammars. It also provides other extensions of the standard notion of Meta-Grammar in order to generate very compact TAG grammars. These extensions include the notion of *guarded nodes*, i.e. nodes whose existence and non-existence depend on the truth value of a guard, and the use of the regular operators provided by DYALOG on nodes, namely disjunction, interleaving and Kleene star. The current release provides a dump/restore mechanism for faster compilations on incremental changes of a meta-grammars.

The current version of `mgcomp` has been used to compile a wide coverage Meta-Grammar FRMG (version 2.0.1) to get a grammar of around 200 TAG trees [132]. Without the use of guarded nodes and regular operators, this grammar would have more than several thousand trees and would be almost intractable. FRMG has been packaged and is freely available.

To ease the design of meta-grammars, a set of tools have been implemented, mostly by Éric Villemonte de La Clergerie, and collected in MGTOOLS (version **2.2.2**). This package includes a converter from a compact format to a XML pivot format, an Emacs mode for the compact and XML formats, a graphical viewer interacting with Emacs and XSLT stylesheets to derive HTML views.

The various tools on Metagrammars are available on Inria GForge. FRMG is used directly or indirectly (through a Web service or by requiring parsed corpora) by several people and actions (ANR Rhapsodie, ANR Chronoline, ...)

5.4. The Bonsai PCFG-LA parser

Participants: Marie-Hélène Candito [correspondant], Djamé Seddah, Benoit Crabbé.

Web page:
http://alpage.inria.fr/statgram/frdep/fr_stat_dep_parsing.html

Alpage has developed as support of the research papers [75], [67], [68], [122] a statistical parser for French, named Bonsai, trained on the French Treebank. This parser provides both a phrase structure and a projective dependency structure specified in [66] as output. This parser operates sequentially: (1) it first outputs a phrase structure analysis of sentences reusing the Berkeley implementation of a PCFG-LA trained on French by Alpage (2) it applies on the resulting phrase structure trees a process of conversion to dependency parses using a combination of heuristics and classifiers trained on the French treebank. The parser currently outputs several well known formats such as Penn treebank phrase structure trees, Xerox like triples and CONLL-like format for dependencies. The parsers also comes with basic preprocessing facilities allowing to perform elementary sentence segmentation and word tokenisation, allowing in theory to process unrestricted text. However it is believed to perform better on newspaper-like text.

The parser is available under a GPL license.

5.5. Alpage's linguistic workbench, including SxPipe and MElt

Participants: Benoît Sagot [correspondant], Kata Gábor, Marion Baranes, Pierre Magistry, Pierre Boullier, Éric Villemonte de La Clergerie, Djamé Seddah.

See also the web page http://lingwb.gforge.inria.fr/.

Alpage's linguistic workbench is a set of packages for corpus processing and parsing. Among these packages, two packages are of particular importance: the SxPipe pre-processing chain, and the MElt part-of-speech tagger.

SxPipe [109] is a modular and customizable chain aimed to apply to raw corpora a cascade of surface processing steps. It is used

- as a preliminary step before Alpage's parsers (e.g., FRMG);
- for surface processing (named entities recognition, text normalization, unknown word extraction and processing...).

Developed for French and for other languages, SxPipe includes, among others, various named entities recognition modules in raw text, a sentence segmenter and tokenizer, a spelling corrector and compound words recognizer, and an original context-free patterns recognizer, used by several specialized grammars (numbers, impersonal constructions, quotations...). It can now be augmented with modules developed during the former ANR EDyLex project for analysing unknown words; this involves in particular (i) new tools for the automatic pre-classification of unknown words (acronyms, loan words...) (ii) new morphological analysis tools, most notably automatic tools for constructional morphology (both derivational and compositional), following the results of dedicated corpus-based studies. New local grammars for detecting new types of entities and improvement of existing ones, developed in the context of the PACTE project, will soon be integrated within the standard configuration.

MElt is a part-of-speech tagger, initially developed in collaboration with Pascal Denis (Magnet, Inria — then at Alpage), which was trained for French (on the French TreeBank and coupled with the Lefff), also trained on English [79], Spanish [88], Italian [124], German [38], Dutch, Polish, Kurmanji Kurdish [138] and Persian [119], [120]. It is state-of-the-art for French. It is now able to handle noisy corpora (French and English only; see below). MElt also includes a lemmatization post-processing step. A preliminary version of MElt which accepts input DAGs has been developed in 2013, and is currently under heavy rewriting and improvement in the context of the PACTE project (see 6.3).

MElt is distributed freely as a part of the Alpage linguistic workbench.

In 2014, additional efforts have been achieved for a better pre-processing of noisy input text. This covers two different scenarios:

- user-generated content (see 6.2); two sets of tools are available for processing user-generated content: (i) very noisy computer-mediated content, such as found on social media, forums or blogs, are adressed within the MElt part-of-speech tagger via a three-step procedure (normalisation, tagging, de-normalisation with tag redistribution); this work is performed in relation with the CoMeRe project, funded by the Institut de Linguistique Française [14]; (ii) less noisy customer data, for preparing shallow semantic analysis; this work is performed in collaboration with the viavoo company [17].
- output of OCR systems, in the context of the PACTE project (see 6.3).

5.6. The Alexina framework: the Lefff syntactic lexicon, the Aleda entity database and other Alexina resources

Participants: Benoît Sagot [correspondant], Laurence Danlos.

See also the web page http://gforge.inria.fr/projects/alexina/.

Alexina is Alpage's Alexina framework for the acquisition and modeling of morphological and syntactic lexical information. The first and most advanced lexical resource developed in this framework is the Le*fff*, a morphological and syntactic lexicon for French.

Historically, the Le*fff* 1 was a freely available French morphological lexicon for verbs that has been automatically extracted from a very large corpus. Since version 2, the Le*fff* covers all grammatical categories (not just verbs) and includes syntactic information (such as subcategorization frames); Alpage's tools, including Alpage's parsers, rely on the Le*fff*. The version 3 of the Le*fff*, which has been released in 2008, improves the linguistic relevance and the interoperability with other lexical models.

Other Alexina lexicons exist, at various stages of development, in particular for Spanish (the Le*ff*e), Polish, Slovak, English, Galician, Persian, Kurdish, Italian, German, as well as for Latin verbs and a subset of Maltese and Khaling verbs. These lexicons are used in various tools, including instances of the MElt POS-tagger, and for studies in quantitative morphology.

Alexina also hosts *Aleda* [128], [118] a large-scale entity database currently developed for French but under development for English, Spanish and German, extracted automatically from Wikipedia and Geonames. It is used among others in the SxPipe processing chain and its NP named entity recognition, as well as in the Nomos named entity linking system.

5.7. The free French wordnet WOLF

Participants: Benoît Sagot [correspondant], Valérie Hanoka.

The WOLF (Wordnet Libre du Français) is a wordnet for French, i.e., a lexical semantic database. The development of WOLF started in 2008 [112], [113]. At this time, we focused on benefiting from available resources of three different types: general and domain-specific bilingual dictionaries, multilingual parallel corpora and Wiki resources (Wikipedia and Wiktionaries). This work was achieved in a large part in collaboration with Darja Fišer (University of Ljubljana, Slovenia), in parallel with the development of a free Slovene wordnet, sloWNet. However, it was also impacted by specific collaborations, e.g., on adverbial synsets [114].

In 2014, updated betas of the new version of the WOLF have been published (now version 1.0b4), which integrates and extends the various efforts performed and published somewhat independently in 2012, together with the result of additional filtering, both manual and semi-automatic.

The WOLF is freely available under the Cecill-C license. It has already been used in various experiments, within and outside Alpage.

5.8. OGRE (Optimized Graph Rewriting Engine)

Participants: Corentin Ribeyre [correspondant], Djamé Seddah, Éric Villemonte de La Clergerie, Marie-Hélène Candito.

OGRE (Optimized Graph Rewriting Engine) is a graph rewriting system specifically designed for manipulating linguistic trees and graphs [105]. It relies on a rule specification language for expressing graph rewriting patterns. The transformation is performed in two steps:

1. First, the system performs simple transformations following the rewriting patterns;
2. Second, constraints can be applied on edges, which applies transformations depending on their environment that are propagated while all constraints are satisfied.

The system has been designd for the analysis and manipulation of attributed oriented and multi-relational graphs.

Web site: http://www.corentinribeyre.fr/projects/view/OGRE

5.9. LexViz

Participants: Mikaël Morardo [maintainer], Éric Villemonte de La Clergerie.

In the context of the industrial collaboration of ALPAGE with the company Lingua & Machina, we have extended their WEB plateform Libellex with a new component used to visualize and collaboratively validate lexical resources. In particular, this extension is used to manage terminological lists and lexical networks. The implemented graph-based representation has proved to be intuitive and quite useful for navigating in such large lexical resources (on the order to 10K to 100K entries).

5.10. Mgwiki

Participants: Paul Bui Quang [maintainer], Éric Villemonte de La Clergerie.

In the context of Inria ADT Mgwiki, Paul Bui Quang has developed a linguistic wiki that may used to discuss linguistic phenomena with the possibility to add annotated illustrative sentences. The work is essentially devoted to the construction of an instance for documenting and discussing FRMG, with the annotations of the sentences automatically provided by parsing them with FRMG. This instance also offers the possibility to parse small corpora with FRMG and an interface of visualization of the results. Large parsed corpora (like French Wikipedia or Wikisource) are also available. The parsed corpora can also be queried through the use of the DPath language. The resulting wiki has been officially opened in 2014 on http://alpage.inria.fr/frmgwiki.

Another instance was deployed for managing the annotation guide for the Deep version of the Sequoia treebank, confirming the potential of the notion of linguistic wiki

The source code of the wiki is available on the GForge.

ALPINES Project-Team

5. New Software and Platforms

5.1. Platforms

5.1.1. FreeFem++, http://www.freefem.org/ff++/

FreeFem++ is a PDE solver based on a flexible language that allows a large number of problems to be expressed (elasticity, fluids, etc) with different finite element approximations on different meshes. There are more than 2000 users, and on the mailing list there are 430 members. Among those, we are aware of at least 10 industrial companies, 8 french companies and 2 non-french companies. It is used for teaching at Ecole Polytechnique, Ecole Centrale, Ecole des Ponts, Ecole des Mines, University Paris 11, University Paris Dauphine, La Rochelle, Nancy, Metz, Lyon, etc. Outside France, it is used for example at universities in Japan (Tokyo, Kyoto, Hiroshima, there is a userguide FreeFem++ in japan), Spain (Sevilla, BCAM, userguide available in spanish), UK (Oxford), Slovenia, Switzerland (EPFL, ETH), China. For every new version, there are 350 regression tests, and we provide a rapid correction of reported bugs. The licence of FreeFem++ is LGPL.

5.1.2. Library for preconditioned iterative methods

In the project-team we develop a library that integrates the direction preserving and low rank approximation preconditioners for both approached factorizations and domain decomposition like methods. It will be available through `FreeFem++` and also as a stand alone library, and we expect to have one version of this library available in 2014.

5.1.3. HPDDM, https://github.com/hpddm

HPDDM is an efficient implementation of various domain decomposition methods (DDM) such as one- and two-level Restricted Additive Schwarz methods, the Finite Element Tearing and Interconnecting (FETI) method, and the Balancing Domain Decomposition (BDD) method. These methods can be enhanced with deflation vectors computed automatically by the framework using methods developed by members of the team:

- Generalized Eigenvalue problems on the Overlap (GenEO), an approach first introduced in the PhD of Nicole Spillane.
- local Dirichlet-to-Neumann operators, an approach first introduced in a paper by Nataf et al. and recently revisited by Conen et al.

This code has been proven to be efficient for solving various elliptic problems such as scalar diffusion equations, the system of linear elasticity, but also frequency domain problems like the Helmholtz equation. A comparison with modern multigrid methods can be found in the thesis of Pierre Jolivet.

HPDDM is a header-only library written in C++11 with MPI and OpenMP for parallelism. While its interface relies on plain old data objects, it requires a modern C++ compiler: g++ 4.7.3 and above, clang++ 3.3 and above, icpc 15.0.0.090 and above. HPDDM has to be linked against BLAS and LAPACK (as found in OpenBLAS, in the Accelerate framework on OS X, in IBM ESSL, or in Intel MKL) as well as a direct solver like MUMPS, SuiteSparse, MKL PARDISO, or PaStiX. At compilation, just define before including HPDDM.hpp one of these preprocessor macros MUMPSSUB, SUITESPARSESUB, MKL_PARDISOSUB, or PASTIXSUB (resp. DMUMPS, DSUITESPARSE, DMKL_PARDISO, or DPASTIX) to use the corresponding solver inside each subdomain (resp. for the coarse operator). Additionally, an eigenvalue solver is recommended. There is an existing interface to ARPACK. Other (eigen)solvers can be easily added using the existing interfaces. For building robust two-level methods, an interface with a discretization kernel like FreeFem++ or Feel++ is also needed. It can then be used to provide, for example, elementary matrices, that the GenEO approach requires. As such HPDDM is not an algebraic solver, unless only looking at one-level methods. Note that for substructuring methods, this is more of a limitation of the mathematical approach than of HPDDM itself.

ANGE Project-Team

5. New Software and Platforms

5.1. FRESHKISS

Although the Saint-Venant system is the cornerstone of flow modelling in geosciences, this does not mean that the transfer of the efficient dedicated simulation tools is achieved in the geoscience community.

