Activity Report 2017

Project-Team SPECFUN

Symbolic Special Functions : Fast and Certified

RESEARCH CENTER
Saclay - Île-de-France

THEME
Algorithmics, Computer Algebra and Cryptology

Table of contents

Project-Team SPECFUN

Creation of the Team: 2012 November 01, updated into Project-Team: 2014 July 01

Keywords:

Computer Science and Digital Science:

A2.1.10. - Domain-specific languages
A2.1.11. - Proof languages
A2.4.3. - Proofs
A4.5. - Formal methods for security
A7.2. - Logic in Computer Science
A8.1. - Discrete mathematics, combinatorics
A8.4. - Computer Algebra
A8.5. - Number theory
A8.9. - Performance evaluation

Other Research Topics and Application Domains:

B9.4.2. - Mathematics
B9.4.3. - Physics

1. Personnel

Research Scientists

Frédéric Chyzak [Team leader, Inria, Researcher, HDR]
Alin Bostan [Inria, Researcher, HDR]
Philippe Dumas [Inria, Senior Researcher, until August 2017, now External Collaborator]
Georges Gonthier [Inria, Senior Researcher]
Pierre Lairez [Inria, Researcher]
Assia Mahboubi [Team co-leader, Inria, Researcher, until September 2017]

PhD Student

Thomas Sibut-Pinote [École Polytechnique]

Technical staff

Maxence Guesdon [Inria, Engineer, 40%, until May 2017]

Interns

Rémy Garnier [École Normale Supérieure Cachan, from March 2017 to August 2017]
Pascal Fong [Université Versailles – Saint-Quentin-en-Yvelines, from March 2017 to August 2017]

Administrative Assistant

Christine Biard [Inria, until Aug 2017]

External Collaborator

Marc Mezzarobba [CNRS]

2. Overall Objectives

2.1. Scientific challenges, expected impact

The general orientation of our team is described by the short name given to it: *Special Functions*, that is, particular mathematical functions that have established names due to their importance in mathematical analysis, physics, and other application domains. Indeed, we ambition to study special functions with the computer, by combined means of computer algebra and formal methods.

Computer-algebra systems have been advertised for decades as software for "doing mathematics by computer" [65]. 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 possibility was recently shown by the computer-algebra component of the team [19]. 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 [14] remains one of the most cited mathematical publications ever and has motivated the 10-year-long project of writing its successor [16]. 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 [44] 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 success of large-scale formalization projects, like the Four-Color Theorem of graph theory [39], the above-mentioned Kepler Conjecture [44], and 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) [19] 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]. Since 2006, the MathComp group has endeavoured to develop computer-checked libraries of formalized mathematics, using the Coq proof assistant [61]. The methodological aim of the project was to understand the design methods leading to successful large-scale formalizations. The work culminated in 2012 with the 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 [61] and on the libraries developed by the MathComp project.

[1] https://www.msr-inria.fr/news/the-formalization-of-the-odd-order-theorem-has-been-completed-the-20-septembre-2012/
[2] http://ddmf.msr-inria.inria.fr/1.9.1/ddmf
[3] https://www.msr-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/

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 [50]: "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 [57] 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."
- Jacques Carette, former head of the maths group at Maplesoft, about a bug [15] 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 [66], 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 [48], [22]. 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 our team 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 strive to maintain 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 algorithmic 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 the design of appropriate libraries of formalized mathematics, including certified implementations of certain computer-algebra algorithms. On the other side, we expect that scrutinizing the implementation and the output of computer-algebra algorithms will shed a new light on their semantics and on their correctness proofs, and help clarifying their documentation.

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) [37].

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 large 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 [17], [20] 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 intend 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 in a systematic way. Indeed 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 validation 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 first, archetypal example we expect to work on is a non-commutative generalization of the normalization procedure for elements of rings [43].

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 properties. 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. In all cases, we strive to design our algorithms with the constant goal of good theoretical complexity, and we observe that our algorithms are also fast in practice.

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 approach [25] to more general classes (algebraic with nested radicals, for example): the idea is to speed up calculations by making use of an analogue of Hermite reduction that avoids considering certificates. Homologous problems for summation will be addressed as well.

2.2.2.2. Applications to experimental mathematics

As a consequence of our complexity-driven approach to algorithms design, the algorithms mentioned in the previous paragraph are of good complexity. Therefore, they 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 enumerative generating functions 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 [24], [29], [27]. 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/1.9.1/ddmf, 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.

