RESEARCH CENTRE

Inria Saclay Centre at Université Paris-Saclay

IN PARTNERSHIP WITH:

Université Paris-Saclay

2024

ACTIVITY REPORT

Project-Team

DEDUCTEAM

DEDUCTEAM

IN COLLABORATION WITH: Laboratoire de Méthodes Formelles

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Proofs and Verification

Contents

Project-Team DEDUCTEAM

Creation of the Project-Team: 2017 January 01

Keywords

Computer sciences and digital sciences

A2.1.4. – Functional programming

A2.1.11. – Proof languages

A2.4.3. – Proofs

A7. – Theory of computation

A7.2. – Logic in Computer Science

Other research topics and application domains

B6.1.1. – Software engineering

B9.5.1. – Computer science

B9.5.2. – Mathematics

B9.7. – Knowledge dissemination

B9.7.1. – Open access

B9.7.2. – Open data

B9.8. – Reproducibility

1 Team members, visitors, external collaborators

Research Scientists

- Frederic Blanqui [Team leader, INRIA, Senior Researcher, from Jun 2024, HDR]
- Gilles Dowek [Team leader, INRIA, Senior Researcher, until May 2024, HDR]
- Bruno Barras [INRIA, Researcher]
- Frederic Blanqui [INRIA, Senior Researcher, until May 2024, HDR]
- Valentin Blot [INRIA, Researcher, until Aug 2024]
- Anthony Bordg [INRIA, Advanced Research Position, until May 2024]
- Gilles Dowek [INRIA, Senior Researcher, from Jun 2024, HDR]
- Theo Winterhalter [INRIA, Researcher]

Faculty Member

- Catherine Dubois [ENSIIE, Professor, on delegation Inria, HDR]

Post-Doctoral Fellows

- Ciarán Dunne [INRIA, Post-Doctoral Fellow, from Apr 2024]
- Claude Stolze-Hubert [INRIA, Post-Doctoral Fellow, until Mar 2024]

PhD Students

- Luc Chabassier [ENS PARIS, until Sep 2024]
- Louise Dubois De Prisque [INRIA, until Aug 2024]
- Thiago Felicissimo Cesar [UNIV PARIS SACLAY, until Sep 2024]
- Yoan Geran [Ecole des Mines de Paris, until Sep 2024]
- Nicolas Margulies [ENS PARIS-SACLAY]
- Melanie Taprogge [UNIV PARIS SACLAY, from Oct 2024]
- Thomas Traversie [CENTRALESUPELEC]
- Rishikesh Hirendu Vaishnav [INRIA]

Technical Staff

- Abdelghani Alidra [INRIA, Engineer]

Interns and Apprentices

- Ewen Broudin-Caradec [ENS PARIS-SACLAY, Intern, from Mar 2024 until Jul 2024]
- Thomas Laure [INRIA, Intern, from Mar 2024 until Aug 2024]
- Amal Makni [INRIA, Intern, from Jun 2024 until Aug 2024]
- Salwa Tabet Gonzalez [INRIA, Intern, from May 2024 until Sep 2024]

Administrative Assistant

- Aissatou-Sadio Diallo [INRIA]

External Collaborators

- Guillaume Burel [ENSIIE]
- Olivier Hermant [ENSMP, HDR]
- Jean-Pierre Jouannaud [Université Paris Saclay, Emeritus, HDR]
- Chantal Keller [IUT Orsay]

2 Overall objectives

2.1 Objectives

Deducteam investigates the design of logical frameworks, that is frameworks where various theories can be defined, and the use of such frameworks for interoperability between proof systems, cross verification of proofs, and the sustainability of proof libraries.

To achieve these goals, we develop

- a logical framework DEDUKTI, where various theories can be expressed,
- several implementations of this framework: DKCHECK, (formerly also called DEDUKTI), that is a small trust base, theory independent, proof-checker, LAMBDAPI, that is a system to develop DEDUKTI proofs interactively, and KONTROLI that is a fast parallel proof-checker for DEDUKTI,
- tools to import proofs developed in external proof systems to DEDUKTI theories,
- tools to translate proofs from one DEDUKTI theory to another,
- tools to export proofs expressed in DEDUKTI theories to an external proof system,
- tools to prove the confluence, the termination, and the consistency of theories expressed in DEDUKTI,
- libraries NUBO and LOGIPEDIA of proofs expressed in various DEDUKTI theories.

2.2 History

The development of computerized proof systems such as COQ, HOL LIGHT, or PVS is a major step forward in the quest of mathematical rigor. But it jeopardizes, once again, the universality of mathematical truth: we used to have proofs of Fermat's little theorem, we now have COQ proofs of Fermat's little theorem, HOL LIGHT proofs of Fermat's little theorem, PVS proofs of Fermat's little theorem, etc., as each proof system defines its own language for mathematical statements and its own truth conditions for these statements. See, for instance, our invited talk at IJCAR 2022: *From the Universality of Mathematical Truth to the Interoperability of Proof Systems*.

One way to address this issue is to express the theories implemented in these systems in a common logical framework and to determine, for each proof, which axioms it depends on. This way, a proof can be used in any system that supports these axioms, independently of the system it has been developed in.

The idea that systems such as Euclidean geometry, non-Euclidean geometries, set theory, with or without the axiom of choice, etc. should be expressed in the same logical framework appeared, in 1928, with the design of the first logical framework in the history of logic: predicate logic. Later, several more powerful logical frameworks have been designed: λ-Prolog, Isabelle, the Edinburgh logical framework, Pure type systems, Deduction modulo theory, etc.

The logical framework that we use is a simple λ-calculus with dependent types and rewrite rules, called the $\lambda\Pi$-calculus modulo theory, or the Martin-Löf logical framework. It generalizes all the mentioned frameworks. Its concrete syntax is the language DEDUKTI.

The first implementation of DEDUKTI, now called DKCHECK, was developed in 2011 by Mathieu Boespflug [39]. Then, new versions of this implementation were developed and several theories were expressed in DEDUKTI, allowing to import proofs developed in MATITA (with the tool KRAJONO), HOL LIGHT (with the tool HOLIDE), FOCALIZE (with the tool FOCALIDE), IPROVER, and ZENON, totalizing several hundred of megabytes of proofs.

We now focus on the translation of proofs from one DEDUKTI theory to another and on the exporting of proofs to other proof systems. In particular the MATITA arithmetic library has been translated to a much weaker theory: constructive simple type theory, allowing to export it to COQ, LEAN, PVS, HOL LIGHT, and ISABELLE/HOL. In the same way, the first book of Euclid's elements, formalized in COQ, has been translated to predicate logic and exported to several systems, and a proof of Bertrand's theorem, originally developed in MATITA, has been translated to predicative type theory, allowing its export to AGDA.

This led us to develop an on-line proof repository NUBO and an on-line encyclopedia LOGIPEDIA, allowing to share and browse this library.

We also focus on the development of new theories in DEDUKTI, such as Simple type theory with predicate subtyping, implemented in the system PVS, several formulations of homotopy type theory, various formulations of set theory, in particular those used in B and TLA+, matching logic, etc.

Finally, we develop an interactive theorem prover LAMBDAPI for DEDUKTI. This interactive theorem prover is also used as a tool in the process of translating proofs from PVS and from automated theorem provers.

3 Research program

3.1 Logical Frameworks

A thesis, which is at the root of our research effort, is that logical systems should be expressed as theories in a logical framework. As a consequence, proof-checking systems should not be focused on one theory, such as Simple type theory, Martin-Löf's type theory, or the Calculus of constructions, but should be theory-independent. In the same way, proof-search algorithms or the algorithmic interpretation of proofs should not depend on a theory, but this theory should just be a parameter. This is, for instance, expressed in the title of our invited talk at ICALP 2012: *A theory independent Curry-De Bruijn-Howard correspondence* [40].

