IN PARTNERSHIP WITH:
ENS Paris-Saclay

Activity Report 2017

Project-Team DEDUCTEAM

DEDUCTEAM

RESEARCH CENTER
Saclay - Île-de-France

THEME
Proofs and Verification

Table of contents

Project-Team DEDUCTEAM

Creation of the Team: 2011 December 01, updated into Project-Team: 2017 January 01

Keywords:

Computer Science and Digital Science:

A2.1.3. - Functional programming
A2.1.11. - Proof languages
A2.4.3. - Proofs
A3.1.1. - Modeling, representation
A7. - Theory of computation
A7.2. - Logic in Computer Science

Other Research Topics and Application Domains:

B7. - Transport and logistics

1. Personnel

Research Scientists

Gilles Dowek [Team leader, Inria, Senior Researcher, HDR]
Frédéric Blanqui [Inria, Researcher, HDR]

Faculty Members

Guillaume Burel [ENSIIE, Associate Professor, delegation Inria since Aug 2017]
Catherine Dubois [ENSIIE, Professor, HDR]
Olivier Hermant [Mines ParisTech, Associate Professor, HDR]

Post-Doctoral Fellows

Rodolphe Lepigre [Inria, from Sep 2017, until Dec 2018]
Simon Martiel [Univ. Paris-Est Créteil, until Mar 2017]

PhD Students

Guillaume Bury [ENS Paris]
Frédéric Gilbert [Ministère de l'Ecologie, de l'Energie, du Développement durable et de la Mer, until Aug 2017]
François Thiré [ENS Paris-Saclay]
Guillaume Genestier [ENS Paris-Saclay since Oct 2017, intern Ecole Polytechnique Apr-Aug 2017]
Gaspard Férey [Mines ParisTech since Sep 2017]

Administrative Assistants

Thida Iem [Inria until Jul 2017]
Emmanuelle Perrot [Inria since Jul 2017]

Visiting Scientists

Jean-Pierre Jouannaud [Prof. Emeritus, Univ. Paris-Saclay, Ecole Polytechnique, HDR]
Chaitanya Leena Subramaniam [Intern Ecole Polytechnique, Mar-Aug 2017]
Rafaël Bocquet [Intern ENS Paris Mar-Aug 2017]
Antoine Defourné [Intern ENSIMAG Feb-Aug 2017]

2. Overall Objectives

2.1. Objectives

The project-team investigates applications of proof theory to the design of logical frameworks, to interoperability between proof systems, and to the development of system-independent proof libraries.

To achieve these goals, we develop a logical framework DEDUKTI, where several theories can be expressed, systems translating proofs between external proof systems and DEDUKTI theories, tools to migrate proofs within DEDUKTI from one theory to another, tools to prove the confluence, the termination, and the consistency of theories expressed in DEDUKTI, and tools to develop proofs directly in DEDUKTI.

2.2. History

Deduction modulo theory is a formulation of predicate logic where deduction is performed modulo an equivalence relation defined on propositions. A typical example is the equivalence relation relating propositions differing only by a re-arrangement of brackets around additions, relating, for instance, the propositions $P((x + y) + z)$ and $P(x + (y + z))$. Reasoning modulo this equivalence relation permits to drop the associativity axiom. Thus, in Deduction modulo theory, a theory is formed with a set of axioms and an equivalence relation. When the set of axioms is empty the theory is called purely computational.

Deduction modulo theory was proposed at the end of the 20th century as a tool to simplify the completeness proof of equational resolution [6]. Soon, it was noticed that this idea was also present in other areas of logic, such as Martin-Löf's type theory, where the equivalence relation is definitional equality, Prawitz' extended natural deduction, etc. [2]. More generally, Deduction modulo theory gives an account on the way reasoning and computation are articulated in a formal proof, a topic slightly neglected by logic, but of prime importance when proofs are computerized.

The early research on Deduction modulo theory focused on the design of general proof search methods—Resolution modulo theory, tableaux modulo theory, etc.—that could be applied to any theory formulated in Deduction modulo theory, to general proof normalization and cut elimination results, to the definitions of models taking the difference between reasoning and computation into account, and to the definition of specific theories—simple type theory, arithmetic, some versions of set theory, etc.—as purely computational theories.

