Activity Report 2018

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
A8.3. - Geometry, Topology
A8.4. - Computer Algebra
A8.5. - Number theory

Other Research Topics and Application Domains:

B9.5.2. - Mathematics
B9.5.3. - Physics

1. Team, Visitors, External Collaborators

Research Scientists

Frédéric Chyzak [Team leader, Inria, Researcher, HDR]
Alin Bostan [Inria, Researcher, HDR]
Georges Gonthier [Inria, Senior Researcher]
Pierre Lairez [Inria, Researcher]

Post-Doctoral Fellow

Svyatoslav Covanov [Inria, from Oct 2018]

Intern

Jiadong Han [Inria, from Apr 2018 until Jul 2018]

Administrative Assistant

Stéphanie Aubin [Inria, until Sep 2018]

External Collaborators

Philippe Dumas [Ministère de l'Éducation Nationale (retired)]
Guy Fayolle [Inria, Senior Researcher (emeritus)]
Marc Mezzarobba [CNRS, Researcher]

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” [60]. 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 [16]. 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 [11] remains one of the most cited mathematical publications ever and has motivated the 10-year-long project of writing its successor [13]. 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 [39] 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 [34], the above-mentioned Kepler Conjecture [39], 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) [16] 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 ÃPI 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 [56]. 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 [56] and on the libraries developed by the MathComp project.

[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/1.9.1/ddmf
[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/

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 [45]: 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 [52] 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 [12] 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 [61], 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 [43], [19]. 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) [32].

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 [14], [17] 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 [38].

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 [22] 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 [21], [25], [24]. 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 [11] 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 [62], 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 [62] 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 [63]. 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 [51], [59], [28], [26], [27], [46]. The past ÉPI Algorithms contributed several implementations (*gfun* [54], *Mgfun* [28]).

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* [50]) 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 [25]. All this is based on subtle algorithms for Hermite–Padé approximants [15]. Moreover, guessing can at times be complemented by proven quantitative results that turn the heuristics into an algorithm [23]. 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 [33], 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. [20].

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 [11], [13], [55] 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. 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 [58], [38], [47] 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 [31]. A discussion about the systematic separation of the search for a solution and the checking of the solution is already clearly outlined in [44].

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

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 [48], which is based on set theory. In particular, the calculus of construction (CoC) [29] and its extension with inductive types (CIC) [30], have been studied for more than 20 years and been implemented by several independent tools (like Lego, Matita, and Agda). Its reference implementation, Coq [56], 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 [32] 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 [35] 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 [57], 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 [40], [41], [42], although some existing developments of interest [18], [49] 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 [37], [36], 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 [53]. For the Coq system, this is an active area of research.

4. Application Domains

4.1. Computer Algebra in Mathematics

Our expertise in computer algebra and complexity-driven design of algebraic algorithms has applications in various domains, including:

- combinatorics, especially the study of combinatorial walks,
- theoretical computer science, like by the study of automatic sequences,
- number theory, by the analysis of the nature of so-called periods.

5. Highlights of the Year

5.1. Highlights of the Year

5.1.1. Awards

Georges Gonthier, Martìn Abadi and Cédric Fournet receiver the 20 year test-of-time award for their LICS 1998 paper *Secure Implementation of Channel Abstractions*, during LICS 2018 in Oxford.

6. New Software and Platforms

6.1. 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/

6.2. 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/

6.3. 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

6.4. 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

6.5. 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/

6.6. 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/

7. New Results

7.1. Computing solutions of linear Mahler equations

Mahler equations relate evaluations of the same function f at iterated bth powers of the variable. They arise in particular in the study of automatic sequences and in the complexity analysis of divide-and-conquer algorithms. Recently, the problem of solving Mahler equations in closed form has occurred in connection with number-theoretic questions. A difficulty in the manipulation of Mahler equations is the exponential blow-up of degrees when applying a Mahler operator to a polynomial. In [3], Frédéric Chyzak and Philippe Dumas, together with Thomas Dreyfus (IRMA, Université de Strasbourg) and Marc Mezzarobba (external collaborator from Sorbonne Université), have presented algorithms for solving linear Mahler equations for series, polynomials, and rational functions, and have obtained polynomial-time complexity under a mild assumption. The article was formally accepted and published this year.