Various limits of Predicate logic have led to the development of various families of logical frameworks: λ-Prolog and Isabelle have allowed terms containing bound variables, the Edinburgh logical framework has allowed proofs to be expressed as λ-terms, Pure type systems have allowed propositions to be considered as terms, and Deduction modulo theory has allowed theories to be defined not only with axioms, but also with computation rules.

The $\lambda\Pi$-calculus modulo theory, that is implemented in the system DEDUKTI, is a synthesis of the Edinburgh logical framework and of Deduction modulo theory, and subsumes them all. Our goal is to express as many theories as possible in DEDUKTI, express proofs in these theories and translate proofs from one theory to another, and from one system to another via Dedukti.

3.2 Interoperability, cross verification and sustainability of proof libraries

Using a single prover to check proofs coming from different systems and translating these proofs from one theory to another naturally leads to investigate how these proofs can be used in a system different from the one they have been developed in.

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 put little effort in the direction of standardization and interoperability.

A more recent trend is to use logical frameworks and proof translations for cross-checking. Checking a proof in several systems introduces some redundancy and hence reduces the probability that an incorrect proof is nevertheless successfully verified because of a bug in the proof-checker. This problem can be mitigated by developing proofs in systems that rely on a small and auditable trust base, that ensure a significantly lower probability for such undesirable events. In practice, however, this is not always possible, and our argument gets stronger when the proof has been developed in a theory that does not enjoy a small proof checker, but, instead, a complex, and sometimes heterogeneous, proof-construction system. This is for instance the case of B set theory, the theory on which the B method is based. There are several powerful tools to build proofs in this theory, but no small independent proof checker. Defining such a theory in a logical framework such as DEDUKTI and translating the proofs built by these tools into this theory permits to increase in a substantial way the trust we can have in these proofs.

Finally, on a more long-term perspective, we know that some proof-checking systems are not maintained anymore (this is, for instance the case of Automath and LCF, the two first proof checkers in history). When such a system disappears, its libraries often disappear with it. We can hope that expressing the proofs in a universal format in place of a system-specific one and preserving these proofs into a system-independent on-line repository such as NUBO or LOGIPEDIA will increase the sustainability of these libraries.

3.3 Interactive theorem proving

We also investigate how the $\lambda\Pi$-calculus modulo theory can be used as the basis of an interactive theorem prover. This leads to new scientific questions: first, how much can a tactic system be theory-independent, and then how does rewriting extend the possibility to write tactics.

This has led to the development of LAMBDAPI, which is an interactive theorem prover for the $\lambda\Pi$-calculus modulo theory. Several tactics have been developed for this system, which are intended to help a human user to write proofs in our system instead of writing proof terms by hand.

Such an interactive theorem prover happens to be very useful when we translate to DEDUKTI proofs coming from laconic systems that output a proof sketch rather than a full proof. In these cases, one first produces a proof skeleton with many gaps, that are filled, in a second step of the translation, with the help of automatic tactics.

3.4 Proof automation

Interoperability between interactive and automatic theorem provers can be fruitful to both systems: results coming from automatic solvers can be checked by a third-party software with an identified kernel, and interactive provers can benefit from more automation. We are pushing towards this last application by extending the SMTCoq plugin for the Coq proof assistant with new logical transformations that encode Coq goals into first-order logic, which is the input logic of the class of automatic provers called SMT solvers. We also develop tools for checking proofs in the TSTP and Alethe formats generated by automated theorem provers and SMT solvers.

4 Application domains

Our research project has lead us to focus on applications directed to the proof-checking community itself rather than to users of proof-checking. Indeed, translating proofs from one system to another, or building a system-independent proof library is more a service to the proof-checking community than to the users of formal methods.

This situation is evolving fast, along with the rise of cross-verification.

Providing a complementary small-trust-base proof checker for B leads us to be in closer connection with the community using formal methods in the railways industry and more generally to the modelization of industrial system community.

This is materialized with the ICSPA ANR project. We also have a long-term collaboration with the air traffic control community through the PVS community.

5 Highlights of the year

Louise Dubois De Prisque [32] and Thiago Felicissimo [33] defended their PhD theses.

5.1 Awards

Gilles Dowek received the Medal in History of Sciences and Epistemology 2024 of the Academy of Sciences.

6 New software, platforms, open data

6.1 New software

6.1.1 Lambdapi

Keywords: Dependent types, Rewriting, Proof assistant

Functional Description: Lambdapi is an interactive proof development system featuring dependent types like in Martin-Lőf's type theory, but allowing to define objects and types using oriented equations, aka rewriting rules, and reason modulo those equations. This allows to simplify some proofs, and formalize complex mathematical objects that are otherwise impossible or difficult to formalize in more traditional proof systems.

Lambdapi comes with Emacs and VSCode support.

Lambdapi can also read and output Dedukti files, and can thus be used as an higher-level intermediate language for translating proofs from one system to Dedukti.

Lambdapi is a logical framework and does not come with a pre-defined logic. However, it is easy to define a logic by declaring a few symbols and rules. A library of pre-defined logic is also provided.

Here are some of the features of Lambdapi: - Emacs and VSCode plugins (based on LSP) - support for unicode (UTF-8) and user-defined infix operators - symbols can be declared commutative, or associative and commutative - some arguments can be declared as implicit: the system will try to find out their value automatically - symbol and rule declarations are separated so that one can easily define inductive-recursive types or turn a proved equation into a rewriting rule - support for interactive resolution of typing goals, and unification goals as well, using tactics - a rewrite tactic similar to the one of SSReflect in Coq - the possibility of calling external automated provers - a command is provided for automatically generating an induction principle for (mutually defined) strictly-positive inductive types - Lambdapi can call external provers for checking the confluence and termination of user-defined rewriting rules by translating them to the XTC and HRS formats used in the termination and confluence competitions

URL: `https://github.com/Deducteam/lambdapi`

Contact: Frederic Blanqui

6.1.2 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.

URL: https://deducteam.github.io/

Publications: hal-01086609, hal-01176715, hal-01441751

Contact: Frederic Blanqui

Participants: Francois Thire, Gaspard Ferey, Guillaume Genestier, Rodolphe Lepigre

6.1.3 hol2dk

Keywords: Interoperability, Proof

Functional Description: Tool making HOL-Light generate proofs, simplifying those proofs, and translating those proofs to Dedukti, Lambdapi and Coq.

URL: https://github.com/Deducteam/hol2dk

Contact: Frederic Blanqui

6.1.4 BiTTs

Keywords: Dependent types, Logical Framework

Functional Description: This is an implementation of the generic bidirectional typing algorithm presented in the paper "Generic bidirectional typing for dependent type theories".

URL: https://github.com/thiagofelicissimo/BiTTs

Contact: Thiago Felicissimo Cesar

6.1.5 Predicativize

Name: Predicativize

Keywords: Dedukti, Proof assistant, Interoperability

Functional Description: Predicativize is a tool allowing for the translation of proofs from a core impredicative type theory to a core predicative theory featuring universe polymorphism. It works by calculating constraints between universe levels, which are then solved using universe level unification, generating then a predicative universe polymorphic definition. The theory behind the tool is provided in the paper "Translating proofs from an impredicative type system to a predicative one", by Thiago Felicissimo, Frédéric Blanqui and Ashish Kumar Barnawal. Predicativize was used to translate Matita's arithmetic library to Agda.