A new turn with Deduction modulo theory was taken when the idea of reasoning modulo an arbitrary equivalence relation was applied to typed λ-calculi with dependent types, that permits to express proofs as algorithms, using the Brouwer-Heyting-Kolmogorov interpretation and the Curry-de Bruijn-Howard correspondence [5]. It was shown in 2007, that extending the simplest λ-calculus with dependent types, the $\lambda\Pi$-calculus, with an equivalence relation (more precisely a congruence), led to a calculus, called the $\lambda\Pi$-calculus modulo theory, that permits to simulate many other λ-calculi, such as the Calculus of Constructions, designed to express proofs in specific theories.

This led to the development of a logical framework based on the $\lambda\Pi$-calculus modulo theory [29], that could be used to verify proofs coming from different proof systems, such as COQ [27], HOL [32], etc. To emphasize the versatility of this system, we called it DEDUKTI—"to deduce" in Esperanto. This system is currently developed together with companion systems, COQINE, KRAJONO, HOLIDE, FOCALIDE, and ZENONIDE, that permits to translate proofs from COQ, HOL, FOCALIZE, and ZENON, to DEDUKTI. Other tools, such as ZENON MODULO, directly output proofs that can be checked by DEDUKTI. DEDUKTI proofs can also be exported to several other systems. All this is presented in [1].

3. Research Program

3.1. Proof checking

A thesis, which is at the root of our research effort, and which was already formulated in [31], is that proof checkers should be theory independent. This is for instance expressed in the title of our invited talk at ICALP 2012: *A theory independent Curry-De Bruijn-Howard correspondence* [30]. Such a theory independent proof checker is called a logical framework.

Part of our research effort is focused on improving the $\lambda\Pi$-calculus modulo theory, for instance allowing to define congruences with associative and commutative rewriting.

Another part of our research effort is focused on the automatic analysis of theories to prove their confluence, termination, and consistency automatically [3].

3.2. Interoperability

Using a single prover to check proofs coming from different provers naturally leads to investigate how these proofs can interact one with another. This issue is of prime importance because developments in proof systems are getting bigger and, unlike other communities in computer science, the proof checking community has given little effort in the direction of standardization and interoperability.

For each proof, independently of the system in which it has been developed, we should be able to identify the systems in which it can be expressed. For instance, we have shown that many proofs developed in the MATITA prover did not use the full strength of the logic of MATITA and could be exported, for instance, to the systems of the HOL family, that are based on a weaker logic.

3.3. Libraries

Rather than importing proofs from one system, transforming them, and exporting them to another system, we can use the same tools to develop system-independant proof librairies. In such a library, each proof is labeled with the logics in which it can be expressed and so with the systems in which it can be used.

3.4. Interactive theorem proving

If our main goal with DEDUKTI is to import, transform, and export proofs developed in other systems, we want also, in some cases, to develop proofs interactively directly in DEDUKTI. This leads to the development of a tactic system, called DEMON, on top of DEDUKTI.

4. Application Domains

4.1. Safety of aerospace systems

In parallel with this effort in logic and in the development of proof checkers and automated theorem proving systems, we always have been interested in using such tools. One of our favorite application domain is the safety of aerospace systems. Together with César Muñoz' team in Nasa-Langley, we have proved the correctness of several geometric algorithms used in air traffic control.

This has led us sometimes to develop such algorithms ourselves, and sometimes to develop tools for automating these proofs.

4.2. Termination certificate verification

Termination is an important property to verify, especially in critical applications. Automated termination provers use more and more complex theoretical results and external tools (e.g. sophisticated SAT solvers) that make their results not fully trustable and very difficult to check. To overcome this problem, a language for termination certificates, called CPF, has been developed. Deducteam develops a formally certified tool, RAINBOW, based on the Coq library COLOR, that is able to automatically verify the correctness of some of these termination certificates.

5. New Software and Platforms

5.1. Autotheo

KEYWORD: Automated deduction
SCIENTIFIC DESCRIPTION: Transformation of axiomatic theories into rewriting systems that can be used by iProverModulo.
FUNCTIONAL DESCRIPTION: Autotheo is a tool that transforms axiomatic theories into polarized rewriting systems, thus making them usable in iProverModulo. It supports several strategies to orient the axioms, some of them being proved to be complete, in the sense that ordered polarized resolution modulo the resulting systems is refutationally complete, some others being merely heuristics. In practice, Autotheo takes a TPTP input file and produces an input file for iProverModulo.
NEWS OF THE YEAR: Used by iProverModulo in its participation at the CASC-26 competition.

