Activity Report 2013

Team Specfun

Symbolic Special Functions: Fast and Certified

RESEARCH CENTER
Saclay - Île-de-France

THEME
Algorithmics, Computer Algebra and Cryptology

Table of contents

1. Members **1**

2. Overall Objectives **1**

2.1. Scientific challenges, expected impact 1

2.1.1. Use Computer Algebra but Convince Users beyond Reasonable Doubt 2

2.1.2. Make Computer Algebra and Formal Proofs Help One Another 3

2.1.3. Experimental Mathematics with Special functions 3

2.2. Research axes 3

2.2.1. Computer Algebra Certified by the Coq System 4

2.2.1.1. Libraries of formalized mathematics 4

2.2.1.2. Manipulation of larger algebraic data in a proof assistant 4

2.2.1.3. Formal-proof-producing normalization algorithms 4

2.2.2. Better Symbolic Computations with Special Functions 4

2.2.2.1. Special-function integration and summation 5

2.2.2.2. Applications to experimental mathematics 5

2.2.3. Interactive and Certified Mathematical Web Sites 5

2.3. Highlights of the Year 5

3. Research Program **6**

3.1. Studying special functions by computer algebra 6

3.1.1. Equations as a data structure 6

3.1.2. Algorithms combining functions 6

3.1.3. Solving functional equations 6

3.1.4. Multi-precision numerical evaluation 7

3.1.5. Guessing heuristics 7

3.1.6. Complexity-driven design of algorithms 7

3.2. Trusted computer-algebra calculations 7

3.2.1. Encyclopedias 7

3.2.2. Computer algebra and symbolic logic 8

3.2.3. Certifying systems for computer algebra 8

3.2.4. Semantics for computer algebra 8

3.2.5. Formal proofs for symbolic components of computer-algebra systems 8

3.2.6. Formal proofs for numerical components of computer-algebra systems 8

3.3. Machine-checked proofs of formalized mathematics 8

3.3.1. Logical foundations and proof assistants 9

3.3.2. Computations in formal proofs 9

3.3.3. Large-scale computations for proofs inside the Coq system 9

3.3.4. Relevant contributions from the Mathematical Component libraries 9

3.3.5. User interaction with the proof assistant 10

4. Application Domains **10**

5. Software and Platforms **10**

5.1. Mgfun 10

5.2. DDMF 11

5.3. DynaMoW 11

5.4. Ring 11

5.5. SSReflect 11

5.6. Coqfinitgroup 11

6. New Results **11**

6.1. Creative telescoping for bivariate hyperexponential functions 11

6.2. Creative telescoping for rational functions 11

6.3. Complexity of the uncoupling of linear functional systems 12

6.4. Computation of integrals related to the Ising model 12
6.5. Non-D-finite excursions in the quarter plane 12
6.6. A human proof of Gessel's lattice path conjecture 12
6.7. Efficient algorithms for rational first integrals 12
6.8. Reactive document checking in Coq 12
6.9. Efficient normalization of ring/field expressions in Coq 13
6.10. Documentation of Coq's canonical structures 13
6.11. Maintenance and development of the SSReflect extension for Coq and its user manual 13
6.12. Efficient proof-search techniques in sequent calculus 13
6.13. A formal proof of the irrationality of $\zeta(3)$ 13
6.14. Documentation of the Mathematical Components libraries 13
7. Bilateral Contracts and Grants with Industry 13
8. Partnerships and Cooperations 14
8.1. Regional Initiatives 14
8.2. National Initiatives 14
8.2.1. ANR 14
8.2.2. Other 14
8.3. European Initiatives 14
9. Dissemination 15
9.1. Scientific Animation 15
9.2. Teaching - Supervision - Juries 15
9.2.1. Teaching 15
9.2.2. Supervision 15
9.2.3. Juries 16
9.3. Popularization 16
10. Bibliography 16

Team Specfun

Keywords: Computational Complexity, Computer Algebra, Experimental Mathematics, Formalization Of Mathematics, Special Functions

Creation of the Team: 2012 November 01.

1. Members

Research Scientists

Alin Bostan [Inria, Researcher]
Frédéric Chyzak [Team leader, Inria, Researcher]
Assia Mahboubi [Team vice-leader, Inria, Researcher]
Enrico Tassi [Inria, Researcher]

Faculty Member

Philippe Dumas [Min. de l'Éducation Nationale, Professor]

PhD Students

Augustin Barillec [ÉNS Lyon, from Sep 2013]
Louis Dumont [École Polytechnique, from Oct 2013]
Pierre Lairez [École Polytechnique]
Basile Morcrette [Univ. Paris VI, until Oct 2013]

Administrative Assistant

Valérie Lecomte [Inria, Administrative Assistant]

2. Overall Objectives

2.1. Scientific challenges, expected impact

Computer-algebra systems have been advertised for decades as software for "doing mathematics by computer" [66]. For instance, computer-algebra libraries can uniformly generate a corpus of mathematical properties about special functions so as to display them on an interactive website. This was recently shown by the computer-algebra component of the team [22]. Such an automated generation significantly increases the reliability of the mathematical corpus, in comparison to the content of existing static authoritative handbooks. The importance of the validity of these contents can be measured by the very wide audience that such handbooks have had, to the point that a book like [17] remains one of the most cited mathematical publications ever and has motivated the 10-year-long project of writing its successor [19]. However, can the mathematics produced "by computer" be considered as *true* mathematics? More specifically, whereas it is nowadays well established that the computer helps in discovering and observing new mathematical phenomenons, can the mathematical statements produced with the aid of the computer and the mathematical results computed by it be accepted as valid mathematics, that is, as having the status of mathematical *proofs*? Beyond the reported weaknesses or controversial design choices of mainstream computer-algebra systems, the issue is more of an epistemological nature. It will not find its solution even in the advent of the ultimate computer-algebra system: the social process of peer-reviewing just falls short of evaluating the results produced by computers, as reported by Th. Hales [45] after the publication of his proof of the Kepler Conjecture about sphere packing.

A natural answer to this deadlock is to move to an alternative kind of mathematical software and to use a proof assistant to check the correctness of the desired properties or formulas. The recent success of large-scale formalization projects, like the Four-Color Theorem of graph theory [40], the above-mentioned Kepler Conjecture [45], and, very recently, the Odd Order Theorem of group theory[1], have increased the understanding of the appropriate software-engineering methods for this peculiar kind of programming. For computer algebra, this legitimates a move to proof assistants now.

The Dynamic Dictionary of Mathematical Functions[2] (DDMF) [22] is an online computer-generated handbook of mathematical functions that ambitions to serve as a reference for a broad range of applications. This software was developed by the computer-algebra component of the team as a project[3] of the MSR–INRIA Joint Centre. It bases on a library for the computer-algebra system Maple, Algolib[4], whose development started 20 years ago in ÉPI Algorithms[5]. As suggested by the constant questioning of certainty by new potential users, DDMF deserves a formal guarantee of correctness of its content, on a level that proof assistants can provide. Fortunately, the maturity of special-functions algorithms in Algolib makes DDMF a stepping stone for such a formalization: it provides a well-understood and unified algorithmic treatment, without which a formal certification would simply be unreachable.

The formal-proofs component of the team emanates from another project of the MSR–INRIA Joint Centre, namely the Mathematical Components project (MathComp)[6]. Over the last six years, the MathComp group endeavoured to develop computer-checked libraries of formalized mathematics, using the Coq proof assistant [62]. The methodological aim of the project was to understand the design methods leading to successful large-scale formalizations. The work culminated with the recent completion of a formal proof of the Odd Order Theorem, resulting in the largest corpus of algebraic theories ever machine-checked with a proof assistant and a whole methodology to effectively combine these components in order to tackle complex formalizations. In particular, these libraries provide a good number of the many algebraic objects needed to reason about special functions and their properties, like rational numbers, iterated sums, polynomials, and a rich hierarchy of algebraic structures.

The present team takes benefit from these recent advances to explore the formal certification of the results collected in DDMF. The aim of this project is to concentrate the formalization effort on this delimited area, building on DDMF and the Algolib library, as well as on the Coq system [62] and on the libraries developed by MathComp.

