Activity Report 2016

Project-Team SPECFUN

Symbolic Special Functions : Fast and Certified

RESEARCH CENTER
Saclay - Île-de-France

THEME
Algorithmics, Computer Algebra and Cryptology

Table of contents

1. Members **1**
2. Overall Objectives **1**
2.1. Scientific challenges, expected impact 1
2.1.1. Use computer algebra but convince users beyond reasonable doubt 3
2.1.2. Make computer algebra and formal proofs help one another 3
2.1.3. Experimental mathematics with special functions 4
2.2. Research axes 4
2.2.1. Computer algebra certified by the Coq system 4
2.2.1.1. Libraries of formalized mathematics 4
2.2.1.2. Manipulation of large algebraic data in a proof assistant 4
2.2.1.3. Formal-proof-producing normalization algorithms 5
2.2.2. Better symbolic computations with special functions 5
2.2.2.1. Special-function integration and summation 5
2.2.2.2. Applications to experimental mathematics 5
2.2.3. Interactive and certified mathematical web sites 6
3. Research Program **6**
3.1. Studying special functions by computer algebra 6
3.1.1. Equations as a data structure 6
3.1.2. Algorithms combining functions 7
3.1.3. Solving functional equations 7
3.1.4. Multi-precision numerical evaluation 7
3.1.5. Guessing heuristics 7
3.1.6. Complexity-driven design of algorithms 7
3.2. Trusted computer-algebra calculations 8
3.2.1. Encyclopedias 8
3.2.2. Computer algebra and symbolic logic 8
3.2.3. Certifying systems for computer algebra 8
3.2.4. Semantics for computer algebra 8
3.2.5. Formal proofs for symbolic components of computer-algebra systems 8
3.2.6. Formal proofs for numerical components of computer-algebra systems 8
3.3. Machine-checked proofs of formalized mathematics 9
3.3.1. Logical foundations and proof assistants 9
3.3.2. Computations in formal proofs 9
3.3.3. Large-scale computations for proofs inside the Coq system 9
3.3.4. Relevant contributions from the Mathematical Component libraries 10
3.3.5. User interaction with the proof assistant 10
4. Highlights of the Year **10**
5. New Software and Platforms **11**
5.1. DDMF 11
5.2. DynaMoW 11
5.3. ECS 11
5.4. Math-Components 11
5.5. Ssreflect 12
6. New Results **12**
6.1. Formally certified computation of definite integrals 12
6.2. Real closed fields 12
6.3. Combinatorial walks with small steps in the quarter plane 12
6.4. Multiple binomial sums 13
6.5. Algebraic diagonals and walks 13

6.6. A human proof of the Gessel conjecture 13
6.7. Enumeration of 3-dimensional lattice walks confined to the positive octant 13
6.8. Computation of the similarity class of the p-curvature 14
6.9. Efficient algorithms for mixed creative telescoping 14
6.10. Fast computation of the Nth term of an algebraic series over a finite prime field 14
6.11. Formal methods for cryptocurrencies 14
6.12. Computing solutions of linear Mahler equations 14
6.13. Formal solutions of singularly perturbed linear differential systems 15
7. Bilateral Contracts and Grants with Industry **15**
8. Partnerships and Cooperations **15**
8.1. National Initiatives 15
8.2. European Initiatives 15
8.3. International Research Visitors 16
9. Dissemination **16**
9.1. Promoting Scientific Activities 16
9.1.1. Scientific Events Organisation 16
9.1.1.1. General Chair, Scientific Chair 16
9.1.1.2. Member of the Organizing Committees 16
9.1.2. Scientific Events Selection 17
9.1.2.1. Chair of Conference Program Committees 17
9.1.2.2. Member of the Conference Program Committees 17
9.1.2.3. Reviewer 17
9.1.3. Journal 17
9.1.3.1. Member of the Editorial Boards 17
9.1.3.2. Reviewer - Reviewing Activities 17
9.1.4. Invited Talks 17
9.1.5. Leadership within the Scientific Community 18
9.1.6. Research Administration 18
9.2. Teaching - Supervision - Juries 18
9.2.1. Teaching 18
9.2.2. Supervision 18
9.2.3. Juries 18
9.3. Popularization 18
10. Bibliography **19**

Project-Team SPECFUN

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

Keywords:

Computer Science and Digital Science:

2.1.10. - Domain-specific languages
2.1.11. - Proof languages
2.4.3. - Proofs
4.5. - Formal methods for security
7.2. - Discrete mathematics, combinatorics
7.4. - Logic in Computer Science
7.6. - Computer Algebra
7.7. - Number theory
7.11. - Performance evaluation

Other Research Topics and Application Domains:

9.4.2. - Mathematics
9.4.3. - Physics

1. Members

Research Scientists

Frédéric Chyzak [Team leader, Inria, Researcher, HDR]
Alin Bostan [Inria, Researcher]
Philippe Dumas [Inria, Researcher]
Georges Gonthier [Inria, Senior Researcher, since October 2016]
Pierre Lairez [Inria, Researcher, since October 2016]
Assia Mahboubi [Team co-leader, Inria, Researcher]

Engineer

Maxence Guesdon [Inria, 40%]

PhD Students

Louis Dumont [École Polytechnique, until October 2016]
Thomas Sibut-Pinote [École Polytechnique]

Post-Doctoral Fellow

Suzy S. Maddah [Inria]

Visiting Scientist

Marc Mezzarobba [CNRS]

Administrative Assistant

Christine Biard [Inria]

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” [72]. 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 [26]. 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 [21] remains one of the most cited mathematical publications ever and has motivated the 10-year-long project of writing its successor [23]. 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 [50] 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 [45], the above-mentioned Kepler Conjecture [50], 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) [26] 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 [68]. 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 [68] 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 [56]: "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 [64] 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 [22] 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 [73], 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 [54], [29]. 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) [43].

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 [24], [27] 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 [49].

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 [32] 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 [31], [36], [34]. 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 [21] 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 [74], 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 [74] 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 [75]. 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 [63], [71], [39], [37], [38], [57]. The past ÉPI Algorithms contributed several implementations (*gfun* [66], *Mgfun* [39]).

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* [62]) 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 [36]. All this is based on subtle algorithms for Hermite–Padé approximants [25]. Moreover, guessing can at times be complemented by proven quantitative results that turn the heuristics into an algorithm [33]. 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 [44], 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. [30].

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 [21], [23], [67] 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 [70], [49], [59] 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 [42]. A discussion about the systematic separation of the search for a solution and the checking of the solution is already clearly outlined in [55].

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 [60], which is based on set theory. In particular, the calculus of construction (CoC) [40] and its extension with inductive types (CIC) [41], have been studied for more than 20 years and been implemented by several independent tools (like Lego, Matita, and Agda). Its reference implementation, Coq [68], 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 [43] 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 [46] 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 [69], 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 [51], [52], [53], although some existing developments of interest [28], [61] 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 [48], [47], 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 [65]. 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 has received the *ISSAC Distinguished Paper Award* for his joint work with T. Vaccon on p-adic differential equations [58].

5. New Software and Platforms

5.1. 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.
This year, Maxence Guesdon started to port DDMF to the new DynaMoW. To this end, a special environment has been set up to be able to use the Inria continuous-integration platform.

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

5.2. DynaMoW

Dynamic Mathematics on the Web
FUNCTIONAL DESCRIPTION

DynaMoW is a programming tool for controlling the generation of mathematical websites that embed dynamical mathematical contents generated by computer-algebra calculations. Implemented in OCaml.
After a complete redesign and rewrite last year, to get more reactiveness and configurability, the implementation of DynaMoW was made more robust this year while porting ECS to this new library. It was next further enhanced, in particular in order have informative and reliable traces of execution, to help with the debugging of asynchronous parallel executions of services.

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

5.3. ECS

Encyclopedia of Combinatorial Structures
FUNCTIONAL DESCRIPTION

ECS is an online 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.
This year, we finalized the port of ECS to the last evolutions of DynaMoW. A new website was setup, and ECS is now again online, after a few years of discontinuation for technical reasons.

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

5.4. 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.

This year we experimented the maintenance of the library using the public repository stored on the github platform since December 2015. This allowed for merging several contributions from external users and improved significantly the communication with the community of users. A new website has also been set up, which includes pointers to various teaching and documentation resources.

- Participants: Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, Francois Garillot, Georges Gonthier, Stéphane Le Roux, Assia Mahboubi, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi and Russell O'connor
- Contact: Assia Mahboubi
- URL: http://www.msr-inria.fr/projects/mathematical-components-2/

5.5. Ssreflect

FUNCTIONAL DESCRIPTION

Ssreflect is a tactic language extension to the Coq system, developed by the Mathematical Components team. This year we improved the manual of the language, in order to document new features and to clarify some older parts of the document.