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

5.2. CoLoR

Coq Library on Rewriting and termination
KEYWORDS: Coq - Formalisation
FUNCTIONAL DESCRIPTION: CoLoR is a Coq library on rewriting theory and termination. It provides many definitions and theorems on various mathematical structures (quasi-ordered sets, relations, ordered semi-rings, etc.), data structures (lists, vectors, matrices, polynomials, finite graphs), term structures (strings, first-order terms, lambda-terms, etc.), transformation techniques (dependency pairs, semantic labeling, etc.) and (non-)termination criteria (polynomial and matrix interpretations, recursive path ordering, computability closure, etc.).
NEWS OF THE YEAR: 2017: Port to Coq 8.6 and 8.7.

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

5.3. Coqine

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

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

5.4. Dedukti

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

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

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

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

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

5.5. Holide

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

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

5.6. HOT

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

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

5.7. iProver Modulo

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

NEWS OF THE YEAR: Participation at the automated-theorem-prover competition CASC-26 Integration of version 2.5 of iProver, adding support for types (TFF0)

- Participant: Guillaume Burel
- Partner: ENSIIE
- Contact: Guillaume Burel
- Publications: A Shallow Embedding of Resolution and Superposition Proofs into the ??-Calculus Modulo - Experimenting with deduction modulo
- URL: http://www.ensiie.fr/~guillaume.burel/blackandwhite_iProverModulo.html.en

5.8. mSAT

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

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

5.9. Rainbow

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

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

5.10. Krajono

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

- Contact: François Thiré

5.11. archsat

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

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

6. New Results

6.1. $\lambda\Pi$-calculus modulo theory

G. Dowek has given a semantic criterion for the termination of the $\lambda\Pi$-calculus modulo theory. This result has been published in [23].

A. Assaf, G. Dowek, J.-P. Jouannaud and J. Liu have given a confluence criterion for untyped higher-order rewrite systems, and demonstrated some applications to the $\lambda\Pi$-calculus modulo theory.

G. Dowek has given an invited talk at PxTP where he has presented a state of the art of the production of system-independent proof libraries. This paper has been published in the proceedings of PxTP [12].

6.2. Dedukti

During his internship [22], A. Defourné extended F. Blanqui's prototype of proof assistant based on Dedukti by developing a tactic for calling external provers through Why3 [28]. He also started to study a simple rewriting tactic.

During his internship, R. Bocquet studied unification in the $\lambda\Pi$-calculus modulo rewriting, and started to implement a prototype.

During his internship [24], G. Genestier studied the possibility to use the Size-Change Principle [34] in order to prove termination in the $\lambda\Pi$-calculus modulo rewriting. This work led to an adaptation of the criterion developped in his thesis by Wahlstedt [40] to a calculus containing dependant types. He also implemented a prototype of a weak version of the criterion.

During the first three months of his postdoc, R. Lepigre proposed a new implementation of Dedukti [36], based on the Bindlib library for the representation of structures with binders [38]. The libraries generated for Dedukti are compatible with this new implementation, and can be type-checked with minor modifications.

During the first months of his PhD, G. Férey adapted the higher-order pattern matching and convertibility checking algorithms to implemented support for rewriting modulo associative-commutative (AC) symbols in Dedukti.

6.3. Interoperability

F. Thiré has finished to implement a translation of an arithmetic library from Matita to OpenTheory. This work can be decomposed in two steps: A first step goes from Matita to a new logic called STTforall while a second step goes from STTforall to OpenTheory. This translation will be described in two separate papers. The first paper that will be submitted to FSCD 2018 describe the logic STTforall and its translation to HOL while the second paper explains the translation from Matita to STTforall. STTforall is a very simple logic and so, it is easy to translate proofs from this logic to other proofs assistants. For example, a translation from STTforall to Coq has also been implemented by F. Thiré. Two new tools have been implemented to make these translations:

- Dkmeta is a tool that translates terms thanks to the rewrite engine of Dedukti
- Ediloh is a tool that translates terms from STTforall them in OpenTheory

F.Gilbert developed a first prototype for the extraction of proofs from the proof assistant PVS that can be verified externally. The system PVS is based on the dichotomy between a *type-checker* and a *prover*. This proof extraction mechanism is built by instrumenting the PVS *prover*, but does not contain any typing information from the *type-checker* at this stage. Proofs can be built for any PVS theory. However, some reasoning steps rely on unverified assumptions. For a restricted fragment of PVS, the proofs are exported to Dedukti, and the unverified assumptions are proved externally using the automated theorem prover MetiTarski. This work has been published and presented in [15].