ANGE collaborates with scientists, laboratories and companies that are interested in scientific advances which makes the valuation and the transfer of results easier.

ANGE aims at developing robust and efficient numerical tools. For the simulation of the free surface Navier-Stokes equations, numerical tools have been developed namely FRESHKISS2D[0] and FRESHKISS3D. These tools are used by several scientists typically in the BIOCORE Inria project-team, at EDF and in public research laboratories.

FRESHKISS3D is a numerical code solving the 3D hydrostatic and incompressible Navier-Stokes equations with variable density. This code was initially dedicated to research activities within the team but we now aim at turning it into a numerical tool being used by non-mathematicians. Indeed, there is a demand in research laboratories and companies to use this tool. A young engineer (R. Hamouda) has been hired (ADT In@lgae funded by Inria) and its assignment is to improve/enrich the code and to make it user-friendly. Notice that FRESHKISS3D is used for teaching (master students in geosciences) at university Denis Diderot Paris 7 and IPGP.

5.2. TSUNAMATHS

TSUNAMATHS is an educational platform aiming at simulating historical tsunamis. Real data and mathematical explanations are provided to enable people to better understand the overall process of tsunamis. It is available on the Internet:

http://ange.raoufhamouda.com/tsunami/en_animation.htm

It was presented in the framework of the 2013 UNESCO year of "Mathematics of Planet Earth" and then exhibited at the ICM 2014 session (see § 9.3).

[0] FRESHKISS: FREe Surface Hydrodynamics using KInetic SchemeS

ANTIQUE Team

5. New Software and Platforms

5.1. The Apron Numerical Abstract Domain Library

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

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

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

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

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

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

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

5.2. The Astrée Static Analyzer of Synchronous Software

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

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

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

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

ASTRÉE discovers all runtime errors including:

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

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

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

ASTRÉE has achieved the following two unprecedented results:

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

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

5.3. The AstréeA Static Analyzer of Asynchronous Software

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

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

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

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

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

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

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

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

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

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

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

5.5. FuncTion: An Abstract Domain Functor for Termination

Participants: Caterina Urban, Antoine Miné [Correspondent].

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

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

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

5.6. HOO: Heap Abstraction for Open Objects

Participant: Arlen Cox [Correspondent].

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

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

5.7. The MemCADstatic analyzer

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

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

5.8. The OpenKappa Modeling Plateform

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

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

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

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

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

5.9. QUICr set abstract domain

Participant: Arlen Cox [Correspondent].

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

5.10. Translation Validation

Participant: Xavier Rival [correspondent].

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

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

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

5.11. Zarith

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

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

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

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

AOSTE Project-Team

5. New Software and Platforms

5.1. TimeSquare

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

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

TimeSquare offers five main functionalities:

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

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

5.2. K-Passa

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

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

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

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

5.3. SynDEx

Participants: Yves Sorel [correspondant], Meriem Zidouni.

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

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

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

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

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

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

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

5.4. Lopht

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

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

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

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

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

5.5. SAS

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

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

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

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

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

ARAMIS Project-Team

5. New Software and Platforms

5.1. SACHA

Participants: Marie Chupin [Correspondant], Ludovic Fillon.

SACHA ("Segmentation Automatisée Compétitive de l'Hippocampe et de l'Amygdale") is a software for the fully automatic segmentation of the hippocampus and the amygdala from MRI 3D T1 brain scans. It has been validated in various populations including healthy controls and patients with Alzheimer's disease, epilepsy and depression. It has been successfully applied to over 3,000 subjects, both controls, from adolescents to elderly subjects, and patients with different types of pathologies. The current stable version is fully automatic and focused on cross-sectional segmentation. The software can be used both as a command-line program or through a graphical user interface (GUI). The core of the program is coded in C++. It has a dependency to the AIMS library (http://www.brainvisa.info) and preprocessing steps rely on processes in Matlab from SPM (http://www.fil.ion.ucl.ac.uk/spm/). The GUI is coded in Python and is based on BrainVISA (http://www.brainvisa.info). The software has been registered at the APP (French agency for software protection).

5.2. WHASA

Participants: Marie Chupin [Correspondant], Ludovic Fillon, Thomas Samaille.

WHASA ("White matter Hyperintensity Automatic Segmentation Algorithm") is a software for the fully automatic segmentation of age-related white matter hyperintensities from MRI FLAIR and 3D T1 brain scans. It has been validated on a population showing a wide range of lesion load, and is being further evaluated on elderly subjects with few clinical abnormalisties and with different acquisition characteristics. The current stable version is fully automatic and focused on cross-sectional segmentation. The software can be used both as a Matlab command-line or through a graphical user interface (GUI). The core of the program is coded in Matlab. It has a dependency to the SPM environment (http://www.fil.ion.ucl.ac.uk/spm/). The GUI is coded in Python and is based on BrainVISA (http://www.brainvisa.info). The software has been registered at the APP (French agency for software protection).

5.3. Deformetrica

Participants: Stanley Durrleman [Correspondant], Alexandre Routier, Pietro Gori, Marcel Prastawa, Ana Fouquier, Joan Glaunès, Benjamin Charlier, Cedric Doucet.

Deformetrica is a software which estimates diffeomorphic deformations between sets of geometric objects in 2D and 3D. Those deformations are estimated either for the registration of two of such objects sets or for the construction of an atlas from several of such sets (a template model set and deformations mapping the template model to each set). Geometric objects could be grey-level images, surface meshes, polygonal lines or unstructured point sets. The method relies on the metric on currents for the comparison of point sets and the sum of squared differences for the comparison of images.

The software is written in C++ and relies on the ITK and VTK libraries. Core functions are coded with CUDA for their parallelization on GPU. It is a command-line software.

The version 2.1 of the software has been released on December 19, 2014. It is freely accessible to the scientific community at www.deformetrica.org.

5.4. qualiCATI

Participants: Marie Chupin [Correspondant], Hugo Dary, Urielle Thoprakarn, Amadou Tall, David Gay, Nicolas Vibet, Aude Costard, Cyril Poupon, Vincent Perlbarg, Mélanie Pélégrini-Issac, Alexandre Vignaud.

qualiCATI is a software designed for comprehensive quality control of multimodal MRI data acquisition in large multicentre clinical studies. The software is built as a platform receiving several modules, developped by several CATI engineers. The first module is dedicated to acquisition requirement checking and conversion to nifti format. The second module aims at making 3DT1 acquisition quality check more systematic, and relies both on visual inspection and quantitative indices. The third module allows a simultaneous evaluation of the clinical part of the CATI acquisition protocol. The fourth module embeds automatic indices to evaluate resting state fMRI acquisition. The fifth module is dedicated to first preprocessings and quality indices for dMRI. The sixth module is dedicated to qMRI, with visual and automated quality control together with preprocessings. The last module is dedicated to data and project management. QualiCATI requires training for the visual parts, and is closely linked with a team of clinical research assistants. It has been used to analyse about 5000 subjects from about 15 multi centre research projects initiated before or after the CATI started. Other modules will be added in the future to embed new aspects of the MRI protocol proposed by the CATI. The Aramis team is in charge of the second and third modules and jointly in charge of the first module. The software is centered on a graphical user interface (GUI). The whole program is coded in Python within the pyPTK environment developped by Cyril Poupon (Neurospin). It has dependencies to SPM (http://www.fil.ion.ucl.ac.uk/spm/) and brainVISA environments as well as specific tools for DICOM management.

5.5. Brain Networks Toolbox

Participants: Mario Chavez, Fabrizio de Vico Fallani [Correspondant].

Brain Networks Toolbox is a collection of Matlab routines developed to quantify topological metrics of complex brain networks. These routines are associated with published publications with application to real data and freely distributed via the FreeBorn (French Brain Networks) consortium https://sites.google.com/site/fr2eborn/download

CASCADE Project-Team (section vide)

CLASSIC Project-Team (section vide)

CLIME Project-Team

5. New Software and Platforms

5.1. Data assimilation library: Verdandi

Participants: Nicolas Claude, Vivien Mallet, Dominique Chapelle [M3DISIM], Philippe Moireau [M3DISIM].

The leading idea is to develop a data assimilation library intended to be generic, at least for high-dimensional systems. Data assimilation methods, developed and used by several teams at Inria, are generic enough to be coded independently of the system to which they are applied. Therefore these methods can be put together in a library aiming at:

- making easier the application of methods to a great number of problems,
- making the developments perennial and sharing them,
- improving the broadcast of data assimilation works.

An object-oriented language (C++) has been chosen for the core of the library. A high-level interface to Python is automatically built. The design study raised many questions, related to high dimensional scientific computing, the limits of the object contents and their interfaces. The chosen object-oriented design is mainly based on three class hierarchies: the methods, the observation managers and the models. Several base facilities have also been included, for message exchanges between the objects, output saves, logging capabilities, computing with sparse matrices.

In 2014, version 1.6 was released with a lot of new unit tests, within the Google Test framework. The extended Kalman filter now supports model error. For users of C++11, a native random perturbation manager has been added and allows to circumvent the use of Newran. The overall compatibility with Clang has been reinforced. The documentation was significantly improved, especially about the installation under Windows and Linux.

5.2. Image processing library: Heimdali

Participants: David Froger [SED], Dominique Béréziat, Isabelle Herlin.

The initial aim of the image processing library Heimdali was to replace an internal Inria library (named Inrimage) by a library based on standard and open source tools, and mostly dedicated to satellite acquisitions.

The leading idea of the library is to allow the following issues:

- making easier the sharing and development of image assimilation softwares. For that purpose, the installation is easily achieved with the package manager Conda.
- developing generic tools for image processing and assimilation based on ITK (Insight Segmentation and Registration Toolkit http://www.itk.org). In reverse providing tools to ITK and contribute to the ITK community. Our software corresponds to issues related to satellite acquisitions but could be of interest for processing medical image sequences.

The main components of Heimdali concern:

- the pre/post processing of image sequences,
- the image assimilation with numerical models,
- the visualization of image sequences.

In 2014, prototypes of the two first items have been defined. The development of the whole library should be available in 2015.

5.3. Polyphemus

Participants: Sylvain Doré, Vivien Mallet, Yelva Roustan [CEREA].

Polyphemus (see the web site http://cerea.enpc.fr/polyphemus/) is a modeling system for air quality. As such, it is designed to yield up-to-date simulations in a reliable framework: data assimilation, ensemble forecast and daily forecasts. Its completeness makes it suitable for use in many applications: photochemistry, aerosols, radionuclides, etc. It is able to handle simulations from local to continental scales, with several physical models. It is divided into three main parts:

- libraries that gather data processing tools (SeldonData), physical parameterizations (AtmoData) and post-processing abilities (AtmoPy);
- programs for physical pre-processing and chemistry-transport models (Polair3D, Castor, two Gaussian models, a Lagrangian model);
- model drivers and observation modules for model coupling, ensemble forecasting and data assimilation.

Fig. 1 depicts a typical result produced by Polyphemus.

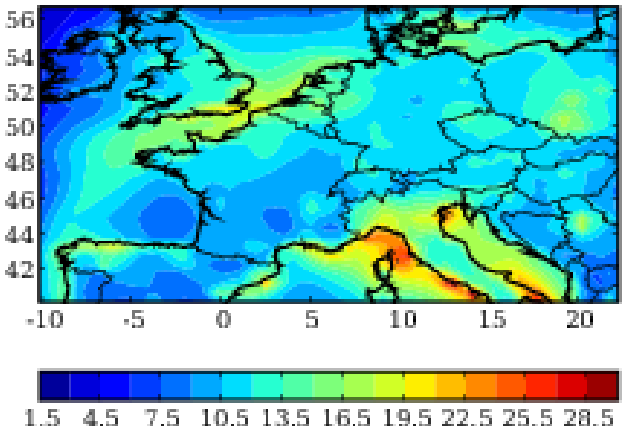

Figure 1. Map of the relative standard deviation (or spread, %) of an ensemble built with Polyphemus (ozone simulations, $\mu g\ m^{-3}$). The standard deviations are averaged over the summer of 2001. They provide an estimation of the simulation uncertainties.

Clime is involved in the overall design of the system and in the development of advanced methods in model coupling, data assimilation and uncertainty quantification (through model drivers and post-processing).

In 2014, Polyphemus was developed to better handle in-cloud and below-cloud scavenging. The interface of its Eulerian model, Polair3D, was extended to allow for detailed sensitivity analysis.

CRYPT Team (section vide)

DEDUCTEAM Exploratory Action

5. New Software and Platforms

5.1. Dedukti

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

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

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

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

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

5.2. Coqine, Holide, Focalide and Sigmaid

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

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

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

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

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

Focalide and Sigmaid have been developed by Raphaël Cauderlier.

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

5.3. iProver Modulo

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

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

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

5.4. Super Zenon and Zenon Modulo

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

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

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

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

5.5. Zipperposition (and extensions) and Logtk

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

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

5.6. CoLoR

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

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

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

5.7. HOT

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

5.8. Moca

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

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

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

Moca supports two kinds of relations:

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

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

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

5.9. Rainbow

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

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

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

5.10. mSAT

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

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

DYOGENE Project-Team

5. New Software and Platforms

5.1. Clones: CLOsed queueing Networks Exact Sampling

Clones is a Matlab toolbox for exact sampling of closed queueing networks.

Details can be found in [18] (best tool-paper award).