- Participants: Cyril Cohen, Yves Bertot, Laurence Rideau, Enrico Tassi, Laurent Thery, Assia Mahboubi and Georges Gonthier
- Contact: Yves Bertot
- URL: http://ssr.msr-inria.inria.fr/

6. New Results

6.1. Formally certified computation of definite integrals

Assia Mahboubi and Thomas Sibut-Pinote, in collaboration with Guillaume Melquiond (Toccata), have developed a Coq library for the computation of intervals approximating the value of definite integrals for elementary mathematical functions. This library provides an automated tool which builds automatically 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. A description of this work was published in the proceeding of the ITP 2016 conference [13]. An extension to domains including singularities of the integrand is in progress.

6.2. Real closed fields

Assia Mahboubi has worked with Henri Lombardi (Université de Franche Comté) on a constructive axiomatization of real closed fields. For this purpose, they have proposed an equational theory based on virtual roots and close to the classical notion of local real closed rings. This is a first step toward a constructive understanding of o-minimal structures. This work has been accepted for publication in Contemporary Mathematics [19].

6.3. Combinatorial walks with small steps in the quarter plane

Alin Bostan and Frédéric Chyzak, together with Mark van Hoeij (Florida State University), Manuel Kauers (Johannes Kepler University), and Lucien Pech (former intern), have applied their algorithms on special functions to generate complete, quantitative results in the enumerative theory of combinatorial walks with small steps in the quarter plane [2]. They gave the first proof that differential equations conjectured years ago by Bostan and Kauers are indeed satisfied by the corresponding generating functions. They also obtained explicit hypergeometric expressions for the latter, and could provably determine which of the generating functions are transcendental or algebraic.

6.4. 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 and also all the sequences with algebraic generating function. Alin Bostan and Pierre Lairez, together with Bruno Salvy (Inria and ENS Lyon), have studied in [5] the representation of the generating functions of binomial sums by integrals of rational functions. The outcome is twofold. Firstly, we show 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 propose 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.

6.5. 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 [35], Alin Bostan and Louis Dumont, together with Bruno Salvy (Inria and ENS Lyon), 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. We propose an algorithm that computes an annihilating polynomial for $\mathsf{Diag}F$. We give a precise bound on the size of this polynomial and show that generically, this polynomial is the minimal polynomial and that its size reaches the bound. The algorithm runs in time quasi-linear in this bound, which grows exponentially with the degree of the input rational function. We then address the related problem of enumerating directed lattice walks. The insight given by our study leads to a new method for expanding the generating power series of bridges, excursions and meanders. We show 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 is presented in [3].

6.6. 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, that move by unit steps in any of the following directions: West, North-East, East and South-West. In 2001, Ira Gessel conjectured a closed-form expression for the number of 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. Alin Bostan, together with Irina Kurkova (Univ. Paris 6) and Kilian Raschel (CNRS and Univ. Tours), have proposed in [4] 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.

6.7. Enumeration of 3-dimensional lattice walks confined to the positive octant

Small step walks in 2D are by now quite well understood, but almost everything remains to be done in higher dimensions. Alin Bostan, together with Mireille Bousquet-Mélou (CNRS and Univ. Bordeaux), Manuel Kauers (Johannes Kepler Univ.) and Stephen Melczer (Univ. of Waterloo and ENS Lyon), have explored in [1] the classification problem for 3-dimensional walks with unit steps confined to the positive octant. The first difficulty is their number: there are 11 074 225 cases (instead of 79 in dimension 2). In our work, we focused on the 35 548 that have at most six steps. We applied to them a combined approach, first experimental and then rigorous. Among the 35 548 cases, we first found 170 cases with a finite group; in the remaining cases, our experiments suggest that the group is infinite. We then rigorously proved D-finiteness of the generating series in all the 170 cases, with the exception of 19 intriguing step sets for which the nature of the generating function still remains unclear. In two challenging cases, no human proof is currently known, and we derived computer-algebra proofs, thus constituting the first proofs for those two step sets.