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 [14] 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 [67], 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, as well as their counterparts for recurrence equations.

3.1.2. Algorithms combining functions

Differential/recurrence equations that define special functions can be recombined [67] 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 [68]. 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 [56], [64], [33], [30], [31], [51]. The past ÉPI Algorithms contributed several implementations (*gfun* [59], *Mgfun* [33]).

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 linear in the output size, with a constant that is not more than a few units. On the implementation side, several original works exist, one of which (*NumGfun* [55]) 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 [29]. All this is based on subtle algorithms for Hermite–Padé approximants [18]. Moreover, guessing can at times be complemented by proven quantitative results that turn the heuristics into an algorithm [26]. 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 [38], 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. [23].

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 [14], [16], [60] 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.

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 [63], [43], [52] 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 [36]. A discussion about the systematic separation of the search for a solution and the checking of the solution is already clearly outlined in [49].

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.

[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/1.9.1/ddmf
[10] http://129.81.170.14/~vhm/Table.html

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 [53], which is based on set theory. In particular, the calculus of construction (CoC) [34] and its extension with inductive types (CIC) [35], have been studied for more than 20 years and been implemented by several independent tools (like Lego, Matita, and Agda). Its reference implementation, Coq [61], 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 exception, which was finished in 2014, 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 project.

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 project 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 [37] 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 [40] is another methodology promoted by the MathComp project. 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 [62], 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 [45], [46], [47], although some existing developments of interest [21], [54] 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 [42], [41], 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 [58]. For the Coq system, this is an active area of research.

4. Highlights of the Year

4.1. Highlights of the Year

4.1.1. Awards

Pierre Lairez was awarded the SIAM/AAG (SIAM Activity Group on Algebraic Geometry) Early Career Prize.

5. New Software and Platforms

5.1. DynaMoW

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

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

5.2. ECS

Encyclopedia of Combinatorial Structures
FUNCTIONAL DESCRIPTION: On-line mathematical encyclopedia with an emphasis on sequences that arise in the context of decomposable combinatorial structures, with the possibility to search by the first terms in the sequence, keyword, generating function, or closed form.

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

5.3. DDMF

Dynamic Dictionary of Mathematical Functions
FUNCTIONAL DESCRIPTION: Web site consisting of interactive tables of mathematical formulas on elementary and special functions. The formulas are automatically generated by OCaml and computer-algebra routines. Users can ask for more terms of the expansions, more digits of the numerical values, proofs of some of the formulas, etc.

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

5.4. Mgfun

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

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

5.5. Ssreflect

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

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

5.6. Math-Components

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

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

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

5.7. CoqInterval

Interval package for Coq
KEYWORDS: Interval arithmetic - Coq
FUNCTIONAL DESCRIPTION: CoqInterval is a library for the proof assistant Coq.

It provides several tactics for proving theorems on enclosures of real-valued expressions. The proofs are performed by an interval kernel which relies on a computable formalization of floating-point arithmetic in Coq.

The Marelle team developed a formalization of rigorous polynomial approximation using Taylor models in Coq. In 2014, this library has been included in CoqInterval.

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

6. New Results

6.1. Efficient Algorithms in Computer Algebra

This year has seen the end of the writing and the publication of a book on computer-algebra algorithms [8]. The course at Master 2 level *Algorithmes efficaces en calcul formel* is a course that Alin Bostan and Frédéric Chyzak have set up progressively since 2005 together with Marc Giusti (LIX), Bruno Salvy (today AriC), as well as, initially, Éric Schost (LIX at the time) and François Ollivier (LIX), and, more recently, Grégoire Lecerf (LIX). The course is very strongly focused to presenting the design of algorithms guided by complexity analysis, with the goal to lead the students to the understanding of all algorithmic aspects that are necessary to the "creative telescoping" used for symbolic computations of sums and integrals. Their lecture notes had been circulating in and used by the (French) computer-algebra community, while they long had the goal of turning them into a book. They could publish it in 2017 (686 pages), after a big finalization effort in 2016 and 2017. The first parts of the book present fast algorithms for basic objects (integers, polynomials, series, matrices, linear recurrences), insisting on general principles to design efficient algorithms. The next parts of the work build on them to address topics that have made recent progress: factorization of polynomials, algorithms for polynomial systems, definite summation and integration. The work [8] is online as a HAL collection[11]. It is available for free in pdf format and is otherwise sold at a very low price (via print-on-demand). Over the first three months after publication, the book has sold roughly 60 printed copies and the pdf has been downloaded 265 times.