Available at: http://www.di.ens.fr/~rovetta/Clones/index.html

GALLIUM Project-Team

5. New Software and Platforms

5.1. OCaml

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

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

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

5.2. CompCert C

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

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

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

5.3. The diy tool suite

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

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

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

5.4. Zenon

Participant: Damien Doligez.

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

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

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

GAMMA3 Project-Team

3. New Software and Platforms

3.1. BLGEOL-V1 software

Participants: Patrick Laug [correspondant], Houman Borouchaki.

BLGEOL-V1 software can generate hex-dominant meshes of geologic structures complying with different geometric constraints: surface topography (valleys, reliefs, rivers), geologic layers and underground workings. First, a reference 2D domain is obtained by projecting all the line constraints into a horizontal plane. Different size specifications are given for rivers, outcrop lines and workings. Using an adaptive methodology, the size variation is bounded by a specified threshold in order to obtain a high quality quad-dominant mesh. Secondly, a hex-dominant mesh of the geological medium is generated by a vertical extrusion, taking into account the surfaces found (interfaces between two layers, top or bottom faces of underground workings). The generation of volume elements follows a global order established on the whole set of surfaces to ensure the conformity of the resulting mesh.

GANG Project-Team (section vide)

HIPERCOM2 Team

5. New Software and Platforms

5.1. OPERA and OCARI Software

Participants: Cédric Adjih, Ichrak Amdouni, Pascale Minet, Ridha Soua, Erwan Livolant.

The OPERA software was developed by the Hipercom2 team in the OCARI project (see https://ocari.org/). It includes EOLSR, an energy efficient routing protocol and OSERENA, a coloring algorithm optimized for dense wireless networks. It was registered by the APP. In 2013, OPERA has been made available for download as an open software from the InriaGForge site: https://gforge.inria.fr/scm/?group_id=4665

In 2014, OPERA has been ported on a more powerful platform based on the Atmel transceiver AT86RF233 and on a 32 bits microcontroler Cortex M3.

More details and documentation about this software are available in the website made by the Hipercom2 team: http://opera.gforge.inria.fr/index.html

Erwan Livolant, Pascale Minet from Inria as well as Tuan Dang from EDF and Maurice Sellin from DCNS showed the wireless sensor network OCARI during the Inria-Industry Meeting devoted to Telecommunications organized by Inria at Rocquencourt in November 2014. Two types of demonstration were done: one illustrating the internal behavior of OCARI and the other one illustrating a fire detection in a DCNS ship.

5.2. SAHARA Software

Participants: Erwan Livolant, Pascale Minet, Ridha Soua.

The software module DimTool developed in 2014 by the Hipercom2 team in the SAHARA project has been registered by the APP in January 2014. It helps to dimension the network parameters of the SAHARA wireless sensor network and evaluate the feasibility of a given application.

5.3. CONNEXION Software

Participants: Ines Khoufi, Pascale Minet, Erwan Livolant.

In 2014, we developped two softwares to compute the positions of:

- sensor nodes that ensure full coverage of a 2-D area with irregular shape and containing obstacles.
- relay nodes that maintain a robust connectivity of each point of interest with the sink. The area may contain obstacles.

With regard to the wireless sensor network OCARI, in 2014 we developed the interface of wireless temperature sensors PT100 with the OCARI stack. We also gave our support to CEA for the development of the OCARI interface of smoke detectors. With Telecom ParisTech, we interconnected OCARI with the industrial facility backbone based on OPC/UA via a gateway implemented on a Raspberry Pi. More precisely, we developed the serial interface between the OCARI network coordinator and the OPC/UA server.

5.4. NS3 Network Coding Software

Participants: Cédric Adjih, Ichrak Amdouni, Hana Baccouch.

One output of the GETRF project, was the creation of a solution for Network Coding, called DragonNet. DragonNet is a complete modular solution. This solution is responsible of: coding, decoding, maintaining necessary information and the associated signaling. It is designed to be extensible. A variant of DragonNet has been specified for wireless sensor networks and implemented.

As a follow-up to the ADT MOBSIM (and the previous module EyWifi), DragonNet was also integrated as a module for the ns-3 simulation tool.

5.5. FIT IoT-LAB Platforms

Participants: Cédric Adjih, Alaeddine Weslati, Vincent Ladeveze, Ichrak Amdouni.

This is a joint work with Emmanuel Baccelli from Inria Saclay.

Period: 2011 - 2021

Partners: Inria (Lille, Sophia-Antipolis, Grenoble), INSA, UPMC, Institut Télécom Paris, Institut Télécom Evry, LSIIT Strasbourg.

- Deployment: During the year 2014, the practical deployment has been finished, at the location planned for this testbed of the EQUIPEX FIT: the basement of building 1 at Rocquencourt. Ten external nodes have also been integrated.

  The testbed is offering most of the planned 344 open nodes, including 120 WSN430 nodes, 200 Cortex A8 based nodes, 24 Cortex M3.
- Opening: The official opening of the Rocquencourt was done in November 2014, at that point all IoT-LAB (and OneLab) users could run experiments in the M3 and A8 nodes from the site. See https://www.iot-lab.info/deployment/rocquencourt/ for more information.
- Support of external projects: Support for RIOT-OS and OpenWSN projects has been developed for IoT-Lab hardware and is being tested.
  - RIOT-OS, a joint work between Inria and FU-Berlin to create an Operating System for the Internet of Things.
  - OpenWSN, an open-source protocol stack for Internet of Things developed by UC Berkley.
- Demonstration: the project IoT-LAB in general had been demonstrated in several events during the year
  - March 2014: Description of IoT-LAB, and test of IoT-LAB nodes at the IETF 6TiSCH plugfest during IETF-89 in London https://bitbucket.org/6tisch/meetings/wiki/140306a_ietf89_london_plugfest
  - July 2014: Demonstration of IoT-LAB (Contiki RPL experiments) at:
    * the Bits-n-Bytes event of IETF-90 in Toronto http://www.ietf.org/meeting/90/ietf-90-bits-n-bites.html
    * the LLN plugfest of IETF-90 in Toronto https://bitbucket.org/6tisch/meetings/wiki/140720a_ietf90_toronto_plugfest
  - November 2014: Demonstration of IoT-LAB for use of scientific experiments, with network coding at MASS 2014 (http://mass2014.eecs.utk.edu/)

Moreover, during 2014, Ichrak Amdouni was in charge of:

- Testing the support of new switches in FIT IoT-LAB Paris-Rocquencourt site.
- Experimenting network coding protocols in the FIT IoT-LAB platform.

LIFEWARE Team

5. New Software and Platforms

5.1. The Biochemical Abstract Machine

Participants: François Fages, François-Marie Floch, Thierry Martinez, Sylvain Soliman, Pauline Traynard.

5.1.1. BIOCHAM V3.6

The Biochemical Abstract Machine (BIOCHAM) is a software environment for modeling and analyzing biochemical reaction systems, making simulations, performing static analyses, specifying behaviors in temporal logic. It is distributed under the GPL license since 2002.

The new features of version v3.6 released in October 2014 include:

- Hybrid boolean-stochastic-continuous models and simulations
- Quantitative temporal logic patterns with dedicated solvers (based on [20])
- Trace simplifications (based on [19])
- `export_sbml3` added
- `curate_model` and `curate_sbml` added (based on [8])
- Refinements of the types for command arguments (no effect on Biocham models)
- Bug fixes

5.1.2. BIOCHAM-web

BIOCHAM-WEB is a web service which makes it possible to try BIOCHAM on line without any installation, through a spreadsheet. A new version BIOCHAM-web was released in March 2014 and is kept evolving since.

That web service will evolve to a complete graphical user interface, named BIOCHAM-gui, and will replace the current graphical user interface.

5.1.3. BIOCHAM-parallel

A (non-distributed) parallel version of BIOCHAM is also maintained for our own use on a cluster of 10000 cores at GENCI CINES. This version speeds-up (linearly in the number of processors) the search of parameter values by parallelizing the evaluation of the fitness function (computed by numerical integration) for both the different parameter sets and the different conditions (perturbations, gene knock down or stress).

5.2. ClpZinc

Participants: François Fages, Thierry Martinez, Philippe Morignot, Sylvain Soliman.

CLP2ZINC is a rule-based modeling language for constraint programming. It extends the MiniZinc modeling language with Horn clauses which can be used to express search strategies as constraints in the model. This system is developed in the framework of the ANR Net-WMS-2 project and is a follow-up of the RULES2CP modeling language.

5.3. CellStar: Long-term tracking of single cells from brightfield microscopy images

Participants: Grégory Batt, Pascal Hersen, Artémis Llamosi, Szymon Stoma.

In close collaboration with Kirill Batmanov, Cédric Lhoussaine and Cristian Versari from the LIFL (CNRS/Lille Univ), we developed CELLSTAR, a tool-chain for image processing and analysis dedicated to segmentation and tracking of yeast cells in brightfield time-lapse microscopy movies. To estimate algorithm quality we developed a benchmark made of manually-verified images illustrating various situations. On this benchmark, CELLSTAR outperformed 5 other state-of-the-art methods. The tool-chain is implemented in MATLAB and is provided together with the Python YEAST IMAGE TOOLKIT benchmark tool.

5.4. Other software

The team also develops several software primarily for internal use. Some of them are specific to particular hardware and are not distributed. Some others are general purpose and currently on the web page for free downloading.

MAMBA Team

5. New Software and Platforms

5.1. Logiciels

5.1.1. TiQuant

Systems biology and medicine on histological scales require quantification of images from histological image modalities such as confocal laser scanning or bright field microscopy. The latter can be used to calibrate the initial state of a mathematical model, and to evaluate its explanatory value, which hitherto has been little recognized ([7]). We generated a software for image analysis of histological material and demonstrated its use in analysing liver confocal micrografts, called TiQuant (Tissue Quantifier) [10]. The software is part of an analysis chain detailing protocols of imaging, image processing and analysis in liver tissue, permitting 3D reconstructions of liver lobules down to a resolution of less than a micrometer. (This work is a collaboration with the group of JG Hengstler, IfADo, Germany)

5.1.2. TiSim

We advanced the complementary software TiSim (Tissue Simulator) that will soon be provided. TiSim permits agent-based simulations of multicellular systems and can be directly fed by processed image data from TiQuant.

MATHERIALS Team (section vide)

MATHRISK Project-Team

5. New Software and Platforms

5.1. PREMIA

Participants: Antonino Zanette, Mathrisk Research Team, Agnès Sulem [correspondant].

5.1.1. Premia: general description

Premia is a software designed for option pricing, hedging and financial model calibration. It is provided with it's C/C++ source code and an extensive scientific documentation. https://www-rocq.inria.fr/mathfi/Premia

The Premia project keeps track of the most recent advances in the field of computational finance in a well-documented way. It focuses on the implementation of numerical analysis techniques for both probabilistic and deterministic numerical methods. An important feature of the platform Premia is the detailed documentation which provides extended references in option pricing.

Premia is thus a powerful tool to assist Research & Development professional teams in their day-to-day duty. It is also a useful support for academics who wish to perform tests on new algorithms or pricing methods without starting from scratch.

Besides being a single entry point for accessible overviews and basic implementations of various numerical methods, the aim of the Premia project is:

1. to be a powerful testing platform for comparing different numerical methods between each other;
2. to build a link between professional financial teams and academic researchers;
3. to provide a useful teaching support for Master and PhD students in mathematical finance.

- AMS: 91B28;65Cxx;65Fxx;65Lxx;65Pxx
- License: Licence Propriétaire (genuine license for the Consortium Premia)
- Type of human computer interaction: Console, interface in Nsp, Web interface
- OS/Middelware: Linux, Mac OS X, Windows
- APP: The development of Premia started in 1999 and 15 are released up to now and registered at the APP agency. Premia 15 has been registered on 18/03/2013 and has the number IDDN.FR.001.190010.012.S.C.2001.000.31000
- Programming language: C/C++ librairie Gtk
- Documentation: the PNL library is interfaced via doxygen
- Size of the software: 280580 lines for the Src part only, that is 11 Mbyte of code, 130400 lines for PNL, 105 Mbyte of PDF files of documentation.
- interfaces : Nsp for Windows/Linux/Mac, Excel, binding Python, and a Web interface.
- Publications: [12], [58], [68], [79], [84], [40], [18].

5.1.2. Content of Premia

Premia contains various numerical algorithms (Finite-differences, trees and Monte-Carlo) for pricing vanilla and exotic options on equities, interest rate, credit and energy derivatives.

1. **Equity derivatives**

   The following models are considered:

   Black-Scholes model (up to dimension 10), stochastic volatility models (Hull-White, Heston, Fouque-Papanicolaou-Sircar), models with jumps (Merton, Kou, Tempered stable processes, Variance gamma, Normal inverse Gaussian), Bates model.

For high dimensional American options, Premia provides the most recent Monte-Carlo algorithms: Longstaff-Schwartz, Barraquand-Martineau, Tsitsklis-Van Roy, Broadie-Glassermann, quantization methods and Malliavin calculus based methods.

Dynamic Hedging for Black-Scholes and jump models is available.

Calibration algorithms for some models with jumps, local volatility and stochastic volatility are implemented.

2. **Interest rate derivatives**

   The following models are considered:

   HJM and Libor Market Models (LMM): affine models, Hull-White, CIR++, Black-Karasinsky, Squared-Gaussian, Li-Ritchken-Sankarasubramanian, Bhar-Chiarella, Jump diffusion LMM, Markov functional LMM, LMM with stochastic volatility.