URL: https://github.com/Deducteam/predicativize

Contact: Thiago Felicissimo Cesar

6.1.6 commutative-diagrams

Name: Commutative diagrams proof assistant

Keyword: Proof assistant

Functional Description: A coq plugin enabling to progress categoretical proofs graphically. It can infer the diagram from the proof context, and display it graphically to the user. The user's action on the diagram are then converted into Coq proofs.

URL: https://github.com/dwarfmaster/commutative-diagrams

Contact: Luc Chabassier

6.1.7 pogtranslator

Keywords: Formal methods, Proof

Functional Description: Translator of Atelier B proof obligations (in the POG format) to the TPTP or SMT-LIB format.

Contact: Claude Stolze

6.1.8 sniper

Keywords: Coq, Automated deduction

Functional Description: Sniper is a Coq plugin that improves its automation.

URL: `https://github.com/smtcoq/sniper`

Contact: Chantal Keller

Partner: Université Paris-Saclay

6.1.9 dkpltact

Keywords: Coq, Interoperability, Proof

Functional Description: A tool to translate proofs from a Dedukti encoding of Predicate logic to the tactic language of Coq. It takes Dedukti files whose terms comply with this encoding and produces the corresponding Coq files.

URL: `https://gitlab.crans.org/geran/dkpltact`

Contact: Yoan Geran

6.1.10 Zenon Modulo

Keywords: First-order logic, Automated theorem proving, Deduction Modulo

Functional Description: Zenon Modulo is an extension of the automated theorem prover Zenon. Compared to Super Zenon, it can 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.

URL: `https://github.com/Deducteam/zenon_modulo`

Contact: Guillaume Burel

Partner: ENSIIE

6.1.11 Agda2Dedukti

Keywords: Compilation, Proof assistant, Higher-order logic, Rewriting systems

Functional Description: Translation of Agda proofs to the Logical Framework Dedukti.

URL: `https://github.com/Deducteam/Agda2Dedukti`

Contact: Thiago Felicissimo Cesar

Partner: Chalmers University

6.1.12 Coqine

Name: 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.

URL: `http://www.ensiie.fr/~guillaume.burel/blackandwhite_coqInE.html.en`

Contact: Guillaume Burel

6.1.13 Krajono

Keyword: Proof

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

Contact: Claudio Sacerdoti Coen

6.1.14 personoj

Keywords: PVS, Automated theorem proving, Dedukti, Machine translation

Functional Description: Personoj comprises a set of PVS patches that may be used to export PVS specific-ations (propositions and definitions) or to export successive sequents of a proof to lambdapi. Another program is able to process these sequents and call automated theorem provers through Why3 to prove the implications of the successive sequents.

Contact: Gabriel Hondet

6.1.15 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.

URL: `https://github.com/Deducteam/Holide`

Contact: Guillaume Burel

6.1.16 Logipedia

Name: Logipedia

Keywords: Formal methods, Web Services, Logical Framework

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

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

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

Release Contributions: This is the beta version of Logipedia. It implements the functionalities mentioned above.

URL: http://www.logipedia.science

Contact: Frederic Blanqui

6.1.17 SKonverto

Name: SKonverto

Keywords: Skolemization, First-order logic, Proof assistant

Functional Description: SKonverto is a tool that transforms Lambdapi proofs containing Skolem symbols into proofs without these symbols.

URL: https://github.com/Deducteam/SKonverto

Contact: Mohamed Yacine El Haddad

Partner: ENSIIE

6.2 Open data

HOL-Light definition of number types in Coq

Project link: https://github.com/Deducteam/coq-hol-light-real

Contact: Frédéric Blanqui

Translation of HOL-Light base library to Coq

Project link: https://github.com/Deducteam/coq-hol-light

Contact: Frédéric Blanqui

Translation of Matita arithmetic library to Agda

Project link: https://github.com/Deducteam/matita_lib_in_agda

Contact: Thiago Felicissimo

7 New results

7.1 Metatheory of proof and computation systems

7.1.1 Ghost types and equality reflection

Participants: Ewen Broudin–Caradec, Théo Winterhalter.

We introduced ghost type theory (GTT) a dependent type theory extended with a new universe for ghost data that can safely be erased when running a program but which is not proof irrelevant like with a universe of (strict) propositions. Instead, ghost data carry information that can be used in proofs or to discard impossible cases in relevant computations. Casts can be used to replace ghost values by others that are propositionally equal, but crucially these casts can safely be ignored for conversion. We provide a type-preserving erasure procedure which gets rid of all ghost data and proofs, a step which may be used as a first step to program extraction. We give a syntactical model of GTT using a program translation akin to the parametricity translation and thus show consistency of the theory. Because it is a parametricity model, it can also be used to derive free theorems about programs using ghost code. We further extend GTT to support equality reflection and show that we can eliminate its use without the need for the usual extra axioms of function extensionality and uniqueness of identity proofs. In particular we validate the intuition that indices of inductive types—such as the length index of vectors—do not matter for computation and can safely be considered modulo theory. Our results have been formalised in Coq and lead to a POPL publication [18].

7.1.2 Type preserving rewrite rules for the Coq proof assistant

Participants: Théo Winterhalter.

We present an implementation of rewrite rules on top of the Coq proof assistant, together with a modular criterion to ensure that the added rewrite rules preserve typing. This criterion, based on bidirectional type checking, is formally expressed in the type theory of Coq. This lead to an ITP publication [28].

7.1.3 Generic bidirectional typing for dependent type theories

Participants: Thiago Felicissimo, Frédéric Blanqui, Gilles Dowek.

Bidirectional typing is a discipline in which the typing judgment is decomposed explicitly into inference and checking modes, allowing to control the flow of type information in typing rules and to specify algorithmically how they should be used. Bidirectional typing has been fruitfully studied and bidirectional systems have been developed for many type theories. However, the formal development of bidirectional typing has until now been kept confined to specific theories, with general guidelines remaining informal. In this work, we give a generic account of bidirectional typing for a general class of dependent type theories. This is done by first giving a general definition of type theories (or equivalently, a logical framework), for which we define declarative and bidirectional type systems. We then show, in a theory-independent fashion, that the two systems are equivalent. This equivalence is then explored to establish the decidability of typing for weak normalizing theories, yielding a generic type-checking algorithm that has been implemented in a prototype and used in practice with many theories. This work was published at ESOP [24] and an extended version has been accepted for publication at TOPLAS.

7.1.4 Second-order Church-Rosser modulo, without normalization

Participants: Thiago Felicissimo.

Rewriting modulo is an alternative to standard rewriting in which one considers not only rewriting rules but also undirected equations, allowing to handle theories defined by axioms that cannot be oriented in a well-behaved manner, such as commutativity. In this setting, the Church-Rosser property must be adapted into *Church-Rosser modulo*. Unfortunately, most criteria for Church-Rosser modulo rely on normalization, yet confluence proofs for dependent type theories are usually carried out on untyped terms, for which normalization does not hold due to rules such as β-reduction. In this work, we investigate criteria for proving Church-Rosser modulo of second-order rewrite systems without relying on normalization. This work was published at IWC [25].

7.2 Graph rewriting

Participants: Jean-Pierre Jouannaud.

With Nachum Dershowitz (U. Tel Aviv) and Fernando Orejas (UTC Barcelona), we have finished and submitted to a journal a first article describing our rewriting model over drags. Drags are directed, ordered graphs whose number of outgoing edges at each vertex is governed by the arity of the function symbol labelling that vertex. Vertices are also endowed with roots. Sprouts are special vertices labelled by variables, of arity zero. A product operation is defined which allows to redirect all edges ending in a sprout to a vertex endowed with (enough) roots, as specified in a set called switchboard. Given a drag rewrite rule (L to R), rewriting a given drag D amounts to find a context drag C and a switchboard xi such that C xi L = D, this is matching, and compute the new graph D' = C xi R. This model has many advantages over the traditionnal categorical approach for graph rewriting, the Double PushOut model: (1) product based matching is more powderful than monomorphism based matching, as used by DPO. (2) rewriting does not generate dangling edges as is the case with DPO. (3) term rewriting appears to be a true particular case of our model, a problem which was still open since raised by Barendregt et al in 1987. (4) the nice term rewriting techniques for proving confluence (critical pairs) and termination (recursive path ordering) scale to the drag model. We are currently extending our model to arbitrary directed, unordered graphs.