6.4. Termination

F. Blanqui revised his paper on “size-based termination of higher-order rewrite systems” submitted to the Journal of Functional Programming [19]. This paper provides a general and modular criterion for the termination of simply-typed λ-calculus extended with function symbols defined by user-defined rewrite rules. Following a work of Hughes, Pareto and Sabry for functions defined with a fixpoint operator and pattern-matching [33], several criteria use typing rules for bounding the height of arguments in function calls. In this paper, we extend this approach to rewriting-based function definitions and more general user-defined notions of size.

R. Lepigre worked on his paper “Practical Subtyping for System F with Sized (Co-)Induction” [39] (joint work with C. Raffalli), which was submitted to the journal Transactions on Programming Languages and Systems (TOPLAS) and is now under revision. This paper proposes a practical type system for a rich, normalizing, extension of (Curry-style) System F. The termination of recursive programs is established using a new mechanism based on circular proofs, which is also used to deal with (sized) inductive and coinductive types (in subtyping). The idea is to build (possibly ill-formed) infinite, circular typing (resp. subtyping) derivations, and to check for their well-foundedness a posteriori. The normalization proof then follows using standard realizability (or reducibility) techniques, the main point being that the adequacy lemma can still be proved by (well-founded) induction on the structure of the “circular” typing (resp. subtyping) derivations.

6.5. Proof theory

G. Burel developed a general framework, focusing with selection, of which various logical systems are instances: ordinary focusing, refinements of resolution, deduction modulo theory, superdeduction and beyond [20]. This strengthens links between sequent calculi and resolution methods.

F. Gilbert developed a constructivization algorithm, taking as input the classical proof of some formula and generating as output, whenever possible, a constructive proof of the same formula. This result has been published and presented in [14].

F. Gilbert submitted his PhD dissertation (work document [25]), centered on the extension of higher-order logic with predicate subtyping. Predicate subtyping is a key feature of the proof assistant PVS, allowing to define types from predicates – for instance, using this feature, the type of even numbers can be defined from the corresponding predicate. The core of this work is the definition of a language of verifiable certificates for predicate subtyping, as well as the proof of two properties of this language: a cut-elimination theorem, a theorem of conservativity over higher-order logic. F. Gilbert presented this language of certificates as well as the cut-elimination theorem at the workshop TYPES 2017.

6.6. Automated theorem proving

G. Bury presented the mSAT library at the OCaml workshop during the International Conference on Functional Programming [21]. This library provides an efficient SAT/SMT solver core written in OCaml, and presented as a functor to allow instantiation with different theories.

6.7. Program verification

R. Lepigre submitted a paper describing the PML_2 programming language and proof assistant [35], which was the main object of his recently defended PhD thesis [37].

6.8. Quantum computing

A. Díaz-Caro and G. Dowek have developed a type system for the λ-calculus that permits to distinguish duplicable terms from non duplicable ones. This work has been presented at Theory and Practice of Natural Computing [13].

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR PROGRAMme

This is an ANR for junior researcher Liesbeth Demol (CNRS, UMR 8163 STL, University Lille 3) to which G. Dowek participates. The subject is: "What is a program? Historical and Philosophical perspectives". This project aims at developing the first coherent analysis and pluralistic understanding of "program" and its implications to theory and practice.

7.2. International Initiatives

7.2.1. Participation in Other International Programs

7.2.1.1. International Initiatives

FoQCoSS

Title: Foundations of Quantum Computation: Syntax and Semantics

International Partners (Institution - Laboratory - Researcher):

Universidad Nacional de Quilmes (Argentina) - Alejandro Díaz-Caro

CNRS (France) - Simon Perdrix

Universidade Federal de Santa Maria (Brazil) - Juliana Kaizer Vizzotto

Duration: 2016 - 2017

Start year: 2016

The design of quantum programming languages involves the study of many characteristics of languages which can be seen as special cases of classical systems: parallelism, probabilistic systems, non-deterministic systems, type isomorphisms, etc. This project proposes to study some of these characteristics, which are involved in quantum programming languages, but also have a more immediate utility in the study of nowadays systems. In addition, from a more foundational point of view, we are interested in the implications of computer science principles for quantum physics. For example, the consequences of the Church-Turing thesis for Bell-like experiments: if some of the parties in a Bell-like experiment use a computer to decide which measurements to make, then the computational resources of an eavesdropper have to be limited in order to have a proper observation of non-locality. The final aim is to open a new direction in the search for a framework unifying computer science and quantum physics.