   Premia provides a calibration toolbox for Libor Market model using a database of swaptions and caps implied volatilities.

3. **Credit derivatives: Credit default swaps (CDS), Collateralized debt obligations (CDO)**

   Reduced form models and copula models are considered.

   Premia provides a toolbox for pricing CDOs using the most recent algorithms (Hull-White, Laurent-Gregory, El Karoui-Jiao, Yang-Zhang, Schönbucher)

4. **Hybrid products**

   A PDE solver for pricing derivatives on hybrid products like options on inflation and interest or change rates is implemented.

5. **Energy derivatives: swing options**

   Mean reverting and jump models are considered.

   Premia provides a toolbox for pricing swing options using finite differences, Monte-Carlo Malliavin-based approach and quantization algorithms.

5.1.3. Premia design

Premia has managed to grow up over a period of fifteen years; this has been possible only because contributing an algorithm to Premia is subject to strict rules, which have become too stringent. To facilitate contributions, a standardized numerical library (PNL) has been developed by J. Lelong under the LGPL since 2009, which offers a wide variety of high level numerical methods for dealing with linear algebra, numerical integration, optimization, random number generators, Fourier and Laplace transforms, and much more. Everyone who wishes to contribute is encouraged to base its code on PNL and providing such a unified numerical library has considerably eased the development of new algorithms which have become over the releases more and more sophisticated. An effort is made to continue and stabilize the development of PNL. J. Ph Chancelier, B. Lapeyre and J. Lelong are using Premia and Nsp for Constructing a Risk Management Benchmark for Testing Parallel Architecture [18].

Development of the PNL in 2014 (J. Lelong) - Release 1.70. and 1.7.1 of the *PNL* library (http://pnl.gforge.inria.fr/).

1. Sampling from new distributions: non central Chi squared, Poisson, Bernoulli.
2. When using quasi Monte Carlo sequences, sampling from any distribution resorts to using the inverse of the cumulative distribution function technique.
3. A CMake module is provided to automatically detect the library when used by third party codes.
4. Add a sparse matrix object with advanced functionalities provided by Blas & Lapack. This new object is handled by the MPI binding.

5. Complete refactoring of the Basis object to considerably speedup the evaluation functions. Multivariate polynomials are represented as tensor products of one variate polynomials. The matrix holding the tensor product now uses a sparse storage which avoids many operations leading to a zero value thus leading to an impressive reduction the computational time.
6. All random number generators are thread–safe.

5.1.4. Algorithms implemented in Premia in 2014

Premia 16 has been delivered to the Consortium Premia in March 2014. In this release we have developed the Haar Wavelets-based approach for quantifying credit portfolio losses, Monte Carlo simulations of Credit Value Adjustment (CVA) using Malliavin techniques, asymptotic and exact pricing options on variance and importance sampling, and multilevel methods for jump models.

It contains the following new algorithms:

5.1.4.1. Commodities, Forex (FX), Insurance, Credit Risk

- Pricing and hedging gap risk. P. Tankov.
  Journal of Computational Finance. Volume 13 Number 3, Spring 2010.
- An Optimal Stochastic Control Framework for Determining the Cost of Hedging of Variable Annuities. P. A. Forsyth K.Vetzal
- Haar Wavelets-Based Approach for Quantifying Credit Portfolio Losses.
  J. J. Masdemont, L. O. Gracia. *Quantitative Finance, to appear*
- Cutting CVA's complexity. P. Henry-Labordère. *Risk Magazine 04 Jul 2012*
- Towards a coherent Monte Carlo simulation of CVA. L. Abbas Turki, A.Bouselmi, M.Mikou, *hal-00873200*
- Stochastic local intensity loss models with interacting particle system.
  A. Alfonsi, C. Labart, J. Lelong *Mathematical Finance, to appear*
- Repricing the Cross Smile: An Analytic Joint Density. P. Austing.

5.1.4.2. Equity Derivatives

- On the Fourier cosine series expansion (COS) method for stochastic control problems. R.F.T. Aalber, C.W. Oosterlee and M.J. Ruijter.
- A multifactor volatility Heston model. J.Da Fonseca M. Grasselli C. Tebaldi.
  Quant. Finance 8 (2008), no. 6, 591—604.
- General approximation schemes for option prices in stochastic volatility models. K.Larsson, *Quantitative Finance Volume 12, Issue 6, 2012*
- A robust tree method for pricing American options with the Cox-Ingersoll-Ross interest rate model. E. Appolloni, L. Caramellino and A. Zanette
  IMA Journal of Management Mathematics 2014, to appear.
- A hybrid tree-finite difference approach for the Heston and Bates model model. M. Briani, L. Caramellino and A. Zanette *Preprint* ArXiv 1307.7178
- A Closed-Form Exact Solution for Pricing Variance Swaps with Stochastic Volatility. S. Zhu and G. Lian, *Mathematical Finance, Volume 21, Issue 2, April 2011*
- Asymptotic and exact pricing options on variance. M.Keller-Ressel J.Muhle-Karbe.
  Finance & Stochastics, Volume 17 (2013), issue 1
- Importance sampling and Statistical Romberg Method for jump models.
  M.B. Alaya, A. Kebaier and K. Hajji
- Smart expansion and fast calibration for jump diffusions E. Benhamou, E. Gobet and M. Miri, *Finance Stochastics Volume 13, Number 4, September, 2009*
- Scaling and multiscaling in financial series: a simple model.
  Andreoli, A., Caravenna, F, Dai Pra, P. Posta, G., *Advances in Applied Probability. (2012), 44(4), 1018-1051.*
- Smooth convergence in the binomial model.
  L.B. Chang, K. Palmer.*Finance Stochastics Volume 11, Number 1,2007.*

MIMOVE Team

5. New Software and Platforms

5.1. Introduction

In order to validate our research results, our research activities encompass the development of related prototypes as surveyed below.

5.2. iCONNECT: Emergent Middleware Enablers

Participant: Valérie Issarny [correspondent].

As part of our research work on Emergent Middleware, we have implemented Enablers (or Enabler functionalities) that make part of the overall CONNECT architecture realizing Emergent Middleware in practice [4]. The focus of our work is on the: *Discovery enabler* that builds on our extensive background in the area of interoperable pervasive service discovery; and *Synthesis enabler* that synthesizes mediators that allow Networked Systems (NSs) that have compatible functionalities to interact despite mismatching interfaces and/or behaviors.

The Discovery Enabler is the component of the overall CONNECT architecture that handles discovery of networked systems, stores their descriptions (NS models), and performs an initial phase of matchmaking to determine which pairs of systems are likely to be able to interoperate. Such pairs are then passed to the Synthesis Enabler so that mediators can be generated. The Discovery Enabler is written in Java and implements several legacy discovery protocols including DPWS and UPnP.

The Synthesis Enabler assumes semantically-annotated system descriptions *à la* OWL-S, which are made available by the Discovery Enabler, together with a domain ontology, and produces the mediators that enable functionally compatible networked systems to interoperate. The semantically-annotated interfaces of the NSs that need to communicate are processed to compute the semantic mapping between their respective operations using a constraint solver. The resulting mapping serves generating a mediator that coordinates the behaviors of the NSs and guarantees their successful interaction. Only when the mediator includes all the details about the communication of NSs, can interoperability be achieved, which calls for the adequate concretization of synthesized mediators.

The *concretization of mediators* bridges the gap between the application level, which provides the abstraction necessary to reason about interoperability and synthesize mediators, and the middleware-level, which provides the techniques necessary to implement these mediators. Concretization entails the instantiation of the data structures expected by each NS and their delivery according to the interaction pattern defined by the middleware, based on which the NS is implemented. Therefore, we have been developing a mediation engine that, besides executing the data translations specified by the mediator, generates composed parsers and composers, which can process complex messages, by relying on existing libraries associated with standard protocols and state-of-the-art middleware solutions.

The data structures defined within OWL-S system descriptions, that is, the types of the inputs and outputs of the OWL-S atomic process were previously defined manually. As part of the prototype implementation that allows the mediator engine to generate composed parsers and composers, we made an extension that enables the inference of correct data-types [17], relieving developers from the time-consuming task of defining them. This prototype takes the form of a JAVA library that can optionally be used by the Synthesis Enabler whenever abstract data structures are unavailable. This library can be obtained from the iCONNECT GIT repository https://gforge.inria.fr/git/iconnect/iconnect.git (under the subproject `mtc`). While the underlying source code closely follows the formal mechanisms (such as tree automata) and algorithms presented in [17], we further concerned ourselves with making this library usable for non-expert developers by adhering to well-established standards. Specifically, data types, which are internally modeled as top-down tree automata, are transformed both on the input and output to RelaxNG (http://relaxng.org/) or XSD (www.w3.org/TR/xmlschema-1/).

The iCONNECT software has been released in the OW2 open source community (http://forge.ow2.org/projects/iconnect/), as part of FISSi, the Future Internet Software and Services initiative (http://www.ow2.org/view/Future_Internet/). OW2 and FISSi will give to this effort the required visibility in order to attract users and developers of the open source community.

5.3. XSB: eXtensible Service Bus for the Future Internet

Participant: Nikolaos Georgantas [correspondent].

The eXtensible Service Bus (XSB) is a development and runtime environment dedicated to complex distributed applications of the Future Internet. Such applications will be based, to a large extent, on the open integration of extremely heterogeneous systems, such as lightweight embedded systems (e.g., sensors, actuators and networks of them), mobile systems (e.g., smartphone applications), and resource-rich IT systems (e.g., systems hosted on enterprise servers and Cloud infrastructures). Such heterogeneous systems are supported by enabling middleware platforms, particularly for their interaction. With regard to middleware-supported interaction, the client-service (CS), publish-subscribe (PS), and tuple space (TS) paradigms are among the most widely employed ones, with numerous related middleware platforms, such as: Web Services, Java RMI for CS; JMS, SIENA for PS; and JavaSpaces, Lime for TS. XSB then provides support for the seamless integration of heterogeneous interaction paradigms (CS, PS and TS).

In a nutshell, our systematic interoperability approach implemented by the proposed XSB is carried out in two stages. First, a middleware platform is abstracted under a corresponding interaction paradigm among the three base ones, i.e., CS, PS and TS. To this aim, we have elicited a connector model for each paradigm, which comprehensively covers its essential semantics. Then, these three models are abstracted further into a single generic application (GA) connector model, which encompasses their common interaction semantics. Based on GA, we build abstract connector converters that enable interconnecting the base interaction paradigms.

Following the above, XSB is an abstract service bus that prescribes only the high-level semantics of the common bus protocol, which is the GA semantics. Furthermore, we provide an implementation of the XSB, building upon existing SOA and ESB realizations. XSB features richer interaction semantics than common ESBs to deal effectively with the increased Future Internet heterogeneity. Moreover, from its very conception, XSB incorporates special consideration for the cross-integration of heterogeneous interaction paradigms. Services relying on different interaction paradigms can be plugged into XSB by employing binding components (BCs) that adapt between their native middleware and the common bus protocol. This adaptation is based on the abstractions, and in particular on the conversion between the native middleware, the corresponding CS/PS/TS abstraction, and the GA abstraction.

Furthermore, we provide a companion implementation, named Light Service Bus (LSB), targeting the Internet of Things (IoT) domain. LSB forms a concrete access solution for IoT systems as it is able to cope with the diversity of the involved interaction protocols and take care of the specifics of IoT services, such as resource constraints, dynamic environments, data orientation, etc. It is implemented to be lightweight in nature and uses REST as the common protocol/bus in place of an ESB solution. In LSB, we confirm the wide use of the aforementioned interaction paradigms (CS/PS/TS) but also underline the existence of an additional paradigm focused on continuous interaction known as Streaming (STR).

Both the XSB and LSB solutions are available for download under open source licenses at http://xsb.inria.fr and http://websvn.ow2.org/listing.php?repname=choreos&path=%2Ftrunk%2Fextensible-service-access%2Flsb%2FLsbBindingComponents%2F respectively.

5.4. MobIoT: Service-oriented Middleware for the Mobile IoT

Participant: Valérie Issarny [correspondent].

MobIoT is a service-oriented middleware aimed at the mobile Internet of Things (IoT), which in particular deals with the ultra-large scale, heterogeneity and dynamics of the target networking environment. MobIoT offers novel probabilistic service discovery and composition approaches, and wraps legacy access protocols to be seamlessly executed by the middleware. The middleware exposes two levels of service abstractions: Thing as a service (on the service provider side); and Things measurements/actions as a service (on the service consumer side).

Key features of MobIoT lie in: (i) the exploitation of ontologies to overcome the heterogeneity of the Things network, (ii) the introduction of probabilistic approaches for both registering and retrieving networked things so as to filter out the ones that are redundant with already known alternatives, and finally, (iii) the exploitation of Thing services composition for responding to user queries asking information about the physical world so as to ease interaction with such a complex and dynamic networking environment.

MobIoT is implemented using Java and the Android platform, and consists of two complementary components: The MobIoT Mobile middleware and the MobIoT Web Service. The MobIoT Mobile middleware is deployed on mobile devices (e.g., smartphones, tablets, sensor devices). It wraps: (i) the Query component that enables the querying of the physical world, (ii) the Registration component that deals with the probabilistic registration of local sensors and actuators, (iii) the domain ontology that allows reasoning about the features of Things, and (iv) the Sensor Access component that enables the sensor data retrieval and exposure. The MobIoT Web Service wraps: (i) the Registry component that keeps tracks of the registered services, (ii) the probabilistic Lookup component that allows retrieving relevant services in a scalable way, and (iii) the Composition & Estimation component to answer queries over the physical world using available Thing services, and finally domain and devices ontologies.