7.3 Confluence of non-left-linear rules in typed lambda calculi

Participants: Jean-Pierre Jouannaud, Thiago Felicissimo.

This question has become important since the introduction of rewrite rules in many proof assistants based on dependent type theory, including Agda, Coq, and Lean. In this work, we extend the technique introduced by Assaf et al, who introduced the notion of confined sort, inhabited by first-order terms. In this work, confined variables in rewrite rules could be non-linear, while non-confined ones could not, and instantiated by expressions inhabiting a confined sort. Our own notion of confined sort is very general, and actually inferred from the sort structure of a specification. Our current results allow us to show confluence of encodings of type theories for which difficult proofs carried by hand. had to be carried out. An example of interest for Dedukti is the encoding of "Deduction modulo", a pure lambda calculus equipped with an equality over the confined sort of natural numbers, including the non-linear rule EQ(n,n) -> True. This work should be ready for submission to CADE 2025.

7.4 Expressing proof systems in Dedukti

7.4.1 Impredicativity, cumulativity and product covariance in Dedukti

Participants: Thiago Felicissimo, Théo Winterhalter.

Proof assistants such as Coq implement a type theory featuring three important features: impredicativity, cumulativity and product covariance. This combination has proven difficult to be expressed in Dedukti, and previous attempts have failed in providing an encoding that is proven confluent, sound and conservative. We solve this longstanding open problem by providing an encoding of these three features that we prove to be confluent, sound and to satisfy a restricted (but, we argue, strong enough) form of conservativity. Our proof of confluence is a contribution by itself, and combines various criteria and proof techniques from rewriting theory. Our proof of soundness also contributes a new strategy in which the result is shown in terms of an inverse translation function, fixing a common flaw made in previous encoding attempts. This work was published at FSCD [26].

7.4.2 A linear Rewrite System to Represent Impredicative and Cumulative Universes with Polymorphism

Participants: Yoan Géran.

Yoan Géran has designed a rewrite system to represent, in the $\lambda\Pi$-calculus modulo theory, type universes that feature cumulativity, impredicativity of the first universe, and universe polymorphism. This system extends previous works and is now confluent, terminating, and left-linear.

7.4.3 Translating Lean to Dedukti

Participants: Rishikesh Vaishnav, Frédéric Blanqui.

This year, Rishikesh Vaishnav made progress on various aspects of the translation from Lean to Dedukti. Notably, he completed the implementation of instantiation scheme for universe level variables in lean2dk (work that was started last year in collaboration with Yoan Geran), extended the translation to handle more cases of definitional equality/reduction, and started to attempt the translation of Lean's standard library's prelude files. In doing so, he identified some tricky cases of definitional equality that are difficult to encode, in particular Lean's support for K-like reduction and proof irrelevance. This prompted an investigation into the possibility of pre-translating Lean into a smaller theory that replaces these definitional rules with axioms in order to ease translation, resulting in the implementation of the tool "Lean4Less" (see section 7.5.1).

7.4.4 Making Leo-III output Lambdapi proofs

Participants: Melanie Taprogge, Frédéric Blanqui.

Melanie Taprogge began her PhD in October, building on the work from her Master thesis [38]. Her project focuses on certifying the proofs generated by the fully automated higher-order logic theorem prover Leo-III through an encoding of its proofs in Lambdapi. She analyzed the particular challenges involved in this encoding and developed strategies to address them. This effort resulted in a general schema that can be systematically applied to encode inference rules and their applications, both for Leo-III and other automated systems. The effectiveness of this approach was demonstrated by deriving encodings for some core inference rules of the EP calculus implemented in Leo-III, as well as a partial implementation of the proof output, yielding a verifiable proof of a popular benchmark problem for testing the capabilities of automated reasoning systems. In collaboration with team members working on the expression of other proof systems in Lambdapi, particularly Alessio Coltellacci, Anne Grieu, Ciarán Dunne and Frédéric Blanqui, Melanie identified several encodings necessary across different projects that had previously been addressed using different strategies in each project. She contributed to the development of more general encodings of such principles, aiming for a more uniform and widely applicable representation across multiple projects and systems.

7.4.5 Translating Eunoia to Dedukti and Lambdapi

Participants: Ciarán Dunne, Guillaume Burel.

Ciarán has been working towards the development of a tool named `eo2dk` for automated translation of specifications and proofs of the Eunoia logical framework to Dedukti and Lambdapi. Eunoia (formerly AletheLF) is a dependently-typed logical framework designed for specifying the theories and proof systems used by SMT solvers (in particular, `cvc5`). Among other features, Eunoia extends the core SMT-LIB language with dependent types (with implicit paramters), rewrite rules, and a mechanism for defining inference rules.

Currently, the only proof system defined in Eunoia is cvc5's *co-operating proof calculus* (CPC). First, our translation tool will allow automatically translating the inference rules of CPC to a library of symbol declarations in Dedukti. In turn, the proofs outputted by `cvc5` can be translated into Dedukti proof terms using those symbols. All together, we provide an external proof-checker for `cvc5` that can be easily updated as the rules of CPC inevitably change.

Furthermore, we have developed a set-theoretic semantics of Eunoia based on the types-as-sets interpretation given by Aczel, Werner and Barras. The aim of this work is gain a understanding of dependently-typed extensions of SMT-LIB 2 (like Eunoia, and the anticipated release of SMT-LIB v3), and to also lay the groundwork for verifying the soundness of the encoding used in `eo2dk`.

7.4.6 Translating B Proof Obligations to Dedukti

Participants: Claude Stolze, Olivier Hermant.

In the framework of the ICSPA ANR project, Claude Stolze has implemented a translator from B proof obligations to why pog3why. The intermediate translation to Why3 allowed to use automated theorem provers on the proofs obligations [35] (joint work with Romain Guillaumé).

7.4.7 Translating Dedukti proofs to Coq with Tactics

Participants: Yoan Geran.

Yoan Géran has implemented a translator, dkpltact, for the GeoCoq library embedding in Dedukti,back to Coq. After a reverse mathematical analyzis and a minimization of the embeddding to minimal logic, he translated the library back to Coq by generating proof scripts. This yielded a back-and-forth translation from Coq to Coq with two properties : the logic has been minimized, and the size of the proof scripts/files is not increased.

7.5 Translation of one theory to another

7.5.1 Implementing a translation from extensional to intensional type theory in Lean

Participants: Rishikesh Vaishnav, Frédéric Blanqui.

In support of his work translating from Lean to Dedukti, Rishikesh Vaishnav implemented an extensional-to-intensional translation framework Lean4Less which is based on a theoretical formalization of an ETT to ITT translation found in ett-to-itt. It implements a specialized case of this translation that replaces uses of definitional proof irrelevance and K-like reduction with type transport using a proof irrelevance axiom, translating Lean to the smaller theory Lean$^-$ which should be a better candidate for translation to Dedukti.

In implementing Lean4Less, he added a number of optimizations to help minimize the output size and avoid redundancy. The tool as currently implemented is capable of translating the entirety of the Lean standard library into Lean$^-$. Further work remains to be done to scale this translation up to Mathlib. Because this implementation should be consistent with the general ETT to ITT translation, it should also be possible to extend and adapt it for use in Lean's elaboration routine to simulate extensional typechecking in Lean.