6.8. Computation of the similarity class of the p-curvature

The p-curvature of a system of linear differential equations in positive characteristic p is a matrix that measures how far the system is from having a basis of polynomial solutions. Alin Bostan, together with Xavier Caruso (CNRS and Univ. Rennes) and Éric Schost (Univ. Waterloo), have showed in [10] that the similarity class of the p-curvature can be determined without computing the p-curvature itself. More precisely, we have designed an algorithm that computes the invariant factors of the p-curvature in time quasi-linear in $\sqrt{p}$. This is much less than the size of the p-curvature, which is generally linear in p. The new algorithm allowed to answer a question originating from the study of the Ising model in statistical physics.

6.9. Efficient algorithms for mixed creative telescoping

Creative telescoping is a powerful computer algebra paradigm –initiated by Doron Zeilberger in the 90's– for dealing with definite integrals and sums with parameters. Alin Bostan and Louis Dumont, together with Bruno Salvy (Inria and ENS Lyon), have addressed in [12] the mixed continuous–discrete case, and have focussed on the integration of bivariate hypergeometric-hyperexponential terms. We have designed a new creative telescoping algorithm operating on this class of inputs, based on a Hermite-like reduction procedure. The new algorithm has two nice features: it is efficient and it delivers, for a suitable representation of the input, a minimal-order telescoper. Its analysis reveals tight bounds on the sizes of the telescoper it produces.

6.10. Fast computation of the Nth term of an algebraic series over a finite prime field

Alin Bostan and Philippe Dumas, together with Gilles Christol (IMJ), have addressed in [11] the question of computing one selected term of an algebraic power series. In characteristic zero, the best algorithm currently known for computing the Nth coefficient of an algebraic series uses differential equations and has arithmetic complexity quasi-linear in $\sqrt{N}$. We show that over a prime field of positive characteristic p, the complexity can be lowered to $O(\log N)$. The mathematical basis for this dramatic improvement is a classical theorem stating that a formal power series with coefficients in a finite field is algebraic if and only if the sequence of its coefficients can be generated by an automaton. We revisit and enhance two constructive proofs of this result for finite prime fields. The first proof uses Mahler equations, whose sizes appear to be prohibitively large. The second proof relies on diagonals of rational functions; we turn it into an efficient algorithm, of complexity linear in $\log N$ and quasi-linear in p.

6.11. Formal methods for cryptocurrencies

Georges Gonthier and Thomas Sibut-Pinote, along with a team of researchers from Microsoft Research and Inria, participated in a hackathon internal to Microsoft Research with the goal to apply formal methods to the verification of the smart contracts involved in the Ethereum platform. They outlined a framework to analyze and verify both the runtime safety and the functional correctness of Ethereum contracts by translation to F*, a functional programming language aimed at program verification. This work was published in the proceedings of the PLAS 2016 conference [9].

6.12. 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 [17], Frédéric Chyzak and Philippe Dumas, together with Thomas Dreyfus (Université Claude Bernard Lyon 1) and Marc Mezzarobba (visiting scientist from UPMC), have presented algorithms for solving linear Mahler equations for series, polynomials, and rational functions, and have obtained polynomial-time complexity under a mild assumption.

6.13. Formal solutions of singularly perturbed linear differential systems

Suzy Maddah, together with Boulay Barkatou (Université de Limoges), has obtained algorithms for computing formal invariants of singularly-perturbed linear differential systems [20].

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

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

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. 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, ÉNS Lyon). Participants: Assia Mahboubi, Th. Sibut-Pinote.
Website: http://fastrelax.gforge.inria.fr/.

8.2. European Initiatives

8.2.1. Collaborations in European Programs, Except FP7 & H2020

- Program: COST
- Project acronym: EUTYPES (CA15123)
- Project title: The European research network on types for programming and verification
- Duration: October 2015 - October 2019
- Coordinator: Herman Geuvers (Radboud University, Nijmegen, the Netherlands)
- Other partners: Czech Republic, Estonia, Macedonia, Germany, Greece, the Netherlands, Norway, Poland, Serbia, Slovenia, United Kingdom.

- Abstract: Types are pervasive in programming and information technology. A type defines a formal interface between software components, allowing the automatic verification of their connections, and greatly enhancing the robustness and reliability of computations and communications. In rich dependent type theories, the full functional specification of a program can be expressed as a type. Type systems have rapidly evolved over the past years, becoming more sophisticated, capturing new aspects of the behaviour of programs and the dynamics of their execution. This COST Action will give a strong impetus to research on type theory and its many applications in computer science, by promoting: (1) the synergy between theoretical computer scientists, logicians and mathematicians to develop new foundations for type theory, for example as based on the recent development of "homotopy type theory", (2) the joint development of type theoretic tools as proof assistants and integrated programming environments, (3) the study of dependent types for programming and its deployment in software development, (4) the study of dependent types for verification and its deployment in software analysis and verification. The action will also tie together these different areas and promote cross-fertilisation. Europe has a strong type theory community, ranging from foundational research to applications in programming languages, verification and theorem proving, which is in urgent need of better networking. A COST Action that crosses the borders will support the collaboration between groups and complementary expertise, and mobilise a critical mass of existing type theory research.

