SCML-2026: International Conference on Symbolic Computation and Machine Learning Extended Abstracts

Bruno Buchberger, François Charton, Matthew England, Cezary Kaliszyk, Manuel Kauers, Hiroshi Kera, Temur Kutsia, Bernhard Moser, Markus Schedl, Wolfgang Schreiner, Martina Seidl, Wolfgang Windsteiger

June 29, 2026

https://scml.risc.jku.at

RISC Proceedings on
Symbolic Computation and Machine Learning
Publication Number: 3

https://doi.org/10.35011/risc-proceedings-scml.3

Cite as: Bruno Buchberger, François Charton, Matthew England, Cezary Kaliszyk, Manuel Kauers, Hiroshi Kera, Temur Kutsia, Bernhard Moser, Markus Schedl, Wolfgang Schreiner, Martina Seidl, Wolfgang Windsteiger: SCML-2026: International Conference on Symbolic Computation and Machine Learning – Extended Abstracts. *RISC Proceedings on Symbolic Computation and Machine Learning*, 3, June 29, 2026. SCML, https://scml.risc.jku.at, doi.org/10.35011/risc-proceedings-scml.3.

The SCML publication forum is dedicated to all research that strives to combine "Symbolic Computation" (SC) and "Machine Learning" (ML) as two major approaches to "Artificial Intelligence", in particular to the application of ML to SC, the application of SC to ML, and the hybrid combination of SC and ML to solving problems. More information about SCML is available at https://scml.risc.jku.at.

Steering Commitee and Editorial Board

Scientific Commitee

SCML-2026: International Conference on Symbolic Computation and Machine Learning

July 6–8, 2026, Hagenberg, Austria

Extended Abstracts

The SCML-2026 Program Committee

Bruno Buchberger François Charton Matthew England
Cezary Kaliszyk Manuel Kauers Hiroshi Kera
Temur Kutsia Bernhard Moser Markus Schedl
Wolfgang Schreiner Martina Seidl Wolfgang Windsteiger

Preface

Bruno Buchberger

A Symbolic Computation Summer

Artificial Intelligence (AI) has never been, and still is not, a clearly defined field. In recent years, at RISC, we have had intensive discussions on this notion, particularly in connection with Symbolic Computation (SC)—namely both the Automated Reasoning (AR) and the Computer Algebra (CA) part of SC—and the various views on the relation between AI and SC.

In fact, in the early days of SC, approximately 1965–1980, SC was regarded as AI. For example, the early SC system MACSYMA had "MAC" ("Machine-Aided Cognition") in its name, indicating that some kind of "machine intelligence" would evolve by creating algorithms for CA and AR, and some special intelligence is necessary for creating such algorithms.

Now, since 2022, with the appearance of ChatGPT ("the LLM flash"), the general view has been that "artificial intelligence" is based on Machine Learning (ML), i.e., by extracting (logically simple, but structurally massive) algorithms from massive data, and the symbolic approach to AI seems to be outdated.

As a result of our discussion, at RISC, we tend to agree that the situation is actually quite simple:

- There is a vague quest to automate (algorithmize) the intellectually demanding process of inventing algorithms for demanding problems. Some people call this "Artificial Intelligence", but in fact this quest is as old as mathematics itself: Automate the solution of problems by systematic procedures ("algorithms") , whose correctness is mathematically proven, and, in an act of self-reflection, automate (algorithmize) the process of inventing such algorithms and the theorems and proofs on which they are based. (Personally, I think that self-reflection in some way is the essence of mathematics and even the essence of intelligence[1].)

- In recent decades (since approx. 1970), the quest to automate the mathematical invention process has taken on tremendous vigor and speed, with two methodologically very different approaches: SC and ML, so that we could summarize our view with the simple formula: AI = SC + ML[2]. Both SC and ML had their "summers" and "winters" over the past sixty years. There have been times when SC seemed quite promising (e.g., during the Japanese Fifth Generation Project based on resolution theorem proving, or during the era of expert systems), and ML was just considered a nice playground for watching "neural networks" do fancy but useless things. Then disappointment with the SC approach prevailed, and, of course, since the "LLM flash", ML is considered the big club that can beat any monster, while SC is left behind as just an esoteric playground for mathematicians who like to do difficult but useless things.

However, recently it has become clear that substantial practical progress across all areas of science and technology is possible with the ML approach in a large majority of applications, but there remain the "10%" of safety-critical contexts where 100% mathematical correctness or at least some coherent argumentation, explanation, or partial verification is indispensable. This has renewed interest in the SC approach to AI. In this situation, at RISC, we decided to set several impulses intended to foster a transition to a new era of AI in which SC and ML interact in multiple ways, achieving new levels of

[1] See my recent book *Science and Meditation*, amazon.de, 2024.

[2] I presented a detailed argument for this view in my paper *Automated Programming: Symbolic Computation and Machine Learning*, Annals of Mathematics and Artificial Intelligence, 91(5), pp. 569–589, 2023.

sophistication in the automation of mathematical problem solving and, through self-reflection, in the automation of the mathematical invention process. The impulses RISC is currently organizing are:

- this conference, "SCML-2026: International Conference on Symbolic Computation and Machine Learning", which is probably the first with this title;
- a second event at RISC, overlapping both in terms of topics and time, the workshop "SCDDE 2026: Symbolic Computation and Differential and Difference Equations", the main research area of our two new full professors Carsten Schneider and Georg Regensburger;
- the "SCML Publishing Forum for Symbolic Computation and Machine Learning" for rapid dissemination of papers in this area; for example, the proceedings of this conference are published in this forum;
- the organization of a special issue "SCML: Symbolic Computation and Machine Learning" in the Journal of Symbolic Computation, in which polished versions of the current conference papers and other papers on this topic will be published;
- the organization of a new International Master's Program "Symbolic Computation and Artificial Intelligence" within the established Master's Program "Computational Mathematics" at the Johannes Kepler University and within our new "RISC European College", which provides fellowships and a vibrant community for international students who want to specialize in this and related areas;
- a major research project in the area of SC and ML, for which the RISC Institute and our own RISC Company will join forces.

With these impulses, we continue our tradition as an institute of the Johannes Kepler University (JKU) to contribute to innovation nationally and internationally, as we have done in earlier years with the creation of the Softwarepark Hagenberg, the University of Applied Science in Hagenberg, and the Journal of Symbolic Computation. We are fortunate that JKU has, in the person of Sepp Hochreiter, a world-renowned representative and founder of the ML approach to AI. We hope that our SC impulse can strengthen JKU as a stronghold in AI research, education, and application in full range.

The SCML-2026 Conference

We are very grateful and pleased that renowned researchers have agreed to work together in the program committee of this conference and also in the editorial board of the above-mentioned SCML Publishing Forum. Without their dedicated work and expertise, it would not have been possible to initiate and realize this conference in such a short time.

I also wish to thank my RISC colleagues Temur Kutsia, Wolfgang Schreiner, and Wolfgang Windsteiger for their focused and enthusiastic work on the scientific part of this conference, and our RISC staff, Tanja Gutenbrunner, Ramona Oehme-Pöchinger, Werner Danielcyk-Landerl, and Ralf Wahner, for the excellent local organization. Special thanks also go to Wolfgang Freiseisen, CEO of the RISC Software Company, who is the experienced bridge between the RISC Institute and the world of industry and business and was instrumental in shaping the format of SCML-2026 and getting the project going. It is our pride that, at RISC, we can span the complete range from foundational research to industrial applications, and so this conference also has the goal of presenting the new and exciting area of SC and ML to the industry in Austria, in particular, Upper Austria.

We are particularly indebted to the following companies and governmental entities for sponsoring this conference:

- The State of Upper Austria
- Green Events Upper Austria
- The Community of Hagenberg
- The City of Linz
- Softwarepark Hagenberg
- Federation of Austrian Industries, Upper Austria
- bluesource mobile solutions
- RISC Software GmbH
- uni software plus
- Software Competence Center Hagenberg
- NXAI
- Dynatrace
- MIC customs solutions

We also appreciate deeply the invaluable contributions of our invited speakers with their impressive expertise and insight into the essence of mathematics in this turbulent and exciting phase of mathematics driven by the interaction and interplay between mathematical human creativity, SC (CA+AR), and ML.

Our sincere thanks also go to all authors of contributed talks for sharing their novel and inspiring ideas at the conference, in the proceedings, and, hopefully, in after-conference full papers in the Publishing Forum and the JSC or other journals.

While the basic idea of bringing SC and ML together stems from the need for logical correctness for what is generated by ML, a wealth of ideas about how this interaction could and should work emerge upon deeper reflection. In the preparation of this conference and through numerous emails and personal interactions with potential contributors, many exciting ideas for such interactions were identified and pushed forward:

A. *The formal, automated logical verification of programs that are generated by ML-based "vibe coding":*

   This is a promising area that has enormous practical implications for the software industry. The requirement for formal automated verification of the results of vibe coding is clear, but there are only a few concrete examples where formal verification could be practically worthwhile and feasible. There are three talks providing such examples as a challenge:

   – Gábor Kusper: Verified Vibe Coding.
   – Steffen Fricke, Jürgen Jasperneite: Approach for the Network Configuration of Wireless Systems using RAG: Fine-Tuning and Formal Verification Component.

- Thomas Mahringer: When the Vibes Fade: An Industry Perspective on LLM Coding Limits — A Discussion Case for Symbolic Computation.

B. *The replacement of well-established human-generated and human-proved algorithms for completely pecified mathematical problems (e.g., algebraic problems) by algorithms trained by ML methods:*

   At first sight, this seems a "logically impossible" and pointless approach: Why should we replace something which is 100% provenly correct by something about whose correctness we only have some hope? Bold people are just trying this and, actually, with a good reason: For a practically important finite range of inputs, the approach may work with good confidence and, actually, dramatically better computational complexity. We have a couple of talks on this approach at our conference:

   - Hiroshi Kera. Computational Algebra with Transformers: What Deep Learning Adds to Computational Algebra (Invited Talk).
   - Yuki Ishihara, Kazuhiro Yokoyama. Efficient Dataset Generation for Bases of Zero-Dimensional Ideals.
   - Yuta Kambe. Zariski-Dense Dataset Generation for Learning to Compute Gröbner Bases.
   - Yuxuan Song, Changbo Chen. Learning to Compute Polynomial Products with Transformers.
   - Meskerem Abebaw Mebratie, Rüdiger Nather, Guido Falk von Rudorff, Werner M. Seiler. Discovering Symbolic Representation of Conservation Laws of Dynamical Systems Using Machine Learning.
   - Constant Le Bezvoët, François Fages, Julien Martinelli. Reactmine-2: a Statistical Beam Search Algorithm for Learning Biochemical Reaction Models from Time Series Data.

C. *Heuristic support in symbolic algorithms*

   Algorithms in SC often leave a wide variety of choices for individual steps open, and the actual complexity of the computations depends heavily on these choices. Every choice is correct, but making choices may greatly benefit from good heuristics, for which ML is a promising approach. Similarly, setting up symbolic models for real-world problems may benefit from ML-based heuristics:

   - Juan Esteban Suarez. Learning-Based Complexity of PDE Solutions (Invited Talk).
   - Cezary Kaliszyk. Formalization and Automated Reasoning in the Age of LLMs (Invited Talk).
   - Lixin Du. Interactive AI for Computer Algebra: A Documentation-Grounded Assistant for ore_algebra.
   - Matthew England. Machine Learning Symbolic Integration Algorithm Selection.
   - Rohit John, Rashid Barket, Matthew England. Replacing Heuristic Rule Ordering in Symbolic Integration with Learned Policies.
   - Rui-Juan Jing, Yuegang Zhao, Changbo Chen. Breaking the Data Barrier in Learning Symbolic Computation: A Case Study on Variable Ordering Suggestion for Cylindrical Algebraic Decomposition.

- François Lemaire, Louis Roussel. Deep Learning for Integro-Differential Modeling.
- Alexei Lisitsa. Towards Quantum-Reservoir Trajectory Signatures for Symbolic Rewriting Dynamics.
- Ali Soltani, Gabriel Kronberger, Fabricio Olivetti de França, Alessandro Lucantonio. Learning to Rank Symbolic Expressions for Model Selection.
- Tereso del Rió. Improving Optimization Formulations in Industrial Settings with LLMs.

D. *The automation of the mathematical invention process (including the invention of concepts, problems, algorithms, theorems, and proofs) has a new paradigm:*

A cycle is organized between

- the human mathematician,
- an ML-trained component that collects relevant informal mathematical texts from the literature,
- and an automated checker that verifies the logical correctness of informal proposals and ideas from the literature.

This approach has recently proved to be incredibly powerful. It promises to realize the dream that has been expressed in the "MKM Movement" (Mathematical Knowledge Management), starting around 2000, of which RISC was an initiator (see also the invited talk by Michael Kohlhase). There are prominent exponents of this approach at our conference, and a couple of impressive examples are given:

- Alex Best. Reinforcement Learning for Theorem Proving (Invited Talk).
- Ido Kaminer. From π to QFT: Symbolic Discovery at Scale (Invited Talk).
- Stav Belyy, Shalev Zuriel, Tomer Raz, Michael Shalyt, Ido Kaminer. Generate, Verify, Refine: A Closed-Loop LLM–CAS Approach to Symbolic Integration.
- Uri Kasher Hitin, Michael Shalyt, Shachar Weinbaum, Hila Barkan, Tali Monderer, Elyasheev Leibtag, Rotem Kalisch, Ido Kaminer. A Computational Framework for Automated Discovery within Conservative Matrix Fields.
- Michael Shalyt, Elyasheev Leibtag, Shachar Weinbaum, Ido Kaminer. Unifying Structure for Ramanujan's 17 Series for $1/\pi$: A Human-AI Discovery.
- Wolfgang Schreiner. On the Rapid Prototyping of a Logical Agent.
- Tetsuo Ida. On the Use of ChatGPT in Symbolic Computation and Mathematical Research: A Case Study in Computational Origami.
- David M. Cerna. Towards Inductive Logic Programming at Scale.
- Verena Praher, Endre Szasz-Revai, Wolfgang Windsteiger. Reasoning over Legal Texts Using Large Language Models and Automated Reasoning.
- Zoltán Kovács, Tomás Recio, Piedad Tolmos, Pilar M. Vélez. Comparing Human Perception, Computer-algebra, and Generative AI Approaches for Ranking Elementary Geometry Statements.
- Koji Nakagawa, Bruno Buchberger. Proof Engineering: Nakano's Light Puzzle Example.

- Hao Shen, Junyu Guo, Junqi Liu, Lihong Zhi. Certified CAS-assisted Polynomial Reasoning in Lean.

E. *The application of symbolic methods in the training algorithms for ML:*

The ML algorithms are typically numerical methods. (In fact, the entire ML approach can be viewed as a variant of traditional numerical methods for interpolation, approximation, fitting, etc.) However, it is near at hand that using symbolic methods at certain stages of learning could save time in some applications. Various realizations of this approach are presented in the following talks:

- Martina Seidl. Reason with SAT for Rule Learning (Invited Talk).
- Josef Urban. Alien Codes and Their Automated and Human Explanations (Invited Talk).
- Mohit Kumar, Bernhard A. Moser, Manuela Geiß. Operator-Theoretic and Complexity-Based Synthesis of a Gradient-Free Federated Kernel Learner.
- Rüdiger Nather. Representing Polynomial Ideals as Heterogeneous Graphs for Inductive Machine Learning.
- Gregoire Sergeant-Perthuis, Jules Tsukahara, Elias Tsigaridas. Exact Algebraic Computation of Learning Coefficients for Two-Dimensional Singular Models.

F. *Views on the field:*

AI, SC (CA+AR), ML, LLMs, MKM, etc.: How does all this go together, and where does this move in the foreseeable future? We have a couple of talks by people who tried to view some or all of these research streams together:

- Bruno Buchberger Vibe Coding, Vibe Proving, Vibe Mathematics (Invited Talk).
- Martin Charles Golumbic. Years of AI and Math (Invited Talk).
- Michael Kohlhase. Machine Learning in Symbolic Computation — A Skeptical Perspective (Invited Talk).
- Stephen Wolfram. Getting Math from the Computational Universe (Invited Talk).
- Erhard Glötzl. The Extended Scientific Method — From DNA to SC and ML.

G. *Bridge to the workshop "SCDDE 2026: Symbolic Computation and Differential and Difference Equations":*

SC is essentially the tight interconnection of Computer Algebra and Automated Reasoning. By intention, we are organizing SCML-2026 and SCDDE 2026 (with an emphasis on CA) in one week, with Wednesday as an overlapping day, and we will have three invited talks of SCDDE also as a part of SCML-2026:

- Johannes Blümlein. Mathematical Methods in Perturbative Quantum Field Theory and the Analytic Integration of Feynman Integrals (Invited Talk).
- Markus Lange-Hegemann. Differential Algebraic Machine Learning in Linear PDE Solution Spaces (Invited Talk).
- Peter Paule. The Unreasonable Effectiveness of Computer Algebra in the Mathematical Sciences (Invited Talk).

H. *Collocated workshop "FGB 2026: Formalization of Gröbner Bases Theory in Automated Reasoning Systems":*

- Junyu Guo, Hao Shen, Junqi Liu, Lihong Zhi. Formalizing Gröbner Basis Theory in Lean.
- Bruno Buchberger. Two Approaches to Gröbner Bases: Reduction Rings and Macaulay Matrices.
- Alexander Maletzky. Formalizations of Gröbner Bases Theory in Isabelle/HOL.

We hope we were able to put together an attractive program for all participants. We welcome all of you, decision-makers, researchers, and developers from academia and industry. We particularly welcome all students. We hope that new ideas, projects, plans, cooperations, and contacts are being generated by SCML-2026 for all of you!

Bruno Buchberger
in the name of the Program Committee and the Conference Team

Table of Contents

Invited Talks

Contributed Talks

Invited Talks

Reinforcement Learning for Theorem Proving

Alex Best
Harmonic, Palo Alto, USA

I'll discuss recent advances training language models to prove theorems at a very advanced level via reinforcement learning on formal languages. I will cover the progression of these systems from proving standalone statements, to building up whole new theories. I'll give a demo of the system we have been developing with these techniques, called Aristotle, and survey interesting use cases so far. Additionally I will discuss the potential ways that a mathematicians work will change when incorporating these tools.

Mathematical Methods in Perturbative Quantum Field Theory and the Analytic Integration of Feynman Integrals

Johannes Blümlein
Deutsches Elektronen-Synchrotron (DESY), Zeuthen, Germany

We present recent advances in the analytic calculation of single scale Feynman integrals in renormalizable quantum field theories. We consider complete physical processes at large-scale high energy colliders, such as the anomalous dimensions, massless and massive Wilson coefficients and the massive form factors to three-loop order in QCD. The fundamental methods to be used are difference and differential equations, which have first to be established by guessing methods. The corresponding equations are huge. In the case of first order factorizing equations iterative-integral solutions are obtained. We also present computer-algebraic methods to solve non first order factorizing differential equations.

Vibe Coding, Vibe Proving, Vibe Mathematics. . . ?

Bruno Buchberger
Johannes Kepler University Linz, Austria

I discuss two recent research areas combining symbolic computation and machine learning:

- learning algorithms for mathematical problems from training data
- proving by generating proof ideas through large language models and checking the ideas with a proof checker.

The first area is concerned with replacing provenly correct algorithms for algebraic problems (as simple as integer addition or as complex as Gröbner bases construction or Cylindrical Algebraic Decomposition) with algorithms (for example, Neural Networks) that have been trained on large sets of input-output examples for the respective problem. Why should one do that – knowing that probably the trained algorithm may be incorrect for certain inputs? Well, this is practically interesting if the computation times of the trained algorithm are drastically better. This has been proved to be the case by recent work by H. Kera and others.

In the talk, I will discuss whether it is sufficient just to compare the computation times of a provenly correct algebraic algorithm and an algorithm trained by machine learning from examples. I argue that one also must take into account the computation time for checking the results generated by the trained algorithm. In the extreme case, this may be equally costly as the generation of the output by an algebraic algorithm. I will analyze this in particular in the case of the Gröbner bases problem, where we do not only have to prove that the result is a Gröbner bases but also that the ideal generated does not change. Both questions may need considerable effort. The second question may need the application of what is called "an extended Gröbner bases algorithm".

The second area is concerned with building up large mathematical knowledge bases (definitions, theorems, algorithms) together with formally checked proofs. This area existed for a long time under the name "Mathematical Knowledge Management" (MKM). The practical impact of work in this area on the daily work of mathematicians or for generating mathematical archives , however, was relatively low. The reason was that proof-checking, as the only algorithmic tool in MKM, still left enormous work for the mathematicians who added knowledge to such mathematical knowledge bases. With the THEOREMA system (around 2000) I tried to improve the situation by coming up proof-generating algorithms. However, still much of the work – finding essential proof ideas – was left to the mathematicians. With the new approach (for which the Aristotle system is a prominent example) much of the creative work of finding proof ideas (and interesting definitions, lemmata, theorems, algorithms) is automated by just "looking to the literature for relevant text expressed in the usual semi-formal mathematical language" using kind of a specifically trained LLM and submit the results found to a proof checker. (In the case of Aristotle, this is LEAN). This step can be viewed as a kind of "vibe proving" (generalizing the recent "vibe coding" approach for programming) or even "vibe mathematics".

In the talk, I will discuss whether, in this approach, it would make sense to replace a proof checker by a proof generator like THEOREMA.

36 Years of AI and Math

Martin Charles Golumbic
University of Haifa, Israel

Research considered to be "artificial intelligence" constantly redefines itself. As the disciplines within AI have evolved and changed over many decades, mathematical methods have provided the solid foundation of these diverse areas. The range of topics appearing in the literature has been impressive.

To help bridge the gap between new applications of technology and foundational mathematical research, we explore research in a variety of disciplines, with a particular emphasis on the foundations of AI and mathematical methods. Artificial intelligence (AI) has contributed to mathematical discovery, by guiding conjecture generation, assisting in formalizing mathematics, and exploiting its power to identify patterns in data and refine relationships between properties.

There has always been a strong relationship between mathematics and artificial intelligence, including applications of each discipline in the other. Computational geometry is central to much of motion planning in robotics. Combinatorial optimization is a primary tool in search, and a main ingredient of machine problem solving. Mathematical logic is arguably the appropriate language for AI, and the theoretical underpinnings of intelligent systems. Automated reasoning systems have provided a further motivation for work on non-standard, non-monotonic logics and belief deduction. Mathematical probability is the foundation of dealing with uncertainty. The mathematical theory of games is the basis for AI game-playing and models of strategy and fairness.

The state of the art and current challenges provide a wealth of opportunities to advance research in the interactions between AI and mathematics through interdisciplinary collaboration. We present a selection of topics from recent symposia as an indication of this, along with some early history of interaction between Math and AI and its development over the years.

Formalization and Automated Reasoning in the Age of LLMs

Cezary Kaliszyk
University of Melbourne, Australia

As LLMs become part of the formalization pipeline, the key question is no longer only how to automate proof, but how to organize collaboration between human guidance, automated reasoning, and machine learning. I will discuss what forms of automation are most useful to humans and to LLMs, the kinds of proofs they tend to generate and learn from, and the role of proof style in effective formal development. I will also cover readability: when are human-created formal proofs hard to understand, and how this changes for generative-model augmented formalization.

From π to QFT: Symbolic Discovery at Scale

Ido Kaminer
Technion — Israel Institute of Technology, Israel

For centuries, formulas for mathematical constants such as π and e appeared sporadically, discovered by figures like Newton, Euler, Gauss, and Ramanujan. Inspired by experimental physics, we established the Ramanujan Machine collaboration, a program of experimental mathematics that discovers formulas at scale, including new identities for π and for Riemann ζ values. This effort runs on a global volunteer network contributing CPU time, enabling the discovery of thousands of formulas—now surpassing the total found by humans over centuries.

In this talk, I will show how combining these symbolic mathematical capabilities with large language models yielded the first example of automated unification of mathematical knowledge, revealing deeper organizing structure among constants. I will then outline potential applications to areas outside of number theory: efficient computation of hypergeometric functions and broader families of D-finite functions, with implications for computing Feynman integrals.

As another application of massive symbolic computation, I will discuss our recent contribution to modern toolkits for automated effective field theory. As a proof of concept, I will present our massive automated scan for quantum completions of gravity, showing that one-loop contributions can reproduce the curvature of gravity from flat-space quantum field theory.