7.2. Becker's conjecture on Mahler functions

In 1994, Becker conjectured that if $F(z)$ is a k-regular power series, then there exists a k-regular rational function $R(z)$ such that $F(z)/R(z)$ satisfies a Mahler-type functional equation with polynomial coefficients where the initial coefficient satisfies $a_0(z) = 1$. In [1], Frédéric Chyzak and Philippe Dumas, together with Jason P. Bell (University of Waterloo, Canada) and Michael Coons (University of Newcastle, Australia) have proved Becker's conjecture in the best-possible form: they have shown that the rational function $R(z)$ can be taken to be a polynomial $z^\gamma Q(z)$ for some explicit non-negative integer γ and such that $1/Q(z)$ is k-regular. The article was formally accepted this year.

7.3. Generalized Hermite reduction, creative telescoping and definite integration of D-finite functions

Hermite reduction is a classical algorithmic tool in symbolic integration. It is used to decompose a given rational function as a sum of a function with simple poles and the derivative of another rational function. Alin Bostan, Frédéric Chyzak, and Pierre Lairez, together with Bruno Salvy (project-team AriC) have extended Hermite reduction to arbitrary linear differential operators instead of the pure derivative. They have also developped efficient algorithms for this reduction, and then applied the generalized Hermite reduction to the computation of linear operators satisfied by single definite integrals of D-finite functions of several continuous or discrete parameters. The resulting algorithm is a generalization of reduction-based methods for creative telescoping. Their article [6] was published at the ISSAC conference.

7.4. Bijections between Łukasiewicz walks and generalized tandem walks

In [9], Frédéric Chyzak, together with Karen Yeats (University of Waterloo, Canada), have studied the enumeration by length of several walk models on the square lattice. They have obtained bijections between walks in the upper half-plane returning to the x-axis and walks in the quarter plane. An ongoing work by Bostan, Chyzak, and Mahboubi has given a bijection for models using small north, west, and south-east steps. The work in [9] has adapted and generalized it to a bijection between half-plane walks using those three steps in two colours and a quarter-plane model over the symmetrized step set consisting of north, north-west, west, south, south-east, and east. They have then generalized their bijections to certain models with large steps: for given $p \geq 1$, a bijection has been given between the half-plane and quarter-plane models obtained by keeping the small south-east step and replacing the two steps north and west of length 1 by the $p + 1$ steps of length p in directions between north and west. An article was submitted this year.

7.5. Putting Fürer's algorithm into practice with the BPAS library

Fast algorithms for integer and polynomial multiplication play an important role in scientific computing as well as in other disciplines. In 1971, Schönhage and Strassen designed an algorithm that improved the multiplication time for two integers of at most n bits to $O(\log n \log \log n)$. Martin Fürer presented a new algorithm that runs in $O(n \log n \cdot 2^{O(\log^* n)})$, where $\log^* n$ is the iterated logarithm of n. In a submitted article, Svyatoslav Covanov, together with Davood Mohajerani, Marc Moreno Maza and Lin-Xiao Wang, have explained how one can put Fürer's ideas into practice for multiplying polynomials over a prime field $\mathbb{Z}/p\mathbb{Z}$, for which p is a Generalized Fermat prime of the form $p = r^k + 1$ where k is a power of 2 and r is of machine word size. When k is at least 8, they have shown that multiplication inside such a prime field can be efficiently implemented via Fast Fourier Transform (FFT). Taking advantage of Cooley-Tukey tensor formula and the fact that r is a $2k$-th primitive root of unity in $\mathbb{Z}/p\mathbb{Z}$, they have obtained an efficient implementation of FFT over $\mathbb{Z}/p\mathbb{Z}$. This implementation outperforms comparable implementations either using other encodings of $\mathbb{Z}/p\mathbb{Z}$ or other ways to perform multiplication in $\mathbb{Z}/p\mathbb{Z}$.