A work-in-progress report was presented at LFMTP 2024 [37], and a complete publication is targeted for FSCD 2025.

7.5.2 Replacement of rewrite rules by axioms

Participants: Thomas Traversié, Valentin Blot, Gilles Dowek, Théo Winterhalter.

We showed that it is possible to replace the rewrite rules of a theory of the $\lambda\Pi$-calculus modulo theory by equational axioms, when this theory features the notions of proposition and proof, while maintaining the same expressiveness. To do so, we introduced in the target theory a heterogeneous equality, and we built a translation that replaces each use of the conversion rule by the insertion of a transport. At the end, the theory with rewrite rules is a conservative extension of the theory with axioms. This work was published at FoSSaCS [21].

7.5.3 Translation from classical logic to intuitionistic logic

Participants: Thomas Traversié, Olivier Hermant.

In 1951, Kuroda defined an embedding of classical first-order logic into intuitionistic logic, such that a formula and its translation are equivalent in classical logic. Recently, Brown and Rizkallah extended this translation to higher-order logic, but did not prove the classical equivalence, and showed that the embedding fails in the presence of functional extensionality. We proved that functional extensionality and propositional extensionality are sufficient to derive the classical equivalence between a higher-order formula and its translation. We emphasized a condition under which Kuroda's translation works with functional extensionality. This work was submitted for publication [36].

The next step was to adapt this work to the $\lambda\Pi$-calculus modulo theory and to Dedukti proofs. Kuroda's translation can be adapted for theories encoded in higher-order logic in the $\lambda\Pi$-calculus modulo theory. As we work with theories—that is with typed constants and rewrite rules—we have to translate them as well. Moreover, we developed a tool Construkti that implements Kuroda's translation for proofs written in DEDUKTI. This work was published at LFMTP [29].

Recently, we started working on extending to higher-order logic several translations that generalize double-negation translations by using monad operators instead of double negations. Such translations eliminate particular axioms, for instance the principle of excluded middle or the principle of exclusion, and therefore embed classical logic into intuitionistic logic or intuitionistic logic into minimal logic.

7.5.4 Generic translation templates for Dedukti

Participants: Thomas Traversié.

Since the $\lambda\Pi$-calculus modulo rewriting is used as a formal middleware for exchanging proofs between different proof systems, it is important to define generic translations between theories of the $\lambda\Pi$-calculus modulo rewriting.

To this end, we defined an interpretation of theories of the $\lambda\Pi$-calculus modulo theory, in which a morphism and its invariants are mutually defined. Such an interpretation allows to transfer proofs between theories that feature the notions of proposition and proof, when the source theory can be embedded into the target theory. This work was published in LFMTP [30].

Several translations templates already exist for LF, for instance theory morphisms [41] and logical relations [42], but all of these developments were done in the absence of rewriting. we extended the existing theory morphisms and logical relations to theories of the $\lambda\Pi$-calculus modulo rewriting. We implemented TranslationTemplates these translation templates in DEDUKTI and formalized some case studies. This work launched a collaboration with Florian Rabe (University of Erlangen).

7.5.5 Translating HOL-Light proofs to Dedukti, Lambdapi and Coq

Participants: Frédéric Blanqui.

In [20], we present a method and a tool, hol2dk, to fully automatically translate proofs from the proof assistant HOL-Light to the proof assistant Coq, by using Dedukti as an intermediate language. Moreover, a number of types, functions and predicates defined in HOL-Light are proved (by hand) to be equal to their counterpart in the Coq standard library. By replacing those types and functions by their Coq counterpart everywhere, we obtain a library of theorems (based on classical logic like HOL-Light) that can directly be used and applied in other Coq developments.

7.5.6 Equivalence of the types of real numbers of HOL-Light and Coq

Participants: Frédéric Blanqui, Anthony Bordg, Amal Makni.

More recently, we formally proved in Coq that the types of real numbers in HOL-Light and in the Coq standard library are isomorphic, and the basic functions (addition, multiplication, etc.) and predicates (ordering) on real numbers are equal extensionally. A paper is in preparation.

7.5.7 Translating TSTP proofs to Lambdapi

Participants: Frédéric Blanqui, Guillaume Burel.

With our help, Geoff Sutcliffe extended his tool GDV for checking the correctness of TSTP proofs generated by automated theorem provers so that it now outputs Lambdapi proofs using ZenonModulo. GDV now subsumes the tool Ekstrakto developped by our former PhD student Mohamed Yacine El Haddad.

7.6 Deductive Verification of programs

Participants: Catherine Dubois.

We formalized in Why3 the data structure called "sparse set" used to represent finite sets of integers and proved the correctness of its operations (membership, union, intersection, etc.). In particular, we have proved that its variant used in constraint programming solvers to represent the domain of integer variables is reversible by providing a formally verified OCaml implementation of an "undo" operation [23]. Furthermore, with the help of Maximilano Cristiá, its basic operations were implemented in three deductive formal verification tools, Event-B, {log} and Why3 and compared regarding specifications and proofs [31].

8 Bilateral contracts and grants with industry

8.1 Nomadic Labs

Participants: Valentin Blot, Louise Dubois De Prisque, Chantal Keller.

Valentin Blot and Chantal Keller got some funding as part of part of the Inria - Nomadic labs partnership for Tezos blockchain for a 4-year project (2021–2025) involving a PhD student (Louise Dubois De Prisque), a research engineer (2 years) and a post-doctoral researcher (2 years).

8.2 Amazon AWS

Participants: Gilles Dowek, Guillaume Burel, Ciarán Dunne.

Gilles Dowek received a grant from Amazon to hire a post-doc (Ciarán Dunne) to work on checking proofs produced by SMT solvers.

9 Partnerships and cooperations

9.1 International initiatives

9.1.1 Inria associate team not involved in an IIL or an international program

ICI

Title: Interoperability of Coq and Isabelle proof systems

Duration: 2024 ->

Coordinator: Frédéric Blanqui

Partners:

- National Institute of Advanced Industrial Science and Technology Tokyo (Japon)

Summary: This project aims at improving the interoperability between the most two used proof systems in the world: Coq and Isabelle. Deducteam is expert in proof systems interoperability but has no expert in Isabelle. AIST uses Coq and Isabelle to prove the correctness of programs and has experts in both langages.

Web site

CARMA

Title: CAtalyzing progRess in smt solving and proof assistants via Modularity, proof trAnslation, and proof reconstruction

Duration: 2024 ->

Coordinator: Sophie Tourret (INRIA Nancy)

Partners:

- Federal Univ. of Minas Gerais (UFMG), Belo Horizonte, Brasil
- Deducteam

Inria contact: Frederic Blanqui

Summary: This collaboration aims at improving the state of the art in SMT solving on three fronts: counter the research slowdown created by the tight interconnection of parts in state-of-the-art SMT solvers, develop the missing components to make higher-order SMT competitive, and provide a tool for translating Alethe proofs originating from TLA specifications to Dedukti.

Web site

9.2 International research visitors

9.2.1 Visits of international scientists

Geoff Sutcliffe

Status Professor

Institution of origin: University of Miami

Country: USA

Dates: 10 June - 19 July 2024

Martina Seidl

Status Professor

Institution of origin: Institute for Symbolic Artificial Intelligence, Johannes Kepler University

Country: Germany

Dates: 19-23 February 2024

Context of the visit: She has great expertise in the design and implementation of solvers for quantified Boolean formulas. These proof tools produce certificates that can be independently verified. The main goal of this research visit was to study with Guillaume Burel and Catherine Dubois if Dedukti can be used to check such proofs.