I will conclude with a broader perspective on AI4Physics vs. Physics4AI, with recent examples from my lab.

1. G. Raayoni, et al., The Ramanujan Machine: Automatically Generated Conjectures on Fundamental Constants, Nature 590, 67–73 (2021)

2. O. Razon, et al., Automated Search for Conjectures on Mathematical Constants using Analysis of Integer Sequences, ICML 202, 28809-28842 (2023)

3. R. Elimelech, et al., Algorithm-assisted discovery of an intrinsic order among mathematical constants, PNAS 121, e2321440121 (2024)

4. M. Shalyt†, U.Seligmann†, et al., Unsupervised Discovery of Formulas for Mathematical Constants, NeurIPS 37, 113156 (2024)

5. Beit Halachmi and I. Kaminer, The Ramanujan Library - Automated Discovery on the Hypergraph of Integer Relations, ICLR (2025); arXiv:2412.12361

6. T. Raz, et al., From Euler to AI: Unifying Formulas for Mathematical Constants, NeurIPS (2025); arXiv:2502.17533

Computational Algebra with Transformers: What Deep Learning Adds to Computational Algebra

Hiroshi Kera
Chiba University, Japan

Computer algebra has advanced with computing. Computing enables experimenting with theories, algorithms, and heuristics of computer algebra, supporting their verification and the study of their limitations. This survey overviews recent applications of deep learning, particularly Transformer models, to arithmetic and symbolic computation, and presents a new axis that deep learning can add to computer algebra. While not always offering interpretable or guaranteed computation, the learning-based approach can extract patterns of computation through observation of instances at enormous scale; exploiting this may lead to faster algorithms and the discovery of new heuristics. In addition, generating training sets for deep learning models gives rise to new symbolic computation tasks. I will also briefly introduce the CALT library as a convenient entry point for interested audiences.

Machine Learning in Symbolic Computation — A Skeptical Perspective

Michael Kohlhase
Friedrich-Alexander Universität Erlangen-Nürnberg, Germany

Driven by an enormous investments and industrial hype machine ML methods and particularly LLMs are making great progress in public/industrial/scientific awareness and practices. Every discipline asks (and has to ask) what can ML/LLMs do for me and how will science change with the new methods?

Generally, we have to come to terms with how to best combine/reconcile methods from subsymbolic AI (ML and LLMs) with symbolic methods in our respective areas. The former are data-driven and excel at global coverage, but have deficits in prediction and explainability whereas the latter are human-driven, precise, and explainable, but do not scale well.

In this talk I will try to survey the shape of symbolic/subsymbolic AI (keeping a distance to the "AI hype" to separate some of the myth from reality) so that we can assess combination opportunities and strategies. Finally I want to conjecture some of the effects of the "subsymbolic AI boom" for traditionally symbolic disciplines like formalized mathematics, symbolic computation, and tertiary STEM education.

Differential Algebraic Machine Learning in Linear PDE Solution Spaces

Markus Lange-Hegermann
Technische Hochschule Ostwestfalen-Lippe, Germany

We consider linear constant-coefficient PDE systems. We present a computational differential-algebraic approach to probabilistic and learning-based models that work directly within their solution spaces. The starting point is the Ehrenpreis–Palamodov fundamental principle, which describes solutions through algebraically computable characteristic varieties and exponential-polynomial Fourier-type modes. We use this representation to construct two learning approaches within the solution spaces of such PDEs. First, it yields Ehrenpreis–Palamodov Gaussian processes: Bayesian priors concentrated on admissible PDE solutions, supporting conditioning on sparse, noisy, initial, or boundary data. Second, it yields exact-by-construction trainable shallow neural networks whose hidden units are themselves solutions of the governing equations, such as Maxwell's equations or the wave equation. The main message is that symbolic structure can make scientific machine learning both more faithful and more efficient: rather than penalizing violations of PDEs after the fact, we learn directly in a computationally tractable space of exact solutions. We illustrate these ideas in applications to PDE solving, system control, and sampling from solution spaces.

The Unreasonable Effectiveness of Computer Algebra in the Mathematical Sciences

Peter Paule
Johannes Kepler University Linz, Austria
Tianjin University, China

Despite the current renaissance of AI, the main theme of the talk is on more traditional lines: namely, to stress the huge potential of algorithmic mathematics, and of respective computer algebra software, for applications in pure mathematics and related fields.

For example, the Ramanujan Machine (Nature 590, 2021) creates mathematical conjectures using AI and computer automation. On the other hand, Cristian-Silviu Radu (RISC) developed a computer algebra algorithm which can be used to discover (and prove!) identities even Ramanujan would have appreciated to see.

In the talk we present a variety of such examples from different areas: optimization of antenna radiation, special functions and Gauss' contiguous relations, linear Diophantine inequalitites and partitions of numbers, symbolic summation in quantum field theory, a.s.o.

Reason with SAT For Rule Learning

Martina Seidl
Johannes Kepler University Linz, Austria

Propositional logic is one of the most successful formalisms in symbolic artificial intelligence. Many hard combinatorial problems can be naturally formulated as decision problems of propositional logic (SAT) and efficiently solved using modern SAT solvers. When using SAT, problems are described symbolically in a compact manner, in contrast to machine learning approaches, which typically rely on large volumes of data to characterize a problem.

In recent work, we have investigated how SAT techniques can be applied to learning tasks that involve examples. In this talk, we report on our experiences, examine the gap between symbolic and subsymbolic approaches, and outline potential directions for future research.

Learning-Based Complexity of PDE Solutions

Juan Esteban Suarez
Ludwig-Maximilians-Universität München, Germany

Partial Differential Equations (PDEs) are fundamental tools for modeling phenomena across physics, biology, and engineering, yet their numerical approximation becomes challenging in the presence of singularities, discontinuities, or complex geometries. While classical numerical methods—such as finite element, finite volume, and spectral schemes—offer strong theoretical guarantees, they typically rely on high regularity assumptions and scale poorly in non-smooth or high-dimensional settings. Recent advances in scientific machine learning have introduced new approximation paradigms based on variational formulations, neural representations, and hybrid surrogate models, raising fundamental questions about the computability and algorithmic complexity of PDE solutions.

In this talk, we explore how modern learning-based models can be used as computational structures for studying the computability and complexity of broad classes of PDE solutions, including those with singularities and discontinuities. We present a variational framework for analyzing the algorithmic complexity of a class of continuous PDE solutions using polynomial approximation schemes, and discuss extensions based on hybrid surrogate models that enable the analysis of discontinuous solutions. This framework provides constructive insights into the design of novel computational architectures for solving PDEs, addressing both the limitations of classical solvers and the energy-efficiency challenges of state-of-the-art AI models.

Alien Codes and Their Automated and Human Explanations

Josef Urban
Czech Technical University in Prague, Czech Republic

We have automatically discovered symbolic explanations for over one-third of the sequences in the Online Encyclopedia of Integer Sequences (OEIS). The talk will describe the neuro-symbolic system consisting of a positive feedback loop that starts from zero knowledge and iterates between guessing the explanations, their verification, and training of the guessing methods. Then I will describe several additions and experiments that led to the current set of solutions found in hundreds of iterations of the feedback loop. I will show some of the solutions discovered. This includes over 50 programs for primes developed as the system self-evolves. I will also discuss a related experiment in automatically proving equivalences of the discovered programs using the SMT solver Z3. Because induction is often needed in such proofs, this leads to another self-learning neuro-symbolic system that repeatedly tries to guess the right instances of induction for Z3. Finally, I will also show some of the recent human explanations of the programs, mostly done by Tom Hales.

Getting Math from the Computational Universe

Stephen Wolfram
Wolfram Research, Champaign, USA

How does computation relate to the mathematics that humans have done, and are interested in? I'll talk about the science of the computational universe, and what it tells us about the foundations and future of mathematics, and what can and cannot be expected for mathematics from AI. I also expect to show some new technology under development, as well as some new science that addresses the foundations of machine learning and their relation to mathematics.

Contributed Talks

Generate, Verify, Refine: A Closed-Loop LLM–CAS Approach to Symbolic Integration

Stav Belyy, Shalev Zuriel, Tomer Raz, Michael Shalyt, and Ido Kaminer

Technion – Israel Institute of Technology, Haifa 3200003, Israel

Integrals are a key tool in science and engineering, used to model physical systems, solve equations, and analyze data. Over the years, many Computer Algebra Systems (CAS) have been developed to tackle this task and provide symbolic solutions, but often struggle with more complex or unconventional integrals. More recently, Large Language Models (LLMs) have demonstrated an impressive ability to perform symbolic reasoning tasks [2], but they are still unreliable, often hallucinating or inventing assumptions. We propose a hybrid reasoning framework that integrates LLMs with CAS to enable verifiable symbolic integration.

At the core of this framework is a closed-loop architecture that orchestrates tasks between a generative model and a deterministic solver. The LLM serves as both orchestrator and solver, contributing creativity by proposing candidate solutions and transformations, coordinates symbolic operations between generative and deterministic components. The CAS provides grounding through deterministic verification. The hybrid solver operates as an **iterative reasoning process** composed of three key stages:

- **Generate:** proposes candidate solutions or symbolic transformations, such as substitutions, decompositions, or heuristic rewrites.
- **Verify:** checks the correctness of each candidate using CAS-based validation.
- **Refine:** uses feedback or symbolic transformations to recover from incorrect solutions.

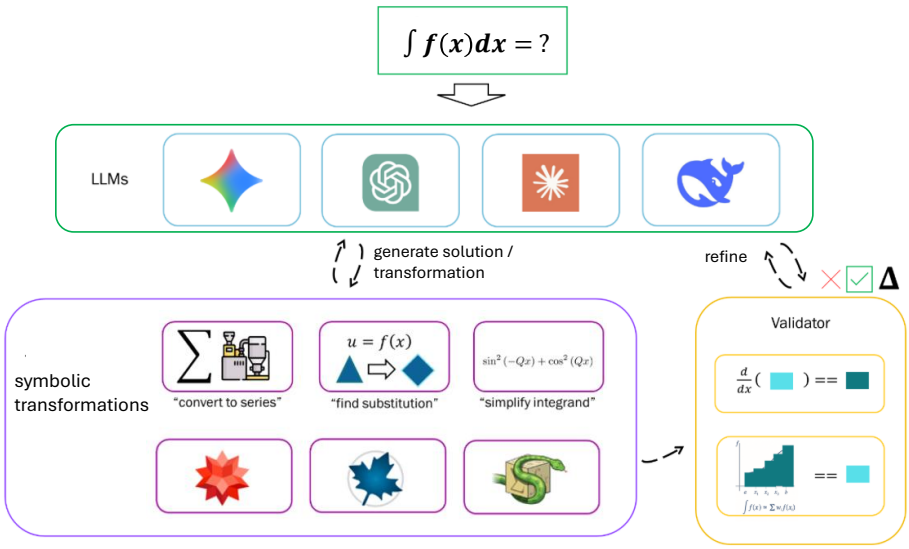


Figure 1: **Hybrid reasoning loop with symbolic validation.** The system iteratively generates, verifies, and refines solutions using CAS validation and feedback.

We propose a hybrid LLM–CAS framework that recovers from common LLM failures in symbolic integration through a layered recovery process: iterative feedback for correcting logical errors, followed by symbolic transformations for re-representing complex integrands.

To assess the effectiveness of this approach, we conducted experiments across multiple LLMs, using 400 challenging integrals from various benchmarks, such as IntegralBench [4] and U-math [1]. The first step is **Iterative Refinement (IR).** Each integral was solved by an LLM and verified using a CAS. We attempt symbolic verification via $\frac{d}{dx}F(x) - f(x) = 0$, but this can be inconclusive for complex or domain-sensitive expressions; in such cases, we use high-precision numerical checks over valid regions. If validation fails, the system initiates a feedback loop, allowing the LLM to refine its answer. Solutions corrected within these retries demonstrate impact of **Iterative Refinement** on model performance.

Transformations. If refinement fails, the orchestrator proposes a symbolic transformation (e.g., variable substitution, algebraic, series expansion and integration by parts), verifies its correctness via the CAS, and re-attempts solving the transformed problem using the same validation procedure. Successes at this stage highlight the role of **symbolic transformations** in overcoming representation bottlenecks.

Example. Consider $I = \int_0^2 \frac{dx}{\sqrt{x}(1+\sqrt{x(2-x)})^{3/2}}$. A direct attempt by the LLM produces an incorrect value, which fails CAS verification. After the feedback loop fails to correct the error, the system applies a symbolic transformation, introducing the substitution $x = 2\sin^2\theta$. After verifying the transformation, the LLM is tries again on the transformed problem and returns the correct result $I = \sqrt{2}$.

Model	Initial	After IR	Final	Improvement
Gemini 3	2%	$< 1\%$	$< 1\%$	$\sim$50%
GPT-5	10%	8%	8%	20%
DeepSeek V3	40%	30%	30%	25%
Gemini 2.5	60%	50%	46%	23%
GPT-4.1	50%	44%	39%	22%

Table 1: **Recovery from LLM errors via feedback and symbolic transformations.** Failure rates across models before and after Iterative Refinement (IR) and symbolic transformations. Values are approximate ($\pm 1\%$).

These results indicate that many failures are not fundamental, but arise from step-wise errors or incorrect transformations that can be corrected by proper use of established symbolic methods. Feedback provides the primary gains for stronger models, suggesting that they can reach correct solutions but often make mistakes during the solving processü [3]. Meanwhile, symbolic transformations contribute additional recovery for weaker models, as they simplify the problem and help overcome their limited reasoning ability.

As LLMs become standard tools in science, we need a way to move past 'blind trust' and toward formal grounding. Our results show that many failure types could be identified and sometimes fixed using established CAS tools that act as guardrails, making hybrid systems a promising path toward reliable symbolic reasoning.

References

[1] Konstantin Chernyshev et al. U-math: A university-level benchmark for evaluating mathematical skills in llms. *arXiv preprint arXiv:2412.03205*, 2024.

[2] Luyu Gao et al. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2022.

[3] Michael Shalyt, Rotem Elimelech, and Ido Kaminer. ASyMOB: Algebraic symbolic mathematical operations benchmark. *arXiv preprint arXiv:2505.23851*, 2025.

[4] Bintao Tang et al. Integralbench: Benchmarking llms with definite integral problems. *arXiv preprint arXiv:2507.21130*, 2025.

2

Towards Inductive Logic Programming at scale

David M. Cerna[1,2*]

[1] Ndea,
dcerna@ndea.com
[2] Czech Academy of Sciences, Institute of Computer Science
dcerna@cas.cs.cz

Inductive Logic programming (ILP) combines knowledge representation and machine learning [9, 4]. The goal is to search a hypothesis space, consisting of logic programs, to find a hypothesis that generalizes the given training examples and background knowledge. Early ILP approaches such as *foil* iteratively specialize rules till only positive examples are accepted and repeat this process until all positive examples are accepted. This process was refined in systems like *Aleph* by restricting the search space of specializations to hypotheses that θ-subsume the so-called *bottom clause*, i.e, the most specific clause which accepts a given positive example.

Modern ILP often follows the *meta-learning* approach, i.e., perform search through an encoding of the hypothesis space. In particular, *Popper* [6] encodes the generation of logic programs modulo the background knowledge as an ASP program (*generate phase*). As *Popper* tests the hypothesis produced by the generate phase, additional constraints are added guiding the search at subsequent steps. This leads to a significant decrease in the number of tested hypotheses while maintaining optimality guarantees. These constraints are based on propositional subsumption and are thus computationally feasible to check. Popper's approach to ILP has led to significant advancements in the tasks such systems can handle. For example, learning with *negative invention* [1], *Higher-order* [10], *very large programs* [7], and competative performance on ARC [8].

Nonetheless, searching through the hypothesis space entailing from a substantial background knowledge remains difficult for complex task. Thus, in such intractable cases, learning will requires additional mechanisms and constraints beyond what current systems implement. In this abstract, we discuss recent investigations into pruning mechanisms based on **symmetry breaking** [3] and **redundancy elimination** [2, 5]

References

[1] David M. Cerna and Andrew Cropper. Generalisation through negation and predicate invention. *AAAI*, 38(9):10467–10475, 2024.

[2] Andrew Cropper and David M. Cerna. Efficient rule induction by ignoring pointless rules. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(23):19003–19011, Mar. 2026.

[3] Andrew Cropper, David M. Cerna, and Matti Järvisalo. Symmetry breaking for inductive logic programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(23):19012–19020, Mar. 2026.

[4] Andrew Cropper and Sebastijan Dumancic. Inductive logic programming at 30: A new introduction. *J. Artif. Intell. Res.*, 74:765–850, 2022.

*This extended abstract concerns joint work with Andrew Cropper.

[5] Andrew Cropper, Filipe Gouveia, and David M. Cerna. Honey, i shrunk the hypothesis space (through logical preprocessing), 2026. To appear JAIR.

[6] Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Mach. Learn.*, 110(4):801–856, 2021.

[7] Céline Hocquette, Andreas Niskanen, Rolf Morel, Matti Järvisalo, and Andrew Cropper. Learning big logical rules by joining small rules. In *IJCAI*, pages 3430–3438, 2024.

[8] Céline Hocquette and Andrew Cropper. Relational decomposition for program synthesis, 2025.

[9] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[10] Stanislaw J. Purgal, David M. Cerna, and Cezary Kaliszyk. Learning higher-order logic programs from failures. In Luc De Raedt, editor, *IJCAI*, pages 2726–2733, 2022.

2

Improving Optimisation Formulations in Industrial Settings with LLMs

Tereso del Río[1]

Research Institute for Symbolic Computation (RISC)
and Institute for Symbolic AI (SAI)
JKU Linz, Austria

Building correct, reusable, and maintainable optimisation models remains a challenge in industrial settings. Industrial users often understand the operational process very well, but may not be expert modellers. In this work, we present a language aimed at industrial users and study whether LLMs can be used to generate formulations of industrial optimisation problems in this language. These specifications can then be translated into constraint models that can be processed by symbolic constraint solvers.

A Pythonic language for industrial optimisation.

Existing modelling languages such as MiniZinc, Pyomo, PuLP, and CPMpy provide powerful ways to formulate optimisation problems and connect them to solvers [8, 2, 7, 3, 4]. Some of these languages are embedded in Python. In contrast, the domain-specific language we present, OptDSL, uses Python syntax itself and is designed to stay close to the way industrial users describe, test, and debug their processes.

The main idea of OptDSL is to describe an optimisation problem in a checker-style form: the user writes a small typed Python-like program which, given a possible solution, checks whether the solution is valid and computes the objective value. This language allows users to use familiar programming constructs such as functions, loops, conditionals and assignments. OptDSL can be translated automatically into MiniZinc, enabling the use of multiple solvers.

Making LLM-generated formulations useful for industry.

The second part of the presentation studies how Large Language Models can create formulation of optimisation problems in OptDSL. Recent work has shown that LLMs can generate optimisation formulations in existing modelling languages [6, 9, 1]. In this work, we focus on requirements that are essential in industrial settings: reusability and checkability.

Instead of asking the LLM to generate a complete formulation in one step, we use Tree-of-Thoughts prompting [10] to generate the formulation in multiple steps: object types, constants, decision variables, objective function, and constraints. This allows the LLM to explore multiple candidate formulations, essential considering that a huge level of expertise is required to know which formulation is best for a given solver/family of instances.

As a case study, we consider the two-dimensional bin-packing problem, on 20 benchmark instances from 2DPackLib [5], around 85% of the generated formulations were valid, and some were up to 14% faster than a handcrafted OptDSL baseline.

Finally, we compare two formulation-generation strategies. The first gives the LLM more freedom, which supports exploration but makes reuse and validation harder. The second fixes the structures of constants and decision variables, so that the LLM only generates the objective function and constraints. This makes the resulting formulations reusable across instances and easier to check automatically.

References

[1] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: optimization modeling using MIP solvers and large language models, 2024. OpenReview.

[2] Michael L. Bynum, Gabriel A. Hackebeil, William E. Hart, Carl D. Laird, Bethany L. Nicholson, John D. Siirola, Jean-Paul Watson, and David L. Woodruff. *Pyomo: optimization modeling in Python*. Springer, 3 edition, 2021.

[3] Tias Guns. Increasing modeling language convenience with a universal n-dimensional array, CPpy as python-embedded example, 2019. Workshop on Constraint Modelling and Reformulation (ModRef) at CP 2019.

[4] Tias Guns. Things we underestimated while developing the CPMpy constraint modelling library, 2023. Workshop on Constraint Modelling and Reformulation (ModRef) at CP 2023.

[5] M. Iori, V. L. de Lima, S. Martello, and M. Monaci. 2DPackLib: a two-dimensional cutting and packing library. *Optimization Letters*, 16(2):471–480, 2022.

[6] Kostis Michailidis, Dimos Tsouros, and Tias Guns. Constraint modelling with LLMs using in-context learning. In Paul Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming (CP 2024)*, volume 307 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:27, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[7] Stuart Mitchell, Michael O'Sullivan, and Iain Dunning. PuLP: a linear programming toolkit for python. Technical report, Optimization Online, 2011.

[8] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. MiniZinc: towards a standard CP modelling language. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming – CP 2007*, volume 4741 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2007.

[9] Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. NL4Opt competition: formulating optimization problems based on their natural language descriptions. In *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 189–203. PMLR, 2022.

[10] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: deliberate problem solving with large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 11809–11822. Curran Associates, Inc., 2023.

2

Interactive AI for Computer Algebra: A Documentation-Grounded Assistant for ore_algebra

Lixin Du

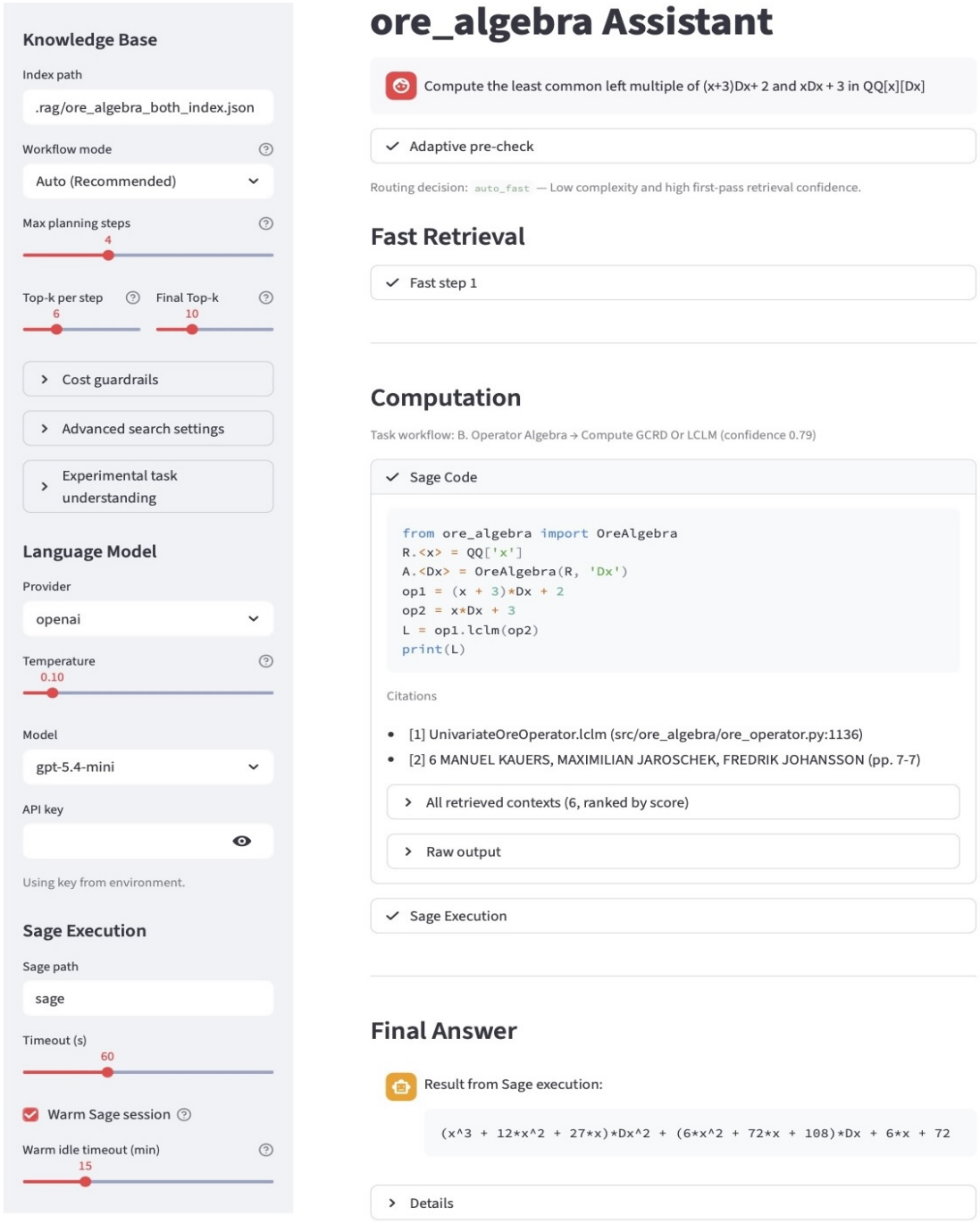
Institute for Algebra, Johannes Kepler University, Linz, Austria
lixindumath@gmail.com

The `ore_algebra package` [1, 2] in SageMath provides tools for computations with linear differential and recurrence operators in the framework of Ore algebras. We present a natural language assistant for the `ore_algebra`. Given a user query, the system retrieves relevant documentation, generates executable SageMath code with a Large Language Model (LLM), executes it, and returns the result together with source-level citations. The assistant combines a local documentation index with an agentic Retrieval Augmented Generation (RAG) pipeline and offers three execution modes (Auto, Fast, and Plan) to balance latency and multi-step reasoning. The current prototype supports a broad set of common `ore_algebra` tasks, including GCRD and LCLM, closure properties of D-finite functions, guessing from data, and Gröbner basis computations. Exposed through an interactive web interface, the system is intended to reduce the need for manual lookup of function usage in specialized symbolic computations. Beyond its functionality, the project also serves as a case study in LLM-assisted software engineering for computer algebra. The software is available at: https://github.com/LixinDu/ore-algebra-agentic-rag-assistant. An illustrative example of the system interface is provided on the next page.