7.6. Fast coefficient computation for algebraic power series in positive characteristic

In [5], Alin Bostan and Philippe Dumas, together with Xavier Caruso (CNRS, Rennes) and Gilles Christol (IMJ, Paris) have studied the algorithmic question of coefficient computation of algebraic power series in positive characteristic. They revisited Christol's theorem on algebraic power series in positive characteristic and proposed another proof for it. Their new proof combines several ingredients and advantages of existing proofs, which make it very well-suited for algorithmic purposes. The construction used in the new proof was then applied to the design of a new efficient algorithm for computing the Nth coefficient of a given algebraic power series over a perfect field of characteristic p. This algorithm has several nice features: it is more general, more natural and more efficient than previous algorithms. Not only the arithmetic complexity of the new algorithm is linear in $\log N$ and quasi-linear in p, but its dependency with respect to the degree of the input is much smaller than in the previously best algorithm. Moreover, when the ground field is finite, the new approach yields an even faster algorithm, whose bit complexity is linear in $\log N$ and quasi-linear in $\sqrt{p}$.

7.7. Counting walks with large steps in an orthant

In the past fifteen years, the enumeration of lattice walks with steps taken in a prescribed set and confined to a given cone, especially the first quadrant of the plane, has been intensely studied. As a result, the generating functions of quadrant walks are now well-understood, provided the allowed steps are *small*. In particular, having small steps is crucial for the definition of a certain group of bi-rational transformations of the plane. It has been proved that this group is finite if and only if the corresponding generating function is D-finite. This group is also the key to the uniform solution of 19 of the 23 small step models possessing a finite group. In contrast, almost nothing was known for walks with arbitrary steps. In [7], Alin Bostan together with Mireille Bousquet-Mélou (CNRS, Bordeaux) and Stephen Melczer (U. Pennsylvania, Philadelphia, USA), extended the definition of the group, or rather of the associated orbit, to this general case, and generalized the above uniform solution of small step models. When this approach works, it invariably yields a D-finite generating function. They applied it to many quadrant problems, including some infinite families. After developing the general theory, the authors of [7] considered the 13 110 two-dimensional models with steps in $\{-2, -1, 0, 1\}^2$ having at least one -2 coordinate. They proved that only 240 of them have a finite orbit, and solve 231 of them with our method. The 9 remaining models are the counterparts of the 4 models of the small step case that resist the uniform solution method (and which are known to have an algebraic generating function). They conjecture D-finiteness for their generating functions (but only two of them are likely to be algebraic!), and proved non-D-finiteness for the 12 870 models with an infinite orbit, except for 16 of them.

7.8. Subresultants of $(x-\alpha)^m$ and $(x-\beta)^n$, Jacobi polynomials and complexity

A previous article in 2017 described explicit expressions for the coefficients of the order-d polynomial subresultant of $(x-\alpha)^m$ and $(x-\beta)^n$ with respect to Bernstein's set of polynomials $\{(x-\alpha)^j(x-\beta)^{d-j},\ 0 \le j \le d\}$, for $0 \le d < \min\{m, n\}$. In [8], Alin Bostan, together with T. Krick, M. Valdettaro (U. Buenos Aires, Argentina) and A. Szanto (U. North Carolina, Raleigh, USA) further developed the study of these structured polynomials and showed that the coefficients of the subresultants of $(x-\alpha)^m$ and $(x-\beta)^n$ with respect to the monomial basis can be computed in *linear* arithmetic complexity, which is faster than for arbitrary polynomials. The result is obtained as a consequence of the amazing though seemingly unnoticed fact that these subresultants are scalar multiples of Jacobi polynomials up to an affine change of variables.