7.3. International Research Visitors

7.3.1. Visits of International Scientists

A. Díaz-Caro (Universidad Nacional de Quilmes, Argentina) visited Deducteam 3 weeks.

7.3.2. Visits to International Teams

7.3.2.1. Research Stays Abroad

F. Thiré has visited the Computation and Logic Group at McGill University for three months.

G. Dowek has visited the university of Quilmes in Buenos Aires for two weeks.

G. Dowek has visited the Pontifical University at Rio for three weeks.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Selection

8.1.1.1. Member of the Conference Program Committees

G. Burel was a member of the 12th International Workshop on the Implementation of Logics.

8.1.1.2. Reviewer

G. Burel, S. Martiel, and F. Gilbert rewiewed submissions to the Logic In Computer Science conference.

G. Burel reviewed submissions to the Formal Structures for Computation and Deduction conference.

8.1.2. Journal

8.1.2.1. Reviewer - Reviewing Activities

G. Burel reviewed articles for the Computer Journal and the Journal of Logic and Computation.

8.1.3. Invited Talks

G. Burel gave an invited lecture at the 28th Journées Francophones des Langages Applicatifs, entitled "Exprimer ses théories en Dedukti, le vérificateur de preuves universel".

8.1.4. Scientific Expertise

G. Dowek is a member of the scientific concil of La Société Inforamatique de France.

G. Dowek is a member of the scientific concil of La Main à la Pâte.

G. Dowek is a member of the Commission de réflexion sur l'Éthique de la Recherche en sciences et technologies du Numérique d'Allistene.

8.1.5. Research Administration

F. Blanqui is co-director of the pole 4 (programming: models, algorithms, languages and architectures) of Paris-Saclay University's doctoral school on computer science.

F. Blanqui is referent of LSV PhD students.

8.2. Teaching - Supervision - Juries

8.2.1. Teaching

F. Blanqui gave a M1 course (16h) on rewriting theory in the MPRI at the ENS Paris-Saclay.

F. Blanqui gave a M1 course (21h) on language theory at the ENSIIE.

G. Dowek's paper *Rules and derivations in an elementary logic course* has been published in the IfCoLog Journal of Logics and their Applications [11].

8.2.2. Supervision

F. Blanqui supervised the internships of A. Defourné and R. Bocquet.

F. Blanqui and O. Hermant supervised the internship of G. Genestier.

F. Blanqui and O. Hermant supervise the PhD of G. Genestier.

G. Dowek supervises the PhD of G. Férey and F. Gilbert.

G. Dowek and D. Delahaye supervise the PhD of G. Bury.

G. Dowek and S. Graham-Lengrand supervise the PhD of F. Thiré.

8.2.3. Juries

F. Blanqui was member of the jury for the best scientific production of the year within Paris-Saclay University's doctoral school on computer science.

F. Blanqui has been in the jury for the PhD of R. Lepigre on "Semantics and Implementation of an Extension of ML for Proving Programs", Chambéry.

9. Bibliography

Major publications by the team in recent years

[1] A. ASSAF, G. BUREL, R. CAUDERLIER, D. DELAHAYE, G. DOWEK, C. DUBOIS, F. GILBERT, P. HALMAGRAND, O. HERMANT, R. SAILLARD. *Expressing theories in the $\lambda\Pi$-calculus modulo theory and in the Dedukti system*, in "22nd International Conference on Types for Proofs and Programs, TYPES 2016", Novi SAd, Serbia, May 2016, https://hal-mines-paristech.archives-ouvertes.fr/hal-01441751

[2] F. BLANQUI. *Definitions by rewriting in the Calculus of Constructions*, in "Mathematical Structures in Computer Science", 2005, vol. 15, n^{o} 1, pp. 37-92 [*DOI :* 10.1017/S0960129504004426], http://hal.inria.fr/inria-00105648/en/

[3] F. BLANQUI, J.-P. JOUANNAUD, A. RUBIO. *The Computability Path Ordering*, in "Logical Methods in Computer Science", October 2015 [*DOI :* 10.2168/LMCS-11(4:3)2015], https://hal.inria.fr/hal-01163091