6.2. Hypergeometric Expressions for Generating Functions of Walks with Small Steps in the Quarter Plane

In [2], Alin Bostan and Frédéric Chyzak, together with Mark van Hoeij (Florida State University), Manuel Kauers (Johannes Kepler University), and Lucien Pech, have studied nearest-neighbors walks on the two-dimensional square lattice, that is, models of walks on $\mathbb{Z}^2$ defined by a fixed step set that consists of non-zero vectors with coordinates 0, 1 or -1. They concerned themselves with the enumeration of such walks starting at the origin and constrained to remain in the quarter plane $\mathbb{N}^2$, counted by their length and by the position of their ending point. In earlier works, Bousquet-Mélou and Mishna had identified 19 models of walks that possess a D-finite generating function, and linear differential equations had then been guessed in these cases by Bostan and Kauers. Here, we have given the first proof that these equations are indeed satisfied by the corresponding generating functions. As a first corollary, we have proved that all these 19 generating functions can be expressed in terms of Gauss' hypergeometric functions, with specific parameters that relate them intimately to elliptic integrals. As a second corollary, we have shown that all the 19 generating functions are transcendental, and that among their 19×4 combinatorially meaningful specializations only four are algebraic functions.

6.3. Multiple Binomial Sums

Multiple binomial sums form a large class of multi-indexed sequences, closed under partial summation, which contains most of the sequences obtained by multiple summation of products of binomial coefficients, as well as all the sequences with algebraic generating function. Alin Bostan and Pierre Lairez, together with Bruno Salvy (AriC), have studied in [7] the representation of the generating functions of binomial sums by integrals of rational functions. The outcome is twofold. Firstly, we have shown that a univariate sequence is a multiple binomial sum if and only if its generating function is the diagonal of a rational function. Secondly, we have proposed algorithms that decide the equality of multiple binomial sums and that compute recurrence relations for them. In conjunction with geometric simplifications of the integral representations, this approach behaves well in practice. The process avoids the computation of certificates and the problem of the appearance of spurious singularities that afflicts discrete creative telescoping, both in theory and in practice.

[11] https://hal.archives-ouvertes.fr/AECF/

6.4. Algebraic Diagonals and Walks

The diagonal of a multivariate power series F is the univariate power series $\mathsf{Diag}F$ generated by the diagonal terms of F. Diagonals form an important class of power series; they occur frequently in number theory, theoretical physics and enumerative combinatorics. In [28], Alin Bostan and Louis Dumont, together with Bruno Salvy (AriC), have studied algorithmic questions related to diagonals in the case where F is the Taylor expansion of a bivariate rational function. It is classical that in this case $\mathsf{Diag}F$ is an algebraic function. They have proposed an algorithm for computing an annihilating polynomial of $\mathsf{Diag}F$. They have given a precise bound on the size of this polynomial and show that generically, this polynomial is the minimal polynomial of $\mathsf{Diag}F$ and that its size reaches the bound. Their algorithm runs in time quasi-linear in this bound, which grows exponentially with the degree of the input rational function. They have also addressed the related problem of enumerating directed lattice walks. The insight given by their study has led to a new method for expanding the generating power series of bridges, excursions and meanders. They have shown that their first N terms can be computed in quasi-linear complexity in N, without first computing a very large polynomial equation. An extended version of this work has been presented in [4].

6.5. A Human Proof of the Gessel Conjecture

Counting lattice paths obeying various geometric constraints is a classical topic in combinatorics and probability theory. Many recent works deal with the enumeration of 2-dimensional walks with prescribed steps confined to the positive quadrant. A notoriously difficult case concerns the so-called *Gessel walks*: they are planar walks confined to the positive quarter plane, which move by unit steps in any of the West, North-East, East, and South-West directions. In 2001, Ira Gessel conjectured a closed-form expression for the number of such 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. This year, Alin Bostan, together with Irina Kurkova (Univ. Paris 6) and Kilian Raschel (CNRS and Univ. Tours), proposed in [6] the first "human proofs" of these results. They are derived from a new expression for the generating function of Gessel walks in terms of special functions.