References

[1] M. Kauers, M. Jaroschek, and F. Johansson. Ore polynomials in Sage. In *Computer Algebra and Polynomials*, pages 105–125. Springer, 2015.

[2] M. Kauers and M. Mezzarobba. Multivariate Ore polynomials in SageMath. *ACM Communications in Computer Algebra*, 53(2):57–60, 2019.


Knowledge Base
Index path
.rag/ore_algebra_both_index.json
Workflow mode
Auto (Recommended)
Max planning steps
4
Top-k per step
6
Final Top-k
10
Cost guardrails
Advanced search settings
Experimental task understanding
Language Model
Provider
openai
Temperature
0.10
Model
gpt-5.4-mini
API key
Using key from environment.
Sage Execution
Sage path
sage
Timeout (s)
60
Warm Sage session
Warm idle timeout (min)
15
ore_algebra Assistant
Compute the least common left multiple of (x+3)Dx+ 2 and xDx + 3 in QQ[x][Dx]
Adaptive pre-check
Routing decision: auto_fast — Low complexity and high first-pass retrieval confidence.
Fast Retrieval
Fast step 1
Computation
Task workflow: B. Operator Algebra → Compute GCRD Or LCLM (confidence 0.79)
Sage Code
from ore_algebra import OreAlgebra
R.<x> = QQ['x']
A.<Dx> = OreAlgebra(R, 'Dx')
op1 = (x + 3)*Dx + 2
op2 = x*Dx + 3
L = op1.lclm(op2)
print(L)
Citations
[1] UnivariateOreOperator.lclm (src/ore_algebra/ore_operator.py:1136)
[2] 6 MANUEL KAUERS, MAXIMILIAN JAROSCHEK, FREDRIK JOHANSSON (pp. 7-7)
All retrieved contexts (6, ranked by score)
Raw output
Sage Execution
Final Answer
Result from Sage execution:
(x^3 + 12*x^2 + 27*x)*Dx^2 + (6*x^2 + 72*x + 108)*Dx + 6*x + 72
Details

2

Machine Learning Symbolic Integration Algorithm Selection

Matthew England

Coventry University, Coventry, United Kingdom
Matthew.England@coventry.ac.uk

This talk will present joint work with Rashid Barket, Jüergen Gerhard, and Uzma Shafiq, on the topic of Machine Learning and Symbolic Integration. The work is either already published, or submitted for review, as referenced below.

Introduction

We will summarise a recently concluded PhD project at Coventry University that was sponsored by the software company Maplesoft. The overall task was to provide guidance to the user-level `int` command for symbolic integration in the Maple computer algebra system. That command is actually a meta-algorithm which can call from a range of sub-algorithms. At the start of the project Maple would try these sub-algorithms in turn until one succeeded (in some cases checking guard code that could quickly conclude if an algorithm would fail). The hypothesis of the project was that Machine Learning (ML) could better guide the choice of sub-algorithm for a given instance. Following the priorities of Maplesoft, we optimised for the size of the output rather than the time taken to produce it. We summarise some of the key lessons learned.

The importance & challenge of datasets for ML training in symbolic computation

We started using the data generation methods of [7], but quickly found these to contain hidden biases and shortcomings: much effort was spent reverse engineering established symbolic computation methods to form new data generators, as summarised in [2, 3].

The importance of independent datasets for validating the quality of ML models

It is well-established ML practice to have different datasets for testing and training; but in the case where data is being generated synthetically this is not sufficient – to build trust in the outputs we must validate them on data other than from those generators. We used an independent test set from Maplesoft to uncover in [1] and then fix in [4] shortcomings in our models that would otherwise have not been detected.

Appropriate embeddings for mathematical expressions

Our initial work in [5] used sequential transformer models to process mathematical expressions, similar to [7]. While these showed strong performance on the synthetic data their generalisation to independent data was weaker than hoped for. We discovered that using tree embeddings offered superior results and generalisation for both LSTM [1] and transformer models [4].

Experimenting to find the optimal architecture

We experiments in framing the problem using a range of paradigms: e.g. classification on whether a method is optimal; regression to predict output size. Such traditional paradigms

were challenged by the imputation problem: how to treat data instances were a particular sub-algorithm failed (which is a common occurence in the dataset).

We discovered that the optimal architecture was a two stage approach: we first trained classifiers on the simpler problem of predicting whether a given method could tackle a problem (effectively replacing the guard code for the methods which had it in Maple, and providing guard code for those which did not [5]); and then trained ranking models to order the methods on output size, but enforcing the ranking of methods predicted admissible above those not [4].

Summary

Taken together, these lessons led us to propose in [4] a two-stage architecture tree-transformer solution as the state-of-the-art for the problem. We hypothesise that many of these lessons above may be useful for similar problems of ML optimisation in symbolic computation.

We finish by noting recent work on an adjacent but different problem: the formation of a sequence of integration rules to evaluate an integral similar to how this would be tackled by hand. The problem was previously approached with supervised learning in [8]. In contrast to the work surveyed here, this problem involves iterative decisions on rules and so we proposes the first reinforcement learning approach in [6]. It remains to be seen whether the results in [6] can be improved by utilising some of the lessons surveyed here.

References

[1] Barket, R., England, M., Gerhard, J.: Symbolic integration algorithm selection with machine learning: LSTMs vs tree LSTMs. In: Buzzard, K., Dickenstein, A., Eick, B., Leykin, A., Ren, Y. (eds.) Mathematical Software – ICMS 2024. Lecture Notes in Computer Science, vol. 14749, pp. 167–175. Springer Nature Switzerland (2024), https://doi.org/10.1007/978-3-031-64529-7_18

[2] Barket, R., England, M., Gerhard, J.: Generating elementary integrable expressions. In: Boulier, F., England, M., Kotsireas, I., Sadykov, T.M., Vorozhtsov, E.V. (eds.) Computer Algebra in Scientific Computing, Lecture Notes in Computer Science, vol. 14139, pp. 21–38. Springer Nature Switzerland (2023), https://doi.org/10.1007/978-3-031-41724-5_2

[3] Barket, R., England, M., Gerhard, J.: The Liouville generator for producing integrable expressions. In: Boulier, F., Mou, C., Sadykov, T.M., Vorozhtsov, E.V. (eds.) Computer Algebra in Scientific Computing (Proc. CASC 2024). Lecture Notes in Computer Science, vol. 14938, pp. 47–62. Springer Nature Switzerland (2024), https://doi.org/10.1007/978-3-031-69070-9_4

[4] Barket, R., England, M., Gerhard, J.: Tree-based deep learning for ranking symbolic integration algorithms. Submitted, Preprint: arXiv:2508.06383 (2025), https://doi.org/10.48550/arXiv.2508.06383

[5] Barket, R., Shafiq, U., England, M., Gerhard, J.: Transformers to predict the applicability of symbolic integration routines. In: The 4th Workshop on Mathematical Reasoning and AI (MATH-AI) at NeurIPS'24 (2024), https://openreview.net/forum?id=b2Ni828As7

[6] John, R., Barket, R., England, M.: Replacing heuristic rule ordering in symbolic integration with learned policies. Submitted (2026)

[7] Lample, G., Charton, D.: Deep learning for symbolic mathematics. In: Mohamed, S., White, M., Cho, K., Song, D. (eds.) Eighth International Conference on Learning Representations (ICLR 2020) (2020), https://iclr.cc/virtual_2020/poster_S1eZYeHFDS.html

[8] Sharma, V., Nagpal, A., Balin, M.: SIRD: Symbolic integration rules dataset. In: Proceedings of the 3rd Workshop on Mathematical Reasoning and AI (MATH-AI 2023) at NIPS 2023 (2023), https://mathai2023.github.io/papers/39.pdf

Approach for the network configuration of wireless systems using RAG, Fine-Tuning and formal verification component

Steffen Fricke[1] and Jürgen Jasperneite[1,2]

[1] Technische Hochschule Ostwestfalen-Lippe, Lemgo, Germany (steffen.fricke, juergen.jasperneite)@th-owl.de
[2] Fraunhofer IOSB-INA, Lemgo, Germany juergen.jasperneite@iosb-ina.fraunhofer.de

Abstract

Managing 5G/Time-Sensitive Networking (TSN) systems requires effective use of contextual knowledge including strict operational constraints and mathematical models. Large Language Models (LLMs) can support this by leveraging Retrieval Augmented Generation (RAG) and fine-tuning to incorporate domain-specific information. This work reviews related approaches and proposes a method for implementing a system that integrates these techniques to efficiently process heterogeneous data sources and incorporates domain-specific verifiers to enable formal reasoning, thereby ensuring consistency, interpretability, and reliability in wireless network management.

1 Introduction

Managing wireless systems using Artificial Intelligence (AI) requires the efficient use of contextual knowledge from heterogeneous data sources. LLMs provide promising capabilities in this domain, particularly when enhanced with domain-specific information. Two key approaches to achieve this are RAG and fine-tuning. Recent research has demonstrated the potential of these methods for domain-specific question answering. However, applying them to technical systems with structured and unstructured data remains challenging. This work reviews relevant approaches and proposes an approach for implementing a system that combines RAG, fine-tuning and verification components to support the management and analysis of wireless systems.

2 Related Work

Several publications address methods for adapting LLMs to domain-specific tasks. Two main approaches are commonly used: RAG and Low-Rank-Adaptation (LoRA) fine-tuning. The study [4] investigates the potential of LLMs for question-answering tasks using RAG. In this implementation the vector-database ChromaDB combined with keyword search and LLM-generated responses to query the most relevant vector embeddings as contextual information is used. A LoRA-based fine-tuning is applied on the small Phi-2 model, evaluated on telecommunication-related questions based on a 3rd Generation Partnership Project (3GPP) Release 18 dataset stored in the database. The work shows the feasibility of using RAG to improve LLMs in performance and accuracy by providing domain-specific knowledge in telecommunications. However, the approach is limited to Q&A and does not address open-ended questions, which require automated reasoning to support coherent, constraint-aware decision-making. Therefore, Q&A alone is not sufficient for the configuration of wireless systems. The work [5] highlights the differences between traditional RAG and Agentic RAG approaches. Traditional RAG is implemented as a single-pass pipeline and has the ability to offer domain knowledge to improve the LLMs performance with low inference latency, but has the inability to review and improve configurations. In contrast, Agentic RAG represents an advanced extension of RAG, integrating external tools and iterative refinement mechanisms, to support

structured reasoning and constraint-aware decision-making. In addition, the publication [2] explores methods to improve the efficiency of RAG systems and to reduce hallucinations by introducing strategic context reordering to mitigate the "lost-in-the-middle" problem, where LLMs tend to pay less attention to information located in the middle of the input. By placing the most relevant passages at the beginning and the query at the end, retrieval effectiveness can be improved, especially for small models with reduced reasoning capabilities and context-ordering component. For practical wireless systems, technical documentation plays a crucial role, often containing both textual and non-textual elements such as images and tables. Study [1] focuses on processing such heterogeneous data by converting Optical Character Recognition (OCR) and non-OCR images to extract relevant information and store it in a vector database. Finally, 5G systems are typically controlled via Application Programming Interfaces (APIs). To address the increasing complexity of manual interface management, work [3] proposes a method to process Network Exposure Function (NEF) and Representational State Transfer (REST) APIs, generating a dataset for system fine-tuning.

3 Methodology/Approach

The proposed approach combines a single-pass pipeline of a vector database, a fine-tuned lightweight large language model, and a formal verification component like Batfish for general network verification, and Open5Gs and OpenAvnu for domain-specific 5G and TSN validation to avoid unintended system behavior. For vector storage and retrieval, Qdrant is selected as the primary database due to its flexible and powerful API, which allows efficient management and modification of multiple collections. The system organizes data into three separate collections: 3GPP standard documents, technical system documentation, and parameters of the wireless system by using the advances of [3] and [1] for practical system integration. This separation ensures structured retrieval and improves the relevance of query results using [2]. For model deployment, Ollama is used as a lightweight and easy-to-integrate framework for running locally hosted models. A compact large language model, like Phi-2 or Qwen-3, fine-tuned using LoRA, is employed to ensure computational efficiency while maintaining strong performance. Overall, the approach achieves modularity, efficiency, and scalability by orchestrating specialized components for diverse data sources and tasks. Future extensions may focus on developing an Agentic RAG framework, enabling formal reasoning capabilities and refinement that support constraint-aware generation and verifiable configuration synthesis.

References

[1] Sukanya Bag, Ayushman Gupta, Rajat Kaushik, and Chirag Jain. Rag beyond text: Enhancing image retrieval in rag systems. In *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET*, pages 1–6, 2024.

[2] Yufei Chen, Zhiliang Huang, and Youyan Zhang. Efficient-rag: Competitive single-pass performance in rag via strategic context reordering. In *2025 18th International Conference on Advanced Computer Theory and Engineering (ICACTE)*, pages 188–194, 2025.

[3] Zainab Khan, Ahmed Hussain, Mukesh Thakur, Arto Hellas, and Panos Papadimitratos. Nefmind: Parameter-efficient fine-tuning of open-source llms for telecom apis automation, 2025.

[4] Andrew Neeser, Christo Kurisummoottil Thomas, Shengzhe Xu, Naren Ramakrishnan, and Walid Saad. Wireless knowledge grounding in smaller llms using retrieval augmented generation and fine-tuning. In *ICC 2025 - IEEE International Conference on Communications*, 2025.

[5] Fnu Neha and Deepshikha Bhati. Traditional rag vs. agentic rag: A comparative study of retrieval-augmented systems. In *2025 IEEE International Conference on Future Machine Learning and Data Science (FMLDS)*, pages 383–387, 2025.

The Extended Scientific Method — From DNA to SC and ML

Erhard Glötzl

JKU, Johannes Kepler University Linz, Austria
erhard.gloetzl@gmail.com

Extended Abstract

1 Introduction and Core Thesis

This presentation advances a provocative thesis: *SCML* is neither an incremental improvement of Symbolic Computation (*SC*) nor of Machine Learning (*ML*) but a fundamentally new, qualitative leap in information technology with profound implications for humanity's future.

To substantiate this, we employ the *Extended Scientific Method (ESM)* as a foundational meta-algorithm. The ESM augments the traditional cycle of the scientific method (observation, hypothesis, testing) with two critical phases: the *search for basic elements* and the *analysis of their structure.* This four-step framework provides a universal blueprint for deciphering complex systems. We trace its application from the foundational code of life (DNA) to the paradigms of SC as AI 1.0 and ML as AI 2.0, culminating in their synthesis SCML as AI 3.0.

2 The Extended Scientific Method: A Universal Framework

We formalize the ESM in four interdependent steps, with (2) and (3) as the core extension:

(1) Observation & Modeling: Isolate phenomena and form hypotheses.
(2) *Search for Basic Elements:* Identify fundamental components (e.g., atoms, words).
(3) *Analysis of Structure*: Discover governing rules and relationships (e.g., Periodic Table, grammar).
(4) Inference & Application: Explain and engineer the behavior of the whole system.

The ESM's power is *generative*. Mapping a structure enables*:*

(1) Gap-filling (predicting missing elements),
(2) Generalization (extending principles to new domains), and
(3) Combination (synthesizing structures to create new fields, e.g., $\mathrm{SC} + \mathrm{ML} \rightarrow \mathrm{SCML}$).

3 Illustrative Examples Across Disciplines

The ESM's universality is demonstrated across fields:

- ***Chemistry:*** (1) Compounds → (*2) Elements → (3) Periodic Table* → (4) predicts new elements: Ga, Ge, Sc (Gap-filling).
- ***Mathematics:*** (1) Linear equation systems → *(2) Eigenvector basis → (3) Spectral Theorem* → (4) solution
  generalize to non-linear systems → Gröbner basis

generalize to all mathematical structures → Langlands programm

- ***Linguistics:*** (1) Language → *(2) phonemes and morphemes → (3) syntax and grammar →* (4) speech and writing.
- ***Classical Mechanics:*** (1) Motion → *(2) mass, canonical variables, potential → (3) Newton, Lagrange, Hamilton* → (4) dynamical behavior.
- ***Economics:*** (1) Economic activity → *(2) economic power, economic variables, utility, constraints → (3) GCD Models* → (4) policy analysis (Glötzl et al., 2019).
- ***General Evolutionary Theory (GET):*** (1) Evolution → *(2) Information Types and Information Technologies→ (3) Triad of Technologies* (Storage, Duplication, Processing) → (4) governs biological, cultural and technological evolution (Glötzl, 2025).

These cases confirm the ESM as the unifying cognitive engine for structuring knowledge.

4 The Evolution of Information Technologies

Applying the ESM via GET yields a periodization of eight eras, each defined by a dominant information type and driven by the core triad of Information Technologies (storage, duplication, processing):

[1] *Structural Information* (Crystalline): No processing.
[2] *Early Biochemical Information* (RNA): No processing.
[3] *Genetic Information* (DNA/Proteins): Processing via sexual reproduction.
[4] *Neural Information* (Electro-Chemical): Processing via the limbic system.
[5] *Complex Neural Information* (Neuronal Network, Brain): Processing via logical reasoning.
[6] *Local-External Information* (Small Data): Processing via Symbolic Computation (**SC**) = AI 1.0.
[7] *Distributed-External Information* (Big Data): Processing via Machine Learning (**ML**) = AI 2.0.
[8] *Future Information***:** Targeted processing via **SCML** = AI 3.0.

5 Synthesis: Positioning SC, ML, and the Path to AI 3.0

Within the GET framework, SC and ML find precise historical and conceptual positions:

- **SC** is the pinnacle of processing from the Local-External (Small Data) Age [6] as AI 1.0, formalizing exact, rule-based reasoning.
- **ML** (including LLMs) defines processing in the Distributed-External (Big Data) Age [7] as AI 2.0, excelling at statistical pattern recognition.
- The synthesis **SC+ML=SCML** heralds the processing paradigm of the Future Age [8]**:** AI 3.0 aims to overcome the limitations of its components by combining the rigorous reasoning of SC with the adaptive pattern recognition of ML, enabling the logical processing of big data with verifiable chains of reasoning.

6 Conclusion and Vision

The journey "From DNA to SC and ML" exemplifies the ESM in action. Their convergence into AI 3.0 is not just a technical milestone but, through the lens of GET, the logical next step in the evolution of information processing. This perspective provides a coherent roadmap: to *consciously apply the ESM* to both understand modern AI and engineer AI 3.0 by design. The

goal is to create intelligent systems that are not only powerful but also *structured, comprehensible, and aligned with the logical frameworks of scientific discovery.*

References

Glötzl, E., Glötzl, F. and Richters O. (2019). *From Constrained Optimization to Constrained Dynamics: Extending Analogies between Economics and Mechanics*. In *Journal of Economic Interaction and Coordination 14 (3)* (pp. 623–642). https://doi.org/10.1007/s11403-019-00252-7.

Glötzl, E. (2025). *Unification of Biological and Cultural Evolution through Natural Periodization.* In *Navigating Complexity in Big History*, edited by David J. LePoire, Leonid Grinin, and Andrey Korotayev. World-Systems Evolution and Global Futures. Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-85410-1_9.

A Computational Framework for Automated Discovery within Conservative Matrix Fields

Uri Kasher Hitin, Michael Shalyt, Shachar Weinbaum, Hila Barkan, Tali Monderer, Elyasheev Leibtag, Rotem Kalisch, and Ido Kaminer

Technion - Israel Institute of Technology, Haifa 3200003, Israel

Historically, discovering elegant recurrence relations for fundamental mathematical constants relied on profound human intuition to invent a specific formula. While these classical approaches have yielded beautiful mathematical results, exploring formulas efficiently has proven notoriously difficult. Recently, Conservative Matrix Fields (CMFs) [1] have emerged as a powerful algebraic concept that condenses infinite summations, continued fractions, and recurrence relations into a single mathematical object [2]. In essence, a CMF can be thought of as a geometric space where each trajectory (a ray) encodes a different recurrence relation. Early algorithmic efforts explored a few specific CMFs for $\zeta(3)$ [1] and π [3]. However, navigating high-dimensional CMFs has remained analytically intractable. Such CMFs relate to higher order recurrence relations – necessary for irrationality-proving formulas for constants like $\zeta(5)$. Consequently, new analytical and computational methods are needed to explore the necessary high-dimensional CMFs.

Here we present a computational framework for systematic exploration of CMFs. While calculating individual trajectories is computationally feasible, achieving sufficiently wide coverage to discover new and optimal formulas has been bottlenecked by the intrinsic geometry of the space: it naturally decomposes into distinct convex regions (shards), and their number grows from tens in 3D to millions in 11D, posing a severe combinatorial barrier.

To overcome this, our system employs a multilayer filtration pipeline (Figure 1). It first exploits mathematical symmetries to prune the shard count, then applies efficient characterization and sampling techniques to the remaining shards (e.g., compute the value the recurrence converges to). After filtering shards without a converging trajectory, it performs deep exploration and extracts numerous number-theoretical properties of the recurrences and their convergence (e.g., error rate). The data acquired during the search is stored for later use, gradually increasing the CMFs dataset. This approach enables a targeted search for optimal formulas.

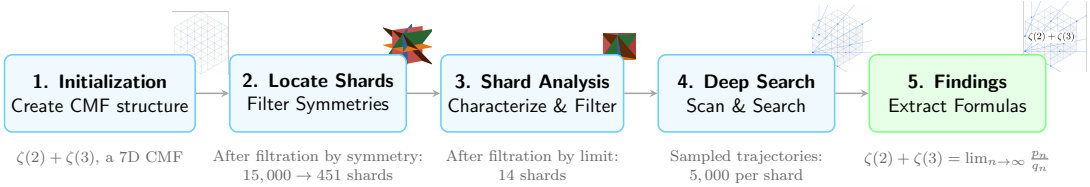


Figure 1: *Advanced shard filtration and space exploration pipeline. Illustrating the exploration process of a 7D CMF containing* $\zeta(2)+\zeta(3)$ *and performing guided search.*

The immediate yield of this structured geometric exploration is twofold. First, it allows finding high-quality rational approximations that prior methods could not reach. Second, the unprecedented volume of extracted data allows observation of previously hidden mathematical phenomena. For instance, plotting this data reveals intriguing geometric structures, such as the continuous variation of the irrationality measure δ as a function of the trajectory angle (Figure 2).