7.9. A numerical transcendental method in algebraic geometry

In "A transcendental method in algebraic geometry", Griffiths emphasized the role of certain multivariate integrals, known as *periods*, "to construct a continuous invariant of arbitrary smooth projective varieties".

Periods often determine the projective variety completely and therefore its algebraic invariants. Translating periods into discrete algebraic invariants is a difficult problem, exemplified by the long standing Hodge conjecture which describes how periods determine the algebraic cycles within a projective variety.

Recent progress in computer algebra makes it possible to compute periods with high precision and put transcendental methods into practice. In [10], Pierre Lairez and Emre Sertöz focus on algebraic surfaces and give a numerical method to compute Picard groups. As an application, they count smooth rational curves on quartic surfaces using the Picard group. It is the first time that this kind of computation is performed.

8. Partnerships and Cooperations

8.1. International Research Visitors

8.1.1. Internships

- Jiadong Han did a Master internship from March to August. Under the supervision of Pierre Lairez, he studied the computation of adaptive grid to improve the computation of the homology of semialgebraic sets.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

- Alin Bostan is part of the Scientific advisory board of the conference series *Effective Methods in Algebraic Geometry* (MEGA).
- Alin Bostan was a member of the Scientific advisory board of the conference *Algèbre, arithmétique et combinatoire des équations différentielles et aux différences*, CIRM (Luminy, France); ~ 60 participants.
- Alin Bostan is part of the scientific committee of the GDR EFI ("Functional Equations and Interactions") dependent on the mathematical institute (INSMI) of the CNRS. The goal of this GDR is to bring together various research communities in France working on functional equations in fields of computer science and mathematics.
- 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 (and current chair) of the steering committee of the *International Symposium on Symbolic and Algebraic Computation* (ISSAC, 3-year term, 2016–2018).
- Georges Gonthier is a member of the steering committee of the *Certified Programs and Proofs* Conference (CPP).

9.1.1.1. Member of the Organizing Committees

- Alin Bostan co-organizes, with Lucia Di Vizio, the *Séminaire Différentiel* between U. Versailles and Inria Saclay, with a bi-annual frequency (~ 30 participants per event).
- Alin Bostan co-organizes, with Lucia Di Vizio, the working group *Marches dans le quart de plan*, at Institut Henri Poincaré (Paris), with a bi-monthly frequency (~ 15 participants per event).

9.1.2. Scientific Events Selection

9.1.2.1. Reviewer

- Frédéric Chyzak has served as reviewer for the selection of the international conference ISSAC 2018.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

- Alin Bostan is on the editorial board of the *Journal of Symbolic Computation*.
- Georges Gonthier is on the editorial board of the *Journal of Formalized Reasoning*.

9.1.3.2. Reviewer - Reviewing Activities

- Alin Bostan has served as a reviewer for the journals: *Journal of Symbolic Computation*, *Journal of Combinatorial Theory, Series A*, *Applicable Algebra in Engineering Communications and Computing*, *Minnesota Journal of Undergraduate Mathematics*.
- Frédéric Chyzak has served multiple times as a reviewer for the *Journal of Symbolic Computation*.
- Pierre Lairez has served as a reviewer for the *Journal of Symbolic Computation*, *Journal of the ACM* and *Journal of Physics A*.