6.6. Subresultants in Multiple Roots

In [3], we have provided explicit formulae for the coefficients of the order-d polynomial subresultant of $(x-\alpha)^m$ and $(x-\beta)^n$ with respect to the set of Bernstein polynomials $\{(x-\alpha)^j(x-\beta)^{d-j},\ 0 \le j \le d\}$. They are given by hypergeometric expressions arising from determinants of binomial Hankel matrices.

6.7. On Matrices with Displacement Structure: Generalized Operators and Faster Algorithms

For matrices with displacement structure, basic operations like multiplication, inversion, and linear-system solving can all be expressed in terms of a single task: evaluating the product AB, where A is a structured $n \times n$ matrix of displacement rank α, and B is an arbitrary $n \times \alpha$ matrix. Given B and a so-called *generator* of A, this product is classically computed with a cost ranging from $O(\alpha^2\mathsf{M}(n))$ to $O(\alpha^2\mathsf{M}(n)\log(n))$ arithmetic operations, depending on the specific structure of A. (Here, M is a cost function for polynomial multiplication.) In [5], Alin Bostan, jointly with Claude-Pierre Jeannerod (AriC), Christophe Mouilleron (ENSIIE), and Éric Schost (University of Waterloo), has generalized classical displacement operators, based on block diagonal matrices with companion diagonal blocks, and has also designed fast algorithms to perform the task above for this extended class of structured matrices. The cost of these algorithms ranges from $O(\alpha^{\omega-1}\mathsf{M}(n))$ to $O(\alpha^{\omega-1}\mathsf{M}(n)\log(n))$, with ω such that two $n \times n$ matrices over a field can be multiplied using $O(n^\omega)$ field operations. By combining this result with classical randomized regularization techniques, he has obtained faster Las Vegas algorithms for structured inversion and linear system solving.

6.8. Quasilinear Average Complexity for Solving Polynomial Systems

How many operations do we need on the average to compute an approximate root of a random Gaussian polynomial system? Beyond Smale's 17th problem that asked whether a polynomial bound is possible, Pierre Lairez has proved in [10] a quasi-optimal bound (input size)$^{1+o(1)}$, which improves upon the previously known (input size)$^{3/2+o(1)}$ bound. His new algorithm relies on numerical continuation along *rigid continuation paths*. The central idea is to consider rigid motions of the equations rather than line segments in the linear space of all polynomial systems. This leads to a better average condition number and allows for bigger steps. He showed that on the average, one approximate root of a random Gaussian polynomial system of n equations of degree at most D in $n+1$ homogeneous variables can be computed with $O(n^5D^2)$ continuation steps. This is a decisive improvement over previous bounds, which prove no better than $\sqrt{2}^{\min(n,D)}$ continuation steps on the average.

6.9. Computing the Homology of Basic Semialgebraic Sets in Weak Exponential Time

In [9], Pierre Lairez, jointly with Peter Bürgisser (TU Berlin) and Felipe Cucker (City University of Hong Kong), has described and analyzed an algorithm for computing the homology (Betti numbers and torsion coefficients) of basic semialgebraic sets. The algorithm works in weak exponential time, that is, out of a set of exponentially small measure in the space of data, the cost of the algorithm is exponential in the size of the data. All algorithms previously proposed for this problem have a complexity that is doubly exponential (and this is so for almost all data).

6.10. Formally Certified Computation of Improper Definite Integrals

Assia Mahboubi and Thomas Sibut-Pinote, in collaboration with Guillaume Melquiond (Toccata), have pursued their work on the certified computation of intervals approximating the values of definite integrals involving elementary mathematical functions. This library provides an automated tool that builds a formal proof of the correctness of the output, that is, a formal proof that the interval contains the mathematical values and a formal proof of the integrability of the input function on the input interval. This tool has been extended this year, and it can now deal with improper integrals, that is, integrals whose bounds are infinite or singularities of the integrand. The methodology, the implementation and benchmarks have been described in [13].

6.11. A Complete Formal Proof of the Irrationality of $\zeta(3)$

Assia Mahboubi and Thomas Sibut-Pinote have completed a formal proof of the irrationality of the constant $\zeta(3)$. The missing step in a previous work [32] with Frédéric Chyzak and Enrico Tassi was to obtain a formal proof of the asymptotic behaviour of the least common multiple of the first n integers. They have written a report on this work, which is included as a chapter in Thomas Sibut-Pinote's PhD manuscript.