9.2.2 Visits to international teams

Research stays abroad

- Rishikesh Vaishnav stayed one week with members of the Lean FRO in Munich, Germany, funded by the COST action EuroProofNet, to work with Sebastian Ullrich on the translation of Lean to Dedukti.
- Théo Winterhalter stayed ten days at AIST Tokyo, Japan, in May, to work with Akihisa Yamada and Reynald Affeldt in the framework of the associated team project ICI.
- Frédéric Blanqui stayed two weeks at AIST Tokyo, Japan, at the end of October, to work with Akihisa Yamada and Reynald Affeldt in the framework of the associated team project ICI.
- Melanie Taprogge was founded by the EuroProofNet to stay at the ENS in Paris for two weeks to work on the verification of higher-order logic automated reasoning with Frédéric Blanqui in March and April while finishing her Master’s degree in Germany. Furthermore, she stayed for one week at AIST Tokyo, Japan, at the end of December, to work with Akihisa Yamada and Reynald Affeldt in the framework of the associated team project ICI.
- At the end of December, Ciarán Dunne stayed one week at Belo Horizonte, Brazil, to work with Haniel Barbosa and his team in the framework of the associated team project CARMA headed by Sophie Tourret in Nancy, and one week at AIST Tokyo, Japan, to work with Akihisa Yamada and Reynald Affeldt in the framework of the associated team project ICI.

9.3 European initiatives

9.3.1 COST action 20111 EuroProofNet

Frédéric Blanqui is the chair of the COST action CA20111 EuroProofNet 2022-2025 which is a research network on proofs gathering more than 500 members from 45 different countries, with an average annual budget of 200,000 euros.

9.4 National initiatives

9.4.1 ICSPA

Participants: Guillaume Burel, Gilles Dowek, Catherine Dubois, Olivier Hermant, Claude Stolze.

The ANR project (2022-2025) ICSPA (Interoperable and Confident Set-based Proof Assistants) has been accepted in the context of the AAPG 2021 call. It is coordinated by Catherine Dubois and has Samovar, Inria Grand Est, Inria Paris-Saclay, LIRMM, IRIT as academic partner, and Clearsy as industrial partner. The project starts on January 1st 2022. This project aims at reinforcing the confidence in proofs carried out mechanically for the set-based specification formalisms B, Event-B, and TLA+ that are used in industry.This will be done by verifying these proofs formally and independently with the proof verifier Dedukti. The project also aims at designing and implementing an exchange framework, through which those three systems can share their proofs and theories, making them effectively interoperable.

9.4.2 PROGRAMme

Participants: Gilles Dowek.

The ANR PROGRAMme 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.

10 Dissemination

10.1 Promoting scientific activities

10.1.1 Scientific events: organisation

Member of steering committees

- Catherine Dubois is the chair of the steering committee of the international conference Test and Proof (TAP). She was also a member of the steering committee of the international Conference on Intelligent Computer Mathematics (CICM) until August 2024.
- Valentin Blot is a member of the steering committee of the Logic In Computer Science (LICS) conference.

Member of the organizing committees

- Théo Winterhalter co-organised the Coq Workshop 2024

10.1.2 Scientific events: selection

Chair of conference program committees

- Théo Winterhalter co-chaired the program of Coq Workshop 2024

Member of the conference program committees

- Théo Winterhalter was a PC member of the international workshop on Logical Frameworks and Meta-Languages: Theory and Practice (LFMTP'24).
- Frédéric Blanqui was a PC member of the 19th international workshop on Logical and Semantic Frameworks with Applications (LSFA'24).
- Catherine Dubois was a PC member of the 17th international Conference on Intelligent Computer Mathematics (CICM 2024), the 19th International Conference on Integrated Formal Methods (iFM'24), the 18th International Conference on Tests and Proofs (TAP 2024), the 6th Formal Methods Teaching Workshop, (FMTea 2024), the 15h International Conference on Interactive Theorem Proving (ITP 2024), the 10th International Conference on Rigorous State-Based Methods (ABZ 2024) and a PC member of the track Software Verification and Testing of the 40th ACM/SIGAPP Symposium On Applied Computing (SAC 2025).

Reviewer

- Théo Winterhalter contributed a review to ITP'24.
- Frédéric Blanqui reviewed a paper at IJCAR'24.

10.1.3 Journal

Reviewer - reviewing activities

- Théo Winterhalter contributed reviews for JAR.

10.1.4 Invited talks

- Théo Winterhalter was invited to give a talk at the CHoCoLa meeting in Lyon, on 21 November.
- Frédéric Blanqui was keynote speaker at the 15th International Conference on Interactive Theorem Proving (ITP'24).
- Frédéric Blanqui was invited lecturer at the 14th International School on Rewriting (ISR'24).
- Valentin Blot was an invited speaker at the Logic Colloquium 2024 (LC'24).

10.1.5 Research administration

- Frédéric Blanqui is chair of the COST action CA20111 EuroProofNet.
- Frédéric Blanqui is Vice Director of the STIC Doctoral School of the University Paris Saclay, in charge of the budget.
- Frédéric Blanqui is a member of the Scientific Committee of Inria Saclay, and Vice President since November 2024.
- Catherine Dubois is one of the two co-chairs of Groupement de Recherche Génie de la Programmation et du Logiciel (Gdr GPL).

10.2 Teaching - Supervision - Juries

10.2.1 Teaching

- Master: Frédéric Blanqui, formal languages, 21h, M1, ENSIIE, France
- Master: Frédéric Blanqui, rewriting theory, 14h, M1, ENS Paris-Saclay, France
- Master: Théo Winterhalter, Proof Assistants, 12h, M2, MPRI, France

- License: Luc Chabassier, Logique, L3, 30h, ENS Paris-Saclay, France
- License: Luc Chabassier, Projet base de données, 22h30, L3, ENS Paris-Saclay, France
- License: Luc Chabassier, Architecture et système, 22h30, L3, ENS Paris-Saclay, France
- License: Nicolas Margulies, Compilation project, 15h, L3, ENS Paris-Saclay, France
- License: Nicolas Margulies, Architecture et système, 22h30, L3, ENS Paris-Saclay, France
- License: Yoan Géran, Compilation project, 15h, L3, ENS Paris-Saclay, France
- License: Yoan Géran, Projet Programmation, 30h, L3, ENS Paris-Saclay, France
- IUT: Luc Chabassier, C++ R101-2, première année, 38h30, IUT d'Orsay, France
- IUT: Luc Chabassier, Projet C++ S102, première année, 10h30, IUT d'Orsay, France
- IUT: Claude Stolze, C++ R101-2, première année, 33h, IUT d'Orsay, France
- Engineering school: Thomas Traversié, Algorithmique et complexité, 18h, first year, CentraleSupélec, France
- Engineering school: Thomas Traversié, Systèmes d'information et Programmation, 22h30, first year, CentraleSupélec, France
- Engineering school: Thomas Traversié, Informatique théorique, 19h30, second year, CentraleSupélec, France
- Engineering school: Thomas Traversié, Modélisation logique et systèmes formels, 10h30, third year, CentraleSupélec, France
- Engineering school: Thomas Traversié, Modélisation logique, 7h30, third year, CentraleSupélec, France

10.2.2 Computer Science Education

- Catherine Dubois is a member of the formal methods teaching committee whose aim is to support a worldwide improvement in learning formal methods committee. In this context she co-authored an article which advocates "FM thinking", that is the application of ideas from Formal Methods applied in informal, lightweight, practical and accessible ways [15].