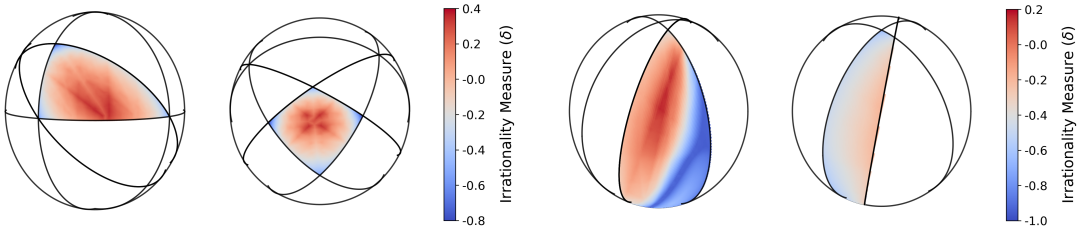


Figure 2: *Geometrical search for irrationality. Left: a CMF containing* $\log(2)$ *recurrences. Right: a CMF containing* π *recurrences. 2 shards plotted for each CMF. The irrationality measure of an approximation* $\frac{p_n}{q_n}$ *of constant* L *is defined as* $\delta = -1 - \lim_{n\to\infty} \frac{\log\left|L - \frac{p_n}{q_n}\right|}{\log|q_n|}$*, indicating that a higher* δ *hints at a better approximation. This makes positive values prime targets for irrationality proofs.*

Crucially, our system extracts formulas across higher-dimensional CMFs (Table 1), with several already breaking the previous record irrationality measure δ, paving the way toward new irrationality proofs. Notably, our framework was able to discover a new record δ for Catalan's constant G and independently reconstruct the known records for $\zeta(2)$ and $\zeta(3)$.

Constant	Generating Function Family	δ	Dimensions
$\log(2)$	${}_2F_1(a_1\, a_2;\, b_1;\, -1)$	0.359	3
$L(2, \chi_{-3})$	${}_4F_3\,(a_1 \ldots a_4;\, b_1 \ldots b_3;\, 1)$	-0.06	7
G	${}_4F_3\,(a_1 \ldots a_4;\, b_1 \ldots b_3;\, 1)$	-0.285	7
$\zeta(2)$	${}_4F_3\,(a_1 \ldots a_4;\, b_1 \ldots b_3;\, 1)$	0.244	7
$\zeta(2) + \zeta(3)$	${}_4F_3\,(a_1 \ldots a_4;\, b_1 \ldots b_3;\, 1)$	-0.0067	7
$\zeta(3)$	$G^{4,2}_{4,4}\,(a_1 \ldots a_4;\, b_1 \ldots b_4;\, 1)$	0.221	8
$\zeta(5)$	${}_6F_5\,(a_1 \ldots a_6;\, b_1 \ldots b_5;\, 1)$	-0.168	11

Table 1: *Examples of recurrence relations discovered using our framework which yield rational approximations (note that higher* δ *indicates a better approximation, as detailed in Figure 2). The 'Generating Function Family' column specifies the function family from whose contiguous relations the CMF recurrences originate. Where* ${}_pF_q$ *and* $G^{m,n}_{p,q}$ *denote the hypergeometric and Meijer-G functions respectively, while* a_i *and* b_i *denote the parameter degrees of freedom.*

Beyond individual formulas, this framework provides rich datasets of recurrence relations and their convergence behavior. The continuous geometric nature of properties like δ within CMF domains suggests that the space is highly learnable: Because δ varies smoothly with respect to the trajectory angle (Figure 2), the space is amenable to both gradient-based optimization and generalization via standard regression models. These ML-ready datasets provide the exact inputs and labels needed to train predictive models (e.g., predicting the resulting δ directly from initial trajectory parameters) or to train reinforcement learning agents to navigate shards toward formulas with the tightest error rates. Ultimately, this work bridges symbolic computation and automated exploration, establishing a foundation for a new era of AI-driven discovery in number theory.

References

[1] Elimelech et al. Algorithm-assisted discovery of an intrinsic order among mathematical constants. *Proceedings of the National Academy of Sciences*, 2024.

[2] Weinbaum et al. On conservative matrix fields: Continuous asymptotics and arithmetic, 2025.

[3] Raz et al. From Euler to AI: Unifying formulas for mathematical constants. In *Advances in Neural Information Processing Systems*, 2025. NeurIPS Poster 117099.

On the Use of ChatGPT in Symbolic Computation and Mathematical Research: A Case Study in Computational Origami

Tetsuo Ida

University of Tsukuba, Tsukuba, Japan
ida@cs.tsukuba.ac.jp

Abstract

This paper reports on the sustained use of ChatGPT in a symbolic computation–based mathematical research workflow, focusing on its roles in expository refinement, conceptual clarification, and natural-language mediation within a Mathematica Notebook environment, as well as on practical issues of stability, reproducibility, and knowledge organization. The study draws on extended practical experience gained during the revision of the monograph *An Introduction to Computational Origami* (Springer, to appear), and aims to identify both the opportunities and limitations of integrating large language models into such workflows.

The central setting of this work is a research environment built around the Mathematica Notebook interface, in which the author's origami system (Eos) is implemented. In this environment, ChatGPT is not used as a standalone system, but as a complementary component supporting tasks such as formalizing definitions, refining mathematical exposition, checking consistency, and restructuring arguments. A key architectural principle emerging from this experience is the separation between *thinking* and *organization*: exploratory reasoning, dialogue, and incremental refinement are carried out interactively in the Notebook, while stable knowledge artifacts—such as indices, summaries, and cross-references—are maintained externally in Markdown. This separation enables a clear distinction between transient cognitive processes and persistent scholarly records.

Within this framework, the interaction with ChatGPT also influenced the conceptual development of the underlying mathematical model. In particular, the focus of the origami formalization shifted from a static description based on face adjacency relations to a more operational perspective centered on face division as a primitive transformation. This shift reflects a transition from structural description to process-oriented modeling and aligns with the view of origami as a sequence of state transitions. ChatGPT played a role in articulating and refining this transition, highlighting its potential as a tool for conceptual clarification as well as linguistic assistance.

The paper analyzes workflows ranging from highly manual interaction—where the researcher guides each step of formalization—to partially automated pipelines, in which ChatGPT outputs are post-processed and integrated into structured repositories. Particular attention is given to Markdown as an intermediate representation for indexing and retrieval, enabling efficient navigation of a growing body of conversational and mathematical content. The combination of Notebook-based symbolic computation and Markdown-based knowledge organization provides a flexible yet disciplined framework for managing complex research materials.

Several practical challenges are examined in detail, including stability (e.g., session persistence and interface reliability), reproducibility (ensuring that generated content can be traced and verified), and knowledge organization (preventing fragmentation and redundancy across conversational threads). While ChatGPT is effective at generating well-formed mathematical text and suggesting structural improvements, careful human oversight remains essential to maintain logical rigor and consistency, particularly in domains involving formal definitions and symbolic reasoning.

From a broader perspective, this experience illustrates how large language models can serve as interactive interfaces to symbolic computation systems, supporting both formalization and communication. This aligns with emerging directions in integrating machine learning and symbolic computation, including the use of language models as co-pilots for mathematical software. The paper

concludes by outlining design principles for such integration, including modular workflow design, explicit separation of concerns, and lightweight but robust indexing mechanisms. These principles are expected to be relevant not only to computational origami but also to a wider range of research areas at the intersection of symbolic computation and machine learning.

Efficient Dataset Generation for Bases of Zero-Dimensional Ideals

Yuki Ishihara[1] and Kazuhiro Yokoyama[2]

[1] Department of Mathematics, College of Science and Technology, Nihon University, Japan.
ishihara.yuki@nihon-u.ac.jp
[2] Department of Mathematics, College of Science, Rikkyo University, Japan.
kazuhiro-yokoyama@rikkyo.ac.jp

1 Introduction

In recent years, there has been growing interest in applying machine learning to the study of symbolic computation. For example, machine learning has been applied to computationally expensive symbolic tasks, such as integration [4], Gröbner basis [2], and cylindrical algebraic decomposition [1]. Machine learning is expected to reduce the complexity from a different perspective from that of traditional symbolic computation algorithms. However, machine learning typically requires sufficiently large datasets, which are often difficult to obtain in symbolic computation. For example, if a single computation requires ten minutes, generating one million samples would amount to ten million minutes (approximately 7,000 days), thereby incurring an extremely high computational cost. Therefore, efficient dataset generation remains a critical challenge in symbolic computation. In this extended abstract, we present an efficient method for the random generation of bases of zero-dimensional ideals. The proposed approach enables effective dataset construction for learning tasks involving ideals, including those related to Gröbner bases and border bases. This approach generalizes the result in [3] by extending it from radical ideals to non-radical ones.

Let K be a field and $K[X] = K[x_1, \dots, x_n]$ the n-variables polynomial ring over K. Let $I = \langle G \rangle$ be an ideal of $K[X]$ generated by $G = (g_1, \dots, g_s)^\top$, that is, $I = \{h_1 g_1 + \cdots + h_s g_s \mid h_1, \dots, h_s \in K[X]\}$. Here, G is called a basis of I. We consider the problem of generating a large number of bases of I from G. For example, if G is a Gröbner basis of I then this task enable us to generate dataset of pairs (F, G) where F is a non-Gröbner basis of I and G is a Gröbner basis of I. For a basis $F = (f_1, \dots, f_r)^\top$ of I, each f_i can be written as $f_i = h_{i1} g_1 + \cdots + h_{is} g_s$ for some $h_{i1}, \dots, h_{is} \in K[X]$. Thus, letting $A = (h_{ij}) \in K[X]^{r \times s}$, we can write $F = AG$. On the other hand, for a polynomial matrix $A \in K[X]^{r \times s}$, AG does not necessarily form a basis of I. We therefore ask: what is the probability that AG is a basis of I? This problem can also be interpreted using parametric ideals as follows. Let d be a positive integer and $\overline{A} = (\overline{h}_{ij})$ an $r \times s$ matrix with $\overline{h}_{ij}$ as (i, j) entry where $\overline{h}_{ij} = \sum_{|\alpha| \le d} a_{\alpha,ij} X^\alpha$ is a parametric polynomial of total degree d with parameters $a_{\alpha,ij}$. Let $\mathcal{A} = \{a_{\alpha,ij}\}$ be the set of all parameters of $\overline{A}$, and let D denote the cardinality of $\mathcal{A}$. For a point $q \in K^D$, we define σ_q to be the specialization map $\sigma_q : K[\mathcal{A}, X] \to K[X]$, given by $\sigma_q\big(f(\mathcal{A}, X)\big) = f(q, X)$ for $f(\mathcal{A}, X) \in K[\mathcal{A}, X]$. Let $K[X]_{\le d} = \{f \in K[X] \mid \mathrm{tdeg}(f) \le d\}$ where $\mathrm{tdeg}(f)$ is the total degree of f. For any $A \in K[X]_{\le d}^{r \times s}$, there exists $q \in K^D$ such that $\sigma_q(\overline{A}) = A$. Thus, it is enough to consider the parametric polynomial matrix $\overline{A}$ and $q \in K^D$ instead of the particular $A \in K[X]_{\le d}^{r \times s}$. Our main theorem establishes that, for almost all random $A \in K[X]_{\le d}^{r \times s}$ satisfying $r > s$, one has $\langle AG \rangle = \langle G \rangle$ when the characteristic of K is 0 or large enough (see Theorem 2.0.1 and Corollary 2.0.2). In other words, by applying randomly chosen matrices $A \in K[X]_{\le d}^{r \times s}$ to G with $r > s$, one can generate a dataset of bases of I, in which, more precisely, most elements are bases of I.

2 Main Results

Theorem 2.0.1. *Let K be a field of characteristic 0 and $I = \langle G \rangle$ a zero-dimensional ideal of $K[X]$. If $r > s$, then there exists a non-empty Zariski-open set $O \subset K^D$ such that*

$$\langle \sigma_q(\overline{A}G) \rangle = \langle G \rangle$$

for any $q \in O$.

Corollary 2.0.2. *Let K be a field of characteristic zero or sufficiently large positive characteristic and $I = \langle G \rangle$ a zero-dimensional ideal of $K[X]$. If $r > s$, then $\langle AG \rangle = \langle G \rangle$ holds with probability almost 1 when the coefficients of each entry h_{ij} of $A \in K[X]_{\leq d}^{r \times s}$ are chosen at random.*

Example 2.0.3. *Let $G = (x^2, y)^\top$, $d = 1$, $\mathcal{A} = \{a_1, \ldots, a_{18}\}$, $X = \{x, y\}$ and*

$$\overline{A} = \begin{pmatrix} a_1x + a_2y + a_3 & a_4x + a_5y + a_6 \\ a_7x + a_8y + a_9 & a_{10}x + a_{11}y + a_{12} \\ a_{13}x + a_{14}y + a_{15} & a_{16}x + a_{17}y + a_{18} \end{pmatrix} \in \mathbb{Q}[a_1, \ldots, a_{18}, x, y]^{3 \times 2}.$$

Then

$$\begin{aligned} \overline{A}G = ((a_1x + a_2y + a_3)x^2 + (a_4x + a_5y + a_6)y, \\ (a_7x + a_8y + a_9)x^2 + (a_{10}x + a_{11}y + a_{12})y, \\ ((a_{13}x + a_{14}y + a_{15})x^2 + (a_{16}x + a_{17}y + a_{18})y)^\top. \end{aligned}$$

If we assign random values $q \in \mathbb{Q}^{18}$ to $a_1, \ldots, a_{18}$, then in most cases $\langle \sigma_q(\overline{A})G \rangle = \langle G \rangle$. For example, for $A = \begin{pmatrix} 2x + 3y + 1 & 5x - y + 7 \\ x - 3 & -2y + 5 \\ 8x + 3y + 1 & 3x + 2y - 1 \end{pmatrix} \in \mathbb{Q}[x, y]^{3 \times 2}$,

$\langle (2x+3y+1)x^2 + (5x-y+7)y, (x-3)x^2 + (-2y+5)y, (8x+3y+1)x^2 + (3x+2y-1)y \rangle = \langle x^2, y \rangle$.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP22K13901 and JST K Program Grant Number JPMJKP24U2, Japan.

References

[1] Changbo Chen, Rui-Juan Jing, Chengrong Qian, Yaru Yuan, and Yuegang Zhao. A dataset for suggesting variable orderings for cylindrical algebraic decompositions. In *Computer Algebra in Scientific Computing*, pages 100–119, Cham, 2024. Springer Nature Switzerland.

[2] Hiroshi Kera, Yuki Ishihara, Yuta Kambe, Tristan Vaccon, and Kazuhiro Yokoyama. Learning to compute Gröbner bases. In *Advances in Neural Information Processing Systems*, volume 37, pages 33141–33187. Curran Associates, Inc., 2024.

[3] Hiroshi Kera, Nico Pelleriti, Yuki Ishihara, Max Zimmer, and Sebastian Pokutta. Computational algebra with attention: Transformer oracles for border basis algorithms. In *Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[4] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

Breaking the Data Barrier in Learning Symbolic Computation: A Case Study on Variable Ordering Suggestion for Cylindrical Algebraic Decomposition *

Rui-Juan Jing[1], Yuegang Zhao[1], and Changbo Chen[2]

[1] Jiangsu University, Zhenjiang, China
rjing@ujs.edu.cn, ygzhao@stmail.ujs.edu.cn
[2] Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences, Chongqing, China
chenchangbo@cigit.ac.cn

Abstract

Symbolic computation, powered by modern computer algebra systems, has important applications in mathematical reasoning through exact deep computations. The efficiency of symbolic computation is largely constrained by such deep computations in high dimension. This creates a fundamental barrier on labelled data acquisition if leveraging supervised deep learning to accelerate symbolic computation. Cylindrical algebraic decomposition (CAD) is a pillar symbolic computation method for reasoning with first-order logic formulas over reals with many applications in formal verification and automatic theorem proving. Variable orderings have a huge impact on its efficiency. Impeded by the difficulty to acquire abundant labelled data, existing learning-based approaches are only competitive with the best expert-based heuristics. In this work, we address this problem by designing a series of intimately connected tasks for which a large amount of annotated data can be easily obtained. We pre-train a Transformer model with these data and then fine-tune it on the datasets for CAD ordering. Experiments on publicly available CAD ordering datasets show that on average the orderings predicted by the new model are significantly better than those suggested by the best heuristic methods.

Main results

An overview of the proposed pre-training and fine-tuning framework for CAD variable ordering selection is illustrated by Figure 1. The entire process begins with generating a large number of random systems that reflect the structural characteristics of those in the CAD order dataset, including the number of constraints, the number of variables, and the exponents of variables. Each generated system is then preprocessed to extract a feature vector, which serves as its label according to a designed pre-training task. The resulting labeled dataset is subsequently tokenized and used to train and evaluate a Transformer model. The model obtained through this pre-training procedure is referred to as the *pre-trained model*. After pre-training, we fine-tune the model on the CAD order dataset. During fine-tuning, we freeze the input embedding layers and some early encoder layers (highlighted in light blue in Figure 1). These layers are expected to encode task-independent structural properties of input systems, as well as basic computational behaviors that facilitate learning effective variable orderings for CAD construction. The remaining unfrozen layers are then trained using the CAD order dataset, enabling the model to specialize in CAD variable ordering prediction. This fine-tuning strategy allows the model to leverage the general representations learned during pre-training while adapting to the specific patterns of the downstream task.

*Corresponding author: Changbo Chen.

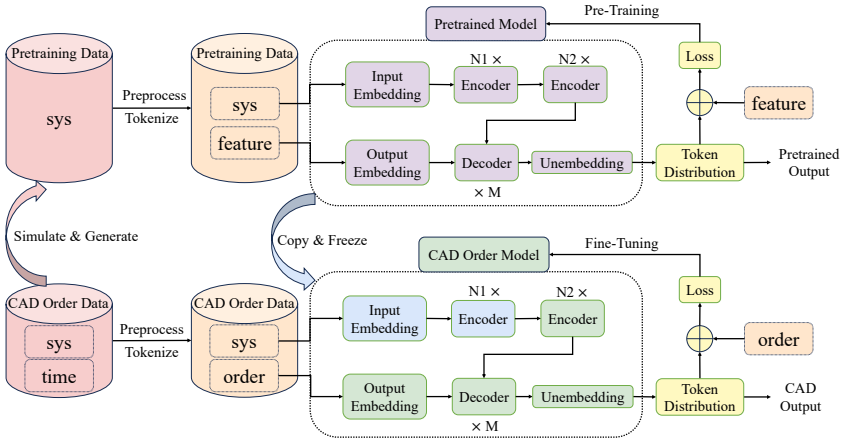
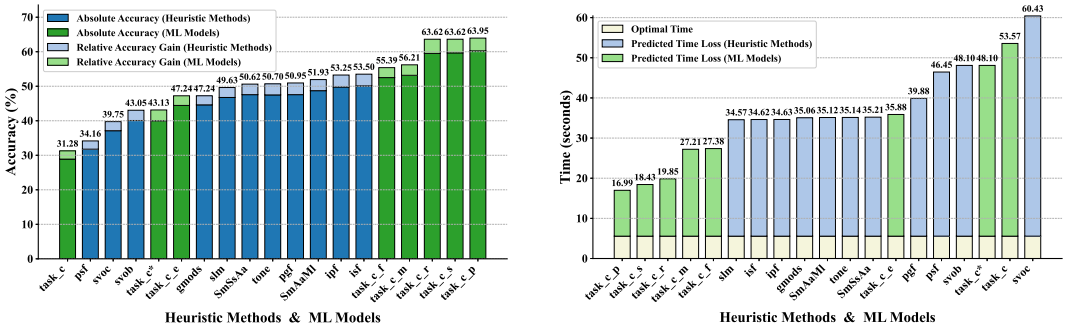


Figure 1: An overview of the proposed framework for CAD variable ordering selection.

Figure 2 compares the performance of ML models and heuristic methods for CAD variable ordering selection on the three variable random dataset DQ-3.



(a) Accuracy results.

(b) Execution time results.

Figure 2: Comparison of heuristic methods and ML models on the DQ-3 testing set.

More results and experiments can be found in the full version of this paper [1].

References

[1] Rui-Juan Jing, Yuegang Zhao, and Changbo Chen. Breaking the data barrier in learning symbolic computation: A case study on variable ordering suggestion for cylindrical algebraic decomposition. *arXiv preprint arXiv:2601.13731*, 2026.

Replacing Heuristic Rule Ordering in Symbolic Integration with Learned Policies*

Rohit John[1], Rashid Barket[2], and Matthew England[2]

[1] University of Bath, Bath, U.K.
rpj31@bath.ac.uk
[2] Coventry University, Coventry, U.K.
{barketr, Matthew.England}@coventry.ac.uk

Abstract

Rule-based symbolic integrators such as SymPy's `manualintegrate` apply transformation rules in a fixed order, often producing inefficient search trees. We cast rule selection as a reinforcement learning problem in which a neural policy ranks integration rules while the Computer Algebra System executes transformations and verifies correctness. Trained on only 50,000 unlabelled expressions, our agent reduces rule attempts by 77% and runtime by 80% on the RUBI benchmark set while improving accuracy over both `manualintegrate` and a supervised learning baseline trained on 2,000,000 labelled examples.

1 Introduction

Symbolic integration remains a challenging problem in Computer Algebra, lacking a single complete algorithmic solution. Rule-based solvers such as SymPy's `manualintegrate` rely on hand-crafted heuristics to select transformation rules, often resulting in long, inefficient derivations. By learning to prioritise promising rules, a neural model can reduce wasted rule attempts and guide the solver toward a solution more efficiently, without replacing the underlying symbolic reasoning engine. Prior work approaches rule selection via supervised behaviour cloning [5], or trains transformers to predict antiderivatives directly [2]. We cast rule selection as a reinforcement learning problem for the first time, using trajectory-level rewards to produce multiple learning signals from a single expression, enabling training on substantially fewer examples without sacrificing correctness.

2 Method

We model rule selection as a finite-horizon Markov Decision Process (S, A, T, R, γ). A state s_t is the current integrand encoded as a prefix-notation token sequence, with a prepended `[CLS]` token [1] as a global expression summary. The action a_t is one of 24 discrete integration rules. Applying a_t to s_t either produces a terminal node, one or more sub-problems each becoming a new state s_{t+1}, or leaves the integrand unchanged on failure, advancing to the next rule in the ranked list. Rewards are assigned per attempted rule: $r_t = -0.05$ for a successful application and $r_t = -0.2$ for a failed attempt. At episode termination, successful-action rewards are post-scaled by an outcome-dependent factor $\lambda(o_T) \in \{-15, +15, +10\}$ for correct, incorrect, and `DontKnowRule` outcomes respectively; since successful-step rewards are negative, $\lambda = -15$ makes them positive for correct episodes while the remaining values amplify the penalty, prioritising correctness over short but wrong derivations. The policy maximises expected discounted return $J(\theta) = \mathbb{E}\big[\sum_n \gamma^n \tilde{r}_n\big]$.

*Code and models are available at https://github.com/rohitpj/RL_for_Integration.

RL architecture. Both actor and critic are built on an encoder-only Transformer: the actor produces logits over the 24 rules from the CLS representation while the critic produces a scalar value estimate. Parameters are optimised using the PPO algorithm [4].

Supervised baseline. Following the approach of [5], we train an encoder-only Transformer of identical architecture to our RL model to classify expression–rule pairs. A linear head over the CLS representation produces a softmax distribution over rules, optimised via cross-entropy loss to imitate SymPy's rule selection.

Data. The RL model was trained on 50,000 unlabelled expressions generated by the FWD method of [2], and the supervised baseline was trained on the SIRD dataset, a collection of 2,000,000 labelled rule decisions extracted from SymPy integration traces.

3 Results

We evaluate all three methods on four datasets: held-out FWD (in-domain), IBP and BWD (out-of-domain), and the RUBI independent test suite [3] as an external benchmark. We compare the RL model against SymPy's `manualintegrate` (*Manual*) and our supervised behaviour-cloning baseline across accuracy (correct integrations), total rule attempts (*steps*), number of branches (successful rule applications), and average runtime.

Table 1: Integration results across datasets.