7. Partnerships and Cooperations

7.1. National Initiatives

7.1.1. ANR

FastRelax (ANR-14-CE25-0018). Goal: Develop computer-aided proofs of numerical values, with certified and reasonably tight error bounds, without sacrificing efficiency. Leader: B. Salvy (Inria, ENS Lyon). Participants: Assia Mahboubi, Th. Sibut-Pinote. Website: http://fastrelax.gforge.inria.fr/.

7.2. International Research Visitors

7.2.1. Visits of International Scientists

- Marni Mishna (Simon Fraser University) visited the team for one week in January.
- Emre Sertöz (Max Planck Institute Leipzig) visited the team for one week in November. He worked with Pierre Lairez on applications to algebraic geometry of two tools developped at Specfun: the computations of periods (Lairez's PhD) and numerical analytic continuation (Mezzarobba's PhD, 2011).
- Karen Yeats (Simon Fraser University) visited the team for a few days in June. She continued a work on bijective combinatorics of words with Frédéric Chyzak. A text is now under writing.

7.2.1.1. Internships

- Pascal Fong did a Master internship from March to August. Under the supervision of Pierre Lairez and Mohab Safey El Din (UPMC), he studied the numerical computation of the length of plane algebraic curves.
- Rémy Garnier did a Master internship from March to July. Under the supervision of Alin Bostan and Frédéric Chyzak, he studied existing algorithms to solve linear differential systems for their rational-function solutions.
- Meissa M'baye did a Master internship from February to June. Under the remote supervision of Assia Mahboubi, he studied the principles of proof assistants and surveyed formalization methodologies for elementary number theory.

7.2.2. Visits to International Teams

- Frédéric Chyzak and Alin Bostan have been invited by the Erwin Schrödinger Institute (Vienna, Austria) for two weeks, to participate to the thematic program "Algorithmic and Enumerative Combinatorics" http://www.mat.univie.ac.at/~kratt/esi4/.
- Pierre Lairez visited Felipe Cucker (City University of Hong Kong) for two weeks. The outcome is a strengthened collaboration on the study of the complexity of numerical algorithms. A publication is in preparation: the second part of [10].
- Georges Gonthier was invited at the Newton Institue, for six weeks, as co-organiser and participant to the Big Proof thematic program.
- Assia Mahboubi visited Sander Dahmen (VU Amsterdam, The Netherlands) for three days. She has started a collaboration with his team, to obtain formal guarantees of computations for number theory.
- Assia Mahboubi has been invited by the Newton Institute (Cambridge, UK) for one month. She participated to the Big Proof thematic program.

8. Dissemination

8.1. Promoting Scientific Activities

8.1.1. Scientific Events Organisation

8.1.1.1. General Chair, Scientific Chair

- Alin Bostan is part of the Scientific advisory board of the conference series *Effective Methods in Algebraic Geometry* (MEGA).
- Frédéric Chyzak is member of the steering committee of the *Journées Nationales de Calcul Formel* (JNCF), the annual meeting of the French computer algebra community.

- Frédéric Chyzak is elected member of the steering committee of the *International Symposium on Symbolic and Algebraic Computation* (ISSAC, 3-year term, 2016–2018).
- Assia Mahboubi has served on the scientific committee of the Journées Scientifiques Inria and of the EUTypes summer school.

8.1.1.2. Member of the Organizing Committees

- Georges Gonthier was co-chair or the organising committee of the Big Proof thematic program held at the Newton Institute (Cambridge, UK) in June-August 2017.
- Assia Mahboubi has organized the TTT workshop, satellite of the POPL'17 conference.

8.1.1.3. Other

The team organizes a regular seminar[12], with roughly 15–20 talks a year. The topics reflect the team's interests: computer algebra, combinatorics, number theory, formal proofs, and related domains.

8.1.2. Scientific Events Selection

8.1.2.1. Member of the Conference Program Committees

- Assia Mahboubi has served as member of the conference program committees of the international conferences CPP 17, CADE 17, ITP 17, and on the program committee of the international workshops TYPES 17, TyDe 17, HoTT/UF 17.

8.1.2.2. Reviewer

- Alin Bostan has served as reviewer for the selection of the international conferences ISSAC 2017 and MEGA 2017.
- Frédéric Chyzak has served as reviewer for the selection of the international conference ISSAC 2017.
- Assia Mahboubi has served as external reviewer for the international conference LICS 17.