10.2.3 Supervision

- Théo Winterhalter co-supervises (30%) Yann Leray who is doing a PhD in the Gallinette team in Nantes.
- Frédéric Blanqui supervises the PhDs of Rishikesh Vaishnav and Melanie Taprogge.
- Catherine Dubois and Valentin Blot are supervising the PhD of Amélie Ledein.
- Catherine Dubois and Burkhart Wolff are supervising the PhD of Benoit Ballenghien.
- Théo Winterhalter supervised the M2 internship of Ewen Broudin–Caradec.
- Catherine supervised the M2 internship of Salwa Combet Gonzalez.
- Frédéric Blanqui supervised the M1 internship of Amal Makni.
- Valetin Blot supervised the M2 internship of Thomas Laure.

10.2.4 Juries

- Théo Winterhalter was an examiner and exam designer for the ENS Computer Science competitive entrance exam 2024.
- Frédéric Blanqui was a reviewer of the PhD of Thibault Hilaire (Bordeaux University).
- Catherine Dubois was a reviewer of the HDR of Frédéric Dabrowski (Université d'Orléeans)
- Catherine Dubois was a reviewer of the PhD of Benjamin Somers (IMT Atlantique).
- Catherine Dubois was a reviewer of the PhD of Olivier Martinot (Université Paris Cité).
- Catherine Dubois was a reviewer of the PhD of Mohammed El Amin Tebib (Université Grenoble Alpes).
- Catherine Dubois was an examiner of the PhD of Mariya Naumcheva (Université de Toulouse).
- Catherine Dubois was an examiner of the PhD of Camilo Correa Restrepo (Sorbonne Université).

11 Scientific production

11.1 Major publications

[1] B. Barras, T. Coquand and S. Huber. 'A generalization of the Takeuti-Gandy interpretation'. In: *Mathematical Structures in Computer Science* 25.5 (2015), pp. 1071–1099. DOI: 10.1017/S0960129514000504. URL: https://doi.org/10.1017/S0960129514000504.

[2] F. Blanqui. 'Definitions by rewriting in the Calculus of Constructions'. Anglais. In: *Mathematical Structures in Computer Science* 15.1 (2005), pp. 37–92. DOI: 10.1017/S0960129504004426. URL: http://hal.inria.fr/inria-00105648/en/.

[3] F. Blanqui. 'Type safety of rewrite rules in dependent types'. In: FSCD 2020 - 5th International Conference on Formal Structures for Computation and Deduction. Vol. 167. Paris, France, 28th June 2020, p. 14. DOI: 10.4230/LIPIcs.FSCD.2020.13. URL: https://inria.hal.science/hal-02981528.

[4] F. Blanqui, G. Dowek, E. Grienenberger, G. Hondet and F. Thiré. 'A modular construction of type theories'. In: *Logical Methods in Computer Science* 19.1 (14th Feb. 2023). DOI: 10.46298/lmcs-19(1:12)2023. URL: https://inria.hal.science/hal-04317047.

[5] F. Blanqui, J.-P. Jouannaud and A. Rubio. 'The Computability Path Ordering'. In: *Logical Methods in Computer Science* (Oct. 2015). DOI: 10.2168/LMCS-11(4:3)2015. URL: https://hal.inria.fr/hal-01163091.

[6] V. Blot. 'An interpretation of system F through bar recursion'. In: *32nd ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 2017.

[7] G. Burel, G. Bury, R. Cauderlier, D. Delahaye, P. Halmagrand and O. Hermant. 'First-Order Automated Reasoning with Theories: When Deduction Modulo Theory Meets Practice'. In: *Journal of Automated Reasoning* (2019). DOI: 10.1007/s10817-019-09533-z. URL: https://hal.archives-ouvertes.fr/hal-02305831.

[8] D. Cousineau and G. Dowek. 'Embedding Pure Type Systems in the $\lambda\Pi$-calculus modulo'. In: *Typed lambda calculi and applications*. Ed. by S. R. della Rocca. Vol. 4583. Lecture Notes in Computer Science. Springer-Verlag, 2007, pp. 102–117.

[9] G. Dowek, T. Hardin and C. Kirchner. 'Theorem proving modulo'. In: *Journal of Automated Reasoning* 31 (2003), pp. 33–73.

[10] O. Hermant. 'Resolution is Cut-Free'. In: *Journal of Automated Reasoning* 44.3 (Mar. 2010), pp. 245–276.

[11] M. Jacquel, K. Berkani, D. Delahaye and C. Dubois. ‘Tableaux Modulo Theories Using Superdeduction’. In: *Global Journal of Advanced Software Engineering (GJASE)* 1 (Dec. 2014), pp. 1–13. DOI: 10.1007/978-3-642-31365-3_26. URL: https://hal.archives-ouvertes.fr/hal-01099338.

[12] M. Jacquel, K. Berkani, D. Delahaye and C. Dubois. ‘Verifying B Proof Rules using Deep Embedding and Automated Theorem Proving’. In: *Software and Systems Modeling (SoSyM)* (June 2013).

11.2 Publications of the year

International journals

[13] C.-C. Andrici, Ș. Ciobâcă, C. Hriţcu, G. Martínez, E. Rivas, É. Tanter and T. Winterhalter. ‘Securing Verified IO Programs Against Unverified Code in F*’. In: *Proceedings of the ACM on Programming Languages* 8.POPL (2024), pp. 2226–2259. DOI: 10.1145/3632916. URL: https://hal.science/hal-04484770.

[14] P. Arrighi, G. Dowek and A. Durbec. ‘Time arrow without past hypothesis: a toy model explanation’. In: *New Journal of Physics* 26.11 (28th Nov. 2024), p. 113019. DOI: 10.1088/1367-2630/ad93f5. URL: https://hal.science/hal-04856924.

[15] B. Dongol, C. Dubois, S. Hallerstede, E. Hehner, C. Morgan, P. Müller, L. Ribeiro, A. Silva, G. Smith and E. de Vink. ‘On formal methods thinking in computer science education’. In: *Formal Aspects of Computing* 37.1 (26th Dec. 2024), pp. 1–23. DOI: 10.1145/3670419. URL: https://hal.science/hal-04896081 (cit. on p. 21).

[16] T. Felicissimo and F. Blanqui. ‘Sharing proofs with predicative theories through universe-polymorphic elaboration’. In: *Logical Methods in Computer Science* (10th Sept. 2024). URL: https://hal.science/hal-04866019.

[17] M. Sozeau, Y. Forster, M. Lennon-Bertrand, J. B. Nielsen, N. Tabareau and T. Winterhalter. ‘Correct and Complete Type Checking and Certified Erasure for Coq, in Coq’. In: *Journal of the ACM (JACM)* (27th Nov. 2024), pp. 1–76. DOI: \url{https://doi.org/10.1145/3706056}. URL: https://inria.hal.science/hal-04077552.

[18] T. Winterhalter. ‘Dependent Ghosts Have a Reflection for Free’. In: *Proceedings of the ACM on Programming Languages* 258 (1st Aug. 2024), pp. 630–658. DOI: 10.1145/3674647. URL: https://hal.science/hal-04163836 (cit. on p. 11).

International peer-reviewed conferences

[19] P. Arrighi, G. Dowek and A. Durbec. ‘A toy model provably featuring an arrow of time without past hypothesis’. In: *LNCS*. RC 2024 - 16th International Conference on Reversible Computation. Vol. 14680. Lecture Notes in Computer Science. Torun, Poland: Springer Nature Switzerland, 29th May 2024, pp. 50–68. DOI: 10.1007/978-3-031-62076-8_4. URL: https://hal.science/hal-04727052.