9.1.4. Invited Talks

- Alin Bostan has been invited to give a talk at the *Workshop on algebraic and analytic aspects of power series*, Universidade Lisboa, Lisbonne, Portugal, Jan. 2018.
- Alin Bostan has been invited to give a talk at the conference *Algebra, Arithmetic and Combinatorics of Differential and Difference Equations*, CIRM (Luminy), France, May 2018.
- Alin Bostan has been invited to give a talk at the conference *Grands réseaux aléatoires et marches contraintes*, in honor of the 75th birthday of Guy Fayolle, Dijon, France, Aug. 2018.
- Alin Bostan has been invited to give a talk at the conference *Combinatorics and Arithmetic for Physics: special days*, IHES, Bures-sur-Yvette, France, Oct. 2018.
- Frédéric Chyzak was invited invited speaker at the conference *Rencontres Arithmétiques du GDR Informatique Mathématique* (RAIM 2018), Gif-sur-Yvette, France, Nov. 2018.
- Georges Gonthier was a plenary keynote speaker at the *Federated Logic Conference (FLoC 2018)* in Oxford, July 2018.
- Georges Gonthier was invited speaker at the *Workshop on Modular Knowledge (Tetrapod)* during FLoC 2018, Oxford, July 2018.
- Georges Gonthier ws the keynote speaker of the *Future of Mathematical Proofs* workshop at the Heidelberg Laureate Forum, September 2018.

9.1.5. Leadership within the Scientific Community

9.1.5.1. Regular Research Seminar

The team organizes a regular seminar, with roughly 15–20 talks a year. The topics reflect the team's interests: computer algebra, combinatorics, number theory, formal proofs, and related domains. This year, we reduced a bit the number of talks in our seminar, as we have invested much time in setting up a working group with a talk every second week (see 9.1.5.2).

9.1.5.2. Research Working Group

This year we have set up a working group *Marches dans le quart de plan* around the study of walks in the quarter plan, a very active research topic in probability theory and enumerative combinatorics in recent years. The working group is organized at Institut Henri Poincaré, with a regularity of two sessions per month. The original purpose was to read the article "On the Nature of the Generating Series of Walks in the Quarter Plane" by T. Dreyfus, C. Hardouin, J. Roques, M. Singer, published in Invent. Math. this year. But the reality exceeded expectations: the working group attracted a dozen of people, working either in computer science or pure mathematics, who began to interact and a very good dynamic was created. Altogether, sixteen sessions have taken place so far, and we have decided to continue in 2019. From the team, Alin Bostan, Frédéric Chyzak, Guy Fayolle, and Pierre Lairez have given a total of 9 talks to this working group.

9.1.6. Scientific Expertise

- Georges Gonthier participated in a review of the software and algorithms of the Tezos blockchain conducted by the Inria Foundation during Spring 2018.

9.1.7. Research Administration

- Georges Gonthier serves on the Conseil de l'École Doctorale de Mathématiques Hadamard.

9.2. Teaching - Supervision - Juries

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

9.2.1. Teaching

Licence:

Pierre Lairez, *Introduction à l'informatique (INF311)*, TD, 40h, 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 (INF550)*, TD, 18h, M2, École polytechnique, France.

Pierre Lairez, *Les bases de la programmation et de l'algorithmique (INF411)*, TD, 40h, M1, École polytechnique, France.

9.2.2. Juries

- Frédéric Chyzak has been a member of the hiring jury at Inria (Concours CRCN 2018).
- Alin Bostan has served as a referee in the PhD jury of Timothée Pecatte, *Bornes inférieures et algorithmes de reconstruction pour des sommes de puissances affines*, ENS Lyon, July 11, 2018.
- Alin Bostan has served as an examiner in the PhD jury of Boris Djalal, *Formalisations en Coq pour la décision de problèmes en géométrie algébrique réelle*, Inria Sophia Antipolis, December 3, 2018.
- Alin Bostan has served as a member of the monitoring PhD committee of Youssef Abdelaziz, Univ. Paris 6.
- Alin Bostan has served as a member of the monitoring PhD committee of Manon Bertin, Univ. Rouen.

9.3. Popularization

9.3.1. Interventions

- Georges Gonthier testified before the *Mission d'information commune sur les blockchains* of the *Assemblée Nationale* in March.
- Georges Gonthier gave a public lecture and debate on blockchains at the *Institut Diderot* in September, jointly with M. Odonnat (Banque de France).