Dataset	Method	Accuracy (%)	# Steps	# Branches	Avg. Time (s)
FWD	Manual	95.7	101.5	22.5	0.44
	Supervised	94.8	70.2	10.1	0.22
	RL	**96.3**	**53.9**	**8.3**	**0.19**
IBP	Manual	91.7	142.9	28.4	0.60
	Supervised	92.1	54.8	10.9	0.24
	RL	**94.5**	**51.2**	**7.5**	**0.22**
BWD	Manual	41.3	271.3	41.1	1.05
	Supervised	60.5	177.0	23.4	0.77
	RL	**69.7**	**75.9**	**9.6**	**0.33**
RUBI	Manual	39.8	209.6	37.7	1.49
	Supervised	43.9	81.5	11.5	0.41
	RL	**44.4**	**62.1**	**8.2**	**0.30**

4 Conclusion

We introduce an RL-based rule selection policy for symbolic integration within SymPy's `manualintegrate` framework, formulating rule selection as a finite-horizon MDP with trajectory-level rewards encouraging correctness and efficiency. Our method requires no labelled intermediate rule supervision, achieves higher accuracy while substantially reducing attempted rules and runtime across all datasets and demonstrates strong out-of-domain generalisation.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186. Association for Computational Linguistics, June 2019.

[2] G. Lample and F. Charton. Deep learning for symbolic mathematics. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[3] A. Rich, P. Scheibe, and N. Abbasi. Rule-based integration: An extensive system of symbolic integration rules. *Journal of Open Source Software*, 3(32):1073, 2018.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[5] V. Sharma, A. Nagpal, and M. F. Balin. SIRD: Symbolic integration rules dataset. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*, 2023.

Zariski-Dense Dataset Generation for Learning to Compute Gröbner Bases

Yuta Kambe[1]

Mitsubishi Electric, 5-1-1, Ofuna, Kamakura, Kanagawa, Japan
Kambe.Yuta@bx.MitsubishiElectric.co.jp

Abstract

Transformer-based methods for Gröbner basis computation have recently attracted growing attention. Kera et al. [3] introduced a dataset generation method for training Transformers on this task, although their experiments were limited to systems with at most 5 variables. More recently, Malhou et al. [6] extended this line of work to instances with up to 13 variables, indicating that Gröbner basis learning is still rapidly developing.

In this presentation, we focus on the algebraic generality of the datasets produced by the backward generation algorithm of [3]. Under a heuristic assumption introduced by Kambe et al. [2], the set of generated systems is Zariski dense in the space of all systems defining the same ideal, for sufficiently large degree. This gives an algebraic sense in which the data are generic. In this talk, we explain the statement and the heuristic assumption, and briefly discuss its meaning for future learning-based symbolic computation.

The Gröbner basis computation problem, discovered by Bruno Buchberger [1], is one of the central problems in symbolic computation. A foundational motivation for applying machine learning to symbolic mathematics goes back to Lample and Charton [5], who emphasized the importance of generality and representativeness of training data. In the case of Gröbner bases, Kera et al. [3] proposed a practical dataset generation algorithm, together with an efficient polynomial embedding scheme, giving the first concrete demonstration that Gröbner basis computation can be learned by machine learning. More recently, Malhou et al. [6] improved this direction by combining the same backward detaset generation framework with a hierarchical attention architecture and a curriculum learning strategy, thereby allowing larger polynomial systems to be handled.

In this presentation, we introduce the algebraic generality of that backward generation procedure. Relatedly, Kera et al. [4] shows that for a generic $m \times s$ polynomial matrix A and a given system G of s polynomials in n variables with $m > n$, one has $\langle AG \rangle = \langle G \rangle$, where the system $F = AG$ possibly lies outside the set $\mathcal{F}$ of systems generating the same ideal as G; by contrast, the construction of Kambe et al. [2] is designed so that the equality $\langle AG \rangle = \langle G \rangle$ is guaranteed from the outset, and the resulting subset inside $\mathcal{F}$ is shown heuristically to be Zariski dense if $m \geq 2s = 2\#(G)$.

Let G be a fixed Gröbner basis, and let $\mathcal{F}$ be the set of all polynomial systems generating the same ideal as G. We denote by $\mathcal{F}_0 \subset \mathcal{F}$ the set of all outputs of the dataset generation algorithm of Kera et al. The main question addressed by Kambe et al. is whether $\mathcal{F}_0$ is large enough to be regarded as a generic part of $\mathcal{F}$.

The Zariski topology on the affine space of coefficient vectors of polynomial systems. For a degree bound $D \geq 0$, write $R_{\leq D}$ for the space of polynomials over K in variables $x_1, x_2, \ldots, x_n$ of degree at most D, and regard $R_{\leq D}^m$ as an affine space $\mathbb{A}^N$.

Theorem 1 ([2])**.** *Assume the heuristic condition in [2]. If $K = \mathbb{Q}$, $m \geq 2s = 2\#(G)$, and D is sufficiently large, then*

$$\mathcal{F}_0 \cap R_{\leq D}^m$$

is Zariski dense in

$$\mathcal{F} \cap R^m_{\leq D}.$$

In particular, the outputs of the algorithm are not confined to a proper closed algebraic subset of the target space. This does not imply statistical uniformity, but it does provide a precise geometric notion of genericity.

Very roughly, the proof studies the image of the backward generation procedure via polynomial matrices and reduces the density statement to an irreducibility-type heuristic for a certain algebraic set. Thus, the main theorem can be read as a geometric justification for why the generated datasets are sufficiently rich for learning.

In this presentation, we will briefly discuss the plausibility of the irreducibility-type heuristic for a certain algebraic set.

References

[1] Bruno. Buchberger. A theoretical basis for the reduction of polynomials to canonical forms. *SIGSAM Bull.*, 10(3):19–29, Augest 1976.

[2] Yuta Kambe, Yota Maeda, and Tristan Vacccon. Geometric generality of transformer-based Gröbner basis computation. In Mee Seong Im and Tony Shaska, editors, *Artificial Intelligence and Mathematics Research*, volume 835 of *Contemporary Mathematics*, pages 195 – 214. American Mathematical Society, 2026.

[3] Hiroshi Kera, Yuki Ishihara, Yuta Kambe, Tristan Vaccon, and Kazuhiro Yokoyama. Learning to compute Gröbner bases. *Advances in Neural Information Processing Systems*, 37:33141–33187, 2024.

[4] Hiroshi Kera, Nico Pelleriti, Yuki Ishihara, Max Zimmer, and Sebastian Pokutta. Computational algebra with attention: Transformer oracles for border basis algorithms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[5] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

[6] Mohamed Malhou, Ludovic Perret, and Kristin E. Lauter. HATSolver: Learning gröbner bases with hierarchical attention transformers. In *The Fourteenth International Conference on Learning Representations*, 2026.

Comparing human perception, computer-algebra, and generative AI approaches for ranking elementary geometry statements*

Zoltán Kovács[1], Tomás Recio[2], Piedad Tolmos[3], and Pilar M. Vélez[4]

[1] Private University of Education, Diocese Linz, Austria
zoltan.kovacs@ph-linz.at
[2] Universidad Antonio de Nebrija, Madrid, Spain
trecio@nebrija.es
[3] Universidad Rey Juan Carlos, Madrid, Spain
piedad.tolmos@urjc.es
[4] Universidad Antonio de Nebrija Madrid, Spain
pvelez@nebrija.es

This communication reports on recent developments of our ongoing work exploring the interest of developing a certain synergy between GeoGebra Discovery's (GGD) computer-algebra-based automated reasoning tools and generative AI programs, that could "learn" from the rigorous, but less "human readable", output of GGD. Specifically, we will describe and analyze generated data comparing the complexity/difficulty/interest (a subtle concept that will be discussed in the presentation) ranking of geometric statements across five contexts:

- answers from a questionnaire asking math teachers and students for their intuitively perceived difficulty of each of the items in a collection of standard, secondary education level, theorems [1],
- performance of teams from some mathematical olympiads when attempting to solve a certain geometry problem [2, 3],
- automated ranking by the GGD `ShowProof` command [7], see below,
- alternative automated ranking approaches by other programs such as JGEX [4], and
- success rate of generative AI programs, such as ChatGPT, in proving the same statements [5, 6].

Let us roughly describe that our GGD `ShowProof` command declares that a statement $\{s_1 = 0, \ldots, s_n = 0\} \Rightarrow \{T = 0\}$ is of complexity d if d is the maximum degree of the syzygies f_i's expressing 1 as a combination of the hypotheses and the negation of the thesis in a proof by contradiction of the given statement: $1 = f_1 s_1 + \ldots + f_n s_n + f_{n+1}(zT - 1)$. This idea is implemented in the `ShowProof` command and it is shown in the last line of its output. See in Figure 1, right, declaring that the Midpoint Theorem (https://en.wikipedia.org/wiki/Midpoint_theorem_(triangle)) has complexity 2, since the polynomials f_i's have maximum degree 2.

*Other people who contributed to this document include Celina A. A. P. Abar (Pontificia Universidade Católica de São Paulo, Brazil), B. Ariño-Morera (Universidad Rey Juan Carlos, Madrid), Angélica Martínez-Zarzuelo (Universidad Complutense, Madrid) and Álvaro Nolla (Universidad Autónoma de Madrid). The authors gratefully acknowledge the support of the project "Augmented intelligence in mathematics education through modeling, automatic reasoning and artificial intelligence" (IAxEM-CM, https://iaxem.es), PHS-2024/PH-HUM-383, granted by the Comunidad de Madrid.

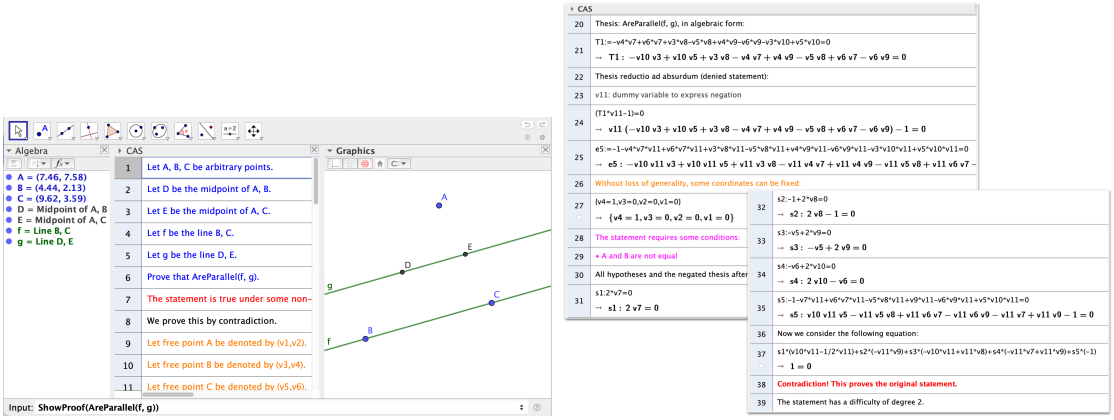


Figure 1: GeoGebra Discovery's `ShowProof` output of the Midpoint Theorem.

The final goal of this research is twofold: to provide educators with a reliable tool for automated difficulty ranking suitable for classroom decision-making, and to enable automatic cooperation between GGD and generative AI when the complexity ranking suggests a high likelihood of error, whereby GGD alerts generative AI to check—using GGD's automated reasoning tool—its own proof steps.

References

[1] Abar, C. A. P. , Almeida, M. V., Recio, T.:"Estimating the Complexity Levels of a Geometric Statement". Boletín de la Sociedad Puig Adam, no. 121, pp. 76–97 (Abril 2026).

[2] Ariño-Morera, B.: GeoGebra Discovery at EGMO 2022. Revista do Instituto GeoGebra Internacional de São Paulo, ISSN-e 2237-9657, Vol. 11, n. 2, pp. 5-16, (2022).

[3] Ariño-Morera, B., Kovács, Z., Recio, T., Tolmos, P.: "Solving with GeoGebra Discovery an Austrian mathematics olympiad problem: Lessons learned". In: Quaresma, P., Kovács, Z. (eds.) Proceedings 14th international conference on automated deduction in geometry, Belgrade, Serbia, 20-22th September 2023. Electronic Proceedings in Theoretical Computer Science, vol. 398, pp. 101109. Open Publishing Association, (2024).

[4] Bartha, S., Kerekes, A., Kovács, Z., Recio, T.: "Automated analysis of the difficulty of secondary school geometry theorems". Ann. Math. Artif. Intell., 93, 1011–1033, (2025).

[5] Botana, F., Recio, T.: "Geometric Loci and ChatGPT: Caveat Emptor!". Computation, 12(2), 30, (2024).

[6] Botana, F., Recio, T., Vélez, M. P.: "On Using GeoGebra and ChatGPT for Geometric Discovery". Computers, 13(8), 187, (2024).

[7] Kovács, Z., Parisse, B., Recio, T., Vélez, M.P., and Jonathan H. Yu: "The ShowProof command in GeoGebra Discovery: Towards the automated ranking of elementary geometry theorems." ACM Communications in Computer Algebra, Vol. 58, No. 2, Issue 228, June 2024. pp. 27–30.

[8] Kovács, Z., Recio, T., Vélez, M.P.: "Automated reasoning tools with GeoGebra: What are they? What are they good for?" In: Richard, P.R., Vlez, M.P., Van Vaerenbergh, S. (eds.) Mathematics education in the age of artificial intelligence: how artificial intelligence can serve mathematical human learning, pp. 23-44. Springer, (2022).

Operator-Theoretic and Complexity-Based Synthesis of a Gradient-Free Federated Kernel Learner

Mohit Kumar, Bernhard A. Moser, and Manuela Geiß

Software Competence Center Hagenberg GmbH, Hagenberg, Austria

Abstract

We present an operator-theoretic and complexity-based analytic framework for *synthesizing* and analyzing a gradient-free federated kernel learner under heterogeneous client data. The point of departure is the population-level problem in $L^2(\mathbb{R}^n, \mathbb{P}_x)$, where the Bayes-optimal class-wise predictor is the mean-squared-error minimizer. Since this object depends on the unknown data-generating law, it is not directly computable from distributed samples. We therefore map the L^2-optimal solution into an RKHS through an invertible forward operator, construct a finite-sample estimator there, and transport it back to L^2 by the inverse operator. The learner is thus derived by an operator-theoretic construction rather than chosen first and analyzed afterwards. The same framework yields a data-dependent hypothesis space, non-asymptotic guarantees on risk, prediction error, robustness, and approximation error, and a communication-efficient federated realization via Kernel Affine Hull Machines (KAHMs). The resulting global rule is gradient-free, exchanges only scalar task-sufficient summaries, and admits privacy- and security-aware extensions.

Problem formulation and synthesis principle. Let (x, y) be a random pair with $x \in \mathbb{R}^n$ and $y \in \{0,1\}^C$, and let y_c denote the indicator of class c. The population-level target is

$$f^*_{x\mapsto y_c} = \underset{g\in L^2(\mathbb{R}^n,\mathbb{P}_x)}{\operatorname{argmin}} \underset{(x,y)\sim\mathbb{P}_{x,y}}{\mathbb{E}} \left[|y_c - g(x)|^2\right] = \underset{y\sim\mathbb{P}_{y|x}}{\mathbb{E}} \left[y_c|x\right].$$

This formulation is canonical but inaccessible in federated settings because $\mathbb{P}_{x,y}$ is unknown and data are distributed across statistically heterogeneous clients. Our question is therefore not merely how to optimize a parameterized model, but how to construct from distributed samples a computable surrogate of the L^2-optimal solution while retaining analytic control of error and communication. For each class c, let $\mathcal{H}_{\Phi_c}$ be an RKHS associated with a generalized kernel. We define an invertible forward operator

$$J^* : L^2(\mathbb{R}^n, \mathbb{P}_x) \to \mathcal{H}_{\Phi_c},$$

map the inaccessible L^2-optimal predictor into RKHS, construct a sample-based estimator $h^{\mathcal{H}_{\Phi_c}}_{x\mapsto y_c} \in \mathcal{H}_{\Phi_c}$ there, and recover a generalized predictor by

$$h_{x\mapsto y_c} = (J^*)^{-1} h^{\mathcal{H}_{\Phi_c}}_{x\mapsto y_c}.$$

Hence the learner is obtained by a forward–approximate–inverse pipeline. This separates population-level optimality from finite-sample approximation and turns the latter into an RKHS estimation problem amenable to kernel and operator-theoretic analysis.

Explicit guarantees. The framework yields a sequence of theorem-level conclusions. First, the RKHS estimator satisfies a non-asymptotic risk bound obtained from kernel and operator theory together with concentration inequalities over operator norms. Under mild regularity assumptions, in particular square-integrability of the class-posterior regression function, this risk scales as $\mathcal{O}(N^{-1/2})$, where N is the total number of training samples across clients. Second, inverse transport yields a generalized learning solution whose risk, prediction error, and approximation error inherit the same $\mathcal{O}(N^{-1/2})$ scaling. Third, the generalized solution is robust in the sense that sufficiently small disturbances cannot induce large prediction error. The operator-theoretic layer therefore provides an explicit analytical route from the inaccessible Bayes-optimal L^2 solution to a finite-sample federated predictor with controlled error.

Complexity-based hypothesis-space derivation. A second pillar is complexity-theoretic. Rather than fixing a hypothesis class *a priori*, we derive a data-dependent hypothesis space from the generalized solution itself by imposing kernel conditions that tune the induced feature map to the scale of the data. The resulting class is tailored to heterogeneous client distributions rather than to a homogeneous population idealization. We then analyze this space via Rademacher complexity and derive additional upper bounds on prediction and approximation error. In this sense, statistical complexity is not merely used to evaluate a chosen learner, it helps determine which class of computable learners should be considered admissible. The operator-theoretic layer synthesizes the learner, while the complexity-theoretic layer identifies and controls the hypothesis space in which that learner lives.

Geometric kernel realization and federated implementation. To realize the analytically derived learner in a communication-efficient federated form, we use Kernel Affine Hull Machines (KAHMs). A KAHM induces a bounded geometric structure associated with the data and a scalar *space folding measure* quantifying how strongly a point must be folded toward the data subspace. This quantity induces a generalized kernel whose feature map aligns with class membership and admits an interpretation as an estimator of class-posterior probability. Since local clients communicate only scalar space-folding summaries rather than gradients or parameter vectors, the global prediction rule can be assembled without repeated rounds of server–client gradient exchange. The resulting federated learner is therefore gradient-free both analytically and operationally. The same low-dimensional structure also supports privacy- and security-aware extensions: scalar summaries can be computed after a single differentially private perturbation of each client data matrix, and the resulting decision rule has a simple comparison-based structure compatible with fully homomorphic encrypted inference.

SCML perspective. The main contribution is not a federated learner with some added theory, but a synthesis program driven by symbolic and analytic methods. Starting from the L^2 population optimum, we transport the learning problem into RKHS by an invertible operator, construct a finite-sample estimator there, map it back to obtain a generalized predictor, derive a data-dependent hypothesis space, and control that space through statistical complexity. KAHM-based space folding then realizes the theory as a communication-efficient federated kernel learner with explicit guarantees on risk, prediction error, robustness, and approximation error. This is precisely the sense in which operator theory, RKHS geometry, and complexity analysis are used not only to explain a machine-learning system, but to derive it.

Verified Vibe Coding

Gábor Kusper

Eszterházy Károly Catholic University
kusper.gabor@uni-eszterhazy.hu

Abstract

Vibe coding accelerates software construction by letting developers express goals in natural language and delegate much of the implementation work to large language models. Its central engineering risk is that the generated codebase can become structurally complex faster than the developer can inspect, understand, and maintain it. We propose Verified Vibe Coding, using "verified" in an operational rather than absolute sense: each development iteration should leave behind some machine-checkable evidence about the artifact. The workflow is organized around four artifacts: documentation, source code, unit tests, and logical specifications such as preconditions, postconditions, assertions, and invariants. Any subset of these artifacts may be written by a human or by an AI tool. The essential point is to make the source of truth explicit and to check generated claims by tests, runtime assertions, model checking, or deductive verification.

1 Motivation

Vibe coding has recently been described as a programming style in which developers interact conversationally with code-generating LLMs and evaluate the resulting artifact through rapid testing, scanning, and repair cycles [1]. This style is attractive because it preserves momentum: the developer can describe a desired feature, ask for changes, paste errors back to the model, and continue iterating. The same property also creates a failure mode. The code grows quickly, while the explicit record of requirements, assumptions, and behavioral constraints often remains weak.

The usual response is to generate more tests. This is useful, but insufficient. Tests sample behavior, they rarely express the full assumptions under which a function is allowed to run, the guarantees it must establish, or the properties that must remain true across a loop or state change. The older foundations of program correctness already identify the relevant logical concepts: assertions, preconditions, postconditions, and invariants [2]. Design-by-contract later made these ideas accessible as explicit software engineering artifacts [3]. Verified Vibe Coding takes these concepts as first-class artifacts in LLM-assisted development.

2 Artifacts and Authorship

Verified Vibe Coding should not be understood as a single automation pattern in which the LLM writes both the program and its specification. It is better described as the controlled co-evolution of four artifacts: documentation (D), code (C), unit tests (T), and logical specification (L). Each component may be human-written, LLM-generated, or jointly edited. What matters methodologically is to record which artifact is treated as the current source of truth and which external checks are used to detect inconsistency.

Several useful configurations follow. In the most direct form of vibe coding, a human supplies D, while the LLM generates C, T, and L. The tests and verifier feedback then become repair signals. In a legacy or human-code setting, the human supplies C, and the LLM proposes T and

L from the code and available documentation. This is useful, but dangerous if the generated specification merely formalizes a bug. A safer documentation-first variant asks the LLM to derive T and L from D before, or independently of, the implementation. Finally, in a contract-first variant, a human writes the crucial parts of L, and the LLM is used to generate or refactor C until the tests and proof obligations are satisfied.

The research problem is therefore not only how to generate logical annotations, but how to avoid circular validation. If the same model generates the code, the tests, and the specification from the same prompt, the artifacts may share the same mistake. A Verified Vibe Coding protocol should therefore make authorship and authority explicit: which artifact expresses the intended behavior, which artifacts are derived from it, and which checks are independent.

3 Target Forms and Tools

The lowest-cost target for L is the ordinary `assert`. Assertions can be generated for many languages without changing the development stack, and they make hidden assumptions executable. However, assertions observe executions rather than covering all executions. They are best understood as a first layer that can later be strengthened into contracts.

A stronger target is a verifier-specific contract language. Dafny is an attractive first experimental platform because clauses such as `requires`, `ensures`, `invariant`, and `assert` become obligations for an automatic verifier [4], and recent work already investigates LLM-based generation of Dafny specifications using verifier feedback [10]. For Java, JML is an older but still active behavioral-interface specification language. OpenJML checks JML-annotated programs by static checking and runtime assertion checking, but experiments should report the exact tool version and supported Java/JML subset [5]. For C, Frama-C/ACSL provides a mature path for annotated C programs and weakest-precondition based proof support [6]. The RISCAL/RISCTP line is relevant on the symbolic-computation side: RISCAL supports executable formal models with contracts and finite-model checking [7], while RISCTP connects such proof problems to theorem proving [8]. This direction is close to recent logical-agent work in which an LLM translates natural-language logical questions into formal proof problems and invokes a prover [9].

4 Research Agenda

The key open problem is whether LLM-assisted development can be shifted from producing plausible code toward producing a logically constrained set of artifacts. We propose to compare workflows by their authorship pattern over D, C, T, L: AI-code/AI-spec generation from documentation, human-code/AI-spec reconstruction, documentation-first generation of tests and contracts, and human-contract/AI-code generation. These workflows should be evaluated not only by whether tests pass, but also by whether annotations are executable, whether static obligations are discharged, and whether the logical specification agrees with the intended documentation rather than merely with the current implementation.

Verified Vibe Coding therefore reframes LLM-assisted programming as a consistency problem among documentation, code, tests, and logical specifications. The verification tool is not replaced by the LLM. It becomes the oracle that distinguishes useful generated logic from convincing but invalid text. This combination offers a practical route from fast AI-assisted construction toward software artifacts that fail early, explain their assumptions, and can gradually be strengthened into formally checked components.

References

[1] A. Sarkar and I. Drosos. Vibe Coding: Programming through Conversation with Artificial Intelligence. *Proceedings of the 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025)*, pp. 87–118, 2025.

[2] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969. doi:10.1145/363235.363259.

[3] B. Meyer. Applying 'design by contract'. *Computer*, 25(10):40–51, 1992. doi:10.1109/2.161279.

[4] K. R. M. Leino. Dafny: an automatic program verifier for functional correctness. In *Logic for Programming, Artificial Intelligence, and Reasoning*, LNCS 6355, pp. 348–370. Springer, 2010. doi:10.1007/978-3-642-17511-4_20.