8.1.3. Journal

8.1.3.1. Member of the Editorial Boards

- Georges Gonthier is on the editorial board of the Journal of Formalized Reasoning.

8.1.3.2. Reviewer - Reviewing Activities

- Alin Bostan has served as a reviewer for the journals: *Journal of Symbolic Computation*; *Linear Algebra and its Applications*; *Journal of Algebra and its Applications*; *Journal of Complexity*; *Advances in Applied Mathematics*; *Journal of Combinatorial Theory, Series A*.
- Assia Mahboubi has served as reviewer for the journals: *Annals of Mathematics and Artificial Intelligence*; *Journal of Automated Reasoning*.

8.1.4. Invited Talks

- Alin Bostan has been invited to give a series of three lectures at the *JNCF – Journées Nationales de Calcul Formel* (CIRM, Luminy, France), January 16–18, 2017, http://jncf2017.lip6.fr.
- Alin Bostan has been invited to give a talk at the workshop *EDATE – Equations différentielles : aspects théoriques et effectifs*, Grenoble, March 13–15, 2017, http://edate2017.sciencesconf.org/.
- Alin Bostan has been invited to give a talk at the workshop *ANT – Automata in Number Theory*, île de Porquerolles, May 30—June 2, 2017, http://indico.math.cnrs.fr/event/2347/.
- Alin Bostan has been invited to give a talk at the workshop *Lattice walks at the Interface of Algebra, Analysis and Combinatorics*, BIRS, Banff, Canada, September 17—22, 2017, http://www.birs.ca/events/2017/5-day-workshops/17w5090.

[12] https://specfun.inria.fr/seminar/

- Alin Bostan has been invited to give a series of introductory lectures at the *Workshop on Computer Algebra in Combinatorics*, Erwin Schrödinger Institut (ESI), Vienna, Austria, November 13–17, 2017, http://www.mat.univie.ac.at/~kratt/esi4/workshop2.html.
- Frédéric Chyzak has been invited to give a talk at the *Second International Conference "Computer Algebra in Moscow"*, Plekhanov Russian University of Economics, Moscow, Russia, October 30 to November 30, 2017, http://www.ccas.ru/ca/conference.
- Frédéric Chyzak has been invited to give a talk at the *Workshop on Computer Algebra in Combinatorics*, Erwin Schrödinger Institut (ESI), Vienna, Austria, November 13–17, 2017, http://www.mat.univie.ac.at/~kratt/esi4/workshop2.html.
- Georges Gonthier gave an invited talk at the Special Session on Computer-Aided Proofs of the *Association for Symbolic Logic 2017 North American Meeting*, Boise, Idaho, USA, March 20–23, 2017.
- Georges Gonthier gave an invited talk at the *Second Conference on Artificial Intelligence and Theorem Proving (AITP'17)*, Obergürgl, Austria, March 26–30, 2017.
- Georges Gonthier gave a talk at the *ERCIM Workshop on Blockchains*, Paris, May 23, 2017.
- Georges Gonthier gave an invited talk at the Workshop *Computer-aided Mathematical Proof*, part of the Big Proof Program, Isaac Newton Institute, Cambridge, U.K.
- Pierre Lairez gave an invited talk at *Effective Methods in Algebraic Geometry (MEGGA 2017)*, Nice.
- Pierre Lairez gave an invited talk at the *Conference on Foundations of Computational Mathematics (FoCM 2017)*, Barcelona, Spain.
- Assia Mahboubi has been invited to give a talk at the General Mathematics Colloquium of the VU Amsterdam, The Netherlands.
- Assia Mahboubi has been invited to give a talk at the Workshop *Computer-aided Mathematical Proof*, part of the Big Proof Program, Isaac Newton Institute, Cambridge, U.K.

8.1.5. Leadership within the Scientific Community

- Assia Mahboubi leads the working group *Type theory based tools* inside the EUTYPES COST project. She is also a member of the management committee for France for this project and a member of its core management group.

8.1.6. Research Administration

- Georges Gonthier serves on the Conseil de l'École Doctorale de Mathématiques Hadamard.
- Assia Mahboubi has been a member of the *Commission Scientifique* of Inria Saclay — Île-de-France, until September 2017.