[4] G. BUREL. *Experimenting with Deduction Modulo*, in "CADE 2011", V. SOFRONIE-STOKKERMANS, N. BJØRNER (editors), Lecture Notes in Artificial Intelligence, Springer, 2011, vol. 6803, pp. 162–176

[5] D. COUSINEAU, G. DOWEK. *Embedding Pure Type Systems in the lambda-Pi-calculus modulo*, in "Typed lambda calculi and applications", S. RONCHI DELLA ROCCA (editor), Lecture Notes in Computer Science, Springer-Verlag, 2007, vol. 4583, pp. 102-117

[6] G. DOWEK, T. HARDIN, C. KIRCHNER. *Theorem proving modulo*, in "Journal of Automated Reasoning", 2003, vol. 31, pp. 33-73

[7] C. DUBOIS, T. HARDIN, V. DONZEAU-GOUGE. *Building certified components within FOCAL*, in "Revised Selected Papers from the Fifth Symposium on Trends in Functional Programming, TFP 2004, München, Germany, 25-26 November 2004", H.-W. LOIDL (editor), Trends in Functional Programming, Intellect, 2006, vol. 5, pp. 33-48

[8] O. HERMANT. *Resolution is Cut-Free*, in "Journal of Automated Reasoning", March 2010, vol. 44, n^{o} 3, pp. 245-276

[9] M. JACQUEL, K. BERKANI, D. DELAHAYE, C. DUBOIS. *Verifying B Proof Rules using Deep Embedding and Automated Theorem Proving*, in "Software and Systems Modeling (SoSyM)", June 2013

[10] M. JACQUEL, K. BERKANI, D. DELAHAYE, C. DUBOIS. *Tableaux Modulo Theories Using Superdeduction*, in "Global Journal of Advanced Software Engineering (GJASE)", December 2014, vol. 1, pp. 1 - 13 [*DOI :* 10.1007/978-3-642-31365-3_26], https://hal.archives-ouvertes.fr/hal-01099338

Publications of the year

Articles in International Peer-Reviewed Journals

[11] G. DOWEK. *Rules and derivations in an elementary logic course*, in "IfColog Journal of Logics and their Applications (FLAP)", January 2017, vol. 4, n^o 1, https://arxiv.org/abs/1601.01483 , https://hal.inria.fr/hal-01252124

Invited Conferences

[12] G. DOWEK. *Analyzing individual proofs as the basis of interoperability between proof systems*, in "PxTP 2017 - Fifth Workshop on Proof eXchange for Theorem Proving", Brasilia, Brazil, September 2017, https://hal.inria.fr/hal-01670394

International Conferences with Proceedings

[13] A. DÍAZ-CARO, G. DOWEK. *Typing Quantum Superpositions and Measurement*, in "TPNC 2017 - 6th International Conference on the Theory and Practice of Natural Computing", Prague, Czech Republic, C. MARTÍN-VIDE, R. NERUDA, M. A. VEGA-RODRÍGUEZ (editors), Lecture Notes in Computer Science, Springer, December 2017, vol. 10687, 13 p. , https://arxiv.org/abs/1601.04294 [*DOI :* 10.1007/978-3-319-71069-3_22], https://hal.inria.fr/hal-01670387

[14] F. GILBERT. *Automated Constructivization of Proofs*, in "FoSSaCS 2017", Uppsala, Sweden, April 2017 [*DOI :* 10.1007/978-3-662-54458-7_28], https://hal.inria.fr/hal-01516788

[15] F. GILBERT. *Proof certificates in PVS*, in "ITP 2017", Brasilia, Brazil, September 2017 [*DOI :* 10.1007/978-3-319-66107-0_17], https://hal.inria.fr/hal-01673517

[16] J.-P. JOUANNAUD, P.-Y. STRUB. *Coq without Type Casts: A Complete Proof of Coq Modulo Theory*, in "LPAR-21: 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning", Maun, Botswana, May 2017, https://hal.inria.fr/hal-01664457

Scientific Popularization

[17] S. ABITEBOUL, G. DOWEK. *Le temps des algorithmes*, Editions Le Pommier, 2017, 192 p. , https://hal.inria.fr/hal-01502505

Other Publications

[18] A. ASSAF, G. DOWEK, J.-P. JOUANNAUD, J. LIU. *Untyped Confluence In Dependent Type Theories: Full version*, April 2017, working paper or preprint, https://hal.inria.fr/hal-01515505