[20] F. Blanqui. ‘Translating HOL-Light proofs to Coq’. In: *Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning*. LPAR-25 - 25th International Conference on Logic for Programming, Artificial Intelligence and Reasoning. Balaclava, Mauritius, 26th May 2024, pp. 1–18. DOI: 10.29007/6k4x. URL: https://inria.hal.science/hal-04613926 (cit. on p. 16).

[21] V. Blot, G. Dowek, T. Traversié and T. Winterhalter. ‘From Rewrite Rules to Axioms in the $\lambda\Pi$-Calculus Modulo Theory’. In: *Lecture Notes in Computer Science, International Conference on Foundations of Software Science and Computation Structures*. FoSSaCS 2024 - 27th International Conference on Foundations of Software Science and Computation Structures. Vol. 14575. Lecture Notes in Computer Science 2. Luxembourg City, Luxembourg: Springer Nature Switzerland, Apr. 2024, pp. 3–23. DOI: 10.1007/978-3-031-57231-9_1. URL: https://hal.science/hal-04275229 (cit. on p. 15).

[22] A. Coltellacci, G. Dowek and S. Merz. 'Reconstruction of SMT proofs with Lambdapi'. In: *CEUR Workshop Proceedings*. SMT 2024 - 22nd International Workshop on Satisfiability Modulo Theories. Vol. 3725. Montréal, Canada, 22nd July 2024, pp. 13–23. URL: https://inria.hal.science/hal-04861898.

[23] C. Dubois. 'Deductive Verification of Sparse Sets in Why3'. In: *Verified Software. Theories, Tools and Experiments - 16th International Conference, VSTTE 2024, Postproceedings*. VSTTE 2024 - 16th International Conference on Verified Software: Theories, Tools, and Experiments. Prague, Czech Republic, 14th Oct. 2024. URL: https://hal.science/hal-04863059 (cit. on p. 16).

[24] T. Felicissimo. 'Generic bidirectional typing for dependent type theories'. In: ESOP 2024 - 33rd European Symposium on Programming. Luxembourg City, Luxembourg, 6th Apr. 2024. URL: https://inria.hal.science/hal-04270368 (cit. on p. 11).

[25] T. Felicissimo. 'Second-order Church-Rosser modulo, without normalization'. In: *Proceedings of the 13th International Workshop on Confluence - IWC 2024*. IWC 2024 – 13th International Workshop on Confluence. Tallinn, Estonia, 9th July 2024. URL: https://hal.science/hal-04835978 (cit. on p. 11).

[26] T. Felicissimo and T. Winterhalter. 'Impredicativity, Cumulativity and Product Covariance in the Logical Framework Dedukti'. In: *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*. Formal Structures for Computation and Deduction. Talinn, Estonia, 2024. URL: https://hal.science/hal-04470850 (cit. on p. 12).

[27] P. G. Haselwarter, B. S. Hvass, L. L. Hansen, T. Winterhalter, C. Hriţcu and B. Spitters. 'The Last Yard: Foundational End-to-End Verification of High-Speed Cryptography'. In: *CPP 2024: Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs*. CPP 2024 - 13th ACM SIGPLAN International Conference on Certified Programs and Proofs. London, United Kingdom: ACM, 9th Jan. 2024, pp. 30–44. DOI: 10.1145/3636501.3636961. URL: https://hal.science/hal-04484598.

[28] Y. Leray, G. Gilbert, N. Tabareau and T. Winterhalter. 'The Rewster: Type Preserving Rewrite Rules for the Coq Proof Assistant'. In: International Conference on Interactive Theorem Proving (ITP 2024). Vol. 15th International Conference on Interactive Theorem Proving (ITP 2024). Tbilisi, Georgia, 2nd Sept. 2024, p. 18. DOI: 10.4230/LIPIcs.ITP.2024.26. URL: https://inria.hal.science/hal-04511667 (cit. on p. 11).

[29] T. Traversié. 'Kuroda's Translation for the λΠ-Calculus Modulo Theory and Dedukti'. In: *Electronic Proceedings in Theoretical Computer Science*. LFMTP 2024 - International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice. Vol. 404. Tallinn, Estonia, 8th July 2024, pp. 35–48. DOI: 10.4204/eptcs.404.3. URL: https://hal.science/hal-04646168 (cit. on p. 15).

[30] T. Traversié. 'Proofs for Free in the λΠ-Calculus Modulo Theory'. In: *Electronic Proceedings in Theoretical Computer Science*. LFMTP 2024 - International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice. Vol. 404. Tallinn, Estonia, 8th July 2024, pp. 49–63. DOI: 10.4204/eptcs.404.4. URL: https://hal.science/hal-04646198 (cit. on p. 15).

National peer-reviewed Conferences

[31] M. Cristiá and C. Dubois. 'Comparing EventB, log and Why3 Models of Sparse Sets'. In: JFLA 2024 - 35es Journées Francophones des Langages Applicatifs. Saint-Jacut-de-la-Mer, France, 30th Jan. 2024. URL: https://hal.science/hal-04407130 (cit. on p. 16).

Doctoral dissertations and habilitation theses

[32] L. Dubois de Prisque. 'Compositional preprocessing in Coq'. Université Paris-Saclay, 10th July 2024. URL: https://theses.hal.science/tel-04696909 (cit. on p. 5).

[33] T. Felicissimo. 'Generic bidirectional typing in a logical framework for dependent type theories'. Université Paris-Saclay, 18th Sept. 2024. URL: https://theses.hal.science/tel-04751633 (cit. on p. 5).

Reports & preprints

[34] G. Dowek. *La dynamique des noms propres*. 2024. URL: https://inria.hal.science/hal-04476056.

[35] C. Stolze, O. Hermant and R. Guillaumé. *Towards Formalization and Sharing of Atelier B Proofs with Dedukti*. 16th Jan. 2024. URL: https://hal.science/hal-04398119 (cit. on p. 14).

[36] T. Traversié. *Kuroda's Translation for Higher-Order Logic*. 27th Apr. 2024. URL: https://hal.science/hal-04561757 (cit. on p. 15).

[37] R. Vaishnav. *A Term-Patching Framework for Eliminating Definitional Equalities in Lean (Work-in-Progress)*. 2nd Dec. 2024. URL: https://inria.hal.science/hal-04813916 (cit. on p. 14).

Other scientific publications

[38] M. Taprogge. 'Computer-Assisted Proof Verification for Higher-Order Automated Reasoning within the Dedukti Framework: A Thesis submitted for the Degree of Master of Science'. Greifswald: Universität Greifswald, 18th July 2024, p. 96. URL: https://inria.hal.science/hal-04733263 (cit. on p. 13).

11.3 Cited publications

[39] M. Boespflug. 'Conception d'un noyau de vérification de preuves pour le $\lambda\Pi$-calcul modulo'. PhD thesis. École Polytechnique, 2011 (cit. on p. 4).

[40] 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*. ICALP'12. Warwick, UK: Springer-Verlag, 2012, pp. 13–15. DOI: 10.1007/978-3-642-31585-5. URL: http://dx.doi.org/10.1007/978-3-642-31585-5 (cit. on p. 4).

[41] R. Harper, D. Sannella and A. Tarlecki. 'Structured theory presentations and logic representations'. In: *Annals of Pure and Applied Logic* 67.1 (1994), pp. 113–160. DOI: https://doi.org/10.1016/0168-0072(94)90009-4. URL: https://www.sciencedirect.com/science/article/pii/0168007294900094 (cit. on p. 15).

[42] F. Rabe and K. Sojakova. 'Logical relations for a logical framework'. In: *ACM Transactions on Computational Logic* 14.4 (Nov. 2013). DOI: 10.1145/2536740.2536741. URL: https://doi.org/10.1145/2536740.2536741 (cit. on p. 15).