8.2. Teaching - Supervision - Juries

- Alin Bostan has served as a jury member of the French *Agrégation de Mathématiques – épreuve de modélisation, option C*.

8.2.1. Teaching

Licence:

Thomas Sibut-Pinote, *Les bases de la programmation et de l'algorithmique*, 32h, L3, École polytechnique, France.

Thomas Sibut-Pinote, *Les principes des langages de programmation*, 32h, L3, École polytechnique, France.

Master:

Frédéric Chyzak, *Algorithmes efficaces en calcul formel*, 18h, M2, MPRI, France.

Alin Bostan, *Algorithmes efficaces en calcul formel*, 40.5h, M2, MPRI, France.

Pierre Lairez, *Algorithmique avancée*, 18h, M1, École polytechnique, France.

Assia Mahboubi, *Algorithmes d'élimination des quantificateurs*, 3h, M2, Université Rennes 1, France.

8.2.2. Supervision

HdR : Alin Bostan, *Computer algebra for lattice path combinatorics* [1], Université Paris 13, December 15, 2017.

PhD : Thomas Sibut-Pinote, *Investigations en Mathématiques Assistées par Ordinateur: Expérimentation, Calcul et Certification*, Université Paris-Saclay, December 4, 2017.

8.2.3. Juries

- Frédéric Chyzak has served as an examiner in the PhD jury of Cyril Hugounenq *Volcans et calcul d'isogénies*, Université de Versailles – Saint-Quentin-en-Yvelines, September 25, 2017.
- Frédéric Chyzak has been a member of the hiring jury at Inria (Concours CR 2017).
- Georges Gonthier served on the Habilitation à diriger des Recherches of Paul-André Melliès *Une étude micrologique de la négation*, Université Paris Diderot, November 20, 2017.
- Assia Mahboubi has been a member of the hiring jury at Inria (Concours CR 2017).
- Assia Mahboubi has served as an examiner in the PhD jury of Evmorfia-Iro Bartzia *Une formalisation des courbes elliptiques pour la cryptographie*, Université Paris-Saclay, February 15, 2017.
- Assia Mahboubi has served as an examiner in the PhD jury of Étienne Miquey *Réalisabilité classique et effets de bord*, Université Paris Diderot, November 17, 2017.
- Assia Mahboubi has been a member of the hiring jury of a Maître de conférence position at Université Paris Diderot.

8.3. Popularization

- Assia Mahboubi has written an article for the MathExpress journal, at the occasion of the *salon Culture & Jeux Mathématiques*. See the Maths Language express volume at http://www.cijm.org/accueil/productions-cijm/90-maths-express.

9. Bibliography

Publications of the year

Doctoral Dissertations and Habilitation Theses

[1] A. BOSTAN. *Computer Algebra for Lattice path Combinatorics*, Université Paris 13, December 2017, Habilitation à diriger des recherches, https://hal.archives-ouvertes.fr/tel-01660300

Articles in International Peer-Reviewed Journals

[2] A. BOSTAN, F. CHYZAK, M. VAN HOEIJ, M. KAUERS, L. PECH. *Hypergeometric Expressions for Generating Functions of Walks with Small Steps in the Quarter Plane*, in "European Journal of Combinatorics", 2017, vol. 61, pp. 242-275, https://arxiv.org/abs/1606.02982 [*DOI :* 10.1016/J.EJC.2016.10.010], https://hal.inria.fr/hal-01332175

[3] A. BOSTAN, C. D'ANDREA, T. KRICK, A. SZANTO, M. VALDETTARO. *Subresultants in multiple roots: an extremal case*, in "Linear Algebra and its Applications", 2017, vol. 529, pp. 185-198, https://arxiv.org/abs/1608.03740 [*DOI :* 10.1016/J.LAA.2017.04.019], https://hal.archives-ouvertes.fr/hal-01588546

[4] A. BOSTAN, L. DUMONT, B. SALVY. *Algebraic Diagonals and Walks: Algorithms, Bounds, Complexity*, in "Journal of Symbolic Computation", 2017, vol. 83, pp. 68–92, https://arxiv.org/abs/1510.04526 [*DOI :* 10.1016/J.JSC.2016.11.006], https://hal.archives-ouvertes.fr/hal-01244914