[19] F. BLANQUI. *Size-based termination of higher-order rewriting*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01424921

[20] G. BUREL. *Linking Focusing and Resolution with Selection*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01670476

[21] G. BURY. *mSAT:An OCaml SAT Solver*, September 2017, OCaml Users and Developers Workshop, Poster, https://hal.inria.fr/hal-01670765

[22] A. DEFOURNÉ. *Proof Tactics in Dedukti*, ENSIMAG, September 2017, https://hal.inria.fr/hal-01661872

[23] G. DOWEK. *Models and termination of proof reduction in the $\lambda\Pi$-calculus modulo theory*, April 2017, working paper or preprint, https://hal.inria.fr/hal-01101834

[24] G. GENESTIER. *Termination checking in the $\lambda\Pi$-calculus modulo theory.From a practical and a theoretical viewpoint*, Université Paris Diderot (Paris 7), September 2017, https://hal.inria.fr/hal-01676409

[25] F. GILBERT. *Extending higher-order logic with predicate subtyping: application to PVS*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01673518

[26] F. THIRÉ. *Exporting an Arithmetic Library from Dedukti to HOL*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01668250

References in notes

[27] Y. BERTOT, P. CASTÉRAN. *Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions*, Springer-Verlag, 2004

[28] F. BOBOT, J.-C. FILLIÂTRE, C. MARCHÉ, A. PASKEVICH. *Why3: Shepherd Your Herd of Provers*, in "First International Workshop on Intermediate Verification Languages", 2011, http://hal.inria.fr/hal-00790310

[29] M. BOESPFLUG. *Conception d'un noyau de vérification de preuves pour le lambda-Pi-calcul modulo*, École Polytechnique, 2011

[30] G. DOWEK. *A Theory Independent Curry-de Bruijn-howard Correspondence*, in "Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part II", Berlin, Heidelberg, ICALP'12, Springer-Verlag, 2012, pp. 13–15, http://dx.doi.org/10.1007/978-3-642-31585-5_2

[31] R. HARPER, F. HONSELL, G. PLOTKIN. *A Framework for Defining Logics*, in "Journal of the association for computing machinery", 1993, pp. 194–204

[32] J. HARRISON. *HOL Light: An Overview*, in "Theorem Proving in Higher Order Logics", S. BERGHOFER, T. NIPKOW, C. URBAN, M. WENZEL (editors), Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2009, vol. 5674, pp. 60-66, http://dx.doi.org/10.1007/978-3-642-03359-9_4

[33] J. HUGHES, L. PARETO, A. SABRY. *Proving the correctness of reactive systems using sized types*, in "Proceedings of the 23th ACM Symposium on Principles of Programming Languages", 1996, http://doi.org/10.1145/237721.240882

[34] C. S. LEE, N. D. JONES, A. M. BEN-AMRAM. *The size-change principle for program termination*, in "Proceedings of the 28th ACM Symposium on Principles of Programming Languages", 2001

[35] R. LEPIGRE. *PML$_2$, programming language with support for program proofs*, 2017, Submitted for the post-proceedings of TYPES, https://lepigre.fr/files/docs/lepigre2017_pml2.pdf

[36] R. LEPIGRE. *Lambdapi, implementation of the $\lambda\Pi$-calculus modulo rewriting*, 2017, https://github.com/rlepigre/lambdapi

[37] R. LEPIGRE. *Semantics and Implementation of an Extension of ML for program proving*, Université Grenoble Alpes, France, 2017, https://github.com/rlepigre/phd/blob/master/manuscript_archived.pdf

[38] R. LEPIGRE, C. RAFFALLI. *Bindlib, representation of binders in OCaml*, 2015, https://github.com/rlepigre/ocaml-bindlib

[39] R. LEPIGRE, C. RAFFALLI. *Practical Subtyping for System F with Sized (Co-)Induction*, 2017, Submitted to the TOPLAS journal (under revision), http://lepigre.fr/files/docs/subtyping2017.pdf

[40] D. WAHLSTEDT. *Dependent Type Theory with Parameterized First-Order Data Types and Well-Founded Recursion*, Chalmers University of Technology, 2007, ISBN 978-91-7291-979-2, http://www.cse.chalmers.se/alumni/davidw/wdt_phd_printed_version.pdf