8.3. International Research Visitors

8.3.1. Research Stays Abroad

- Thomas Sibut-Pinote has spent two months at Microsoft Research Cambridge, visiting Georges Gonthier and working on mathematical libraries for the Lean proof assistant. He also participated in a hackathon internal to Microsoft Research with the goal to apply formal methods to the verification of the smart contracts involved in the Ethereum framework for cryptocurrency.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

- 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 has been elected member of the steering committee of the *International Symposium on Symbolic and Algebraic Computation* (ISSAC, 3-year term).
- Alin Bostan is part of the Scientific advisory board of the conference series *Effective Methods in Algebraic Geometry* (MEGA).
- Assia Mahboubi serves in the scientific advisory board of the Mathematics, Algorithms and Proofs community (MAP).
- Assia Mahboubi has served in the scientific advisory board of the École du GDR Informatique – Mathématiques 2016.
- Georges Gonthier is a member of the steering committee of the *Certified Programs and Proofs* conference (CPP).

9.1.1.2. Member of the Organizing Committees

- Assia Mahboubi has co-organized the *MAP'16 conference* at CIRM (Marseille), with B. Spitters (Aarhus University, Denmark) and P. Schuster (University of Verona, Italy): http://scientific-events.weebly.com/1508.html.

- Assia Mahboubi has co-organized, with E. Tassi (Marelle), the workshop *Mathematical Components: an introduction*, satellite of the conference ITP 2016: https://itp2016.inria.fr/workshops/#mc.
- Assia Mahboubi has co-organized, with K. Nakata (FireEye, Germany), the workshop *TTT*, satellite of the POPL'17 conference: http://popl17.sigplan.org/track/TTT-2017.
- Suzy Maddah has co-organized the gathering *Functional Equations in Limoges (FELIM 2016)*: https://indico.math.cnrs.fr/event/919/.
- Suzy Maddah has co-organized a session on software for the symbolic study of functional equations at the *International Congress on Mathematical Software (ICMS 2016)*: http://icms2016.zib.de/.
- Alin Bostan has co-organized, together with Bruno Salvy (Inria and ENS Lyon) and Conrado Martinez (UPC BarcelonaTech), the conference *ALEA 2016* at CIRM (Marseille): http://scientific-events.weebly.com/1406.html.

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

- Alin Bostan has served as Symbolic Computation track chair for the international conference SYNASC 2016.

9.1.2.2. Member of the Conference Program Committees

- Assia Mahboubi has served as a member of the program committee for the international conferences with proceedings CPP 2017, ITP 2016, CSL 2016, CICM 2016 and SCSS 2016. She has also served as member of the program committee for the MAP 2016 conference and for the HaTT workshop.
- Alin Bostan has served as a member of the program committee of the ISSAC 2016 and of the SYNASC 2016 international conferences.

9.1.2.3. Reviewer

- Assia Mahboubi has served as reviewer for the proceedings of the international conferences CPP 2017, ITP 2016, CSL 2016, CICM 2016 and SCSS.
- Alin Bostan has served as reviewer for the proceedings of the international conferences FPSAC 2016, ISSAC 2016, AofA 2016 and SYNASC 2016.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

- Georges Gonthier is a member of the editorial board of the *Journal of Formalized Reasoning*.

9.1.3.2. Reviewer - Reviewing Activities

- Assia Mahboubi has served as a reviewer for the *Journal of Automated Reasoning*.
- Frédéric Chyzak has served as a reviewer for the journals: *Applicable Algebra in Engineering, Communication and Computing*; *Journal of Symbolic Computation*; *Journal of Algebra*; and *Electronic Journal of Combinatorics*.
- Alin Bostan has served as a reviewer for the journals: *Journal of Complexity*; *Mathematics of Computation*; *Linear Algebra and its Applications*; *Journal of Physics A: Mathematical and Theoretical*; *Journal of Algebra and its Applications*; *Journal of Symbolic Computation*; *Advances in Applied Mathematics*.