The MobIoT middleware is available for download under an open source license at http://mobiot.ow2.org.

5.5. Srijan: Data-driven Macroprogramming for Sensor Networks

Participant: Animesh Pathak [correspondent].

Macroprogramming is an application development technique for wireless sensor networks (WSNs) where the developer specifies the behavior of the system, as opposed to that of the constituent nodes. As part of our work in this domain, we are working on *Srijan*, a toolkit that enables application development for WSNs in a graphical manner using data-driven macroprogramming.

It can be used in various stages of application development, *viz.,*

1. Specification of the application as a task graph,
2. Customization of the auto-generated source files with domain-specific imperative code,
3. Specification of the target system structure,
4. Compilation of the macroprogram into individual customized runtimes for each constituent node of the target system, and finally
5. Deployment of the auto generated node-level code in an over-the-air manner to the nodes in the target system.

The current implementation of *Srijan* targets both the Sun SPOT sensor nodes and larger nodes with J2SE. Most recently, *Srijan* also includes rudimentary support for incorporating Web services in the application being designed.

The software is released under open source license, and available as an Eclipse plug-in at http://code.google.com/p/srijan-toolkit/.

5.6. Dioptase and Spinel: Lightweight Streaming Middleware for the IoT

Participant: Valérie Issarny [correspondent].

Dioptase is a service-oriented middleware for developing stream-based applications which produce, process, store and consume data streams in complex environments such as the Internet of Things (IoT) and Wireless Sensor and Actuator Networks (WSAN). Dioptase leverages a novel service-oriented architecture for continuous processing, in order to deal with the large scale and the heterogeneity of the IoT. Once Dioptase is deployed onto a device, it enables developers to manage it as a generic pool of resources that can execute tasks provided over time. Those tasks are described as compositions of both standard continuous processing operators and customized computations written using a new lightweight stream processing language, called *DiSPL*.

The IoT infrastructure is composed of various devices (sensors, registries, proxies, clusters, etc.), and Dioptase is intended to be deployed onto all of them. To this end, Dioptase is highly modular and can be customized depending on the targeted devices and their roles in the IoT:

- *Dioptase core* is the base version of Dioptase that enables developers to (i) manage embedded sensors and actuators of a device through services and (ii) deploy tasks onto the device at any time.
- *Dioptase task mapper* is a server that implements our research on task mapping and automated deployment. Given a task graph, this server computes where to deploy each task according to the characteristics of tasks and available devices, and then manages the execution of the deployed tasks over time.
- *Dioptase proxy* is a pub/sub broker that enables interactions between Things that can not communicate directly, because of non-compatible networking interfaces/protocols or the use of address translation techniques.
- *Dioptase exchange* is a privacy proxy that manages the data and the services provided by a network of Things. It authenticates outsider Things and users, enabling them to request data streams or services according to access control and data-accuracy policies.

As part of the design of Dioptase, we have been investigating how to open legacy sensors to the future IoT. Toward this end, we propose to take advantage of the multi-modal connectivity as well as the mobility of smartphones, using them as mobile proxies that opportunistically discover close-by static sensors and act as intermediaries between them and the IoT. Spinel is a prototype of such an opportunistic proxy for mobile phones, which monitors the smartphone's mobility and further infers when to discover and register the sensors to an IoT discovery infrastructure, for instance the MobIoT registry(§ 5.4). Spinel collects data from the close-by sensors and pushes the collected data to an IoT stream processing infrastructure, for instance the Dioptase middleware. We will shortly release both Dioptase and Spinel under open source licenses.

5.7. Yarta: Middleware for supporting Mobile Social Applications

Participant: Animesh Pathak [correspondent].

With the increased prevalence of advanced mobile devices (the so-called "smart" phones), interest has grown in *Mobile Social Ecosystems* (MSEs), where users not only access traditional Web-based social networks using their mobile devices, but are also able to use the context information provided by these devices to further enrich their interactions. We are developing a middleware framework for managing mobile social ecosystems, having a multi-layer middleware architecture consisting of modules, which will provide the needed functionalities, including:

- Extraction of social ties from context (both physical and virtual),
- Enforcement of access control to protect social data from arbitrary access,
- A rich set of MSE management functionalities, which can be used to develop mobile social applications.

Our middleware adopts a graph-based model for representing social data, where nodes and arcs describe socially relevant entities and their connections. In particular, we exploit the Resource Description Framework (RDF), a basic Semantic Web standard language that allows representing and reasoning about social vocabulary, and creating an interconnected graph of socially relevant information from different sources.

The current implementation of the Yarta middleware targets both desktop/laptop nodes running Java 2 SE, as well as Android smart phones.

The software is released under open source license at https://gforge.inria.fr/projects/yarta/.

MOKAPLAN Team

5. New Software and Platforms

5.1. ALG2 for Monge Mean-Field Games, Monge problem and Variational problems under divergence constraint

5.1.1. Platforms

A generalisation of the ALG2 algorithm [53] corresponding to the paper [18] ha been implemented in FreeFem++. The scripts and numerical simulations are available at https://team.inria.fr/mokaplan/augmented-lagrangian-simulations/.

We still plan to implement a parallel version on Rocquencourt Inria cluster. We are waiting for FreeFem to be installed on the cluster.

5.2. Mokabajour

5.2.1. Platforms

Following the pioneering work of Caffarelli and Oliker [42], Wang [85] has shown that the inverse problem of freeforming a *convex* reflector which sends a prescribed source to a target intensity is a particular instance of Optimal Mass Transportation. The method developed in [7] has been used by researchers of TU Eindhoven in collaboration with Philips Lightning Labs to compute reflectors [81] in a simplified setting. The industrial motivation is the automatic design of reflector given prescribed source and target illuminance. From the mathematical point of view there is a hierarchy of Optimal Mass Transportation reflector and lenses problems and only the simplest "far field" one can be solved with state of the art Monge-Ampère solvers. We will adapt the Monge-Ampère solvers and also attempt to build real optimized reflector prototypes. We plan on investigating the more complicated near field models and design numerical methods. Finally Monge-Ampère based Optimal Mass Transportation solvers will be made available. This could be used for example in Mesh adaptation.

The web site is under construction https://project.inria.fr/mokabajour/, preliminary results are available.

This ADT (Simon Legrand) on the numerical free forming of specular reflectors started in december. We implement different types of MA solvers in collaboration with Quentin Mérigot (CEREMADE), Boris Thibert (LJK Grenoble) and Vincent Duval. See https://project.inria.fr/mokabajour/.

MUSE Team

5. New Software and Platforms

5.1. Fathom v2.0

Contributors: Anna-Kaisa Pietilainen, Stephane Archer

Available at: https://muse.inria.fr/fathom

Fathom [9] is a Firefox browser extension that explores the browser as a platform for network measurement and troubleshooting. It provides a wide range of networking primitives directly to in-page JavaScript including raw TCP/UDP sockets, higher-level protocol APIs such as DNS, HTTP, and UPnP, and ready-made functionality such as pings and traceroutes. Fathom v2.0 is a complete rewrite of the original Fathom using the new add-on SDK from Mozilla. In addition to javascript APIs, we have improved and added new built-in network measurement tools to Fathom such as 'Debug my Connection', 'Homenet Discovery' and 'Baseline Monitoring' that allow users to troubleshoot home network problems and share with us data for further research on home networks usage and diagnosis.

5.2. OpenWRT Packages for Network Measurements

Contributors: Anna-Kaisa Pietilainen, Sarthak Grover

Available at: https://github.com/apietila/browserlab

OpenWRT is a version of the Linux operating system to run on embedded devices, in particular, in home routers and access points. We have developed an OpenWRT package repository that provides fixes for and new ports of several existing network measurement tools (including iperf, iperf3, shaperprobe and pathload) for OpenWRT and an extended JSON RPC API for LuCI (OpenWRT control interface) to collaborate with the Fathom extension on home network diagnosis.

5.3. TagIt

Contributors: Sara El Aouad, Christophe Diot (Technicolor), Renata Teixeira

Available at: https://drive.google.com/file/d/0B-OcOkKOXok2b3lZYmt4QUw0cGM/view?usp=sharing
Video demo available at:
https://drive.google.com/file/d/0B-OcOkKOXok2VUxQR1NmRlg5VE0/view?usp=sharing

TagIt is an android app that makes it easy for users to enter movie reviews and to summarise the set of reviews of each movie. TagIt uses tags (or a short sequence of words) for entering user's opinions about a movie, so it is easy and quick for users to enter their feedback. TagIt also allows users to quickly see the opinion of other users about a movie with a tag cloud that summarises the set of reviews of a movie.

5.4. Where is the fault?

Contributors: Srikanth Sundaresan (Georgia Tech), Nick Feamster (Georgia Tech), Renata Teixeira

Where's The Fault? (WTF) is a system that localizes performance problems in home and access networks. We implement WTF as custom firmware that runs in an off-the-shelf home router. WTF uses timing and buffering information from passively monitored traffic at home routers to detect both access link and wireless network bottlenecks. We presented a demo of WTF at the ACM SIGCOMM conference in 2014. [4]

5.5. WeBrowse

Contributors: Giuseppe Scavo, Zied Ben Houidi (Alcatel-Lucent Bell Labs), Stefano Traverso (Politecnico di Torino), Marco Mellia (Politecnico di Torino), Renata Teixeira

Available at: http://tstat.polito.it/netcurator/

WeBrowse is the first passive crowdsource-based content curation system. Content curation is the act of assisting users to identify relevant and interesting content in the Internet. WeBrowse requires no active user engagement to promote content. Instead, it extracts the URLs users visit from traffic traversing an ISP network to identify popular content. WeBrowse contains a set of heuristics to identify the set of URLs users visit and to select the subset that are interesting to users [7]. The system proposes the interesting content in a web page available to all users.

5.6. UCN Data Collection

Contributors: Anna-Kaisa Pietilainen, Tom Logde (University of Nottingham), Richard Mortier (University of Nottingham), Peter Tolmie (University of Notthingham), Renata Teixeira

Available at: https://muse.inria.fr/ucn, code https://github.com/ucn-eu

The User-Centric Networking (UCN) project is seeking to understand how people consume various kinds of content when using computer networks. Within this project we are undertaking a detailed user study across a range of environments in order to understand the practices involved in consuming media and other content according to context. For the study, we have set up the following tools and software:

- **Registration and management website:** we have developed a website containing information about the experiment, and user and device registration interfaces.
- **VPN server and clients for network traffic data collection:** we are using OpenVPN open-source VPN server and available free clients on multiple platforms (OpenVPN for Linux, OpenVPN for Android, Tunnelblick for OS X, OpenVPN Connect for iOS) to collect network traffic traces from the participating devices. The VPN server is running on a secure Inria server, and we collect packet headers using tcpdump and http traffic logs with a Squid HTTP proxy. Collected data is stored on another server not directly accessible from the Internet.
- **Activity logging software:** we have developed a small Android application to log additional activity details such as list of running applications, foreground application, screen state, network connectivity details, and system resources (cpu, memory, network, battery) usage.
- **Data collection from Moves and Google Calendar:** we have written some code to import user data from Moves application and a Google Calendar based diary to add user location and daily activity logs to the data set.
- **Data visualisation:** the website contains a section to visualise all the collected data (network traffic as a function of location, time of day, activity) to support interviews with an ethnographer.

  We have obtained Ethics approval from Inria's COERLE for conducting the data collection and the user study and have done the CNIL declaration for this data collection. Our data collection and user study will start early 2015.

MUTANT Project-Team

5. New Software and Platforms

5.1. Antescofo

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

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

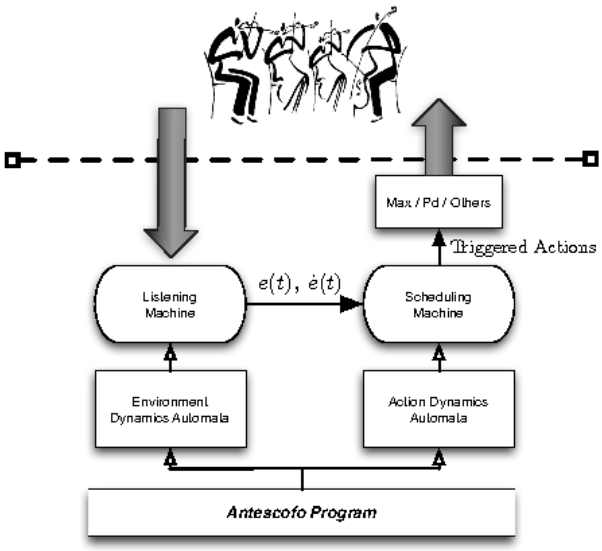

Figure 4. General scheme of Antescofo virtual machine

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

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

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

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

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

5.2. Ascograph: Antescofo Visual Editor

Participants: Thomas Coffy [ADT], Arshia Cont.

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

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

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

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

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

5.3. Antescofo Timed Test Platform

Participants: Florent Jacquemard, Clément Poncelet.