2.1.1. Use Computer Algebra but Convince Users beyond Reasonable Doubt

The following few opinions on computer algebra are, we believe, typical of computer-algebra users' doubts and difficulties when using computer-algebra systems:

- Fredrik Johansson, expert in the multi-precision numerical evaluation of special functions and in fast computer-algebra algorithms, writes on his blog [51]: "Mathematica is great for cross-checking numerical values, but it's not unusual to run into bugs, so *triple checking is a good habit*." One answer in the discussion is: "We can claim that Mathematica has [...] *an impossible to understand semantics*: If Mathematica's output is wrong then change the input. If you don't like the answer, change the question. That seems to be the philosophy behind."
- A professor's advice to students [58] on using Maple: "You may wish to use Maple to check your homework answers. If you do then keep in mind that Maple sometimes gives the *wrong answer, usually because you asked incorrectly, or because of niceties of analytic continuation*. You may even be bitten by an occasional Maple bug, though that has become fairly unlikely. Even with as powerful a tool as Maple you will still *have to devise your own checks* and you will still have to think."

[1] http://www.msr-inria.inria.fr/news/the-formalization-of-the-odd-order-theorem-has-been-completed-the-20-septembre-2012/
[2] http://ddmf.msr-inria.inria.fr/
[3] http://www.msr-inria.inria.fr/projects/dynamic-dictionary-of-mathematical-functions/
[4] http://algo.inria.fr/libraries/
[5] http://algo.inria.fr/
[6] http://www.msr-inria.fr/projects/mathematical-components/

- Jacques Carette, former head of the maths group at Maplesoft, about a bug [18] when asking Maple to take the limit `limit(f(n) * exp(-n), n = infinity)` for an undetermined function `f`: "The problem is that there is an *implicit assumption in the implementation* that unknown functions do not 'grow too fast'."

As explained by the expert views above, complaints by computer-algebra users are often due to their misunderstanding of what a computer-algebra systems is, namely a purely syntactic tool for calculations, that the user must complement with a semantics. Still, robustness and consistency of computer-algebra systems are not ensured as of today, and, whatever Zeilberger may provocatively say in his opinion 94 [67], a firmer logical foundation is necessary. Indeed, the fact is that many "bugs" in a computer-algebra system cannot be fixed by just the usual debugging method of tracking down the faulty lines in the code. It is sort of "by design": assumptions that too often remain implicit are really needed by the design of symbolic algorithms and cannot easily be expressed in the programming languages used in computer algebra A similar certification initiative has already been undertaken in the domain of numerical computing, in a successful manner [49], [25]. It is natural to undertake a similar approach for computer algebra.

2.1.2. Make Computer Algebra and Formal Proofs Help One Another

Some of the mathematical objects that interest us are still totally untouched by formalization. When implementing them and their theory inside a proof assistant, we have to deal with the pervasive discrepancy between the published literature and the actual implementation of computer-algebra algorithms. Interestingly, this forces us to clarify our computer-algebraic view on them, and possibly make us discover holes lurking in published (human) proofs. We are therefore convinced that the close interaction of researchers from both fields, which is what we do in this team, is a strong asset.

For a concrete example, the core of Zeilberger's creative telescoping manipulates rational functions up to simplifications. In summation applications, checking that these simplifications do not hide problematic divisions by 0 is most often left to the reader. In the same vein, in the case of integrals, the published algorithms do not check the convergence of all integrals, especially in intermediate calculations. Such checks are again left to the readers. In general, we expect to revisit the existing algorithms to ensure that they are meaningful for genuine mathematical sequences or functions, and not only for algebraic idealizations.

Another big challenge in this project originates in the scientific difference between computer algebra and formal proofs. Computer algebra seeks speed of calculation on *concrete instances* of algebraic data structures (polynomials, matrices, etc). For their part, formal proofs manipulate symbolic expressions in terms of *abstract variables* understood to represent generic elements of algebraic data structures. In view of this, a continuous challenge is to develop the right, hybrid thinking attitude that is able to effectively manage concrete and abstract values simultaneously, alternatively computing and proving with them.

2.1.3. Experimental Mathematics with Special functions

Applications in combinatorics and mathematical physics frequently involve equations of so high orders and so large sizes, that computing or even storing all their coefficients is impossible on existing computers. Making this tractable is an extraordinary challenge. The approach we believe in is to design algorithms of good, ideally quasi-optimal, complexity in order to extract precisely the required data from the equations, while avoiding the computationally intractable task of completely expanding them into an explicit representation.

Typical applications with expected high impact are the automatic discovery and proof of results in combinatorics and mathematical physics for which human proofs are currently unattainable.

2.2. Research axes

The implementation of certified symbolic computations on special functions in the Coq proof assistant requires both investigating new formalization techniques and renewing the traditional computer-algebra viewpoint on these standard objects. Large mathematical objects typical of computer algebra occur during formalization, which also requires us to improve the efficiency and ergonomics of Coq. In order to feed this interdisciplinary activity with new motivating problems, we additionally pursue a research activity oriented towards experimental mathematics in application domains that involve special functions. We expect these applications to pose new algorithmic challenges to computer algebra, which in turn will deserve a formal-certification effort. Finally, DDMF is the motivation and the showcase of our progress on the certification of these computations. While striving to provide a formal guarantee of the correctness of the information it displays, we remain keen on enriching its mathematical content by developing new computer-algebra algorithms.

2.2.1. Computer Algebra Certified by the Coq System

Our formalization effort consists in organizing a cooperation between a computer-algebra system and a proof assistant. The computer-algebra system is used to produce efficiently algebraic data, which are later processed by the proof assistant. The success of this cooperation relies on three main ingredients.

2.2.1.1. Libraries of formalized mathematics

The appropriate framework for the study of efficient algorithms for special functions is *algebraic*. Representing algebraic theories as Coq formal libraries takes benefit from the methodology emerging from the success of ambitious projects like the formal proof of a major classification result in finite-group theory (the Odd Order Theorem) [38].

Yet, a number of the objects we need to formalize in the present context has never been investigated using any interactive proof assistant, despite being considered as commonplaces in computer algebra. For instance there is up to our knowledge no available formalization of the theory of non-commutative rings, of the algorithmic theory of special-functions closures, or of the asymptotic study of special functions. We expect our future formal libraries to prove broadly reusable in later formalizations of seemingly unrelated theories.

2.2.1.2. Manipulation of larger algebraic data in a proof assistant

Another peculiarity of the mathematical objects we are going to manipulate with the Coq system is their size. In order to provide a formal guarantee on the data displayed by DDMF, two related axes of research have to be pursued. First, efficient algorithms dealing with these large objects have to be programmed and run in Coq. Recent evolutions of the Coq system to improve the efficiency of its internal computations [20], [23] make this objective reachable. Still, how to combine the aforementioned formalization methodology with these cutting-edge evolutions of Coq remains one of the prospective aspects of our project. A second need is to help users *interactively* manipulate large expressions occurring in their conjectures, an objective for which little has been done so far. To address this need, we work on improving the ergonomics of the system in two ways: first, ameliorating the reactivity of Coq in its interaction with the user; second, designing and implementing extensions of its interface to ease our formalization activity. We expect the outcome of these lines of research to be useful to a wider audience, interested in manipulating large formulas on topics possibly unrelated to special functions.

2.2.1.3. Formal-proof-producing normalization algorithms

Our algorithm certifications inside Coq intends to simulate well-identified components of our Maple packages, possibly by reproducing them in Coq. It would however not have been judicious to re-implement them inside Coq, since for a number of its components, the output of the algorithm is more easily checked than found, like for instance the solving of a linear system. Rather, we delegate the discovery of the solutions to an external, untrusted oracle like Maple. Trusted computations inside Coq then formally validate the correctness of the a priori untrusted output. More often than not, this consists in implementing and executing normalization procedures *inside* Coq. A challenge of this automation is to make sure they go to scale while remaining efficient, which requires a Coq version of non-trivial computer-algebra algorithms. A good example we expect to work on is a non-commutative generalization of the normalization procedure for elements of rings [44].

2.2.2. Better Symbolic Computations with Special Functions

Generally speaking, we design algorithms for manipulating special functions symbolically, whether univariate or with parameters, and for extracting algorithmically any kind of algebraic and analytic information from them, notably asymptotic. Beyond this, the heart of our research is concerned with parametrised definite summations and integrations. These very expressive operations have far-ranging applications, for instance, to the computation of integral transforms (Laplace, Fourier) or to the solution of combinatorial problems expressed via integrals (coefficient extractions, diagonals). The algorithms that we design for them need to really operate on the level of linear functional systems, differential and of recurrence.

2.2.2.1. Special-function integration and summation

Our long-term goal is to design fast algorithms for a general method for special-function integration (*creative telescoping*), and make them applicable to general special-function inputs. Still, our strategy is to proceed with simpler, more specific classes first (rational functions, then algebraic functions, hyperexponential functions, D-finite functions, non-D-finite functions; two variables, then many variables); as well, we isolate analytic questions by first considering types of integration with a more purely algebraic flavor (constant terms, algebraic residues, diagonals of combinatorics). In particular, we expect to extend our recent new approach [28] to more general classes (algebraic with nested radicals, for example). Homologous problems for summation will be addressed as well.