9.1.4. Invited Talks

- Assia Mahboubi has given an invited talk at the special trimester Mathematics – Computer Science – Philosophy CIPPMI in Toulouse in March 2016.
- Assia Mahboubi has given an invited lecture for the students of École Normale Supérieure Paris-Saclay in September 2016.

- Alin Bostan has been invited to give a series of five lectures at the summer school *Algorithmic and Enumerative Combinatorics* (RISC, Hagenberg, Austria), August 1–5, 2016.
- Philippe Dumas has given an invited lecture about divide-and-conquer recurrences at the *Journées Aléa* (CIRM, Marseille, France), March 7–11, 2016 [18].

9.1.5. Leadership within the Scientific Community

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

9.1.6. Research Administration

- Assia Mahboubi is member of the *Commission Scientifique* of Inria Saclay — Île-de-France.
- Georges Gonthier is a member of the board of the *École doctorale de mathématiques Hadamard (EDMH)*.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

License:

Louis Dumont, two L1 maths courses, 64h, Université Paris-Sud, France.

Master:

Assia Mahboubi, *Proof Assistants*, 18h, M2, Denis Diderot University (Paris), France.

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.

9.2.2. Supervision

PhD in progress: Thomas Sibut-Pinote, “Calcul numérique et démonstrations mathématiques: de la rigueur à la preuve formelle”, September 2014, Assia Mahboubi

PhD in progress: Louis Dumont, “Algorithmes rapides pour le calcul symbolique de certaines intégrales de contour à paramètre”, started in September 2013, supervised by Alin Bostan and B. Salvy.

Master intership (M1): G. Boisseau and Th. Huffschmitt, *Combination of decision procedures in presence of meta-variables*, École Polytechnique, supervised by Assia Mahboubi (jointly with S. Graham-Lengrand from LIX).

9.2.3. Juries

- Alin Bostan has served as a jury member of the French *Agrégation de Mathématiques – épreuve de modélisation, option C*.
- Alin Bostan has served as an examiner in the PhD jury of Aladin Virmaux, *Théorie des représentations combinatoires de tours de monoïdes, Application à la catégorification et aux fonctions de parking*, Université Paris-Saclay, June 13, 2016.

9.3. Popularization

- Assia Mahboubi has written a paper [8] for the quarterly journal of the Royal Dutch Mathematical Society.
- Alin Bostan has given a talk at the *Mathematic Park* seminar at IHP, Paris, on January 23rd 2016.

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journals

[1] A. BOSTAN, M. BOUSQUET-MÉLOU, M. KAUERS, S. MELCZER. *On 3-dimensional lattice walks confined to the positive octant*, in "Annals of Combinatorics", October 2016, 36 p. , First Online: 14 October 2016 [*DOI :* 10.1007/S00026-016-0328-7], https://hal.archives-ouvertes.fr/hal-01063886

[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", 2016, 29 p. [*DOI :* 10.1016/J.EJC.2016.10.010], https://hal.inria.fr/hal-01332175

[3] A. BOSTAN, L. DUMONT, B. SALVY. *Algebraic Diagonals and Walks: Algorithms, Bounds, Complexity*, in "Journal of Symbolic Computation", 2016 [*DOI :* 10.1016/J.JSC.2016.11.006], https://hal.archives-ouvertes.fr/hal-01244914

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

[5] A. BOSTAN, P. LAIREZ, B. SALVY. *Multiple binomial sums*, in "Journal of Symbolic Computation", 2016 [*DOI :* 10.1016/J.JSC.2016.04.002], https://hal.archives-ouvertes.fr/hal-01220573

[6] P. LAIREZ. *Computing periods of rational integrals*, in "Mathematics of Computation", November 2016, vol. 85, pp. 1719-1752 [*DOI :* 10.1090/MCOM/3054], https://hal.inria.fr/hal-00981114

[7] A. MAHBOUBI. *An Induction Principle over Real Numbers*, in "Archive for Mathematical Logic", 2016 [*DOI :* 10.1007/S00153-016-0513-8], https://hal.inria.fr/hal-01376054

[8] A. MAHBOUBI. *Machine-checked mathematics*, in "Nieuw Archief voor Wiskunde", September 2016, vol. 5/17, n^o 3, 5 p. , https://hal.inria.fr/hal-01363284