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

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

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

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

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

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

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

MYCENAE Project-Team

5. New Software and Platforms

5.1. Platforms

5.1.1. DynPeak

In collaboration with the SED (George Rosca) and Serge Steer (SISYPHE), we have deployed a web resource version of our algorithm for the detection of peaks in pulsatile hormone patterns, DynPeak, that is accessible at the https://dynpeak.inria.fr url.

PARKAS Project-Team

5. New Software and Platforms

5.1. Lucid Synchrone

Participant: Marc Pouzet [contact].

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

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

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

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

Three major release of the language has been done and the current version is V3 (dev. in 2006). As of 2014, the language is still used for teaching and in our research but we do not develop it anymore. Nonetheless, we have integrated several features from Lucid Synchrone in new research prototypes described below. The Heptagon language and compiler are a direct descendent of it. The new language Zélus for hybrid systems modeling borrows many features originaly introduced in Lucid Synchrone.

5.2. ReactiveML

Participant: Guillaume Baudart [contact].

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

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

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

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

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

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

5.3. Heptagon

Participants: Adrien Guatto, Marc Pouzet [contact].

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

With Cédric Pasteur, Léonard Gérard, and Brice Gelineau.

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

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

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

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

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

Participants: Albert Cohen, Adrien Guatto, Marc Pouzet.

With Louis Mandel (IBM Watson, USA).

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

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

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

5.5. ML-Sundials

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

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

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

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

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

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

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

5.6. Zélus

Participants: Timothy Bourke, Marc Pouzet [contact].

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

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

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

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

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

5.7. GCC

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

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

Licence: GPLv3+ and LGPLv3+

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

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

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

5.8. isl

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

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

Licence: MIT

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

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

5.9. ppcg

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

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

Licence: MIT

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

5.10. Tool support for the working semanticist

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

Languages, semantics, tool support, theorem prouvers.

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

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

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

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

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

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

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

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

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

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

5.11. Cmmtest: a tool for hunting concurrency compiler bugs

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

Languages, concurrency, memory models, C11/C++11, compiler, bugs.

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

1. compiles the program using the compiler and compiler optimisations that are being tested;
2. runs the compiled program in an instrumented execution environment that logs all memory accesses to global variables and synchronisations;
3. compares the recorded trace with a reference trace for the same program, checking if the recorded trace can be obtained from the reference trace by valid eliminations, reorderings and introductions.

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

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

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

PI.R2 Project-Team

4. New Software and Platforms

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

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

4.1.1. Version 8.5

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

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

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

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

4.1.2. Evaluation algorithms

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

4.1.3. Internal representation of projections

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

4.1.4. Universes

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

4.1.5. Internal architecture of the Coq software

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

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

4.1.6. Efficiency

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

4.1.7. Documentation generation

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

4.1.8. Maintenance and coordination

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

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

4.1.9. The Coq extraction

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

4.1.10. Parametricity for the Coq proof assistant

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

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

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

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

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

4.1.11. Formalisation in Coq

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

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

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

4.1.12. Systematic development of programs for parallel and cloud computing

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

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

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

4.1.13. Proofs of algorithms on graphs

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

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

4.2. Other software developments

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

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

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

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

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

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

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

POLSYS Project-Team

5. New Software and Platforms

5.1. FGb

Participant: Jean-Charles Faugère [contact].

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

5.2. GBLA

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

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

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

5.3. RAGlib

Participant: Mohab Safey El Din [contact].

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

5.4. Epsilon

Participant: Dongming Wang [contact].

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

5.5. SLV

Participant: Elias Tsigaridas [contact].

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

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

POMDAPI Project-Team

4. New Software and Platforms

4.1. FreeFem++

Participants: Martin Vohralík, Martin Čermák, Zuqi Tang.

The scientific calculation code FreeFem++ is an example of a complex software numerical simulation tool. It in particular encompasses all specifications of the problem, the choice and implementation of the numerical method, the choice and implementation of the linearization method (nonlinear solver), and the choice and implementation of the method of solution of the associated linear systems (linear solver). In the post-doc stays of M. Čermák and Z. Tang, we integrated there the most recent advances of the theory of a posteriori error estimation and of adaptive algorithms. In particular, adaptive stopping criteria for the linear and nonlinear solvers were implemented.

Version 3.33

Programming language: C++

http://www.freefem.org/ff++/

4.2. Oqla, Qpalm

Participants: Jean Charles Gilbert, Émilie Joannopoulos.

OQLA and QPALM aim at minimizing a large scale convex quadratic function on a polyhedron by an augmented Lagrangian method. The original features of the approach are its capacity to solve the problem without factorization, which makes them adapted to large scale problems, and to deal with unbounded and infeasible problems. In case the problem is infeasible, the codes solve the *closest feasible problem* with a global linear rate of convergence [3]. In case the problem is unbounded, the solvers build a feasible direction of unboundedness for the closest feasible problem. The solvers OQLA and QPALM only differ by their programming language; they are documented in [16], [14], [15].

Versions: 0.6 (OQLA), 0.5 (QPALM)

Programming languages: `C++` (OQLA), `Matlab` (QPALM)

4.3. Ref-image

Participants: Hend Ben Ameur, François Clément, Pierre Weis.

Ref-image is an image segmentation program using optimal control techniques. Slogan is “no gestalt inside”. Ref-image implements the refinement indicator algorithm, specialized to the case of the inversion of the identity map. It is a first step towards the implementation of a generic inversion platform using the refinement indicator algorithm.

Version: 1.1+pl0 (2014/02/28)

Programming language: OCaml

http://refinement.inria.fr/ref-image/

4.4. Ref-indic

Participants: Hend Ben Ameur, François Clément, Pierre Weis.

Ref-indic is an adaptive parameterization platform using refinement indicators. Slogan is “details only where they are worth it”. Ref-indic implements a generic version of the refinement indicator algorithm that can dock specific programs provided they conform to the generic algorithm API.

Version: 1.4+pl0 (2014/07/01)

Programming language: OCaml

http://refinement.inria.fr/ref-indic/

4.5. Sklml

Participants: François Clément, Pierre Weis.

Sklml is a functional parallel skeleton compiler and programming system for OCaml programs. Slogan is “easy coarse grain parallelization”.

Version: 1.1+pl0 (2014/01/21)

Programming language: OCaml

http://sklml.inria.fr/

PROSECCO Project-Team

5. New Software and Platforms

5.1. ProVerif

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

PROVERIF (proverif.inria.fr) is an automatic security protocol verifier in the symbolic model (so called Dolev-Yao model). In this model, cryptographic primitives are considered as black boxes. This protocol verifier is based on an abstract representation of the protocol by Horn clauses. Its main features are:

- It can handle many different cryptographic primitives, specified as rewrite rules or as equations.
- It can handle an unbounded number of sessions of the protocol (even in parallel) and an unbounded message space.

The PROVERIF verifier can prove the following properties:

- secrecy (the adversary cannot obtain the secret);
- authentication and more generally correspondence properties, of the form "if an event has been executed, then other events have been executed as well";
- strong secrecy (the adversary does not see the difference when the value of the secret changes);
- equivalences between processes that differ only by terms.

PROVERIF is widely used by the research community on the verification of security protocols (see http://proverif.inria.fr/proverif-users.html for references).

PROVERIF is freely available on the web, at proverif.inria.fr, under the GPL license.

5.2. CryptoVerif

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

CRYPTOVERIF(cryptoverif.inria.fr) is an automatic protocol prover sound in the computational model. In this model, messages are bitstrings and the adversary is a polynomial-time probabilistic Turing machine. CRYPTOVERIF can prove secrecy and correspondences, which include in particular authentication. It provides a generic mechanism for specifying the security assumptions on cryptographic primitives, which can handle in particular symmetric encryption, message authentication codes, public-key encryption, signatures, hash functions, and Diffie-Hellman key agreements.

The generated proofs are proofs by sequences of games, as used by cryptographers. These proofs are valid for a number of sessions polynomial in the security parameter, in the presence of an active adversary. CRYPTOVERIF can also evaluate the probability of success of an attack against the protocol as a function of the probability of breaking each cryptographic primitive and of the number of sessions (exact security).

CRYPTOVERIF has been used in particular for a study of Kerberos in the computational model, and as a back-end for verifying implementations of protocols in F# and C.

CRYPTOVERIF is freely available on the web, at cryptoverif.inria.fr, under the CeCILL license.

5.3. Cryptosense Analyzer

Participants: Graham Steel [correspondant], Romain Bardou.

See also the web page http://cryptosense.com.

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

In June 2013 we submitted a patent application (13 55374) on the reverse engineering process. We also concluded a license agreement between Inria PROSECCO and the nascent spin-off company Cryptosense to commercialize the tool.

5.4. miTLS

Participants: Karthikeyan Bhargavan [correspondant], Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Markulf Kohlweiss [Microsoft Research], Alfredo Pironti, Pierre-Yves Strub [IMDEA], Santiago Zanella-Béguelin [Microsoft Research], Jean Karim Zinzindohoue.

miTLS is a verified reference implementation of the TLS security protocol in F#, a dialect of OCaml for the .NET platform. It supports SSL version 3.0 and TLS versions 1.0-1.2 and interoperates with mainstream web browsers and servers. miTLS has been verified for functional correctness and cryptographic security using the refinement typechecker F7.

A paper describing the miTLS library was published at IEEE S&P 2013, and two updates to the software were released in 2013. The software and associated research materials are available from http://mitls.rocq.inria.fr.

5.5. WebSpi

Participants: Karthikeyan Bhargavan [correspondant], Chetan Bansal [Microsoft], Antoine Delignat-Lavaud, Sergio Maffeis [Imperial College London].

WebSpi is a library that aims to make it easy to develop models of web security mechanisms and protocols and verify them using ProVerif. It captures common modeling idioms (such as principals and dynamic compromise) and defines a customizable attacker model using a set of flags. It defines an attacker API that is designed to make it easy to extract concrete attacks from ProVerif counterexamples.

WebSpi has been used to analyze social sign-on and social sharing services offered by prominent social networks, such as Facebook, Twitter, and Google, on the basis of new open standards such as the OAuth 2.0 authorization protocol.

WebSpi has also been used to investigate the security of a number of cryptographi web applications, including password managers, cloud storage providers, an e-voting website and a conference management system.

WebSpi is under development and released as an open source library at http://prosecco.inria.fr/webspi/

5.6. Defensive JavaScript

Participants: Antoine Delignat-Lavaud [correspondant], Karthikeyan Bhargavan, Sergio Maffeis [Imperial College London].

Defensive JavaScript (DJS) is a subset of the JavaScript language that guarantees the behaviour of trusted scripts when loaded in an untrusted web page. Code in this subset runs independently of the rest of the JavaScript environment. When propertly wrapped, DJS code can run safely on untrusted pages and keep secrets such as decryption keys. DJS is especially useful to write security APIs that can be loaded in untrusted pages, for instance an OAuth library such as the one used by "Login with Facebook". It is also useful to write secure host-proof web applications, and more generally for cryptography that happens on the browser.

The DJS type checker and various libraries written in DJS are available from http://www.defensivejs.com.

5.7. F*

Participants: Nikhil Swamy [Microsoft Research], Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet [Microsoft Research], Catalin Hritcu, Chantal Keller, Aseem Rastogi, Pierre-Yves Strub.

F* is a new higher order, effectful programming language (like ML) designed with program verification in mind. Its type system is based on a core that resembles System Fω (hence the name), but is extended with dependent types, refined monadic effects, refinement types, and higher kinds. Together, these features allow expressing precise and compact specifications for programs, including functional correctness properties. The F* type-checker aims to prove that programs meet their specifications using an automated theorem prover (usually Z3) behind the scenes to discharge proof obligations. Programs written in F* can be translated to OCaml, F#, or JavaScript for execution.

A detailed description of F* (circa 2011) appeared in the Journal of Functional Programming [88]. F* has evolved substantially since then. The latest version of F* is written entirely in F*, and bootstraps in OCaml and F#. It is under active development at GitHub: https://github.com/FStarLang and the official webpage is at http://fstar-lang.org.

QUANTIC Team (section vide)

RAP Project-Team (section vide)

REGAL Project-Team

4. New Software and Platforms

4.1. NumaGiC

Participants: Lokesh Gidra, Marc Shapiro, Julien Sopena [correspondent], Gaël Thomas.

NumaGiC is a version of the HotSpot garbage collector (GC) adapted to many-core computers with very large main memories. In order to maximise GC throughput, it manages the trade-off between memory locality (local scans) and parallelism (work stealing) in a self-balancing manner. Furthemore, the collector features several memory placement heuristics that improve locality. NumaGiC is described in a paper accepted for publication at ASPLOS 2015 [29].

4.2. G-DUR

Participants: Masoud Saeida Ardekani, Dastagiri Reddy Malikireddy, Marc Shapiro [correspondent].

A large family of distributed transactional protocols have a common structure, called Deferred Update Replication (DUR). DUR provides dependability by replicating data, and performance by not re-executing transactions but only applying their updates. Protocols of the DUR family differ only in behaviors of few generic functions. Based on this insight, we offer a generic DUR middleware, called G-DUR, along with a library of finely-optimized plug-in implementations of the required behaviors. This paper presents the middleware, the plugins, and an extensive experimental evaluation in a geo-replicated environment. Our empirical study shows that:

1. G-DUR allows developers to implement various transactional protocols under 600 lines of code;
2. It provides a fair, apples-to-apples comparison between transactional protocols;
3. By replacing plugs-ins, developers can use G-DUR to understand bottlenecks in their protocols;
4. This in turn enables the improvement of existing protocols; and
5. Given a protocol, G-DUR helps evaluate the cost of ensuring various degrees of dependability.