2.2.2.2. Applications to experimental mathematics

The algorithms of good complexity mentioned in the previous paragraphs naturally help us deal with applications that involve equations of high orders and large sizes.

With regard to combinatorics, we expect to advance the algorithmic classification of combinatorial classes like walks and urns. Here, the goal is to determine if enumerating generating series are rational, algebraic, or D-finite, for example. Physical problems whose modelling involves special-function integrals comprise the study of models of statistical mechanics, like the Ising model for ferro-magnetism, or questions related to Hamiltonian systems.

Number theory is another promising domain of applications. Here, we attempt an experimental approach to the automated certification of integrality of the coefficients of mirror maps for Calabi–Yau manifolds. This could also involve the discovery of new Calabi–Yau operators and the certification of the existing ones. We also plan to algorithmically discover and certify new recurrences yielding good approximants needed in irrationality proofs.

It is to be noted that in all of these application domains, we would so far use general algorithms, as was done in earlier works of ours [27], [31], [30]. To push the scale of applications further, we plan to consider in each case the specifics of the application domain to tailor our algorithms.

2.2.3. Interactive and Certified Mathematical Web Sites

In continuation of our past project of an encyclopedia at http://ddmf.msr-inria.inria.fr/, we ambition to both enrich and certify the formulas about the special functions that we provide online. For each function, our website shows its essential properties and the mathematical objects attached to it, which are often infinite in nature (numerical evaluations, asymptotic expansions). An interactive presentation has the advantage of allowing for adaption to the user's needs. More advanced content will broaden the encyclopedia:

- the algorithmic discussion of equations with parameters, leading to certified automatic case analysis based on arithmetic properties of the parameters;
- lists of summation and integral formulas involving special functions, including validity conditions on the parameters;
- guaranteed large-precision numerical evaluations.

2.3. Highlights of the Year

This year, we complete a first work emblematic of the interdisciplinary activity of the team: a computer-algebra based formal proof of irrationality of the mathematical constant $\zeta(3)$, that is, the evaluation at 3 of the Riemann zeta function of number theory. This motivated collateral enhancements of libraries for the interactive theorem prover Coq. This is described in more details in the new results.

3. Research Program

3.1. Studying special functions by computer algebra

Computer algebra manipulates symbolic representations of exact mathematical objects in a computer, in order to perform computations and operations like simplifying expressions and solving equations for "closed-form expressions". The manipulations are often fundamentally of algebraic nature, even when the ultimate goal is analytic. The issue of efficiency is a particular one in computer algebra, owing to the extreme swell of the intermediate values during calculations.

Our view on the domain is that research on the algorithmic manipulation of special functions is anchored between two paradigms:

- adopting linear differential equations as the right data structure for special functions,
- designing efficient algorithms in a complexity-driven way.

It aims at four kinds of algorithmic goals:

- algorithms combining functions,
- functional equations solving,
- multi-precision numerical evaluations,
- guessing heuristics.

This interacts with three domains of research:

- computer algebra, meant as the search for quasi-optimal algorithms for exact algebraic objects,
- symbolic analysis/algebraic analysis;
- experimental mathematics (combinatorics, mathematical physics, ...).

This view is made explicit in the present section.

3.1.1. Equations as a data structure

Numerous special functions satisfy linear differential and/or recurrence equations. Under a mild technical condition, the existence of such equations induces a finiteness property that makes the main properties of the functions decidable. We thus speak of *D-finite functions*. For example, 60 % of the chapters in the handbook [17] describe D-finite functions. In addition, the class is closed under a rich set of algebraic operations. This makes linear functional equations just the right data structure to encode and manipulate special functions. The power of this representation was observed in the early 1990s [68], leading to the design of many algorithms in computer algebra. Both on the theoretical and algorithmic sides, the study of D-finite functions shares much with neighbouring mathematical domains: differential algebra, D-module theory, differential Galois theory, well as their counterparts for recurrence equations.

3.1.2. Algorithms combining functions

Differential/recurrence equations that define special functions can be recombined [68] to define: additions and products of special functions; compositions of special functions; integrals and sums involving special functions. Zeilberger's fast algorithm for obtaining recurrences satisfied by parametrised binomial sums was developed in the early 1990s already [69]. It is the basis of all modern definite summation and integration algorithms. The theory was made fully rigorous and algorithmic in later works, mostly by a group in RISC (Linz, Austria) and by members of the team [57], [65], [34], [32], [33], [52]. The past ÉPI Algorithms contributed several implementations (*gfun* [60], *Mgfun* [34]).

3.1.3. Solving functional equations

Encoding special functions as defining linear functional equations postpones some of the difficulty of the problems to a delayed solving of equations. But at the same time, solving (for special classes of functions) is a sub-task of many algorithms on special functions, especially so when solving in terms of polynomial or rational functions. A lot of work has been done in this direction in the 1990s; more intensively since the 2000s, solving differential and recurrence equations in terms of special functions has also been investigated.

3.1.4. Multi-precision numerical evaluation

A major conceptual and algorithmic difference exists for numerical calculations between data structures that fit on a machine word and data structures of arbitrary length, that is, *multi-precision* arithmetic. When multi-precision floating-point numbers became available, early works on the evaluation of special functions were just promising that "most" digits in the output were correct, and performed by heuristically increasing precision during intermediate calculations, without intended rigour. The original theory has evolved in a twofold way since the 1990s: by making computable all constants hidden in asymptotic approximations, it became possible to guarantee a *prescribed* absolute precision; by employing state-of-the-art algorithms on polynomials, matrices, etc, it became possible to have evaluation algorithms in a time complexity that is not more than a few times the output size. On the implementation side, several original works exist, one of which (*NumGfun* [56]) is used in our DDMF.

3.1.5. Guessing heuristics

"Differential approximation", or "Guessing", is an operation to get an ODE likely to be satisfied by a given approximate series expansion of an unknown function. This has been used at least since the 1970s and is a key stone in spectacular applications in experimental mathematics [31]. All this is based on subtle algorithms for Hermite–Padé approximants [21]. Moreover, guessing can at times be complemented by proven quantitative results that turn the heuristics into an algorithm [29]. This is a promising algorithmic approach that deserves more attention than it has received so far.

3.1.6. Complexity-driven design of algorithms

The main concern of computer algebra has long been to prove the feasibility of a given problem, that is, to show the existence of an algorithmic solution for it. However, with the advent of faster and faster computers, complexity results have ceased to be of theoretical interest only. Nowadays, a large track of works in computer algebra is interested in developing fast algorithms, with time complexity as close as possible to linear in their output size. After most of the more pervasive objects like integers, polynomials, and matrices have been endowed with fast algorithms for the main operations on them [39], the community, including ourselves, started to turn its attention to differential and recurrence objects in the 2000s. The subject is still not as developed as in the commutative case, and a major challenge remains to understand the combinatorics behind summation and integration. On the methodological side, several paradigms occur repeatedly in fast algorithms: "divide and conquer" to balance calculations, "evaluation and interpolation" to avoid intermediate swell of data, etc. [26].

3.2. Trusted computer-algebra calculations

3.2.1. Encyclopedias

Handbooks collecting mathematical properties aim at serving as reference, therefore trusted, documents. The decision of several authors or maintainers of such knowledge bases to move from paper books [17], [19], [61] to websites and wikis [7] allows for a more collaborative effort in proof reading. Another step toward further confidence is to manage to generate the content of an encyclopedia by computer-algebra programs, as is the case with the Wolfram Functions Site [8] or DDMF [9]. Yet, due to the lingering doubts about computer-algebra systems, some encyclopedias propose both cross-checking by different systems and handwritten companion paper proofs of their content [10]. As of today, there is no encyclopedia certified with formal proofs.

[7] for instance http://dlmf.nist.gov/ for special functions or http://oeis.org/ for integer sequences

[8] http://functions.wolfram.com/

[9] http://ddmf.msr-inria.inria.fr/

[10] http://129.81.170.14/~vhm/Table.html

3.2.2. Computer algebra and symbolic logic

Several attempts have been made in order to extend existing computer-algebra systems with symbolic manipulations of logical formulas. Yet, these works are more about extending the expressivity of computer-algebra systems than about improving the standards of correctness and semantics of the systems. Conversely, several projects have addressed the communication of a proof system with a computer-algebra system, resulting in an increased automation available in the proof system, to the price of the uncertainty of the computations performed by this oracle.