[5] D. R. Cok. JML and OpenJML for Java 16. In *Proceedings of the 23rd ACM International Workshop on Formal Techniques for Java-like Programs (FTfJP 2021)*, pp. 65–67. ACM, 2021. doi:10.1145/3464971.3468417.

[6] P. Baudin, F. Bobot, D. Buehler, L. Correnson, F. Kirchner, N. Kosmatov, A. Maroneze, V. Perrelle, V. Prevosto, J. Signoles, and N. Williams. The dogged pursuit of bug-free C programs: the Frama-C software analysis platform. *Communications of the ACM*, 64(8):56–68, 2021. doi:10.1145/3470569.

[7] W. Schreiner. Theorem and algorithm checking for courses on logic and formal methods. In *Proceedings of ThEdu'18*, EPTCS 290, pp. 56–75, 2019. doi:10.4204/EPTCS.290.5.

[8] W. Schreiner. The RISCTP theorem proving interface: Tutorial and reference manual. Technical Report 22-07, Research Institute for Symbolic Computation, Johannes Kepler University Linz, 2022. doi:10.35011/risc.22-07.

[9] W. Schreiner. Building a logical agent with LangChain ... and quite some vibe coding. Technical Report 26-02, Research Institute for Symbolic Computation, Johannes Kepler University Linz, 2026. doi:10.35011/risc.26-02.

[10] J. P. Faria, E. Trigo, V. Honorato, and R. Abreu. Automatic generation of formal specification and verification annotations using LLMs and test oracles. *arXiv:2601.12845*, 2026.

Reactmine-2: a statistical beam search algorithm for learning biochemical reaction models from time series data

Constant Le Bezvoët[1], François Fages[1], and Julien Martinelli[2]

[1] Inria Saclay, Lifeware Group, France, francois.fages@inria.fr
[2] Aalto University, Espoo, Finland, julien.martinelli@aalto.fi

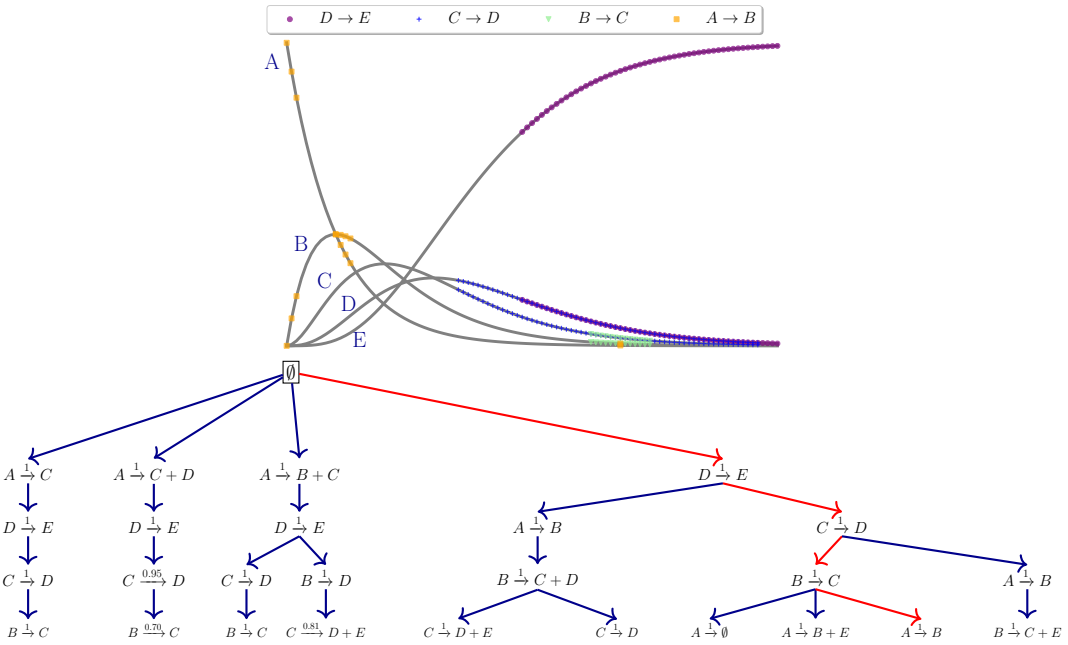
Chemical Reaction Networks (CRNs) are a formalism widely used in computational systems biology to model high-level cell processes in terms of molecular interactions. A CRN is a finite set of reactions over a finite set of molecular species, possibly given with rate functions such as mass action law kinetics (monomial rate function) or Michaelis-Menten kinetics (rational function). A CRN has a bipartite graph structure with vertices for species and reactions and several dynamical interpretations. Depending on the question at hand, a same CRN model can be interpreted in a hierarchy of semantics: by ordinary differential equations (ODE continuous semantics), by continuous time Markov chains (CTMC stochastic semantics), or by ignoring kinetics, by discrete (Petri Net semantics) or Boolean asynchronuous transition system (Boolean semantics), the last three interpretations being formally related by abstract interpretation Fages and Soliman (2008). The Systems Biology Markup Language (SBML) is a model exchange format used in model repositories such as Biomodels which contains a thousand of curated CRN models of various sizes up to more than a thousand molecular species.

While in computational Synthetic Biology, symbolic computation methods are developed to compile mathematical functions in elementary CRNs Fages *et al.* (2025), in Systems Biology, biological modelling remains an art which is currently limited in its applications by the number of available modellers. Automating the process of model building is thus a very desirable goal to attack new applications, develop patient-tailored therapeutics, and also design experiments that can now be largely automated with a gain in both the quantification and the reliability of the observations, at both the single cell and cell population levels. Machine learning is revolutionarizing the statistical methods in biological data analytics, data classification and clustering, and for making predictions from static measurements. However, learning dynamical models from temporal data is highly challenging.

Inferring chemical reaction networks (CRN) from concentration time series is encouraged by the growing availability of quantitative temporal data at the cellular level. This motivates the design of algorithms to infer reactions between the molecular species observed in a given biochemical process, and build CRN structure and kinetics models. State-of-the-art ODE-based inference methods such as SINDy Brunton *et al.* (2016) resort to least square regression combined with sparsity-enforcing penalization, such as Lasso. However, we observe that these methods fail to learn sparse models when the input time series are only available in *wild type* conditions, i.e. without the possibility to play with combinations of zeroes in the initial conditions.

In this abstract, we present Reactmine, a bounded-depth tree search algorithm to infer CRNs from time series data. Sparsity is enforced by inferring reactions with their kinetics in a sequential fashion, with the depth of the search tree bounding the number of inferred reactions. At each node, the reaction candidates are ranked according to the variance of the kinetics inferred on their transition support, and the best candidates are used as choice points at that node. At each successor node, one selected reaction is added and its effect is subtracted from the trace. Each leaf in the search tree represents a CRN candidate. In a final pass, the kinetic parameters of the leaf CRNs are globally re-optimized on the whole trace transitions, and the CRN candidates are ranked according to the quadratic loss between the predicted and observed

state transitions. For example, Reactmine infers the chain CRN $A \xrightarrow{1} B$, $B \xrightarrow{1} C$, $C \xrightarrow{1} D$, $D \xrightarrow{1} E$ from one single trace, by developing the following search tree:



On a benchmark of hidden CRNs of increasing difficulty, including one model of MAPK signal transduction levels (Qiao *et al.*, 2007), Reactmine is able to recover the hidden CRN, or an essentially equivalent form of it, from one single ODE simulation trace. On the opposite, sparse regression algorithms for non linear dynamical systems like SINDy fail to infer sparse ODE systems in that setting. The behavior of sparse regression algorithms is indeed conditional to assumptions about the low level of correlation between predictors (Zhao and Yu, 2006) which are typically not satisfied in our context. Reactmine solves those issues by not relying on sparse regression but on a statistical tree search algorithm which ensures sparsity by construction. The code of the previous version of Reactmine without beam search is available at https://gitlab.inria.fr/julmarti/crninf/ together with introductory notebooks.

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, pages 3932–3937.

Fages, F. and Soliman, S. (2008). Abstract interpretation and types for systems biology. *Theoretical Computer Science*, **403**(1), 52–70.

Fages, F., Hemery, M., and Soliman, S. (2025). On biocham symbolic computation pipeline for compiling mathematical functions into biochemistry. *ACM Commun. Comput. Algebra*, **58**(2), 15–22.

Qiao, L., Nachbar, R. B., Kevrekidis, I. G., and Shvartsman, S. Y. (2007). Bistability and oscillations in the huang-ferrell model of mapk signaling. *PLOS Comp. Biology*, **3**(9), 1–8.

Zhao, P. and Yu, B. (2006). On model selection consistency of lasso. *Journal of Machine Learning Research*, **7**, 2541–2563.

Deep Learning for Integro-Differential Modelling

François Lemaire[1] and Louis Roussel[1]

Univ. Lille, CNRS, Centrale Lille, UMR 9189 CRIStAL, F-59000 Lille, France
{francois.lemaire,louis.roussel}@univ-lille.fr

Abstract

We investigate the use of deep learning techniques for computing integral equations from systems of nonlinear differential equations. The class of integral equations we consider are very general and difficult to manipulate with purely symbolic computations. In order to rapidly explore and experiment with such complicated expressions, we have trained deep learning models to get valuable insight on the type of expressions that can be expected to be calculated using symbolic methods. The first contribution is a novel algorithm for converting an integral equation into a differential equation. The second is the discovery and resolution, via our trained models, of systems which are currently beyond the reach of classical computer algebra.

1 Extended Abstract

This extended abstract is a summary of the article [4] published at the SCML forum. Many models in biology and control theory are written as systems of non differential equations involving unknown parameters. Under suitable assumptions, partial data measurements are sufficient to estimate the values of the parameters. There exists an extensive literature on the parameter estimation problem. This work is motivated by the parameter estimation problem using both computer algebra and deep learning techniques.

We illustrate the parameter estimation problem on the simple example from Figure 1 which only involves two variables $x(t)$ and $y(t)$ and a parameter θ. Assuming that experimental data are only available for $y(t)$ (i.e. $x(t)$ is unknown), can we estimate the value of the parameter θ? One classical approach consists of eliminating $x(t)$ from the system S to obtain Input/Output (I/O) equations only involving θ and $y(t)$. This elimination is usually performed using computer algebra algorithms, which are designed for symbolically manipulating equations. Figure 1 presents two I/O equations $f = 0$ and $F = 0$ obtained by eliminating $x(t)$ from S. For brevity the term Equation f (resp. F) actually means Equation $f = 0$ (resp. $F = 0$). Equation f is a differential I/O equation, while F is an integral I/O equation where the $\int$ is the primitive operator (i.e. $\int u = \int_0^t u(\tau)\mathrm{d}\tau$). Those two I/O equations may be used to estimate the value of θ using the experimental data for $y(t)$.

In case of noisy experimental data (i.e. in case of errors on the measurements), Equation F is usually more suitable since the integration tends to filter the noise. In Figure 1, Equation F can be computed in two ways: either by differential elimination (Algorithm DE) followed by integration (Algorithm INT), or directly by integral elimination (Algorithm IE). Although the DE algorithm is complete (solid arrow on Figure 1), INT and IE are partial algorithms (dashed arrows) implying that integral I/O equations cannot always be computed for general systems.

Improving Algorithms IE and INT is a very complex and time-consuming task. For this reason, we experiment in this work with the use of deep learning techniques to compute integral I/O equations and help improve these algorithms.

The contributions presented in this work demonstrate that the use of deep learning techniques to provide improvement suggestions is a promising approach for integro-differential elimination. Indeed, thanks to the development of a new algorithm I2D, we have *trained 4 models*

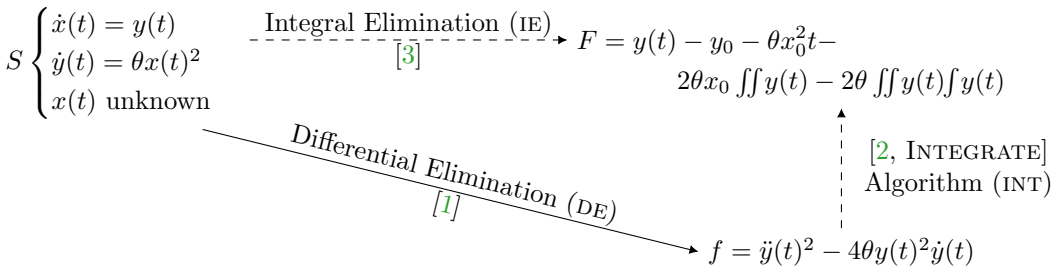


Figure 1: Computing the differential I/O equation $f = 0$ and the integral I/O equation $F = 0$ from the system S.

that revealed and solved *new striking systems* that Algorithms IE and INT could not solve. Those new examples constitute important results since they seem very difficult to create by hand computations. Moreover, those new examples are already a source of inspiration for improving Algorithm IE since logarithms were added in [5] following this work.

The source code associated to the experiment presented in this work is available at https://codeberg.org/louis-roussel/IntegroDifferentialDeepLearning.

Acknowledgments

This work has been partially supported by the THIA ANR Program "AI_PhD@Lille", from the French National Research Agency, grant ANR-20-THIA-0014.

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] François Boulier. *The DifferentialAlgebra project*. 2023. URL: https://codeberg.org/francois.boulier/DifferentialAlgebra.

[2] François Boulier et al. "Additive Normal Forms and Integration of Differential Fractions". In: *Journal of Symbolic Computation* 77 (2016), pp. 16–38. DOI: 10.1016/j.jsc.2016.01.002.

[3] François Lemaire and Louis Roussel. "Contribution to Integral Elimination". In: *Computer Algebra in Scientific Computing*. Ed. by François Boulier et al. Vol. 14938. Rennes, France: Springer Nature Switzerland, Sept. 2024, pp. 215–235. DOI: 10.1007/978-3-031-69070-9_13.

[4] François Lemaire and Louis Roussel. "Deep Learning for Integro-Differential Modelling". In: *RISC Proceedings on Symbolic Computation and Machine Learning*. SCML, Mar. 2026, p. 15. DOI: 10.35011/risc-proceedings-scml.2.

[5] François Lemaire and Louis Roussel. "Recent Advances on Integral Elimination". In: *Proceedings of the 2025 International Symposium on Symbolic and Algebraic Computation*. ISSAC '25. Association for Computing Machinery, 2025, pp. 249–257. ISBN: 9798400720758. DOI: 10.1145/3747199.3747568.

Towards Quantum-Reservoir Trajectory Signatures for Symbolic Rewriting Dynamics

Alexei Lisitsa[1]

University of Liverpool, UK
a.lisitsa@liverpool.ac.uk

Motivation and setting. We investigate trajectory-level representations of symbolic rewriting systems using quantum-reservoir (QR) feature maps [2]. Building on recent work on static QR embeddings of rewriting-state clouds [4], we view rewriting trajectories as time series in QR feature space and analyse them using recurrence statistics and matrix-profile methods [5]. The goal is to explore whether machine learning methods applied to continuous trajectory representations can reveal structures relevant to symbolic reachability and non-reachability.

Our experiments focus on rewriting systems motivated by Andrews–Curtis transformations from combinatorial group theory [1]. We consider states given by pairs of words (r_1, r_2) over generators a, b and their inverses A, B. The rewriting rules include inversion $(r_1, r_2) \mapsto (r_1^{-1}, r_2)$, multiplication $(r_1, r_2) \mapsto (r_1 r_2, r_2)$, and conjugation $(r_1, r_2) \mapsto (g^{-1} r_1 g, r_2)$, where $g \in \{a, A, b, B\}$, together with symmetric variants acting on the second relator.

These operations form the Andrews–Curtis rewriting framework: the conjecture states that every balanced presentation of the trivial group is reachable from (a, b) by such transformations. Several natural families remain unresolved and are strongly suspected to be non-reachable. We also study restricted systems without conjugation or multiplication rules, including cases with known non-reachability invariants [3].

Quantum-reservoir trajectories. Starting from an initial state, we generate rewriting trajectories $s_0, s_1, \ldots, s_T$. Each state is embedded independently into a fixed untrained QR feature space. Words are encoded symbol-by-symbol using local unitary operations interleaved with fixed mixing layers, and selected measurement readouts of the resulting reservoir state produce feature vectors $\Phi(s_t) \in \mathbb{R}^d$. This yields a feature trajectory $\Phi(s_0), \Phi(s_1), \ldots, \Phi(s_T)$.

We study two QR readout granularities: a simple 16-dimensional probability-based representation and a richer 43-dimensional observable-derived representation. The experiments were implemented in Python/Jupyter notebooks using Qiskit for simulation of the quantum-reservoir components and standard scientific Python libraries for the analysis.

Temporal structure analysis. To analyse trajectory structure, we compare genuine rewriting trajectories with shuffled trajectory baselines obtained by permuting local trajectory windows, destroying long-range temporal organisation while retaining local feature statistics.

For a trajectory $x_0, x_1, \ldots, x_T$, we define local windows $W_i = (x_i, \ldots, x_{i+m-1})$ and measure nearest-neighbour distances between windows. For a trajectory statistic $S(\cdot)$, such as mean nearest-window distance or mean matrix-profile value, we evaluate the recurrence ratio $R = S(\text{real trajectory})/S(\text{shuffled trajectory})$. Values $R < 1$ indicate stronger recurrence or revisitation structure in genuine rewriting trajectories relative to shuffled trajectory baselines.

We also compute matrix profiles [5]. Matrix profiles record, for each window W_i, the distance to its closest matching window elsewhere in the trajectory. Low matrix-profile values correspond to repeated motifs or revisitation patterns.

We analyse both raw and z-normalised QR trajectories to distinguish large-scale geometric revisitation effects from scale-independent local motif structure.

Experimental results. Across all systems, genuine trajectories exhibited substantially stronger temporal organisation than shuffled trajectory baselines.

For the 16-dimensional probability-based readout, typical raw matrix-profile ratios were 0.85–0.92; since ratios below 1 indicate stronger recurrence than in shuffled trajectories, this already shows temporal organisation at the level of basic measurement probabilities.

The richer 43-dimensional observable-based readout produced lower ratios, 0.43–0.61, suggesting stronger sensitivity to recurrent symbolic structure.

Length-only trajectory representations also produced low ratios, showing that word growth itself contributes substantial temporal regularity. However, QR representations distinguish rewriting systems in ways not explained purely by length dynamics, indicating sensitivity to additional symbolic structure.

The "no-multiplication" system consistently exhibited the weakest recurrence structure, indicating that multiplication rules are the primary source of large-scale expansion-reduction motifs in rewriting dynamics. Conjugation restrictions also altered trajectory signatures significantly, consistent with orbit-like revisitation behaviour.

Discussion. The results suggest that QR feature trajectories preserve meaningful dynamical information about symbolic rewriting systems. Different rewriting systems produce distinct recurrence signatures, indicating sensitivity not merely to state distributions but to the structure of rewriting evolution itself.

While the present work does not provide formal invariants or proofs of non-reachability, the observed recurrence and matrix-profile patterns suggest a possible connection between temporal structure and reachability complexity. From a machine learning perspective, the framework combines representation learning, trajectory analysis, and time-series motif detection: QR embeddings transform symbolic states into continuous feature spaces where trajectories can be analysed by clustering, nearest-neighbour classification, anomaly detection, and motif discovery. More broadly, the results suggest that rewriting dynamics may admit stable feature-level structures correlated with reachability behaviour, potentially enabling ML-assisted discovery of candidate dynamical invariants for symbolic rewriting systems.

References

[1] J. J. Andrews and M. L. Curtis. Free groups and handlebodies. *Proceedings of the American Mathematical Society*, 16:192–195, 1965.

[2] Keisuke Fujii and Kohei Nakajima. Quantum reservoir computing: A reservoir approach toward quantum machine learning on near-term quantum devices. In *Reservoir Computing*, 2020.

[3] Alexei Lisitsa. The Andrews-Curtis Conjecture, Term Rewriting and First-Order Proofs. In *Mathematical Software - ICMS 2018 - 6th International Conference, South Bend, IN, USA, July 24-27, 2018, Proceedings*, pages 343–351, 2018.

[4] Alexei Lisitsa. Quantum-Reservoir Representations of Rewriting Dynamics, 2026. submitted.

[5] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 1317–1322, 2016.

When the Vibes Fade: *An Industry Perspective on LLM Coding Limits — A Discussion Case for Symbolic Computation*

Thomas Mahringer
metnoia e.U., Austria
thomas@mindruptive.com

1 Context and Empirical Background

Over the past year, we evaluated vibe coding (LLM-assisted software engineering) across several real-world industry projects — practical enterprise applications such as logistics software, ranging from dedicated micro-tools to comprehensive solutions spanning multiple systems and API integrations. They share the typical layered architecture of modern web systems (frontend/backend libraries, database access, data transformations) and exhibit only moderate algorithmic complexity, which made them appear to be good candidates for vibe coding.

2 The Problem: The “Senior Developer Bottleneck” and the Limits of Vibe Coding

While LLMs are highly capable of generating code for these tasks, the sheer size of the applications and the intricate interdependencies between system modules quickly turn development into a challenge. Controlling the development flow and the application lifecycle proved to be non-trivial.

Our projects revealed a bottleneck: LLMs produced recurring errors whose detection was tedious and time-consuming. We estimate that vibe coding saved about 75% of a junior developer's time. However, roughly a third of those savings were consumed by the senior developer tasked with reviewing, debugging, and correcting the generated code.

The senior developer is forced into the role of a "vibe checker" — manually enforcing guardrails, checking pre- and post-conditions, and ensuring the code aligns with broader architectural constraints. Because the reviewer must be considerably more experienced than the AI-assisted developer who generated the code, the bottleneck is shifted rather than eliminated.

3 The Proposed Direction: From Prompting Loops to Symbolic Computation

To make AI-assisted coding truly scalable, we should support — or preferably automate — the senior programmer's task of verifying the correctness of generated code. Today's deterministic safety nets — compilers, static analyzers and software tomography tools, test suites — catch syntactic errors and parts of the logic, but they all operate after the fact: the code has already been generated, the architectural decision already taken. The truly challenging semantic errors and logical hallucinations that survive these checks lie beyond what any amount of prompting can systematically prevent.

Pragmatic, deterministic guardrails are already emerging — for example constraining the output of the LLM (controlled, schema- or grammar-guided, type- and contract-aware generation), and structuring the context in which it operates through deterministically constructed architectural and structural maps of the codebase. These reduce errors but do not prove anything about the code; they mark the pragmatic end of a longer spectrum, whose natural continuation is symbolic computation — formal specifications and verification methods that can prove, or at least check, meaningful properties of the generated code. Automated verification at this end is only possible with a complete or "good enough" formal specification, and the industry faces two massive hurdles here:

- The practical availability and formulation of such specifications.
- The availability of sufficiently strong automated reasoning tools capable of handling modern web architectures.

Both define open research questions that the SCML community is uniquely positioned to address — and the central question of this talk is where, along this spectrum, symbolic computation can most productively connect to industrial vibe-coding workflows..

4 Talk Positioning and Discussion (Food for Thought)

Rather than proposing a specific formal method, this talk is designed as a position piece and discussion case for the SCML community. The author comes from industrial software engineering, not from formal methods, and contributes empirical observations and a problem framing rather than a verification calculus.

Key questions to be discussed with the audience include:

- What is the right way to formulate specifications, contracts, or architectural rules for AI-generated enterprise code? Is a fully formal language strictly necessary, or are there hybrid approaches that already deliver value?
- Where can symbolic methods realistically be inserted into LLM-based coding workflows — at generation time, at review time, or as a structural layer between codebase and model?
- How much "senior approver" time, compute, and API/token cost could realistically be saved if symbolic computation replaced part of the iterative LLM-prompting loop, even short of full verification?

Discovering Symbolic Representation of Conservation Laws of Dynamical Systems using Machine Learning

Meskerem Abebaw Mebratie[1], Rüdiger Nather[2], Guido Falk von Rudorff[3], and Werner M. Seiler[1]

[1] Institute of Mathematics, University of Kassel, 34109 Kassel, Germany
meskerem.mebratie@uni-kassel.de, seiler@mathematik.uni-kassel.de
[2] Department of Electrical Engineering, University of Kassel, 34121 Kassel, Germany
r.nather@uni-kassel.de
[3] Institute of Chemistry, University of Kassel, 34109 Kassel, Germany
vonrudorff@uni-kassel.de

Abstract

Conservation laws of dynamical systems are of great theoretical and practical interest. On a more theoretical side, knowledge of a conservation law almost always provides important insight into a system. Practically, conservation laws allow for a model reduction. Some studies use neural networks to discover conservation laws of dynamical systems, followed by a symbolic regression to obtain closed-form expressions, typically trained on large datasets.