[5] A. BOSTAN, C.-P. JEANNEROD, C. MOUILLERON, E. SCHOST. *On Matrices With Displacement Structure: Generalized Operators and Faster Algorithms*, in "SIAM Journal on Matrix Analysis and Applications", 2017, vol. 38, n° 3, pp. 733-775, https://arxiv.org/abs/1703.03734 [*DOI :* 10.1137/16M1062855], https://hal.archives-ouvertes.fr/hal-01588552

[6] A. BOSTAN, I. KURKOVA, K. RASCHEL. *A human proof of Gessel's lattice path conjecture*, in "Transactions of the American Mathematical Society", 2017, vol. 369, n° 2, February 2017, pp. 1365-1393, https://arxiv.org/abs/1309.1023 - Published electronically: April 14, 2016, https://hal.archives-ouvertes.fr/hal-00858083

[7] A. BOSTAN, P. LAIREZ, B. SALVY. *Multiple binomial sums*, in "Journal of Symbolic Computation", 2017, vol. 80, n° 2, pp. 351–386 [*DOI :* 10.1016/J.JSC.2016.04.002], https://hal.archives-ouvertes.fr/hal-01220573

Scientific Books (or Scientific Book chapters)

[8] A. BOSTAN, F. CHYZAK, M. GIUSTI, R. LEBRETON, G. LECERF, B. SALVY, E. SCHOST. *Algorithmes Efficaces en Calcul Formel*, published by the Authors, 2017, Voir la page du livre à l'adresse https://hal.archives-ouvertes.fr/AECF/, https://hal.inria.fr/hal-01431717

Other Publications

[9] P. BÜRGISSER, F. CUCKER, P. LAIREZ. *Computing the Homology of Basic Semialgebraic Sets in Weak Exponential Time*, June 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01545657

[10] P. LAIREZ. *Rigid continuation paths I. Quasilinear average complexity for solving polynomial systems*, November 2017, working paper or preprint, https://hal.inria.fr/hal-01631778

[11] H. LOMBARDI, A. MAHBOUBI. *Théories géométriques pour l'algèbre des nombres réels*, April 2017, working paper or preprint, https://hal.inria.fr/hal-01426164

[12] S. MADDAH, M. A. BARKATOU. *Formal Solutions of Singularly Perturbed Linear Differential Systems*, January 2017, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01417265

[13] A. MAHBOUBI, G. MELQUIOND, T. SIBUT-PINOTE. *Formally Verified Approximations of Definite Integrals*, February 2017, working paper or preprint, https://hal.inria.fr/hal-01630143

References in notes

[14] M. ABRAMOWITZ, I. A. STEGUN (editors). *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, Dover, New York, 1992, xiv+1046 p. , Reprint of the 1972 edition

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

[16] F. W. J. OLVER, D. W. LOZIER, R. F. BOISVERT, C. W. CLARK (editors). *NIST Handbook of mathematical functions*, Cambridge University Press, 2010

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] A. BOSTAN, L. DUMONT, B. SALVY. *Algebraic Diagonals and Walks*, in "ISSAC'15 International Symposium on Symbolic and Algebraic Computation", Bath, United Kingdom, ACM Press, July 2015, pp. 77–84 [*DOI :* 10.1145/2755996.2756663], https://hal.archives-ouvertes.fr/hal-01240729

[29] 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° 9, pp. 3063–3078, With an appendix by Mark van Hoeij

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

[31] 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

[32] F. CHYZAK, A. MAHBOUBI, T. SIBUT-PINOTE, E. TASSI. *A Computer-Algebra-Based Formal Proof of the Irrationality of ζ(3)*, in "ITP - 5th International Conference on Interactive Theorem Proving", Vienna, Austria, 2014, https://hal.inria.fr/hal-00984057

[33] F. CHYZAK, B. SALVY. *Non-commutative elimination in Ore algebras proves multivariate identities*, in "J. Symbolic Comput.", 1998, vol. 26, n° 2, pp. 187–227

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

[35] 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

[36] D. DELAHAYE, M. MAYERO. *Dealing with algebraic expressions over a field in Coq using Maple*, in "J. Symbolic Comput.", 2005, vol. 39, n° 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

[37] 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

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

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

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

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

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

[43] 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

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

[45] 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

[46] 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

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

[48] 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

[49] 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

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

[51] 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

[52] 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

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

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

[55] 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

[56] 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

[57] B. PETERSEN. *Maple*, Personal web site

[58] 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

[59] 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

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

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

[62] 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/

[63] 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

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

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

[66] 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

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

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