3.2.3. Certifying systems for computer algebra

More ambitious projects have tried to design a new computer-algebra system providing an environment where the user could both program efficiently and elaborate formal and machine-checked proofs of correctness, by calling a general-purpose proof assistant like the Coq system. This approach requires a huge manpower and a daunting effort in order to re-implement a complete computer-algebra system, as well as the libraries of formal mathematics required by such formal proofs.

3.2.4. Semantics for computer algebra

The move to machine-checked proofs of the mathematical correctness of the output of computer-algebra implementations demands a prior clarification about the often implicit assumptions on which the presumably correctly implemented algorithms rely. Interestingly, this preliminary work, which could be considered as independent from a formal certification project, is seldom precise or even available in the literature.

3.2.5. Formal proofs for symbolic components of computer-algebra systems

A number of authors have investigated ways to organize the communication of a chosen computer-algebra system with a chosen proof assistant in order to certify specific components of the computer-algebra systems, experimenting various combinations of systems and various formats for mathematical exchanges. Another line of research consists in the implementation and certification of computer-algebra algorithms inside the logic [64], [44], [53] or as a proof-automation strategy. Normalization algorithms are of special interest when they allow to check results possibly obtained by an external computer-algebra oracle [37]. A discussion about the systematic separation of the search for a solution and the checking of the solution is already clearly outlined in [50].

3.2.6. Formal proofs for numerical components of computer-algebra systems

Significant progress has been made in the certification of numerical applications by formal proofs. Libraries formalizing and implementing floating-point arithmetic as well as large numbers and arbitrary-precision arithmetic are available. These libraries are used to certify floating-point programs, implementations of mathematical functions and for applications like hybrid systems.

3.3. Machine-checked proofs of formalized mathematics

To be checked by a machine, a proof needs to be expressed in a constrained, relatively simple formal language. Proof assistants provide facilities to write proofs in such languages. But, as merely writing, even in a formal language, does not constitute a formal proof just per se, proof assistants also provide a proof checker: a small and well-understood piece of software in charge of verifying the correctness of arbitrarily large proofs. The gap between the low-level formal language a machine can check and the sophistication of an average page of mathematics is conspicuous and unavoidable. Proof assistants try to bridge this gap by offering facilities, like notations or automation, to support convenient formalization methodologies. Indeed, many aspects, from the logical foundation to the user interface, play an important role in the feasibility of formalized mathematics inside a proof assistant.

3.3.1. Logical foundations and proof assistants

While many logical foundations for mathematics have been proposed, studied, and implemented, type theory is the one that has been more successfully employed to formalize mathematics, to the notable exception of the Mizar system [54], which is based on set theory. In particular, the calculus of construction (CoC) [35] and its extension with inductive types (CIC) [36], have been studied for more than 20 years and been implemented by several independent tools (like Lego, Matita, and Agda). Its reference implementation, Coq [62], has been used for several large-scale formalizations projects (formal certification of a compiler back-end; four-color theorem). Improving the type theory underlying the Coq system remains an active area of research. Other systems based on different type theories do exist and, whilst being more oriented toward software verification, have been also used to verify results of mainstream mathematics (prime-number theorem; Kepler conjecture).

3.3.2. Computations in formal proofs

The most distinguishing feature of CoC is that computation is promoted to the status of rigorous logical argument. Moreover, in its extension CIC, we can recognize the key ingredients of a functional programming language like inductive types, pattern matching, and recursive functions. Indeed, one can program effectively inside tools based on CIC like Coq. This possibility has paved the way to many effective formalization techniques that were essential to the most impressive formalizations made in CIC.

Another milestone in the promotion of the computations-as-proofs feature of Coq has been the integration of compilation techniques in the system to speed up evaluation. Coq can now run realistic programs in the logic, and hence easily incorporates calculations into proofs that demand heavy computational steps.

Because of their different choice for the underlying logic, other proof assistants have to simulate computations outside the formal system, and indeed fewer attempts to formalize mathematical proofs involving heavy calculations have been made in these tools. The only notable, but still unfinished, exception, the Kepler conjecture, required a significant work to optimize the rewriting engine that simulates evaluation in Isabelle/HOL.

3.3.3. Large-scale computations for proofs inside the Coq system

Programs run and proved correct inside the logic are especially useful for the conception of automated decision procedures. To this end, inductive types are used as an internal language for the description of mathematical objects by their syntax, thus enabling programs to reason and compute by case analysis and recursion on symbolic expressions.

The output of complex and optimized programs external to the proof assistant can also be stamped with a formal proof of correctness when their result is easier to *check* than to *find*. In that case one can benefit from their efficiency without compromising the level of confidence on their output at the price of writing and certify a checker inside the logic. This approach, which has been successfully used in various contexts, is very relevant to the present research team.

3.3.4. Relevant contributions from the Mathematical Component libraries

Representing abstract algebra in a proof assistant has been studied for long. The libraries developed by the MathComp team for the proof of the Odd Order Theorem provide a rather comprehensive hierarchy of structures; however, they originally feature a large number of instances of structures that they need to organize. On the methodological side, this hierarchy is an incarnation of an original work [38] based on various mechanisms, primarily type inference, typically employed in the area of programming languages. A large amount of information that is implicit in handwritten proofs, and that must become explicit at formalization time, can be systematically recovered following this methodology.

Small-scale reflection [41] is another methodology promoted by the MathComp team. Its ultimate goal is to ease formal proofs by systematically dealing with as many bureaucratic steps as possible, by automated computation. For instance, as opposed to the style advocated by Coq's standard library, decidable predicates are systematically represented using computable boolean functions: comparison on integers is expressed as program, and to state that $a \leq b$ one compares the output of this program run on a and b with $true$. In many cases, for example when a and b are values, one can prove or disprove the inequality by pure computation.

The MathComp library was consistently designed after uniform principles of software engineering. These principles range from simple ones, like naming conventions, to more advanced ones, like generic programming, resulting in a robust and reusable collection of formal mathematical components. This large body of formalized mathematics covers a broad panel of algebraic theories, including of course advanced topics of finite group theory, but also linear algebra, commutative algebra, Galois theory, and representation theory. We refer the interested reader to the online documentation of these libraries [63], which represent about 150,000 lines of code and include roughly 4,000 definitions and 13,000 theorems.

Topics not addressed by these libraries and that might be relevant to the present project include real analysis and differential equations. The most advanced work of formalization on these domains is available in the HOL-Light system [46], [47], [48], although some existing developments of interest [24], [55] are also available for Coq. Another aspect of the MathComp libraries that needs improvement, owing to the size of the data we manipulate, is the connection with efficient data structures and implementations, which only starts to be explored.

3.3.5. User interaction with the proof assistant

The user of a proof assistant describes the proof he wants to formalize in the system using a textual language. Depending on the peculiarities of the formal system and the applicative domain, different proof languages have been developed. Some proof assistants promote the use of a declarative language, when the Coq and Matita systems are more oriented toward a procedural style.

The development of the large, consistent body of MathComp libraries has prompted the need to design an alternative and coherent language extension for the Coq proof assistant [43], [42], enforcing the robustness of proof scripts to the numerous changes induced by code refactoring and enhancing the support for the methodology of small-scale reflection.

The development of large libraries is quite a novelty for the Coq system. In particular any long-term development process requires the iteration of many refactoring steps and very little support is provided by most proof assistants, with the notable exception of Mizar [59]. For the Coq system, this is an active area of research.

4. Application Domains

4.1. Experimental mathematics with special functions

Applications in combinatorics and mathematical physics frequently involve equations of so high orders and so large sizes, that computing or even storing all their coefficients is impossible on existing computers. Making this tractable is another challenge of our project. The approach we believe in is to design algorithms of good, ideally quasi-optimal, complexity in order to extract precisely the required data from the equations, while avoiding the computationally intractable task of completely expanding them into an explicit representation.

Typical applications with expected high impact are the automatic discovery and proof of results in combinatorics and mathematical physics for which human proofs are currently unattainable.

5. Software and Platforms

5.1. Mgfun

(1994–): Maple package for symbolic summation, integration, and other closure properties of multivariate special functions.

Now distributed as part of Algolib, a collection of packages for combinatorics and manipulations of special functions, available at http://algo.inria.fr/libraries/.

5.2. DDMF

(2007–): Web site consisting of interactive tables of mathematical formulas on elementary and special functions. The formulas are automatically generated by OCaml and computer-algebra routines. Users can ask for more terms of the expansions, more digits of the numerical values, proofs of some of the formulas, etc. See http://ddmf.msr-inria.inria.fr/.

5.3. DynaMoW

(2007–): Programming tool for controlling the generation of mathematical websites that embed dynamical mathematical contents generated by computer-algebra calculations. Written in OCaml. See http://ddmf.msr-inria.inria.fr/DynaMoW/.