9.3.2. Internal action

- Georges Gonthier gave a presentation at the *Journées Scientifiques Inria 2018* in Bordeaux.

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journals

[1] J. P. BELL, F. CHYZAK, M. COONS, P. DUMAS. *Becker's conjecture on Mahler functions*, in "Transactions of the American Mathematical Society", 2018, 17 p. , In press, https://hal.inria.fr/hal-01885598

[2] P. BÜRGISSER, F. CUCKER, P. LAIREZ. *Computing the Homology of Basic Semialgebraic Sets in Weak Exponential Time*, in "Journal of the ACM (JACM)", December 2018, vol. 66, n$^{\mathrm{o}}$ 1, pp. 1-30 [*DOI :* 10.1145/3275242], https://hal.archives-ouvertes.fr/hal-01545657

[3] F. CHYZAK, T. DREYFUS, P. DUMAS, M. MEZZAROBBA. *Computing solutions of linear Mahler equations*, in "Mathematics of Computation", July 2018, vol. 87, pp. 2977-3021 [*DOI :* 10.1090/MCOM/3359], https://hal.inria.fr/hal-01418653

[4] A. MAHBOUBI, G. MELQUIOND, T. SIBUT-PINOTE. *Formally Verified Approximations of Definite Integrals*, in "Journal of Automated Reasoning", March 2018, pp. 1-20 [*DOI :* 10.1007/S10817-018-9463-7], https://hal.inria.fr/hal-01630143

International Conferences with Proceedings

[5] A. BOSTAN, X. CARUSO, G. CHRISTOL, P. DUMAS. *Fast Coefficient Computation for Algebraic Power Series in Positive Characteristic*, in "Thirteenth Algorithmic Number Theory Symposium ANTS-XIII", Madison, United States, Algorithmic Number Theory, July 2018, https://arxiv.org/abs/1806.06543 , https://hal.archives-ouvertes.fr/hal-01816375

[6] A. BOSTAN, F. CHYZAK, P. LAIREZ, B. SALVY. *Generalized Hermite Reduction, Creative Telescoping and Definite Integration of D-Finite Functions*, in "ISSAC 2018 - International Symposium on Symbolic and Algebraic Computation", New York, United States, July 2018, pp. 1-8 [*DOI :* 10.1145/3208976.3208992], https://hal.inria.fr/hal-01788619

Other Publications

[7] A. BOSTAN, M. BOUSQUET-MÉLOU, S. MELCZER. *Counting walks with large steps in an orthant*, May 2018, https://arxiv.org/abs/1806.00968 - working paper or preprint, https://hal.archives-ouvertes.fr/hal-01802706

[8] A. BOSTAN, T. KRICK, A. SZANTO, M. VALDETTARO. *Subresultants of $(x-\alpha)^m$ and $(x-\beta)^n$, Jacobi polynomials and complexity*, December 2018, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01966640

[9] F. CHYZAK, K. YEATS. *Bijections between Łukasiewicz walks and generalized tandem walks*, October 2018, working paper or preprint, https://hal.inria.fr/hal-01891792

[10] P. LAIREZ, E. CAN SERTÖZ. *A numerical transcendental method in algebraic geometry*, November 2018, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01932147

References in notes

[11] 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

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

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

[14] 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

[15] 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

[16] 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

[17] 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

[18] 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

[19] 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

[20] 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

[21] 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

[22] 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

[23] 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

[24] 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

[25] 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

[26] 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)

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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

[32] 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

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

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

[35] 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

[36] 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

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

[38] 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

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

[40] 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

[41] 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

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

[43] 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

[44] 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

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

[46] 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

[47] 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

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

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

[50] 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

[51] 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

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

[53] 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

[54] 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

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

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

[57] 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/

[58] 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

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

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

[61] 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

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

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