G-DUR and the results of the comparison campaign are described in a paper to Middleware 2014 [33]. This research is supported in part by ConcoRDanT ANR project (Section 7.1.7) and by the FP7 grant SyncFree (Section 7.2.1.1).

Jessy is freely available on github under http://Github.com/msaeida/jessy under an Apache license.

4.3. SwiftCloud

Participants: Mahsa Najafzadeh, Marc Shapiro [correspondent], Serdar Tasiran, Marek Zawirski.

Client-side (e.g., mobile or in-browser) apps need local access to shared cloud data, but current technologies either do not provide fault-tolerant consistency guarantees, or do not scale to high numbers of unreliable and resource-poor clients, or both. Addressing this issue, the SwiftCloud distributed object database supports high numbers of client-side partial replicas. SwiftCloud offers fast reads and writes from a causally-consistent client-side cache. It is scalable, thanks to small and bounded metadata, and available, tolerating faults and intermittent connectivity by switching between data centres. The price to pay is a modest amount of staleness. A recent Inria Research Report (submitted for publication) presents the SwiftCloud algorithms, design, and experimental evaluation, which shows that client-side apps enjoy the same guarantees as a cloud data store, at a small cost.

SwiftCloud is supported by the ConcoRDanT ANR project (Section 7.1.7), by a Google Research Award, and by the FP7 grant SyncFree (Section 7.2.1.1).

The code is freely available on http://gforge.inria.fr/ under a BSD license.

4.4. Antidote

Participants: Tyler Crain, Marc Shapiro [correspondent], Serdar Tasiran, Alejandro Tomsic.

Antidote is the flexible cloud database platform currently under development in the SyncFree European project (Section 7.2.1.1). Antidote aims to be both a research platform for studying replication and consistency at the large scale, and an instrument for exploiting research results. The platform supports replication of CRDTs, in and between sharded (partitioned) data centres (DCs). The current stable version supports strong transactional consistency inside a DC, and causal transactional consistency between DCs. Ongoing research includes support for explicit consistency [23], for elastic version management, for adaptive replication, for partial replication, and for reconfigurable sharding.

REO Project-Team

5. New Software and Platforms

5.1. FELiScE

Participants: Grégory Arbia, Cédric Doucet, Miguel Ángel Fernández Varela, Justine Fouchet-Incaux, Benoit Fabrèges, Axel Fourmont, Jean-Frédéric Gerbeau [correspondant], Mikel Landajuela Larma, Damiano Lombardi, Elisa Schenone, Saverio Smaldone, Marina Vidrascu, Irène Vignon-Clementel, Vincent Martin.

FELiScE – standing for "Finite Elements for Life Sciences and Engineering" – is a finite element code which the M3DISYM and REO project-teams have decided to jointly develop in order to build up on their respective experiences concerning finite element simulations. One specific objective of this code is to provide in a unified software environment all the state-of-the-art tools needed to perform simulations of the complex respiratory and cardiovascular models considered in the two teams – namely involving fluid and solid mechanics, electrophysiology, and the various associated coupling phenomena. FELISCE is written in C++, and may be later released as an opensource library.

It was registered in July 2014 at the *Agence pour la Protection des Programmes* under the Inter Deposit Digital Number IDDN.FR.001.350015.000.S.P.2014.000.10000.

Gforge web site: https://gforge.inria.fr/projects/felisce/

5.2. LiFE-V library

Participants: Miguel Ángel Fernández Varela [correspondant], Jean-Frédéric Gerbeau.

LiFE-V [0] is a finite element library providing implementations of state of the art mathematical and numerical methods. It serves both as a research and production library. LiFE-V is the joint collaboration between three institutions: Ecole Polytechnique Fédérale de Lausanne (CMCS) in Switzerland, Politecnico di Milano (MOX) in Italy and Inria (REO) in France. It is a free software under LGPL license.

5.3. SHELDDON

Participant: Marina Vidrascu [correspondant].

SHELDDON (SHELls and structural Dynamics with DOmain decomposition in Nonlinear analysis) is a finite element library based on the Modulef package which contains shell elements, nonlinear procedures and PVM subroutines used in domain decomposition or coupling methods, in particular fluid-structure interaction.

Gforge web site: https://gforge.inria.fr/projects/shelddon

[0] http://www.lifev.org/

RITS Team

5. New Software and Platforms

5.1. New Software

The following software items have been submitted very recently to the APP; some already have IDDN references. Some of these are under improvement and other have already been transferred to industrial partners.

- **SODA**: This software has been developed in the context of the French ABV [0] project. This package contains the functions that are necessary to automate the vehicle navigation in its secured lane. This software has been purchased by a private partner (Valeo Group) aiming at developing its own automated vehicle.
- **MELOSYM**: this is the latest laser based Hierarchical ML-SLAM algorithm developed by RITS. It contains all the functions needed to perform the vehicle localization and the mapping of the environment. Windows compatible, it was initially developed under the ^{RT}MAPS platform but the version includes a standalone version. This software has been evaluated by a private partner aiming at developing its own automated vehicle for indoor applications. It is currently evaluated by another private partner aiming at implementing our solution on its outdoor automated shuttles.
- **STEREOLOC**: this is the package performing stereovision based localization and mapping. It performs semi-dense mapping of outdoor large environments and provides real-time estimates of the vehicle position. The software was tested and validated using ^{RT}MAPS like databases as well as the KITTI benchmark.

5.1.1. DOLAR

This software performs real-time obstacle detection and tracking using laser data scanned with one or several laser sensors with different geometric configurations. Obstacle detection is based on laser data segmentation while obstacle tracking uses PHD-based filtering techniques. The software is currently evaluated by a private partner aiming at implementing our solution on its outdoor automated unmanned vehicles.

5.1.2. FEMOT

FEMOT (Fuzzy Embedded MOTor) is an experimental motor for implementing fuzzy logic controllers, including all the fuzzy stages (fuzzification, inference, and defuzzification). This library has been compiled in Microsoft Visual (MVS) Studio and RTMaps. The proposed library is modular and adaptable to different situations and scenarios, especially for autonomous driving applications. FEMOT allows the development of the fuzzy rules to be written as sentences in an almost natural language. It allows the user to define variables and their fuzzy rules and to join them with other variables in rules to yield crisp signals for the controllers. The APP deposit was delivered May 2014. The Properties defined in FEMOT shows the number of inputs, outputs and fuzzy rules that the controller needs.
This software is used for the arbitration and control for fully automated functions. The behaviour of a human driver can be emulated with this technique. First simulations are showing promising results, and the library allows an easy adaptation in decision marking situations.

5.1.3. V2Provue

It is a software developed for the Vehicle-to-Pedestrian (V2P) communications, risk calculation, and alarming pedestrians of collision risk [34]. This software is made of an Android application dedicated to pedestrians and RtMaps modules for the vehicles.

[0] Automatisation Basse Vitesse

On the pedestrian side, the application is relying on GPS data to localize the user and Wi-Fi communications are used to receive messages about close vehicles and send information about the pedestrian positioning. Besides, a service has been developed to evaluate the collision risk with the vehicles near the pedestrian and an HMI based on OpenStreetMap displays all the useful information such as pedestrian and vehicles localization and, collision risk.

On the vehicle side, RtMaps modules allowing V2X communications have been developed. These modules contain features such as TCP/UDP socket transmissions, broadcast, multicast, unicast communications, routing, forwarding algorithms, and application specific modules. In the V2ProVu software, a particular application module has been implemented to create data packets containing information about the vehicle state (position, speed, yaw rate,...) and the V2X communication stack is used to broadcast these packets towards pedestrians. Moreover, the V2proVu application can also receive data from pedestrians and create objects structures that can be shared with the vehicle perception tools.

SECRET Project-Team

5. New Software and Platforms

5.1. New Software

5.1.1. CFS Implementation

Participants: Grégory Landais, Nicolas Sendrier.

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

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

5.1.2. Collision Decoding

Participants: Grégory Landais, Nicolas Sendrier.

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

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

SIERRA Project-Team

4. New Software and Platforms

4.1. SAG

Participant: Mark Schmidt [correspondent].

SAG: Minimizing Finite Sums with the Stochastic Average Gradient.

The SAG code contains C implements (via Matlab mex files) of the stochastic average gradient (SAG) method detailed below, as well as several related methods, for the problem of L2-regularized logistic regression with a finite training set.

The specific methods available in the package are: SGD: The stochastic gradient method with (user-supplied) step-sizes, (optional) projection step, and (optional) (weighted-)averaging. ASGD: A variant of the above code that supports less features, but efficiently implements uniform averaging on sparse data sets. PCD: A basic primal coordinate descent method with step sizes set according the (user-supplied) Lipschitz constants. DCA: A dual coordinate ascent method with a numerical high-accuracy line-search. SAG: The stochastic average gradient method with a (user-supplied) constant step size. SAGlineSearch: The stochastic average gradient method with the line-search described in the paper. SAG-LipschitzLS: The stochastic average gradient method with the line-search and adaptive non-uniform sampling strategy described in the paper.

4.2. fMRI

Participant: Fabian Pedregosa [correspondent].

We showed that HRF estimation improves sensitivity of fMRI encoding and decoding models and propose a new approach for the estimation of Hemodynamic Response Functions from fMRI data. This is an implementation of the methods described in the paper.

SISYPHE Project-Team

4. New Software and Platforms

4.1. The Cardiovascular Waves Analysis toolbox for Scilab

Participants: Lisa Guigue, Claire Médigue, Michel Sorine, Serge Steer.

The work about Heart Failure with preserved Ejection Fraction required the development of a set of tools for ECG signal manipulation and analysis. These tools, developed by Serge Steer, have been included in a Scilab toolbox named Cardiovascular Waves Analysis toolbox that will be available soon as a Scilab module. It extends the former Cardiovascular Toolbox and provides functions for:
- ECG reading multi channel ECG files in various formats (ISHNE, MIT, TMS32),
- Handling huge ECG files obtained through Holter devices,
- ECG pretreatment (filtering, subsampling, power line interference and base line wander removal),
- ECG events detection (P, Q, R, S, T) peaks, onset and end, based on former tools developed by Qinghua Zhang,
- Cardiovascular signals analysis using various approaches like frequency or time-frequency analysis, complex demodulation, non parametric, multilevel and multifractal methods,
- Specialized plotting facilities.

SMIS Project-Team

5. New Software and Platforms

5.1. Introduction

In our research domain, developing software prototypes is mandatory to validate research solutions and is an important vector for publications, demonstrations at conferences as well as for cooperations with industry. Our software strategy is also driven by our ambition to see our research results produce a real societal impact. To reach this goal, we integrate our prototypes in experiments in the field - notably in the healthcare domain and with scientists of other disciplines - and we recently set up educational platforms to raise students awareness of privacy protection problems and embedded programming.

This prototyping task is however difficult because it requires specialized hardware platforms, themselves sometimes at an early stage of development. For a decade, we have developed successive prototypes relying on different hardware platforms provided by Schlumberger then Gemalto, e.g., PicoDBMS a full-fledged DBMS embedded in a smart card [37] [26], Chip-Secured Data Access (C-SDA) a tamper-resistant mediator between a client and an untrusted server hosting encrypted data [32], Chip-Secured XML Access (C-SXA) an XML-based access rights controller, recipient of the e-gate open 2004 Silver Award and SIMagine 2005 Gold award [33]. Today, most of our software development efforts are organized around a unified platform named PlugDB and we are designing our own hardware platforms, that are produced by electronic SMEs. This opens up new research and experiment opportunities and we are engaged in an open-source/open hardware initiative to disseminate our results at a larger scale, both for scientific, educational and business purposes.

The next subsections detail the two prototypes we are focusing on today.

5.2. PlugDB engine

Participants: Nicolas Anciaux [correspondent], Luc Bouganim, Aydogan Ersoz, Quentin Lefebvre, Philippe Pucheral.

More than a stand-alone prototype, PlugDB is part of a complete architecture dedicated to a secure and ubiquitous management of personal data. PlugDB aims at providing an alternative to a systematic centralization of personal data. To meet this objective, the PlugDB architecture lies on a new kind of hardware device called Secure Portable Token (SPT). Roughly speaking, a SPT combines a smart-card and a micro-controller with a large external Flash memory (Gigabyte sized). The SPT can host data on Flash (e.g., a personal folder) and safely run code embedded in the micro-controller. PlugDB engine is the cornerstone of this embedded code. PlugDB engine manages the database on Flash (tackling the peculiarities of NAND Flash storage), enforces the access control policy defined on this database, protects the data at rest against piracy and tampering, executes queries (tackling low RAM constraint) and ensures transaction atomicity. Part of the on-board data can be replicated on a server (then synchronized) and shared among a restricted circle of trusted parties through crypto-protected interactions. PlugDB engine has been registered at APP (Agence de Protection des Programmes) in 2009 [27] and a new version is registered each year. PlugDB has been experimented in the field in the Yvelines District to implement a secure and portable medical-social folder helping the coordination of medical care and social services provided at home to dependent people. This field experiment is being audited by ARS-Ile de France (the Regional Healthcare Agency) and CG78 (General Council of Yvelines District), in order to envision the opportunity of a larger deployment. In parallel, we are improving the PlugDB prototype to overcome the limitations identified during the experiment. Notably, we have integrated a Bluetooth module to communicate in wireless with the token, a fingerprint module to authenticate users and a microphone to record voice messages. These are key elements in the perspective of a generalization. Link: https://project.inria.fr/plugdb/.