5.4. Ring

(2004–): Coq normalization tool and decision procedure for expressions in commutative ring theories. Written in Coq and OCaml. Integrated in the standard distribution of the Coq proof assistant since 2005.

5.5. SSReflect

(2006–): Extension of the language of the Coq system. Originally written by G. Gonthier for his formal proof of the Four-Color Theorem. A. Mahboubi and E. Tassi participate to its development, maintenance, distribution, user support and have written its user manual. See http://www.msr-inria.fr/projects/mathematical-components/.

5.6. Coqfinitgroup

(2006–): Coq libraries that cover the mechanization of the proof of the Odd Order Theorem. Stable libraries are distributed with the SSReflect extension. A. Mahboubi is one of the main contributors to the code and its documentation. E. Tassi contributed to the design of core data structures and to parts of the formalization. A formal proof was completed in September 2012, and the content of the libraries, under continued improvements in view of potential reuse, is available online at http://www.msr-inria.fr/projects/mathematical-components/.

6. New Results

6.1. Creative telescoping for bivariate hyperexponential functions

In [8], we gave a new algorithm for the symbolic integration of bivariate hyperexponential functions, which outperforms state-of-the-art implementations like Maple's function *DEtools[Zeilberger]*. The approach was to extend Hermite's reduction for rational functions and the Hermite-like reduction for hyperexponential functions in a suitable way. A key feature of the algorithm is that it can avoid the costly computation of certificates.

6.2. Creative telescoping for rational functions

In [10] we described a precise and elementary algorithmic version of the Griffiths–Dwork method for the creative telescoping of rational functions. This leads to bounds on the order and degree of the coefficients of the differential equation, and to the first complexity result which is single exponential in the number of variables. One of the important features of the algorithm is that it does not need to compute certificates. The approach is vindicated by a prototype implementation.

6.3. Complexity of the uncoupling of linear functional systems

Uncoupling algorithms transform a linear differential system of first order into one or several scalar differential equations. We examined in [9] two approaches to uncoupling: the cyclic-vector method (*CVM*) and the Danilevski-Barkatou-Zürcher algorithm (*DBZ*). We gave tight size bounds on the scalar equations produced by *CVM*, and designed a fast variant of *CVM* whose complexity is quasi-optimal with respect to the output size. We exhibited a strong structural link between *CVM* and *DBZ* enabling to show that, in the generic case, *DBZ* has polynomial complexity and that it produces a single equation, strongly related to the output of *CVM*. We proved that algorithm *CVM* is faster than *DBZ* by almost two orders of magnitude, and provided experimental results that validate the theoretical complexity analyses.

6.4. Computation of integrals related to the Ising model

We showed in [2] that the n-fold integrals of the magnetic susceptibility of the Ising model, as well as various other n-fold integrals of the "Ising class", or n-fold integrals from enumerative combinatorics, like lattice Green functions, correspond to a distinguished class of functions generalising algebraic functions: they are actually diagonals of rational functions. This algebraic structure explains many remarkable properties of the integrals of the Ising class.

6.5. Non-D-finite excursions in the quarter plane

The number of excursions (finite paths starting and ending at the origin) having a given number of steps and obeying various geometric constraints is a classical topic of combinatorics and probability theory. We proved in [3] that the sequence of numbers of excursions in the quarter plane corresponding to a nonsingular step set $S \subseteq \{0, \pm 1\}^2$ with infinite group does not satisfy any nontrivial linear recurrence with polynomial coefficients. Accordingly, in those cases, the trivariate generating function of the numbers of walks with given length and prescribed ending point is not D-finite. This solves an open problem in the field of lattice path combinatorics.

6.6. A human proof of Gessel's lattice path conjecture

Gessel walks are planar walks confined to the positive quarter plane, that move by unit steps in any of the following directions: West, North-East, East, and South-West. In 2001, Ira Gessel conjectured a closed-form expression for the number of Gessel walks of a given length starting and ending at the origin. In 2008, Kauers, Koutschan, and Zeilberger gave a computer-aided proof of this conjecture. The same year, Bostan and Kauers showed, using again computer algebra tools, that the trivariate generating function of Gessel walks is algebraic. We proposed in [15] the first "human proofs" of these results. They are derived from a new expression for the generating function of Gessel walks.

6.7. Efficient algorithms for rational first integrals

We presented in [14] fast algorithms for computing rational first integrals with bounded degree of a planar polynomial vector field. Our approach is inspired by an idea of Ferragut and Giacomini. We improve upon their work by proving that rational first integrals can be computed via systems of linear equations instead of systems of quadratic equations. This leads to a probabilistic algorithm with arithmetic complexity $\tilde{O}(N^{2\omega})$ and to a deterministic algorithm solving the problem in $\tilde{O}(d^2 N^{2\omega+1})$ arithmetic operations, where N denotes the given bound for the degree of the rational first integral, and where $d \leq N$ is the degree of the vector field, and ω the exponent of linear algebra. By comparison, the best previous algorithm uses at least $d^{\omega+1} N^{4\omega+4}$ arithmetic operations. The new algorithms are very efficient in practice.

6.8. Reactive document checking in Coq

In an effort to improve the reactivity of Coq, the way it processes and checks a single document has been completely redesigned [7]. The current development version is able to reschedule the tasks to be performed in order to minimize the time required to give interactive feedback to the user. On typical documents taken from the formal proof of the Odd Order Theorem, the worst reaction time of the tool dropped from 5 minutes to 9 seconds. This improvement will be part of the next stable release of the Coq system.

6.9. Efficient normalization of ring/field expressions in Coq

The implementation of Coq's proof commands for manipulation of ring/field expressions has been improved in response to the demand for better efficiency that emerged in the formalization of Apéry's irrationality proof of $\zeta(3)$. The data structure used for the abstract syntax tree of ring/field expressions has been refined to enable a more efficient and more precise interpretation into concrete ring/field expressions. Moreover the collection of non-nullity conditions for denominators in a field expressions has been speeded up, making the type-checking time of a field normalization proof not be dominated by this collecting phase.

6.10. Documentation of Coq's canonical structures

The device employed to model a hierarchy of algebraic structures with overloaded notations in Coq has been documented in [6] and in the user manual of the tool.

6.11. Maintenance and development of the SSReflect extension for Coq and its user manual

The Small Scale Reflection extension of Coq has been maintained together with its user manual. Some new linguistic constructs to model non-structural reasoning and to enable the user to better factor out repeated arguments have been developed and documented. Some language constructs have been made compatible with the type-classes mechanism offered by Coq. The release of version 1.5 has been prepared.

6.12. Efficient proof-search techniques in sequent calculus

We have proposed in [11] a sequent calculus which is focussed, polarized, and parameterized by an abstract notion of theory. This new combination of features aims at proposing a framework which is adapted to the simulation in sequent calculus of efficient, general-purpose decision procedures (tableaux methods, satisfiability, ...) that can interact with theory-specific decision procedures (for linear arithmetics, arrays, ...). In particular we propose a tight simulation of the Davis–Putnam–Logemann–Loveland algorithm modulo theory, and show how to simulate some advanced optimizations that are crucial to realistic implementations of SMT solvers.

6.13. A formal proof of the irrationality of $\zeta(3)$

We have obtained a formal proof, machine-checked by the Coq proof assistant, of the irrationality of the constant $\zeta(3)$, under the single assumption of the asymptotic behavior of the least common multiple of the first n natural numbers. The core of this formal proof is based on (untrusted) computer-algebra calculations performed outside the proof assistant with the Algolib Maple library. Then, we verify formally and a posteriori the desired properties of the objects computed by Maple and complete the proof of irrationality.

6.14. Documentation of the Mathematical Components libraries

The approach to finite-group theory adopted in the libraries formalizing in Coq the proof of the Odd Order Theorem has been documented in [5].

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Grants with Industry

The team is involved in two Common Research Agreements in the MSR–INRIA Joint Centre:

- *DDMF (Dynamic Dictionary of Mathematical Functions).*
  Goal: Automate exact computations of the mathematical formulas on the special functions of mathematical analysis and present them on an interactive mathematical dictionary online.

Leader: F. Chyzak. Participants: A. Bostan, P. Lairez.
Website: http://ddmf.msr-inria.inria.fr/.

- *Mathematical Components*.
  Goal: Investigate the design of large-scale, modular and reusable libraries of formalized mathematics. Developed using the Coq proof assistant. This project successfully formalized the proof of the Odd Order Theorem, resulting in a corpus of libraries related to various areas of algebra.
  Leader: G. Gonthier (MSR Cambridge). Participants: A. Mahboubi, E. Tassi.
  Website: http://www.msr-inria.fr/projects/mathematical-components/.