In this work, we present an approach for learning conservation laws of finite-dimensional dynamical systems from trajectory data using kernel methods, more precisely kernel ridge regression with an inhomogeneous polynomial kernel. The method identifies polynomial conservation laws and provides directly explicit symbolic representations without the need of a symbolic regression. This approach is computationally more efficient than neural networks and requires substantially less training data, and this is crucial if we are working with experimental data, which may be much harder to get in large quantities. We also show how to discover multiple, functionally independent conservation laws from a single data set and present a sparsification approach that improves the interpretability and usefulness of the results.

The approach has been tested on various examples of dynamical systems and has also been applied to discrete dynamical systems. Furthermore, it can be used for computing the implicitization of curves and surfaces, as well as for integral manifolds of vector field distributions. The overall process combines a range of algebraic concepts with machine learning, demonstrating both the application of machine learning to symbolic computation and the potential of a hybrid approach that integrates symbolic computation with machine learning techniques.

Proof Engineering: Nakano's Light Puzzle Example

Koji Nakagawa[1] and Bruno Buchberger[2]

[1] Rigor Drive, Japan koji@rigordrive.fun
[2] Research Institute for Symbolic Computation, Johannes Kepler University, Austria
buchberger.bruno@gmail.com

The rapid progress of LLMs (Large Language Models, e.g., ChatGPT, Claude, Gemini) has drastically impacted mathematics recently, with neuro-symbolic theorem proving systems [1] advancing from IMO2025 (International Mathematical Olympiad) gold medal level to handling proofs at the level of mathematical research [2]. In the age of AI, a promising approach of exploring mathematics is to establish a proving cycle for a human, a LLM (Large Language Model), and an ARS (Automated Reasoning System). Figure 1 illustrates a concise overview of this proving process, which in this paper we call "Human-LLM-ARS proving cycle". ARS encompasses various tools, generally categorized as proof checkers (e.g., Lean, Rcoq, Agda) or proof generators (e.g., Theorema https://risc.jku.at/sw/theorema).

In Figure 1, a mathematical theory in natural language is formalized into a suitable formal language using an LLM⟪A⟫and is refined if necessary⟪B⟫. Unproved theorems are then fed to the LLM which yields incomplete ⟦Partial Proofs α⟧⟪C⟫. The ARS checks this, and based on the errors, the LLM continues to improve the ⟦Partial Proofs β⟧⟪D⟫. Humans can suggest directions and ideas for the proofs and LLM yields ⟦Partial Proofs γ⟧⟪E⟫. The ARS checks it and the LLM reports the ⟦Partial Proofs δ⟧⟪F⟫. Finally, humans verify the proof. If incorrect⟪G⟫ , it reverts to ⟦Partial Proofs α⟧; if correct⟪H⟫, it becomes the ⟦Full Proofs⟧.

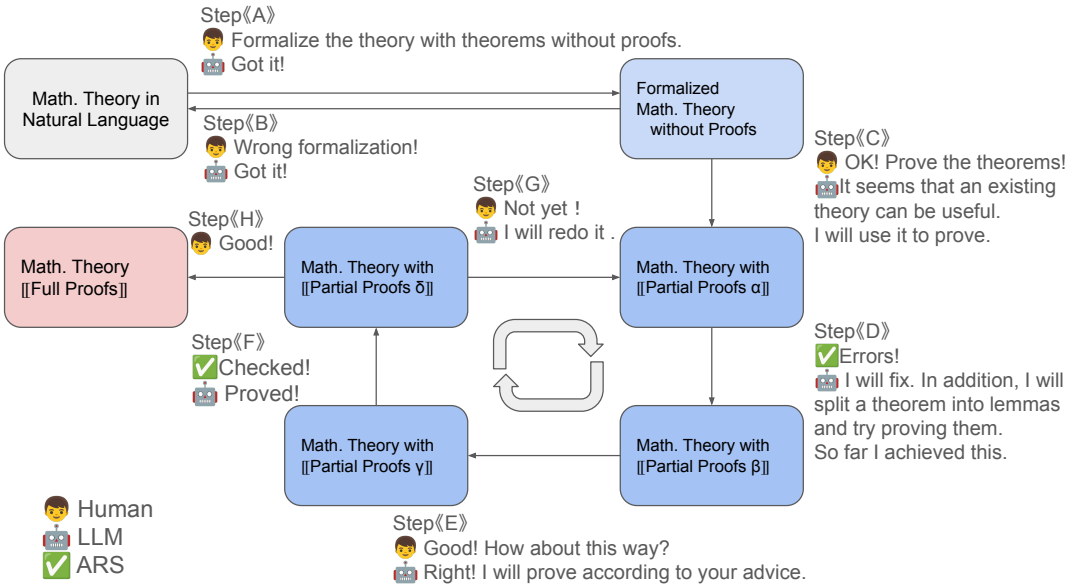


Figure 1: Human-LLM-ARS Proving Cycle

This collaborative process is termed 'Proof Engineering', that is, for example, defined in [4]. The core tasks of proof engineering now encompass the following:

- Environment and System Setup: This involves selecting a suitable LLM or combination of LLMs, and carefully designing the collaboration system. This includes:
  - Prompt Engineering: Crafting effective prompts, especially elaborating on system prompts.
  - Context Engineering: Managing the relevant context provided to the LLM.
  - Harness Engineering: Building a robust system for the collaboration cycle.
- Formal Theory Development: Using LLMs as aids to formally articulate and refine mathematical theories within rigorous languages.
- Theorem Proving and Refinement: Rigorously proving theorems and enhancing the proofs towards more elegance.
- Human Comprehension Transformation: Converting the rigorous formal language description into a style more easily understood by humans, often utilizing well-designed visual aids like pictures and diagrams, in particular also our idea of logico-graphic symbols [5].

During exploring a theory by driving a Human-LLM-ARS proving cycle, we observed the following pitfall. Although the proof parts of theories are automatically generated by LLMs and checked by ARS without errors, the generated formalized definitions and theorems may tacitly include some properties (e.g. commutativity) which have to be proved explicitly.

The main contribution of this talk is to report on a concrete instance of this pitfall for the case of a light puzzle example theory posed in a workshop TPP2025 [6]. One can play the puzzle by smart phones in https://www.rigordrive.fun/lab/scml2026, and access via QR-code next to the title of this paper. In addition, we will give the results of experimenting with the Human-LLM-ARS proving cycle for a new solution of Nakano's problem given in [3] that avoids the above pitfall.

References

[1] Jeremy Avigad. Mathematics in the Age of AI. LMS/BCS-FACS Seminar, London Mathematical Society. https://www.lms.ac.uk/events/lms-bcs-facs-seminar-jeremy-avigad, Nov. 2025. https://youtu.be/2fxXPq8SSkQ.

[2] Jeremy Avigad. Mathematicians in the Age of AI. arXiv preprint, https://arxiv.org/abs/2603.03684, 2026.

[3] Bruno Buchberger. Nakano's Light Puzzle: A Correctness Proof for Segawa's Algorithm. RISC Technical Note 26-05, Research Institute for Symbolic Computation, Johannes Kepler University, 2026. https://epub.jku.at/doi/10.35011/risc.26-05.

[4] Chelsea L. Edmonds. A Proof Engineering Perspective on Formalising Combinatorics in Isabelle/HOL. Workshop on AI and Theorem Provers in Mathematics (AITPM), University of Exeter, April 2026. Apr. 8–10. https://aitpm.github.io. Slides: https://aitpm.github.io/slides/Edmonds.pdf.

[5] Koji Nakagawa. Logicographic Symbols. *Journal of Symbolic Computation*, 41(3–4):411–434, 2006. https://doi.org/10.1016/j.jsc.2005.02.005.

[6] Keisuke Nakano. TPPmark 2025. TPP 2025: 21st Theorem Proving and Provers Meeting, Tohoku University, December 2025. Dec. 3–4. https://tpp2025.blogspot.com/2025/09/tppmark-2025.html.

Representing Polynomial Ideals as Heterogeneous Graphs for Inductive Machine Learning

Rüdiger Nather[1]

University of Kassel, Hesse, Germany
r.nather@uni-kassel.de

Abstract

We present a construction that faithfully encodes a generating set of a polynomial ideal as a heterogeneous attributed graph, enabling machine learning on ideals in a manner that is information-preserving with respect to the polynomial ring. The construction places the ambient ring, its variables, their ordering, and the term order, explicitly inside the graph, rather than treating these as fixed global parameters, thereby enabling training a single Graph Neural Network on ideals from different rings.

1 Introduction

The application of machine learning to computational commutative algebra has attracted growing interest [1–3]. Existing approaches have demonstrated that learned heuristics can meaningfully accelerate Buchberger's algorithm, but many of these approaches share a common structural limitation: the polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$ is typically fixed at training time [3]. The number of variables n and the term order $<$ are treated as external to the model, and representations of polynomials become inherently ring-dependent. A model trained on $\mathbb{K}[x_1, \ldots, x_n]$ cannot be directly applied to $\mathbb{K}[x_1, \ldots, x_{n'}]$, for $n \neq n'$, without retraining.

This limitation is not merely inconvenient. It means that the substantial computational cost of both generating training data, and training models, cannot be amortized across different rings, and that generalisation across rings is formally unavailable to the learned models. We address this by proposing a graph representation that encodes the ring itself as part of the input. Our construction enables applying common Graph Neural Network (GNN) models (e.g., Zhang [5]), to generating sets without flattening the underlying algebraic structures.

2 Construction

Let $R = \mathbb{K}[x_1, \ldots, x_n]$ be a polynomial ring with a fixed term order $\prec$, and let $F = \{f_1, \ldots, f_m\} \subset R$ be a generating set of an ideal $I = \langle F \rangle$. We construct a heterogeneous graph $G = (V, E, \mathcal{T}_V, \mathcal{T}_E)$, where V denotes the set of nodes, $E \subseteq V \times V$ represents the (multi)set of edges, $\mathcal{T}_V = \{\textsc{Var}, \textsc{Mon}, \textsc{Poly}\}$ and $\mathcal{T}_E = \{\textsc{TermOrder}, \textsc{VarOrder}, \textsc{MonEdge}, \textsc{PolyEdge}\}$ are the sets of node types and edge types respectively. Note that we allow nodes and edges to additionally carry attributes (see also [5]).

Variables and term order: For each variable x_i, introduce a vertex v_i of type Var. To represent the term order, we use that every term order on R can be represented as a matrix order via a matrix $M \in \mathbb{R}^{n \times n}$ (see [4]). Therefore, we encode the term order as a weighted complete directed graph on the variable vertices as follows: For each pair (i, j), introduce a directed edge (v_i, v_j, M_{ij}) of type TermOrder. Additionally, introduce a directed path $(v_1, v_2), \ldots, (v_{n-1}, v_n)$ of VarOrder-edges to encode the variable ordering explicitly. This "spine" is necessary: consider lexicographic order, whose matrix is the identity regardless of variable permutation, yet different variable orderings yield different elimination orders and thus genuinely different rings.

Monomials: For a monomial $\mathbf{x}^\mu$, introduce a vertex u_μ of type Mon. For each variable x_i with $\mu_i > 0$, introduce a directed edge (u_μ, v_i, μ_i) of type MonEdge, carrying the degree of x_i in $\mathbf{x}^\mu$ as its attribute. The constant monomial $\mathbf{x}^0$ is represented by a Mon-vertex without MonEdges, requiring no special case. Note that distinct occurrences of the same monomial $\mathbf{x}^\mu$, appearing in different generators f and g, receive distinct vertices. This is a deliberate choice: the same monomial may be the leading term in one generator and a lower term in another, and should therefore be permitted to be treated differently by any model trained on the graph.

Polynomials: For each generator $f_j = \sum_\mu \alpha_\mu \mathbf{x}^\mu$, introduce a vertex w_j of type Poly. For each monomial vertex u_μ belonging to f_j, introduce an edge $(w_j, u_\mu, \varphi(\alpha_\mu))$ of type PolyEdge, where $\varphi : \mathbb{K} \to \mathbb{R}^d$ is a suitable encoding for the field $\mathbb{K}$. E.g., for $\mathbb{K} = \mathbb{Q}$, one practical choice would be to encode numerator and denominator separately. In settings where the coefficients are not relevant, this attribute may be omitted without significant loss.

3 Discussion and Future work

A central property of the construction is its injectivity (up to coefficient encoding): given G, one can recover n, the term order $\prec$, and the generating set F. The variable vertices and VarOrder spine recover n and the variable ordering; the TermOrder edge weights recover M and hence $\prec$; the MonEdge weights recover each monomial; and the PolyEdge structure recovers the generating set. No information present in F as a generating set of an ideal in R is lost. Furthermore, since the polynomial ring R is encoded in the graph structure rather than as a fixed architectural parameter, a model trained on generating sets in rings with $n \leq \tilde{n}$ variables is structurally applicable to generating sets in rings with $n > \tilde{n}$ variables.

The construction opens several concrete research directions. Most importantly, the central empirical claim that a model trained on rings with few variables yields useful predictions on rings with many variables remains to be validated. We conjecture that the structural patterns underlying, for instance, the Hilbert function or the leading ideal are sufficiently local in the graph representation to transfer; an implementation is in progress and experimental results will be reported subsequently. Next, with a canonical graph representation in hand, one can now ask which GNN architectures are best suited to which algebraic tasks. Predicting the degree of a Gröbner basis, selecting reduction pairs in Buchberger's algorithm, and detecting membership in special classes of ideals likely reward different inductive biases. Furthermore, one can consider ideal-invariant pretraining schemes. As the same ideal I admits many generating sets F, each yielding a different graph G, a pretraining objective that encourages identical representations for graphs arising from different generating sets of the same ideal could provide a powerful initialisation for downstream tasks, analogous to data augmentation in geometric deep learning.

References

[1] J. Kavitha, V.S. Triveni, S. Aruna, Shilpi Singhal, N. Vithya, and G. Geetha Ramani. Graph neural networks in the study of algebraic and topological structures. In *IEEE ICTBIG*, 2025.

[2] H. Kera, Y. Ishihara, Y. Kambe, TT Vaccon, and K. Yokoyama. Learning to compute gröbner bases. In *Advances in Neural Information Processing Systems*, 2024.

[3] D. Peifer, M. Stillman, and D. Halpern-Leistner. Learning selection strategies in buchberger's algorithm. In *International Conference on Machine Learning*. PMLR, 2020.

[4] L. Robbiano. Term orderings on the polynomial ring. In *European Conference on Computer Algebra*. Springer, 1985.

[5] C. Zhang, D. Song, C. Huang, A. Swami, and N.V. Chawla. Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD*, 2019.

Reasoning over Legal Texts Using Large Language Models and Automated Reasoning*

Verena Praher[1], Endre Szasz-Revai[2], and Wolfgang Windsteiger[3]

[1] RISC Software GmbH
verena.praher@risc-software.at
[2] Rise2Reality Software
Endre.Szasz-Revai@rise2reality.com
[3] Johannes Kepler University Linz (JKU)
Research Institute for Symbolic Computation (RISC)
Linz, Austria
Wolfgang.Windsteiger@risc.jku.at

Abstract

This presentation describes an ongoing project in the area of tax law. The goal of the project is to bring computer-support to the decision process regarding taxation for international businesses. In particular, in a first prototype, the focus lies on *transfer pricing*, which is a non-trivial price-determination and taxation process that concerns businesses with divisions distributed over different countries or economies. We describe the ideas pursued in the project, we do not yet have results nor can we report on their critical evaluation by practitioners.

1 The Problem and the Requirements

Transfer pricing is the process of calculating certain prices based on laws and regulations from various sources, e.g., EU, OECD, federal laws, but also company-internal guidelines. The decision, which combination of rules finally determines the prices, is non-trivial for human experts and should be supported by software. The main requirements for the system are:

- *Trustworthiness*: the user must rely on logically sound reasoning.
- *Explainability*: the user should be able to track from which sources a decision is derived.
- *Scalability*: it should be capable of dealing with huge datasets of legal and non-legal texts.

2 The Proposed Design of the Software Prototype

The proposed design combines Large Language Models (LLM) with Symbolic Reasoning. Regulatory texts are formulated in natural language whereas reasoning tools are based on some sort of logic language. In order to make these texts accessible to logical reasoning it requires the formalization of all norms in the language of the prover. This can be seen as a translation from one language into another, and this is where LLMs demonstrate strong capability. The target language is essentially predicate logic. It will become clearer after the first examples whether a subset will be sufficient or certain generalizations are required. In order to guarantee sound

*This work is supported by the Austrian Research Promotion Agency (FFG) in the frame of project "TaxoLogic: Explainable Hybrid Artificial Intelligence for Transfer Pricing", project no. FO999936923/66571492.

application of the norms, the reasoning itself will be purely symbolic without any assistance from an LLM. However, turning the symbolic proof into a chain of logical arguments comprehensible for a domain expert needs another layer of translation. Depending on the reasoning tools in use this might be done by an LLM as well.

3 Symbolic Methods & Machine Learning Techniques

In this approach, we do not enhance LLMs by symbolic computation nor do we empower symbolic algorithms through machine learning. We rather combine the two worlds on the "problem level", i.e., we split the real-world problem into pieces that can then be tackled by symbolic and machine learning methods separately.

Machine Learning Techniques The machine learning components are modeled as a pipeline to bridge the gap between natural and formal language. This pipeline should prepare and translate unstructured legal norms into predicate logic expressions. The LLMs have to be capable of translating legal norms into predicate logic. We will benchmark larger models with explicit reasoning capabilities and smaller models that are specifically fine-tuned on tax-related legal norms, evaluating both extraction quality and suitability for on-premise deployment to ensure data sovereignty.

In the first pipeline step we apply information extraction techniques to structure the input and try to build an ontology or knowledge graph of the terminology and relationships between concepts. We intend to use named entity recognition to identify concepts and context engineering to segment text into manageable parts. This intermediate form should reduce ambiguity and allow us to control semantics for the translation.

In a subsequent step, the LLMs translate the structured natural language into predicate logic expressions. To ensure well-formed outputs, constraints are enforced during generation, e.g., by restricting the output format to valid logical expressions using guardrails. Since this translation is inherently imperfect, the resulting formulas must be reviewed and corrected. To guarantee trustworthiness, we employ a human-in-the-loop verification. Experts review and correct the proposed expressions to ensure that the legal norms are correct before handing them over to the symbolic reasoning engine.

Symbolic Methods One of the challenges for formal reasoning based on propositions extracted from laws and guidelines is their inherent inconsistency. When building up the body of formulas we try to identify contradictions and, if possible, resolve them by consulting available court-rulings. In addition, since certain contradictions might not be detected a-priori, we need to restrict the reasoning engine such that it would not apply certain standard inference rules based on the law of contradiction, e.g., any goal is proven as soon as the knowledge base contains a contradiction.

For a prototype we base the reasoning engine on the Theorema system because Theorema by design explains all reasoning steps in a human-comprehensible form. In addition, we have access to the implementation of the inference mechanism so that we can easily adapt to special requirements coming from the application domain. Finally, another advantage is that the Theorema language is untyped and the system is not tied to a certain underlying logic. In its standard form it supports a certain variant of higher-order predicate logic, but it is possible with minimum effort to include parts of modal or deontic logic into the language and add appropriate reasoning capabilities, if this is indicated by the domain of tax law.

On the Rapid Prototyping of a Logical Agent

Wolfgang Schreiner

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University Linz, Austria
Wolfgang.Schreiner@risc.jku.at

We report on our experience with the rapid prototyping of an "agentic AI" that helps a human to answer logical questions in a trustworthy way [4]. This agent combines a Large Language Model (LLM) (which interacts with the human in natural language) with a logical software (which automatically proves formal theorems). The LLM engages in a dialogue with the human in order to translate their logical question from natural language to a formal proof problem. Once the human is satisfied with the formalization, the LLM invokes the prover to automatically solve the problem and thus answer the question; then the LLM also offers the user the possibility to inspect the successful proof or the unsuccessful proof attempt by calling the prover in an interactive mode. Furthermore, we describe how much of the source code (which is based on on the agent construction framework LangChain [1]) has been "vibe coded", i.e., itself generated with the help of an LLM.

This agent employs some state-of-the-art commercial LLMs of various vendors and our own logical software, the RISCTP theorem proving interface [3]; we have previously developed RISCTP to complement the model checking capabilities of the formal specification and verification system RISCAL [2] with theorem proving capabilities.

Our basic idea for the logical agent has been as follows (see also Figure 1):

1. A formal language "FOL-PRE" for describing proof problems is defined in a document that includes examples which explain how logical problems in natural language can be translated to FOL-PRE syntax; this document is made this accessible in the web. We deliberately used a new notation for describing proof problems, rather than an existing

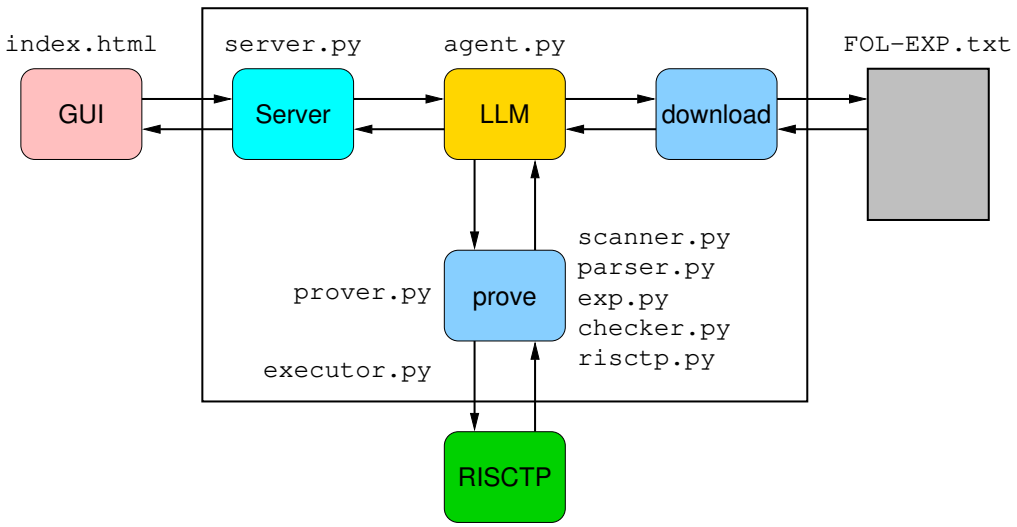


Figure 1: The Logical Agent

notation such as SMT-LIB or the language of RISCTP itself, in order to investigate how well the LLM interprets descriptions of novel languages that it has not been previously trained with (although clearly the LLM has seen many examples of first-order logic in numerous similar, but not identical, notations).

2. A tool "prove" accepts a proof problem in the FOL-PRE format, checks its syntactic correctness and semantic consistency, translates it into the RISCTP language and sends this translation to an external process that executes the RISCTP software. The tool returns as a result the output of RISCTP, i.e., the confirmation of the construction of a successful proof or the report of the failure to find such a proof, respectively. The tool may be also called in an interactive mode where RISCTP opens a GUI in which the proof (attempt) may be inspected (and potentially completed with the help of the user).

3. A system prompt instructs the agent to respond to a logical question of the human by downloading (via an auxiliary "download" tool) the language definition of FOL-PRE from the web and studying its content to translate the question of the human to a formal proof problems in FOL-PRE notation. The agent then presents the translation to the human who may ask for corrections or additions. Once the human is satisfied with the formal description, the agent forwards it to the "prove" tool in its automatic mode in order to solve the problem; based on the outcome of the tool, the agent produces its answer to the question. The agent then also offers to invoke the prover in its interactive mode in order to allow the human to inspect the successful proof or the failed proof attempt, respectively.

The logical agent is equipped with a simple chatbot interface that allows the user to interact with the LLM to generate the formal proof problem and inspect it before it is ultimately submitted to the external prover.

While the agent prototype has been developed with very limited effort (in about two weeks), first experiments have demonstrated its usefulness for constructing first-order proof problems from natural language descriptions as arise in teaching introductory classes on formal logic. This software development experiment also demonstrates how with the help of modern agent frameworks (such as LangChain) quite easily existing symbolic computation software can be integrated with LLMs; thus it may serve as a blueprint for other applications in this domain.