5.3. Eagle Tree

Participants: Matias Bjørling, Philippe Bonnet, Luc Bouganim, Niv Dayan [correspondent].

Solid State Drives (SSDs) are a moving target for system designers: they are black boxes, their internals are undocumented, and their performance characteristics vary across models. There is no appropriate analytical model and experimenting with commercial SSDs is cumbersome, as it requires a careful experimental methodology to ensure repeatability. Worse, performance results obtained on a given SSD cannot be generalized. Overall, it is impossible to explore how a given algorithm, say a hash join or LSM-tree insertions, leverages the intrinsic parallelism of a modern SSD, or how a slight change in the internals of an SSD would impact its overall performance. In this paper, we propose a new SSD simulation framework, named EagleTree, which addresses these problems, and enables a principled study of SSD-Based algorithms. EagleTree is an extensible, customizable SSD simulator designed to enable deep analyses of the interplay between the FTL, block management scheme, IO scheduling policy and application workload. It is able to generate visual illustrations of a host of performance metrics. EagleTree is available for Linux, and is licensed under GPL. EagleTree's git repository is : https://github.com/nivdayan/EagleTree.

TEMPO Team

5. New Software and Platforms

5.1. SimSoC

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

WHISPER Team

5. New Software and Platforms

5.1. Platforms

5.1.1. Coccinelle

Our recent research is in the area of code manipulation tools for C code, particularly targeting Linux kernel code. This work has led to the Coccinelle tool that we are continuing to develop. Coccinelle serves both as a basis for our future research and the foundation of our interaction with the Linux developer community.

The need to find patterns of code, and potentially to transform them, is pervasive in software development. Examples abound. When a bug is found, it is often fruitful to see whether the same pattern occurs elsewhere in the code. For example, the recent Heartbleed bug in OpenSSL partly involves the same fragment of code in two separate files. [0] Likewise, when the interface of an API function changes, all of the users of that function have to be updated to reflect the new usage requirements. This generalizes to the case of code modernization, in which a code base needs to be adapted to a new compiler, new libraries, or a new coding standards. Finding patterns of code is also useful in code understanding, *e.g.*, to find out whether a particular function is ever called with a particular lock held, and in software engineering research, *e.g.*, to understand the prevalence of various kinds of code structures, which may then be correlated with other properties of the software. For all of these tasks, there is a need for an easy to use tool that will allow developers to express patterns and transformations that are relevant to their source code, and to apply these patterns and transformations to the code efficiently and without disrupting the overall structure of the code base.

To meet these needs, we have developed the Coccinelle program matching and transformation tool for C code. Coccinelle has been under development for over 7 years, and is mature software, available in a number of Linux distributions (Ubuntu, Debian, Fedora, etc.). Coccinelle allows matching and transformation rules to be expressed in terms of fragments of C code, more precisely in the form of a *patch*, in which code to add and remove is highlighted by using + and −, respectively, in the leftmost column, and other, unannotated, code fragments may be provided to describe properties of the context. The C language is extended with a few operators, such as metavariables, for abstracting over subterms, and a notion of positions, which are useful for reporting bugs. The pattern matching rules can interspersed with rules written in Python or OCaml, for further expressiveness. The process of matching patterns against the source code furthermore takes into account some semantic information, such as the types of expressions and reachability in terms of a function's (intraprocedural) control-flow graph, and thus we refer to Coccinelle matching and transformation specifications as *semantic patches*.

Coccinelle was originally motivated by the goal of modernizing Linux 2.4 drivers for use with Linux 2.6, and was originally validated on a collection of 60 transformations that had been used in modernizing Linux 2.4 drivers [8]. Subsequent research involving Coccinelle included a formalization of the logic underlying its implementation [1] and a novel mechanism for identifying API usage protocols [50]. More recently, Coccinelle has served as a practical and flexible tool in a number of research projects that somehow involve code understanding or transformation. These include identifying misuses of named constants in Linux code [52], extracting critical sections into procedures to allow the implementation of a centralized locking service [58], generating a debugging interface for Linux driver developers [31], detecting resource release omission faults in Linux and other infrastructure software [68], and understanding the structure of device driver code in our current DrGene project [70].

[0] http://git.openssl.org/gitweb/?p=openssl.git;a=commitdiff;h=96db902

Throughout the development of Coccinelle, we have also emphasized contact with the developer community, particularly the developers of the Linux kernel. We submitted the first patches to the Linux kernel based on Coccinelle in 2007. Since then, over 2000 patches have been accepted into the Linux kernel based on the use of Coccinelle, including around 700 by around 90 developers from outside our research group. Over 40 semantic patches are available in the Linux kernel source code itself, with appropriate infrastructure for developers to apply these semantic patches to their code within the normal `make` process. Many of these semantic are also included in a 0-day build-testing system for Linux patches maintained by Intel. [0] Julia Lawall was invited to the Linux Kernel Summit as a core attendee (invitation only) in 2010 and 2014, and has been invited to the internal 2014 SUSE Labs Conference. She has also presented Coccinelle at developer events such as LinuxCon Europe, Kernel Recipes (Paris), FOSDEM (Brussels), and RTWLS, and has supervised a summer intern financed by the Linux Foundation, as part of the GNOME Foundation's Outreach Program for Women.

Finally, we are aware of several companies that use Coccinelle for modernizing code bases. These include Metaware in Paris, with whom we have had a 5-month contract in 2013-2014 for the customization and maintenance of Coccinelle. We hope to be able to organize other such contracts in the future.

5.1.2. Better Linux

Over the past few years, Julia Lawall and Gilles Muller have designed and developed of a number of tools such as Coccinelle, Diagnosys [31] [30] and Hector [68], to improve the process of developing and maintaining systems code. The BtrLinux action aims to increase the visibility of these tools, and to highlight Inria's potential contributions to the open source community. We will develop a web site `https://BtrLinux.inria.fr`, to centralize the dissemination of the tools, collect documentation, and collect results. This action is supported by Inria by the means of a young engineer (ADT), Quentin Lambert. In the case of Coccinelle, we will focus on enhancing its visibility and its dissemination, by using it to find and fix faults in Linux kernel code, and by submitting the resulting patches to the Linux maintainers. We now present the other tools considered in the BtrLinux action in more detail.

Diagnosys is a hybrid static and dynamic analysis tool that first collects information about Linux kernel APIs that may be misused, and then uses this information to generate wrapper functions that systematically log at runtime any API invocations or return values that may reflect such misuse. A developer can then use a specific make-like command to build an executable driver that transparently uses these wrapper functions. At runtime, the wrappers write log messages into a crash resilient region of memory that the developer can inspect after any crash. Diagnosys is complementary to Coccinelle in the kind of information that it provides to developers. While Coccinelle directly returns a report for every rule match across the code base, often including false positives that have to be manually isolated by the developer, Diagnosys only reports on conditions that occur in the actual execution of the code. Diagnosys thus produces less information, but the information produced is more relevant to the particular problem currently confronting the developer. As such, it is well suited to the case of initial code development, where the code is changing frequently, and the developer wants to debug a specific problem, rather than ensuring that the complete code base is fault free. Diagnosys is a complete functioning system, but it needs to be kept up to date with changes in the kernel API functions. As part of the BtrLinux action, we will regularly run the scripts that collect information about how to create the wrappers, and then validate and make public the results.

Hector addresses the problem of leaking resources in error-handling code. Releasing resources when they are no longer needed is critical, so that adequate resources remain available over the long execution periods characteristic of systems software. Indeed, when resource leaks accumulate, they can cause unexpected resource unavailability, and even single leaks can put the system into an inconsistent state that can cause crashes and open the door to possible attacks. Nevertheless, developers often forget to release resources, because doing so often does not make any direct contribution to a program's functionality. A major challenge in detecting resource-release omission faults is to know when resource release is required. Indeed, the C language does not provide any built-in support for resource management, and thus resource acquisition and release are typically implemented using ad hoc operations that are, at best, only known to core developers.

[0] E.g., http://comments.gmane.org/gmane.linux.kernel.kbuild/269

Previous work has focused on mining sequences of such functions that are used frequently across a code base, [43], [56] but these approaches have very high rates of false negatives and false positives. [53] We have proposed Hector, a static analysis tool that finds resource-release omission faults based on inconsistencies in the operations performed within a single function, rather than on usage frequency. This strategy allows Hector to have a low false positive rate, of 23% in our experiments, while still being able to find hundreds of faults in Linux and other systems.

Hector was developed as part of the PhD thesis of Suman Saha and was presented at DSN 2013, where it received the William C. Carter award for the best student paper. Hector is complementary to Coccinelle, in that it has a more restricted scope, focusing on only one type of fault, but it uses a more precise static analysis, tailored for this type of fault, to ensure a low false positive rate. Hector, like Coccinelle, is also complementary to Diagnosys, in that it exhaustively reports on faults in a code base, rather than only those relevant to a particular execution, and is thus better suited for use by experienced developers of relatively stable software. Over 70 patches have been accepted into Linux based on the results of Hector. The current implementation, however, is somewhat in a state of disarray. As part of the BtrLinux action, we will first return the code to working condition and then actively use it to find faults in Linux. Based on these results, we will either submit appropriate patches to the Linux developers or notify the relevant developer when the corresponding fix is not clear.

WILLOW Project-Team

5. New Software and Platforms

5.1. SPArse Modeling Software (SPAMS)

SPAMS v2.5 was released as open-source software in May 2014 (v1.0 was released in September 2009, v2.0 in November 2010). It is an optimization toolbox implementing algorithms to address various machine learning and signal processing problems involving

- Dictionary learning and matrix factorization (NMF, sparse PCA, ...)
- Solving sparse decomposition problems with LARS, coordinate descent, OMP, SOMP, proximal methods
- Solving structured sparse decomposition problems (ℓ_1/ℓ_2, ℓ_1/ℓ_∞, sparse group lasso, tree-structured regularization, structured sparsity with overlapping groups,...).

The software and its documentation are available at http://www.di.ens.fr/willow/SPAMS/.

5.2. Efficient video descriptors for action recognition

This package contains source code for highly-efficient extraction of local space-time video descriptors for action recognition. The accuracy of descriptors measured at standard benchmarks for action recognition is comparable to the state-of-the-art dense trajectory features, while being more than 100 times faster on standard CPU. The previous version of this code was evaluated in our recent work [12]. The package is available from http://www.di.ens.fr/~laptev/download/fastvideofeat-1.0.zip. Earlier version of our space-time video features is available at http://www.di.ens.fr/~laptev/download/stip-2.0-linux.zip.

5.3. Weakly Supervised Action Labeling in Videos Under Ordering Constraints

This is a package of Matlab code implementing weakly-supervised learning of actions from input videos and corresponding sequences of action labels. The code finds optimal alignment of action labels and video intervals during training. Along the optimization, the method trains corresponding action model. The package is available at http://www.di.ens.fr/willow/research/actionordering/. The method corresponding to this code package has been described and evaluated in Bojanowski *et al.* ECCV 2014 [10].

5.4. Visual Place Recognition with Repetitive Structures

Open-source release of the software package for visual localization in urban environments has been made publicly available in May 2014. The software package implements the method [A. Torii et al., CVPR 2013] for representing visual data containing repetitive structures (such as building facades or fences), which often occur in urban environments and present significant challenge for current image matching methods. This is an extended version that includes geometric verification. The original version was released in 2013. The software is available at http://www.di.ens.fr/willow/research/repttile/download/repttile_demo_ver03.zip.

5.5. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models

Open-source release of the software package for 2D-3D category-level alignment has been made publicly available. The software package implements newly developed method [9] for category-level recognition that not only outputs the bounding box of the object but predicts an approximate 3D model aligned with the input image. The software is available at http://www.di.ens.fr/willow/research/seeing3Dchairs/.

5.6. Painting-to-3D Model Alignment Via Discriminative Visual Elements

Open-source release of the software package for alignment of 3D models to historical and non-photographic depictions has been made publicly available. The software package implements the method of [9] for alignment of 3D models to input historical and non-photographic depictions such as paintings, drawings or engravings, where standard local feature-based method fail. The software is available at http://www.di.ens.fr/willow/research/painting_to_3d/.

5.7. Painting recognition from wearable cameras

Open-source release of the software package for painting recognition from wearable cameras has been made publicly available. The software implements a method described in [20] that recognizes 2D paintings on a wearable Google Glass device, for example, for a virtual museum guide application. The software runs directly on Google Glass without sending images to external servers for processing and recognizes a query painting in a database of 100 paintings in one second. The report and software are publicly available at http://www.di.ens.fr/willow/research/glasspainting/.

5.8. Learning and transferring mid-level image representations using convolutional neural networks

The first version of the open source software package for convolutional neural networks [13] has been released online. The software package is based on the cuda-convnet implementation of convolutional neural networks and includes a pre-trained convolutional neural network that can be applied to visual object classification as in the Pascal VOC 2012 set-up, where it achieves state-of-the-art single network results. The package also includes functions for visualization of object localization. The software is publicly available at http://www.di.ens.fr/willow/research/cnn/code/voc12-cvpr-reproduce.tar.