8. Partnerships and Cooperations

8.1. Regional Initiatives

- Project *Coquelicot*, funded jointly by the Fondation de Coopération Scientifique "Campus Paris-Saclay" and Digiteo.
  Goal: Create a new Coq library for real numbers of mathematics.
  Leader: S. Boldo (INRIA Saclay, Toccata). Participant: A. Mahboubi.
  Website: http://coquelicot.saclay.inria.fr/.

8.2. National Initiatives

8.2.1. ANR

- *Psi* (ANR-09-JCJC-0006).
  Duration: 2009-2013. Goal: Proof-Search control in Interaction with domain-specific methods. Coordinator: Stéphane Lengrand (CNRS, LIX).
  Participant: A. Mahboubi.
  Website: http://www.lix.polytechnique.fr/~lengrand/PSI/.
- *ParalITP* (ANR-11-INSE-001).
  Goal: Improve the performances and the ergonomics of interactive provers by taking advantage of modern, parallel hardware.
  Leader: B. Wolff (University of Orsay, Paris XI). Participants: A. Mahboubi, E. Tassi.

8.2.2. Other

- PEPS Grant *Holonomix*.
  Goal: Asymptotics of special functions arising in physics, computer science, and number theory.
  Leader: Cyril Banderier (CNRS, LIPN). Participant: A. Bostan, F. Chyzak.
  Website: http://www.cnrs.fr/ins2i/spip.php?article143.

8.3. European Initiatives

8.3.1. FP7 Projects

- *Formalisation of Mathematics* (*ForMath*, EU FP7 STREP FET-open project).
  Partners: University of Gothenburg (Sweden); Radboud University Nijmegen (The Netherlands); Inria (France); Universidad de La Rioja (Spain).
  Goal: Investigate how recent advances in the methodology and design of computer-checked libraries of formalized mathematics apply to so-far-unexplored areas of mathematics, like real analysis or certified efficient computations.
  Leader: Th. Coquand (University of Gothenburg, Sweden). Participant: A. Mahboubi (work package leader for WP1).
  Website: http://wiki.portal.chalmers.se/cse/pmwiki.php/ForMath/ForMath.

9. Dissemination

9.1. Scientific Animation

The team started a seminar, first on an irregular basis, but with the view of running more regular sessions. It attracted researchers from teams in the neighbouring environment and had 8 sessions in 2013.

F. Chyzak is part of the scientifique committee of the *Journées Nationales de Calcul Formel*, the annual meeting of the French computer algebra community.

A. Mahboubi and E. Tassi have organized the 5th edition of the Coq international workshop (satellite of the Itp 2013 conference, Rennes, July 2013).

A. Mahboubi has participated to the organization of the Lix Colloquium (November 2013) and of the satellite PSATT international workshop.

A. Mahboubi has served in the program committee of the 5th edition of the Coq international workshop.

A. Mahboubi has served in the program committee of the ITP 2013 international conference.

A. Bostan has served as the Poster Committee Chair for the ISSAC 2013 international conference.

A. Bostan has served in the program committee of the MEGA 2013 international conference.

A. Bostan has served in the program committee of the FPSAC 2013 international conference.

A. Bostan has served in the program committee of the SYNASC 2013 international conference.

A. Bostan is part of the Scientific advisory board of the MEGA conference series.

A. Mahboubi and E. Tassi have given an invited tutorial at the ITP 2013 international conference (Rennes, France).

A. Mahboubi has given an invited talk at the Calculemus 2013 conference (Bath, United Kingdom).

A. Mahboubi has given an invited talk at the Colloquium of the Institute of Mathematics at the University of Nantes (France).

A. Mahboubi has given an invited talk, joint with G. Gonthier at the Dutch Mathematical Congress 2013 (Nijmegen, Netherlands).

A. Mahboubi has given an invited talk at the British Colloquium for Theoretical Computer Sciences (Bath, United Kingdom).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

- Master : A. Bostan, *Algorithmes efficaces en calcul formel*, 12h, M2, MPRI, France
- Master : F. Chyzak, *Algorithmes efficaces en calcul formel*, 12h, M2, MPRI, France
- Master : A. Mahboubi, *Assistants de preuves*, 18h, M2, MPRI, France
- Agrégation de Mathématiques : A. Bostan, *Préparation épreuve de modélisation, option C*, 12h, ÉNS Cachan, France

9.2.2. Supervision

- PhD: B. Morcrette, *Combinatoire analytique et modèles d'urnes*, June 2013, Ph. Flajolet, M. Soria and Ph. Dumas
- PhD in progress: A. Barillec, *Asymptotique automatique certifiée des fonctions spéciales*, September 2013, F. Chyzak and A. Mahboubi
- PhD in progress: L. Dumont, *Algorithmique efficace pour les diagonales, applications en combinatoire, physique et théorie des nombres*, September 2013, A. Bostan and B. Salvy

- PhD in progress: P. Lairez, *Algorithmique efficace pour la création télescopique, et ses applications*, September 2011, A. Bostan and B. Salvy
- L3: D. Rouhling, ÉNS Lyon, *Proof search modulo a theory in sequent calculus*, June–July 2013, S. Graham-Lengrand (CNRS, LIX) and A. Mahboubi

9.2.3. Juries

- A. Mahboubi has served as examiner in the PhD jury of Mahfuza Farooque, *Automated reasoning techniques as proof-search in sequent calculus*, December 19, 2013.
- A. Bostan has served as examiner in the PhD jury of Basile Morcrette, *Combinatoire analytique et modèles d'urnes*, Université Paris 6, June 26, 2013.

9.3. Popularization

- A. Mahboubi has given a lecture to laureates of the *Olympiades académiques de mathématiques, académie de Créteil.*
  http://maths.ac-creteil.fr/spip/spip.php?article463.
- A. Mahboubi has been involved in the scientific committee for the elaboration of the board game *Mémoire Vive* produced by the Inria communication services.
- A. Mahboubi has given a talk at the forum STIC Paris-Saclay (Palaiseau, France) in November 2013.

10. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] B. MORCRETTE. , *Combinatoire analytique et modèles d'urnes*, Université Pierre et Marie Curie - Paris VI, June 2013, version du 25 Juin 2013, http://hal.inria.fr/tel-00843046

Articles in International Peer-Reviewed Journals

[2] A. BOSTAN, S. BOUKRAA, G. CHRISTOL, S. HASSANI, J.-M. MAILLARD. *Ising n-fold integrals as diagonals of rational functions and integrality of series expansions*, in "Journal of Physics A: Mathematical and Theoretical", April 2013, vol. 46, pp. 185202-185245 [*DOI :* 10.1088/1751-8113/46/18/185202], http://hal.inria.fr/hal-00780422

[3] A. BOSTAN, K. RASCHEL, B. SALVY. *Non-D-finite excursions in the quarter plane*, in "Journal of Combinatorial Theory, Series A", October 2013, vol. 121, pp. 45-63 [*DOI :* 10.1016/J.JCTA.2013.09.005], http://hal.inria.fr/hal-00697386

[4] P. DUMAS. *Joint Spectral Radius, Dilation Equations, and Asymptotic Behavior of Radix-Rational Sequences*, in "Linear Algebra and its Applications", March 2013, vol. 438, n^{o} 5, pp. 2107-2126 [*DOI :* 10.1016/J.LAA.2012.10.013], http://hal.inria.fr/hal-00780568

Invited Conferences

[5] A. MAHBOUBI. *The Rooster and the Butterflies*, in "CICM 2013 - Conference on Intelligent Computer Mathematics - 2013", Bath, United Kingdom, J. CARETTE, D. ASPINAL, C. LANGE, P. SOJKA, W. WINDSTEIGER (editors), Lecture Notes in Artificial Intelligence, Springer, July 2013, vol. 7961, pp. 1-18 [*DOI :* 10.1007/978-3-642-39320-4_1], http://hal.inria.fr/hal-00825074

[6] A. MAHBOUBI, E. TASSI. *Canonical Structures for the working Coq user*, in "ITP 2013, 4th Conference on Interactive Theorem Proving", Rennes, France, S. BLAZY, C. PAULIN, D. PICHARDIE (editors), LNCS, Springer, 2013, vol. 7998, pp. 19-34 [*DOI :* 10.1007/978-3-642-39634-2_5], http://hal.inria.fr/hal-00816703

International Conferences with Proceedings

[7] B. BARRAS, L. D. C. GONZALEZ HUESCA, H. HERBELIN, Y. RÉGIS-GIANAS, E. TASSI, M. WENZEL, B. WOLFF. *Pervasive Parallelism in Highly-Trustable Interactive Theorem Proving Systems*, in "MKM/Calculemus/DML", Bath, United Kingdom, 2013, pp. 359-363, http://hal.inria.fr/hal-00908980