International Conferences with Proceedings

[9] K. BHARGAVAN, A. DELIGNAT-LAVAUD, C. FOURNET, A. GOLLAMUDI, G. GONTHIER, N. KOBEISSI, N. KULATOVA, A. RASTOGI, T. SIBUT-PINOTE, N. SWAMY, S. ZANELLA-BÉGUELIN. *Formal Verification of Smart Contracts: Short Paper*, in "ACM Workshop on Programming Languages and Analysis for Security", Vienna, Austria, October 2016 [*DOI :* 10.1145/2993600.2993611], https://hal.inria.fr/hal-01400469

[10] A. BOSTAN, X. CARUSO, É. SCHOST. *Computation of the Similarity Class of the p-Curvature*, in "ISSAC 2016", Waterloo, ON, Canada, ACM Press, July 2016, pp. 111–118 [*DOI :* 10.1145/2930889.2930897], https://hal.archives-ouvertes.fr/hal-01321043

[11] A. BOSTAN, G. CHRISTOL, P. DUMAS. *Fast Computation of the Nth Term of an Algebraic Series over a Finite Prime Field*, in "ISSAC '16", Waterloo, Ontario, Canada, July 2016 [*DOI :* 10.1145/2930889.2930904], https://hal.inria.fr/hal-01389421

[12] A. BOSTAN, L. DUMONT, B. SALVY. *Efficient Algorithms for Mixed Creative Telescoping*, in "ISSAC 2016", Waterloo, Canada, Proceedings ISSAC'16, pp. 127–134, ACM Press, 2016., July 2016, 8 p. [*DOI :* 10.1145/2930889.2930907], https://hal.inria.fr/hal-01317940

[13] A. MAHBOUBI, G. MELQUIOND, T. SIBUT-PINOTE. *Formally Verified Approximations of Definite Integrals*, in "Interactive Theorem Proving", Nancy, France, J. C. BLANCHETTE, S. MERZ (editors), Lecture Notes in Computer Science, August 2016, vol. 9807 [*DOI :* 10.1007/978-3-319-43144-4_17], https://hal.inria.fr/hal-01289616

Scientific Books (or Scientific Book chapters)

[14] A. BOSTAN, F. CHYZAK, M. GIUSTI, R. LEBRETON, G. LECERF, B. SALVY, E. SCHOST. *Algorithmes Efficaces en Calcul Formel*, January 2017, Cette version est la prépublication de l'ouvrage du même nom à paraître dans la collection https://hal.archives-ouvertes.fr/AECF/, https://hal.inria.fr/hal-01431717

Research Reports

[15] G. GONTHIER, A. MAHBOUBI, E. TASSI. *A Small Scale Reflection Extension for the Coq system*, Inria Saclay Ile de France, 2016, n° RR-6455, https://hal.inria.fr/inria-00258384

Other Publications

[16] A. BOSTAN. *Algebraicity and transcendence of power series: combinatorial and computational aspects*, August 2016, Lecture, https://hal.inria.fr/cel-01391907

[17] F. CHYZAK, T. DREYFUS, P. DUMAS, M. MEZZAROBBA. *Computing solutions of linear Mahler equations*, 2016, working paper or preprint, https://hal.inria.fr/hal-01418653

[18] P. DUMAS. *Divide-and-Conquer Algebra and Analysis : Lecture given at Journées ALÉA 2016*, March 2016, 80 p. , Lecture, https://hal.inria.fr/cel-01388741

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

[20] 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

References in notes

[21] 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

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

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

[24] 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

[25] 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° 3, pp. 804–823

[26] 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

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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° 12, 50 p. , http://dx.doi.org/10.1088/1751-8113/42/12/125206

[32] 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

[33] 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

[34] 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

[35] 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

[36] 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

[37] 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)

[38] 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

[39] 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

[40] 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

[41] 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

[42] 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

[43] 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

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

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

[46] 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

[47] 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

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

[49] 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

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

[51] 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

[52] 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

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

[54] 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

[55] 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

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

[57] 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

[58] P. LAIREZ, T. VACCON. *On p-adic differential equations with separation of variables*, in "ISSAC 2016", Waterloo, Canada, Proceedings ISSAC'16, pp. 127–134, ACM Press, 2016., July 2016, 8 p. , http://dx.doi.org/10.1145/2930889.2930907

[59] 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

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

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

[62] 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

[63] 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

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

[65] 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

[66] 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

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

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

[69] 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/

[70] 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

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

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

[73] 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

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

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