[8] A. BOSTAN, S. CHEN, F. CHYZAK, Z. LI, G. XIN. *Hermite Reduction and Creative Telescoping for Hyperexponential Functions*, in "ISSAC'13 - 38th International Symposium on Symbolic and Algebraic Computation", Boston, United States, Northeastern University, Boston, Massachusetts, USA, 2013, pp. 77-84 [*DOI :* 10.1145/2465506.2465946], http://hal.inria.fr/hal-00780067

[9] A. BOSTAN, F. CHYZAK, É. DE PANAFIEU. *Complexity Estimates for Two Uncoupling Algorithms*, in "ISSAC'13 - 38th International Symposium on Symbolic and Algebraic Computation", Boston, United States, Northeastern University, Boston, Massachusetts, USA, 2013, pp. 85-92 [*DOI :* 10.1145/2465506.2465941], http://hal.inria.fr/hal-00780010

[10] A. BOSTAN, P. LAIREZ, B. SALVY. *Creative telescoping for rational functions using the Griffiths-Dwork method*, in "ISSAC'13 - 38th International Symposium on Symbolic and Algebraic Computation", Boston, United States, Northeastern University, Boston, Massachusetts, USA, 2013, pp. 93-100 [*DOI :* 10.1145/2465506.2465935], http://hal.inria.fr/hal-00777675

[11] M. FAROOQUE, S. LENGRAND, A. MAHBOUBI. *A bisimulation between DPLL(T) and a proof-search strategy for the focused sequent calculus*, in "LFMTP - International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice - 2013", Boston, United States, A. MOMIGLIANO, B. PIENTKA, R. POLLACK (editors), ACM, September 2013 [*DOI :* 10.1145/2503887.2503892], http://hal.inria.fr/hal-00854426

[12] G. GONTHIER, A. ASPERTI, J. AVIGAD, Y. BERTOT, C. COHEN, F. GARILLOT, S. LE ROUX, A. MAHBOUBI, R. O'CONNOR, S. OULD BIHA, I. PASCA, L. RIDEAU, A. SOLOVYEV, E. TASSI, L. THÉRY. *A Machine-Checked Proof of the Odd Order Theorem*, in "ITP 2013, 4th Conference on Interactive Theorem Proving", Rennes, France, S. BLAZY, C. PAULIN, D. PICHARDIE (editors), LNCS, Springer, 2013, vol. 7998, pp. 163-179 [*DOI :* 10.1007/978-3-642-39634-2_14], http://hal.inria.fr/hal-00816699

Scientific Books (or Scientific Book chapters)

[13] P. ACZEL, B. AHRENS, T. ALTENKIRCH, S. AWODEY, B. BARRAS, A. BAUER, Y. BERTOT, M. BEZEM, T. COQUAND, E. FINSTER, D. GRAYSON, H. HERBELIN, A. JOYAL, D. LICATA, P. LUMSDAINE, A. MAHBOUBI, P. MARTIN-LÖF, S. MELIKHOV, A. PELAYO, A. POLONSKY, M. SHULMAN, M. SOZEAU, B. SPITTERS, B. VAN DEN BERG, V. VOEVODSKY, M. WARREN, C. ANGIULI, A. BORDG, G. BRUNERIE, C. KAPULKIN, E. RIJKE, K. SOJAKOVA, J. AVIGAD, C. COHEN, R. CONSTABLE, P.-L. CURIEN, P. DYBJER, M. ESCARDÓ, K.-B. HOU, N. GAMBINO, R. GARNER, G. GONTHIER, T. HALES, R. HARPER, M. HOFMANN, P. HOFSTRA, J. KOCH, N. KRAUS, N. LI, Z. LUO, M. NAHAS, E. PALMGREN, E. RIEHL, D. SCOTT, P. SCOTT, S. SOLOVIEV. , *Homotopy Type Theory: Univalent Foundations of Mathematics*, Aucun, 2013, 448 p. , http://hal.inria.fr/hal-00935057

Other Publications

[14] A. Bostan, G. Chèze, T. Cluzeau, J.-A. Weil. , *Efficient Algorithms for Computing Rational First Integrals and Darboux Polynomials of Planar Polynomial Vector Fields*, October 2013, http://hal.inria.fr/hal-00871663

[15] A. Bostan, I. Kurkova, K. Raschel. , *A human proof of Gessel's lattice path conjecture*, 2013, 23 pages, 3 figures, http://hal.inria.fr/hal-00858083

[16] P. Dumas. , *Rational series and asymptotic expansion for linear homogeneous divide-and-conquer recurrences*, 2013, http://hal.inria.fr/hal-00840659

References in notes

[17] M. Abramowitz, I. A. Stegun (editors). , *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, DoverNew York, 1992, xiv+1046 p. , Reprint of the 1972 edition

[18] , *Computer Algebra Errors*, Article in mathematics blog MathOverflow, http://mathoverflow.net/questions/11517/computer-algebra-errors

[19] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, C. W. Clark (editors). , *NIST Handbook of mathematical functions*, Cambridge University Press, 2010

[20] M. Armand, B. Grégoire, A. Spiwack, L. Théry. *Extending Coq with Imperative Features and its Application to SAT Verication*, in "Interactive Theorem Proving, international Conference, ITP 2010, Edinburgh, Scotland, July 11–14, 2010, Proceedings", Lecture Notes in Computer Science, Springer, 2010

[21] B. Beckermann, G. Labahn. *A uniform approach for the fast computation of matrix-type Padé approximants*, in "SIAM J. Matrix Anal. Appl.", 1994, vol. 15, n^o 3, pp. 804–823

[22] A. Benoit, F. Chyzak, A. Darrasse, S. Gerhold, M. Mezzarobba, B. Salvy. *The Dynamic Dictionary of Mathematical Functions (DDMF)*, in "The Third International Congress on Mathematical Software (ICMS 2010)", K. Fukuda, J. van der Hoeven, M. Joswig, N. Takayama (editors), Lecture Notes in Computer Science, 2010, vol. 6327, pp. 35–41, http://dx.doi.org/10.1007/978-3-642-15582-6_7

[23] M. Boespflug, M. Dénès, B. Grégoire. *Full reduction at full throttle*, in "First International Conference on Certified Programs and Proofs, Taiwan, December 7–9", Lecture Notes in Computer Science, Springer, 2011

[24] S. Boldo, C. Lelay, G. Melquiond. *Improving Real Analysis in Coq: A User-Friendly Approach to Integrals and Derivatives*, in "Certified Programs and Proofs", C. Hawblitzel, D. Miller (editors), Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, vol. 7679, pp. 289-304, http://dx.doi.org/10.1007/978-3-642-35308-6_22

[25] S. Boldo, G. Melquiond. *Flocq: A Unified Library for Proving Floating-point Algorithms in Coq*, in "Proceedings of the 20th IEEE Symposium on Computer Arithmetic", Tübingen, Germany, July 2011, pp. 243–252

[26] A. Bostan. *Algorithmes rapides pour les polynômes, séries formelles et matrices*, in "Actes des Journées Nationales de Calcul Formel", Luminy, France, 2010, pp. 75–262, Les cours du CIRM, tome 1, numéro 2, http://ccirm.cedram.org:80/ccirm-bin/fitem?id=CCIRM_2010__1_2_75_0

[27] A. Bostan, S. Boukraa, S. Hassani, J.-M. Maillard, J.-A. Weil, N. Zenine. *Globally nilpotent differential operators and the square Ising model*, in "J. Phys. A: Math. Theor.", 2009, vol. 42, n^o 12, 50 p. , http://dx.doi.org/10.1088/1751-8113/42/12/125206

[28] A. Bostan, S. Chen, F. Chyzak, Z. Li. *Complexity of creative telescoping for bivariate rational functions*, in "ISSAC'10: Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation", New York, NY, USA, ACM, 2010, pp. 203–210, http://doi.acm.org/10.1145/1837934.1837975

[29] A. Bostan, F. Chyzak, G. Lecerf, B. Salvy, É. Schost. *Differential equations for algebraic functions*, in "ISSAC'07: Proceedings of the 2007 international symposium on Symbolic and algebraic computation", C. W. Brown (editor), ACM Press, 2007, pp. 25–32, http://dx.doi.org/10.1145/1277548.1277553

[30] A. Bostan, F. Chyzak, M. van Hoeij, L. Pech. *Explicit formula for the generating series of diagonal 3D rook paths*, in "Sém. Loth. Comb.", 2011, vol. B66a, 27 p. , http://www.emis.de/journals/SLC/wpapers/s66bochhope.html

[31] A. Bostan, M. Kauers. *The complete generating function for Gessel walks is algebraic*, in "Proceedings of the American Mathematical Society", September 2010, vol. 138, n^o 9, pp. 3063–3078, With an appendix by Mark van Hoeij

[32] F. Chyzak. *An extension of Zeilberger's fast algorithm to general holonomic functions*, in "Discrete Math.", 2000, vol. 217, n^o 1-3, pp. 115–134, Formal power series and algebraic combinatorics (Vienna, 1997)

[33] F. Chyzak, M. Kauers, B. Salvy. *A Non-Holonomic Systems Approach to Special Function Identities*, in "ISSAC'09: Proceedings of the Twenty-Second International Symposium on Symbolic and Algebraic Computation", J. May (editor), 2009, pp. 111–118, http://dx.doi.org/10.1145/1576702.1576720

[34] F. Chyzak, B. Salvy. *Non-commutative elimination in Ore algebras proves multivariate identities*, in "J. Symbolic Comput.", 1998, vol. 26, n^o 2, pp. 187–227

[35] T. Coquand, G. P. Huet. *The Calculus of Constructions*, in "Inf. Comput.", 1988, vol. 76, n^o 2/3, pp. 95-120, http://dx.doi.org/10.1016/0890-5401(88)90005-3

[36] T. Coquand, C. Paulin-Mohring. *Inductively defined types*, in "Proceedings of Colog'88", P. Martin-Löf, G. Mints (editors), Lecture Notes in Computer Science, Springer-Verlag, 1990, vol. 417

[37] D. Delahaye, M. Mayero. *Dealing with algebraic expressions over a field in Coq using Maple*, in "J. Symbolic Comput.", 2005, vol. 39, n^o 5, pp. 569–592, Special issue on the integration of automated reasoning and computer algebra systems, http://dx.doi.org/10.1016/j.jsc.2004.12.004

[38] F. Garillot, G. Gonthier, A. Mahboubi, L. Rideau. *Packaging Mathematical Structures*, in "Theorem Proving in Higher-Order Logics", S. Berghofer, T. Nipkow, C. Urban, M. Wenzel (editors), Lecture Notes in Computer Science, Springer, 2009, vol. 5674, pp. 327–342

[39] J. VON ZUR. GATHEN, J. GERHARD. , *Modern computer algebra*, 2nd, Cambridge University PressNew York, 2003, xiv+785 p.

[40] G. GONTHIER. *Formal proofs—the four-colour theorem*, in "Notices of the AMS", 2008, vol. 55, n^{o} 11, pp. 1382-1393

[41] G. GONTHIER, A. MAHBOUBI. *An introduction to small scale reflection in Coq*, in "Journal of Formalized Reasoning", 2010, vol. 3, n^{o} 2, pp. 95–152

[42] G. GONTHIER, A. MAHBOUBI, E. TASSI. , *A Small Scale Reflection Extension for the Coq system*, Inria, 2008, n^{o} RR-6455, http://hal.inria.fr/inria-00258384

[43] G. GONTHIER, E. TASSI. *A language of patterns for subterm selection*, in "ITP", LNCS, 2012, vol. 7406, pp. 361–376

[44] B. GRÉGOIRE, A. MAHBOUBI. *Proving Equalities in a Commutative Ring Done Right in Coq*, in "Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings", Lecture Notes in Computer Science, Springer, 2005, vol. 3603, pp. 98–113

[45] T. HALES. *Formal proof*, in "Notices of the AMS", 2008, vol. 55, n^{o} 11, pp. 1370-1380

[46] J. HARRISON. *A HOL Theory of Euclidean space*, in "Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005", Oxford, UK, J. HURD, T. MELHAM (editors), Lecture Notes in Computer Science, Springer-Verlag, 2005, vol. 3603

[47] J. HARRISON. *Formalizing an analytic proof of the prime number theorem*, in "Journal of Automated Reasoning", 2009, vol. 43, pp. 243–261, Dedicated to Mike Gordon on the occasion of his 60th birthday

[48] J. HARRISON. , *Theorem proving with the real numbers*, CPHC/BCS distinguished dissertations, Springer, 1998, I p.

[49] J. HARRISON. *A Machine-Checked Theory of Floating Point Arithmetic*, in "Theorem Proving in Higher Order Logics: 12th International Conference, TPHOLs'99", Nice, France, Y. BERTOT, G. DOWEK, A. HIRSCHOWITZ, C. PAULIN, L. THÉRY (editors), Lecture Notes in Computer Science, Springer-Verlag, 1999, vol. 1690, pp. 113–130

[50] J. HARRISON, L. THÉRY. *A Skeptic's Approach to Combining HOL and Maple*, in "J. Autom. Reason.", December 1998, vol. 21, n^{o} 3, pp. 279–294, http://dx.doi.org/10.1023/A:1006023127567

[51] F. JOHANSSON. , *Another Mathematica bug*, Article on personal blog, http://fredrik-j.blogspot.fr/2009/07/another-mathematica-bug.html

[52] C. KOUTSCHAN. *A fast approach to creative telescoping*, in "Math. Comput. Sci.", 2010, vol. 4, n^{o} 2-3, pp. 259–266, http://dx.doi.org/10.1007/s11786-010-0055-0

[53] A. MAHBOUBI. *Implementing the cylindrical algebraic decomposition within the Coq system*, in "Mathematical Structures in Computer Science", 2007, vol. 17, n^{o} 1, pp. 99–127

[54] R. MATUSZEWSKI, P. RUDNICKI. *Mizar: the first 30 years*, in "Mechanized Mathematics and Its Applications", 2005, vol. 4, 2005 p.

[55] M. MAYERO. , *Problèmes critiques et preuves formelles*, Université Paris 13, novembre 2012, Habilitation à Diriger des Recherches

[56] M. MEZZAROBBA. *NumGfun: a package for numerical and analytic computation and D-finite functions*, in "ISSAC 2010—Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation", New York, ACM, 2010, pp. 139–146, http://dx.doi.org/10.1145/1837934.1837965

[57] P. PAULE, M. SCHORN. *A Mathematica version of Zeilberger's algorithm for proving binomial coefficient identities*, in "J. Symbolic Comput.", 1995, vol. 20, n^{o} 5-6, pp. 673–698, Symbolic computation in combinatorics Δ_1 (Ithaca, NY, 1993), http://dx.doi.org/10.1006/jsco.1995.1071

[58] B. PETERSEN. , *Maple*, Personal web site, http://people.oregonstate.edu/~peterseb/maple/

[59] P. RUDNICKI, A. TRYBULEC. *On the Integrity of a Repository of Formalized Mathematics*, in "Proceedings of the Second International Conference on Mathematical Knowledge Management", London, UK, MKM '03, Springer-Verlag, 2003, pp. 162–174, http://dl.acm.org/citation.cfm?id=648071.748518

[60] B. SALVY, P. ZIMMERMANN. *Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable*, in "ACM Trans. Math. Software", 1994, vol. 20, n^{o} 2, pp. 163–177

[61] N. J. A. SLOANE, S. PLOUFFE. , *The Encyclopedia of Integer Sequences*, Academic Press, San Diego, 1995

[62] THE COQ DEVELOPMENT TEAM. , *The Coq Proof Assistant: Reference Manual*, http://coq.inria.fr/doc/

[63] THE MATHEMATICAL COMPONENT TEAM. , *A Formalization of the Odd Order Theorem using the Coq proof assistant*, September 2012, http://www.msr-inria.fr/projects/mathematical-components/

[64] L. THÉRY. *A Machine-Checked Implementation of Buchberger's Algorithm*, in "J. Autom. Reasoning", 2001, vol. 26, n^{o} 2, pp. 107-137, http://dx.doi.org/10.1023/A:1026518331905

[65] K. WEGSCHAIDER. , *Computer generated proofs of binomial multi-sum identities*, RISC, J. Kepler University, May 1997, 99 p.

[66] S. WOLFRAM. , *Mathematica: A system for doing mathematics by computer (2nd ed.)*, Addison-Wesley, 1992, I p.

[67] D. ZEILBERGER. , *Opinion 94: The Human Obsession With "Formal Proofs" is a Waste of the Computer's Time, and, Even More Regretfully, of Humans' Time*, 2009, http://www.math.rutgers.edu/~zeilberg/Opinion94.html

[68] D. ZEILBERGER. *A holonomic systems approach to special functions identities*, in "J. Comput. Appl. Math.", 1990, vol. 32, n^{o} 3, pp. 321–368

[69] D. ZEILBERGER. *The method of creative telescoping*, in "J. Symbolic Comput.", 1991, vol. 11, n^{o} 3, pp. 195–204