However, the experiment has been very limited in its scope. Further work is needed to address many open questions, in particular how the translation be scaled to more complex problems that exceed the capabilities of the "few-shot" prompting technique exhibited by the examples given in the FOL-PRE document, e.g., by applying the "Three-of-Thought" paradigm.

References

[1] LangChain: Observe, Evaluate, and Deploy Reliable AI Agents. LangChain, Inc., San Francisco, CA, USA, 2026. https://www.langchain.com.

[2] The RISC Algorithm Language (RISCAL). Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, 2019. https://www.risc.jku.at/research/formal/software/RISCAL.

[3] The RISCTP Theorem Proving Interface. Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria, 2024. https://www.risc.jku.at/research/formal/software/RISCTP.

[4] Wolfgang Schreiner. Building a Logical Agent with LangChain ... and Quite Some Vibe Coding. Technical Report 26-02, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz, Austria, March 2026. doi:10.35011/risc.26-02.

Exact Algebraic Computation of Learning Coefficients for Two-Dimensional Singular Models

Gregoire Sergeant-Perthuis, Jules Tsukahara, and Elias Tsigaridas

Abstract

Given a statistical model $\mathcal{M}$ with parameter space $\Theta \subseteq \mathbb{R}^d$ and likelihood L, the Bayesian Information Criterion (BIC) [9], $\mathrm{BIC}(\mathcal{M}) = -\log(\max_{\theta\in\Theta} L(\theta)) + \frac{d}{2}\log n$, approximates the log marginal likelihood under regularity assumptions on the model. These assumptions—identifiability and positive-definite Fisher information at the MLE—are generically violated by overparametrized architectures such as deep neural networks [13, 12], making parameter count a poor proxy for effective complexity and BIC-based selection inconsistent [2]. In Singular Learning Theory [12], the correct asymptotic expansion is the Widely Applicable BIC, $\mathrm{WBIC}(\mathcal{M}) = -\log(\max_{\theta\in\Theta} L(\theta)) + \lambda \log n$, where the *learning coefficient* $\lambda \leq d/2$ coincides, when the model is analytic, with the *real log canonical threshold* (RLCT) of the Kullback–Leibler divergence of $\mathcal{M}$. The RLCT is a birational invariant of the singularities of f and can be much smaller than $d/2$. Computing it generally requires a resolution of singularities, whose existence is guaranteed by Hironaka's theorem [3] but whose computation is notoriously expensive. We focus on the bivariate case: the local RLCT at the origin of a polynomial $f \in \mathbb{Q}[x, y]$. Following [11, 8, 1], we express $\mathrm{RLCT}_0(f)$ via the Newton polygon $\mathcal{N}(f) \subseteq \mathbb{N}^2$. The *Newton distance* $\delta_f \in \mathbb{Q}_{\geq 0}$ is the value at which the diagonal $\{\alpha = \beta\}$ meets $\partial\mathcal{N}(f)$. Two power series are *right equivalent* if related by an automorphism of $\mathbb{K}[[x, y]]$; right equivalence preserves the RLCT. The polynomial f is *normalized* when, on every facet Δ_i of $\mathcal{N}(f)$, every root of the facet polynomial $F_{\Delta_i}(z)$ has multiplicity at most δ_f; in that case $\mathrm{RLCT}_0(f) = 1/\delta_f$.

Proposition 0.1 ([8, 1])**.** *Any $f \in \mathbb{K}\{x, y\}$ is right equivalent to a normalized power series via a change of variable $(x, y) \mapsto (x, y - P_f(x))$ or $(x, y) \mapsto (x - Q_f(y), y)$, with P_f, Q_f of strictly positive order. The series P_f is unique, and lies in $\mathbb{Q}[[x]]$ whenever $f \in \mathbb{Q}[x, y]$.*

We call P_f the *normalizing power series* of f. It is either the singular part of a Puiseux y-root of f—hence a polynomial—or a y-root of f lying in $\bar{\mathbb{Q}}[[x]] \setminus \mathbb{Q}[x]$, with infinitely many terms. The Newton–Puiseux algorithm computes P_f term by term but is not effective in the second case, and prior algorithmic treatments [7] share this limitation and carry no complexity bound. We show that to compute $\mathrm{RLCT}_0(f)$ it suffices to know P_f up to a *finite*, computable degree.

Definition 0.2 (Finite part)**.** Let $P_f \in \mathbb{Q}[[x]]$ be the normalizing power series of $f \in \mathbb{Q}[x, y]$. If P_f equals a y-root ϕ of f, the *finite part* $P_f^{\mathrm{fin}}(x)$ is the singular part of ϕ (its truncation at the regularity index, beyond which it agrees with no other y-root of f). Otherwise, $P_f^{\mathrm{fin}}(x) \coloneqq P_f \in \mathbb{Q}[x]$.

Our first result makes Varchenko's approach effective:

Theorem 0.3. *Let $f \in \mathbb{Q}[x, y]$ with normalizing series P_f and finite part $P_f^{fin}(x)$. Set $\tilde{f}(x, y) = f(x, y - P_f^{fin}(x))$, and let Δ be the main face of $\mathcal{N}(\tilde{f})$. If $\tilde{f}$ is normalized, then $\mathrm{RLCT}_0(f) = 1/\delta_{\tilde{f}}$. Otherwise, $\mathrm{RLCT}_0(f) = 1/m$, where $m > \delta_f$ is the largest multiplicity of a root of the facet polynomial $\tilde{F}_\Delta(z)$.*

Our second result bounds the degree of $P_f^{\mathrm{fin}}(x)$, and hence the number of substitutions needed. We obtain this bound by generalizing a univariate root separation bound due to [10] to the bivariate case.

Theorem 0.4. *Let $f \in \mathbb{Q}[x, y]$ with $\deg_x f = d_x$ and $\deg_y f = d_y$. Then the degree d of $P_f^{fin}(x)$ satisfies*

$$d < (d_y + \tfrac{1}{2})\, d_x.$$

Combining Theorems 0.3 and 0.4 gives a terminating, effective algorithm (`ComputeRLCT2D`) that iteratively constructs $P_f^{\mathrm{fin}}(x)$ via $\lceil (d_y + \frac{1}{2})d_x \rceil$ rational changes of variables, each requiring only a square-free factorization of a univariate polynomial over $\mathbb{Q}$. The algorithm has been implemented in `SageMath` and returns the exact value of $\mathrm{RLCT}_0(f) \in \mathbb{Q}$ for any input.

Theorem 0.5 (Correctness and complexity)**.** *For every $f \in \mathbb{Q}[x, y]$, `ComputeRLCT2D` returns $\mathrm{RLCT}_0(f)$ in fewer than $(d_y + \frac{1}{2})d_x$ iterations.*

Finally, we illustrate the algorithm on biparametric polynomial neural networks [4] of depth L and activation degree r with a repeated weight, and offer a comparison with sampling-based approaches to learning coefficient approximation [5]. Our algorithm is the first deterministic, terminating procedure for computing exact local RLCTs of arbitrary bivariate rational polynomials, with an explicit $O(d_x d_y)$ iteration bound. Being model-agnostic—requiring only contact equivalence between the Kullback–Leibler divergence and a polynomial—it broadens the class of two-dimensional models for which the learning coefficient is computable in closed form. Our work falls within a growing trend of works using algebraic geometry and symbolic computation to understand machine learning models [6].

References

[1] Tristan C. Collins. Log-canonical thresholds in real and complex dimension 2. *Annales de l'Institut Fourier*, 68(7):2883–2900, 2018.

[2] Mathias Drton and Martyn Plummer. A bayesian information criterion for singular models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(2):323–380, 2017.

[3] Heisuke Hironaka. Resolution of singularities of an algebraic variety over a field of characteristic zero: 2. *Annals of Mathematics*, 79(2):205–326, 1964.

[4] Joe Kileel, Matthew Trager, and Joan Bruna. On the expressive power of deep polynomial neural networks. *Advances in neural information processing systems*, 32, 2019.

[5] Edmund Lau, Zach Furman, George Wang, Daniel Murfet, and **Wei, Susan**. The local learning coefficient: A singularity-aware complexity measure. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[6] Giovanni Luca Marchetti, Vahid Shahverdi, Stefano Mereta, Matthew Trager, and Kathlén Kohn. Algebra unveils deep learning–an invitation to neuroalgebraic geometry. *arXiv preprint arXiv:2501.18915*, 2025.

[7] Erik Paemurru. Reading the log canonical threshold of a plane curve singularity from its newton polyhedron. *ANNALI DELL'UNIVERSITA'DI FERRARA*, pages 1–14, 2024.

[8] Duong H Phong, Elias M Stein, and Jacob A Sturm. On the growth and stability of real-analytic functions. *American Journal of Mathematics*, 121(3):519–554, 1999.

[9] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[10] Elias P Tsigaridas and Ioannis Z Emiris. On the complexity of real root isolation using continued fractions. *Theoretical Computer Science*, 392(1-3):158–173, 2008.

[11] Alexander N Varchenko. Newton polyhedra and estimation of oscillating integrals. *Functional analysis and its applications*, 10(3):175–196, 1976.

[12] Sumio Watanabe. *Algebraic geometry and statistical learning theory*, volume 25. Cambridge university press, 2009.

[13] Susan Wei, Daniel Murfet, Mingming Gong, Hui Li, Jesse Gell-Redman, and Thomas Quella. Deep learning is singular, and that's good. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Unifying Structure for Ramanujan's 17 Series for $1/\pi$: A Human-AI Discovery

Michael Shalyt, Elyasheev Leibtag, Shachar Weinbaum, and Ido Kaminer

Technion - Israel Institute of Technology, Haifa 3200003, Israel

Advances in AI are rapidly transforming mathematical research. In this work, we present a human-AI discovery case study, proposing a hybrid methodology built on AI $\rightarrow$ symbolic computation $\rightarrow$ human mathematician iterations. We apply this methodology to a long-standing challenge: Ramanujan's 17 remarkable series for $1/\pi$, published in 1914, some converging at extraordinary rate [1]. These formulas - later extended by the Borweins and the Chudnovsky brothers - underpin modern computation of π [2]. Despite a century of study, a unified structural explanation for *all* 17 series has remained elusive. We give a new, elementary proof that all 17 formulas arise from a single algebraic template: a Conservative Matrix Field (CMF) [3], where varying just a few parameters recovers each of Ramanujan's formulas.

In essence, a CMF can be viewed as a discrete geometric space in which each linear trajectory (a ray) encodes a recurrence relation. Combined with specific initial conditions, this structure yields formulas for mathematical constants. A CMF can be defined by a D-finite generating function (and its contiguous relations) and a small set of parameters [4] - together determining the constants appearing in the limit formulas and their convergence behavior. Recently, CMFs have emerged as a powerful unifying framework, condensing hundreds of formulas for π from diverse sources and across centuries into a family of 2^{nd}-order recurrences [5]. The Ramanujan formulas, however, resisted this condensation - thus becoming the next unification goal.

We tackled this challenge through tight collaboration with a custom LLM-based assistant - the "Ramanujan Agent" - which was taught skills based on the group's papers, code, private notes, notebooks, previous experiments, and working norms: test generated code, check hypotheses

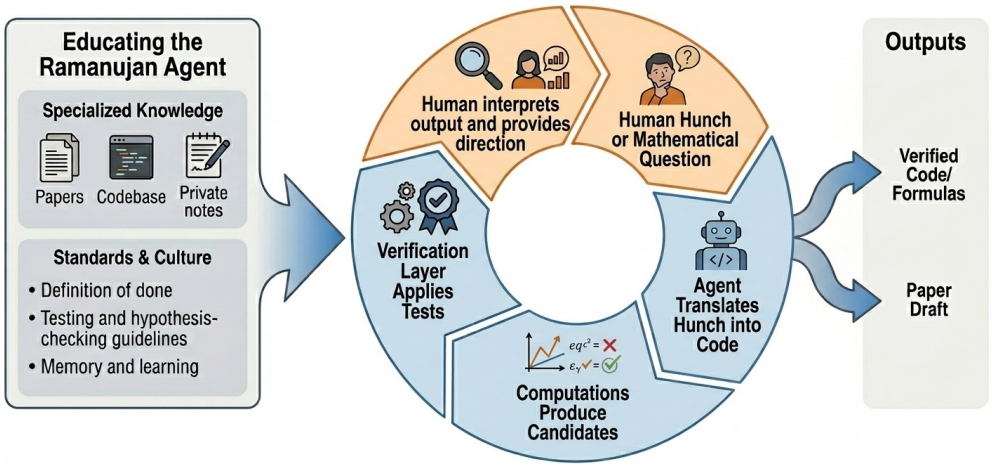


Figure 1: **Human-AI discovery loop:** A mathematician proposes a research idea or "hunch". The agent writes exploratory symbolic or numeric code, which produces result candidates (e.g. formulas, CMFs, coboundary relations). The agent then validates the candidates and the code that generated them through a sequence of tests. Finally, the validated results are returned to the human for interpretation and further directions.

symbolically or numerically, and learn from intermediate states and failures (see Fig. 1). Proficient with a variety of symbolic tools - both custom group libraries and publicly available ones - it became the de facto interface between researchers and computational experimentation. The workflow was iterative rather than prompt-based: Human questions and ideas (e.g. a candidate CMF-generating function, an equivalence between formulas) were turned into scripts, experiments, symbolic checks, and short explorations (e.g. scanning thousands of trajectories and initial conditions, profiling candidate formulas); the results were interpreted and fed into the next conjecture cycle.

The central outcome is a proven mapping between each of Ramanujan's 17 formulas and a trajectory in a CMF. Notably, every formula was shown to emerge from CMFs constructed from the same generator function - the ${}_4F_3(a_1, a_2, a_3, a_4;\, b_1, b_2, b_3;\, z)$ generalized hypergeometric function - and all lying on a single common trajectory. The formulas differ only in the starting point (a shift in the hypergeometric parameters), the evaluation point z, and the initial conditions for the generated recurrence (see Table 1). This unified pattern not only offers a plausible hint at how the formulas may have been originally discovered, but more importantly, suggests that the CMF pattern can be extended to yield even more efficient formulas for π.

Table 1: Mapping of Ramanujan's formulas to CMF trajectories. The initial point is $I = (1, a_2, a_3, a_4; 1, 1, 1)$, determined by the shift. A, B are recurrence initial conditions.

#	formula	shift (a_2, a_3, a_4)	z_0	A	B
1	$\frac{4}{\pi} = \sum_{n\geq 0}(6n+1)\binom{2n}{n}^3 \frac{1}{2^{6n}}$	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\frac{1}{4}$	1	6
2	$\frac{16}{\pi} = \sum_{n\geq 0}(42n+5)\binom{2n}{n}^3 \frac{1}{2^{12n}}$	$(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$	$\frac{1}{2^6}$	5	42
	...				
16	$\frac{2}{\pi\sqrt{11}} = \sum_{n\geq 0}(280n+19)\binom{2n}{n}^2\binom{4n}{2n} \frac{1}{2^{8n}99^{2n+1}}$	$(\frac{1}{4}, \frac{1}{2}, \frac{3}{4})$	$\frac{1}{99^2}$	19	280
17	$\frac{1}{2\pi\sqrt{2}} = \sum_{n\geq 0}(26390n+1103)\binom{2n}{n}^2\binom{4n}{2n} \frac{1}{2^{8n}99^{4n+2}}$	$(\frac{1}{4}, \frac{1}{2}, \frac{3}{4})$	$\frac{1}{99^4}$	1103	26390

These results connect to the RISC catalogue of Ramanujan-type series [6], offering additional unification targets and further motivation for CMF exploration. As a case study in hybrid mathematical discovery, the project demonstrates an efficient division of labor: the human supplies problem selection and interpretation, while the agent supplies rapid experimentation and symbolic checking - lowering the cost of early-stage creativity. The initial hunch, too immature to justify heavy manual investment, became cheap to test, refine, and validate through the prepared agent and codebase - a blueprint for future research.

[1] Srinivasa Ramanujan. Modular equations and approximations to π. *Quarterly Journal of Mathematics*, 45:350–372, 1914.

[2] Jonathan M. Borwein et al. Ramanujan and computation of pi. *The American Mathematical Monthly*, 96(3):201–219, 1989.

[3] Rotem Elimelech et al. Algorithm-assisted discovery of an intrinsic order among mathematical constants. *Proceedings of the National Academy of Sciences*, 121(25):e2321440121, 2024.

[4] Shachar Weinbaum et al. On conservative matrix fields: Continuous asymptotics and arithmetic, 2025.

[5] Tomer Raz et al. From Euler to AI: Unifying formulas for mathematical constants. In *Advances in Neural Information Processing Systems*, 2025.

[6] Peter Paule et al. Ramanujan-type series. *RISC Report Series*, (25-01), 2025.

Certified CAS-assisted Polynomial Reasoning in Lean 4

Hao Shen[1], Junyu Guo[2], Junqi Liu[1], and Lihong Zhi[1]

[1] State Key Laboratory of Mathematical Sciences, Academy of Mathematics and Systems Science, University of Chinese Academy of Science, Beijing, 100190, China
{shenhao24,liujunqi}@amss.ac.cn, lzhi@mmrc.iss.ac.cn

[2] Institute of Logic and Cognition, Department of Philosophy, Sun Yat-sen University, 510275, Guangdong, China
guojy228@mail2.sysu.edu.cn

Symbolic computation and formal verification fulfill distinct but complementary roles. Symbolic computation provides efficient tools for exact algebraic manipulation, whereas formal verification ensures the correctness of proofs. A powerful mathematical assistant should be able to combine efficient computation with trustworthy reasoning.

In this work, we present a certificate-based framework for automated polynomial reasoning in Lean 4 [5] that connects symbolic computation, machine learning, and formal verification. As a foundation, we formalize core Gröbner basis theory [1, 2] in Lean 4 [3]. Rather than carrying out symbolic computation directly inside Lean, we delegate it to external computer algebra systems such as SageMath [6] and SymPy [4], and import the results back into Lean as certificates that can be verified in Lean's trusted kernel [7]. This architecture preserves soundness while achieving practical efficiency.

To support this workflow, we develop a Lean-CAS interface that extracts algebraic data from Lean expressions, serializes it for external computation, and reconstructs the returned results as proof certificates. Building on this interface, we develop three tactics for polynomial reasoning:

- `idealeq`: It proves the equality of polynomial ideals by reducing the goal to mutual inclusion.
- `gb_solve`: It serves as a unified tactic for remainder certification, Gröbner basis verification, ideal membership, and radical membership.
- `add_gb_hyp`: It computes a Gröbner basis externally, certifies it in Lean, and introduces it as a reusable hypothesis.

The system supports three interchangeable backends: local SageMath, SageMath via API, and local SymPy. The source code is available at https://github.com/WuProver/GroebnerTactic.

To further demonstrate the synergy between LLMs and formal verification, we developed a skill for Claude Code that enables it to autonomously invoke our tactics to complete algebraic proofs in Lean 4. Given a statement, Claude Code autonomously attempts to construct a proof in Lean 4. When it encounters a subgoal that involves polynomial reasoning that can be automatically discharged by our tactics, it invokes the appropriate one accordingly. This provides a concrete realization of CAS, Lean, and LLM working in concert: the LLM orchestrates the proof strategy, the CAS handles the heavy algebraic computation, and Lean's kernel guarantees the correctness of every step. The source code can be found at https://github.com/tsuki8/GBProver.

We outline two directions for future work.

- We plan to integrate further symbolic computation methods into Lean 4 beyond Gröbner bases, such as the Wu–Ritt characteristic set method and cylindrical algebraic decomposition (CAD), via the same certificate-based architecture. In the longer term, we aim to implement verified CAS algorithms natively within Lean itself.
- We envision a multi-agent proof architecture in which an LLM serves as the top-level reasoning orchestrator, invoking CAS tools for exact algebraic computation and passing results to Lean for formal verification when correctness is critical. This layered architecture would combine the flexibility of LLMs, the computational power of symbolic systems, and the trustworthiness of formal proof, realizing a scalable pipeline for automated mathematical reasoning.

References

[1] Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal (An Algorithm for Finding the Basis Elements in the Residue Class Ring Modulo a Zero Dimensional Polynomial Ideal). Ph.D. thesis, Mathematical Institute, University of Innsbruck, Austria (1965), english translation in J. of Symbolic Computation, Special Issue on Logic, Mathematics, and Computer Science: Interactions. Vol. 41, Number 3-4, Pages 475–511, 2006

[2] Buchberger, B.: Ein algorithmisches Kriterium fuer die Loesbarkeit eines algebraischen Gleichungssystems (An Algorithmic Criterion for the Solvability of Algebraic Systems of Equations). Aequationes mathematicae **3**, 374–383 (1970), (english transl.: B. Buchberger, F. Winkler: Groebner Bases and Applications, Proc. of the International Conference "33 Years of Groebner Bases", 1998, RISC, Austria, London Math. Society Lecture Note Series 251, Cambridge Univ. Press, 1998, pp.535 -545)

[3] Guo, J., Shen, H., Liu, J., Zhi, L.: Formalizing Gröbner basis theory in Lean (2026), https://arxiv.org/abs/2602.12772

[4] Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., et al.: SymPy: symbolic computing in Python. PeerJ Computer Science **3**, e103 (2017)

[5] Moura, L., Ullrich, S.: The Lean 4 theorem prover and programming language. In: Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings. p. 625–635. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-79876-5_37, https://doi.org/10.1007/978-3-030-79876-5_37

[6] Sage Developers: SageMath, the sage mathematics software system (2020)

[7] Shen, H., Guo, J., Liu, J., Zhi, L.: Automated tactics for polynomial reasoning in lean 4 (2026), https://arxiv.org/abs/2604.13514

Extended Abstract: Learning to Rank Symbolic Expressions for Model Selection

Ali Soltani[1], Gabriel Kronberger[2], Fabricio Olivetti de França[3], and Alessandro Lucantonio[1]

[1] Aarhus University, Department of Mechanical Engineering, Aarhus, Denmark
asoltani@mpe.au.dk, a.lucantonio@mpe.au.dk
[2] Heuristic and Evolutionary Algorithms Laboratory (HEAL), University of Applied Sciences Upper Austria, Hagenberg, Austria gabriel.kronberger@fh-hagenberg.at
[3] Federal University of ABC, Center for Mathematics, Computing, and Cognition, Brazil
folivetti@ufabc.edu.br

Introduction Symbolic regression (SR) searches for explicit mathematical expressions that approximate relationships between input variables and a target variable, making it useful for scientific and engineering applications where interpretability is important. Genetic programming (GP) is an evolutionary method to solve SR tasks and represents expressions as trees and evolve them through variation and selection operators [1, 2]. A key challenge is that SR algorithms typically produce a set of candidate expressions rather than a single clearly optimal model. Selecting the final expression therefore requires balancing predictive accuracy, structural complexity, number of parameters, and robustness to noise. Existing approaches often use fixed criteria such as validation error, information criteria, minimum description length, or Bayesian approximations, which we compared several of such criteria in our previous work [3]. However, a single fixed rule may not provide the best trade-off across datasets, noise levels, sample sizes, and expression structures. This motivates a data-driven meta-selector that learns to rank candidate symbolic expressions according to their expected generalization quality.

Method The proposed framework consists of four stages: candidate expression generation, feature extraction, relevance label construction, and meta-selector training. We first construct a meta-dataset from symbolic regression runs on a heterogeneous collection of regression problems, including synthetic benchmark functions, engineering datasets, and selected real-world regression datasets from symbolic regression benchmark suites [4]. For each problem, a pool of candidate expressions represented as trees and fitted to the training data are generated using a genetic programming based symbolic regression algorithm. Each candidate expression is then described by structural features, performance-related features, dataset-level features, and standard model selection criteria such as Akaike information criterion (AIC), corrected-AIC (AICc), Bayesian information criterion (BIC), and minimum description length (MDL). Relevance labels are assigned according to the ranking induced by test error, with the best-generalizing expressions receiving the highest relevance. Finally, we train a gradient-boosted learning-to-rank model using normalized discounted cumulative gain (NDCG), which emphasizes placing highly relevant expressions near the top of the predicted ranking [5].

Results We test whether the proposed meta-selector improves the ranking of symbolic expressions compared with fixed model selection criteria. The main hypothesis is that a meta-selector can learn dataset-dependent trade-offs between accuracy and complexity. The designed meta-selector will be compared against standard baselines such as training error, AIC, AICc, BIC, and MDL. Ranking performance will be evaluated using NDCG, mean reciprocal rank [6], precision at k, and Spearman's rank correlation.

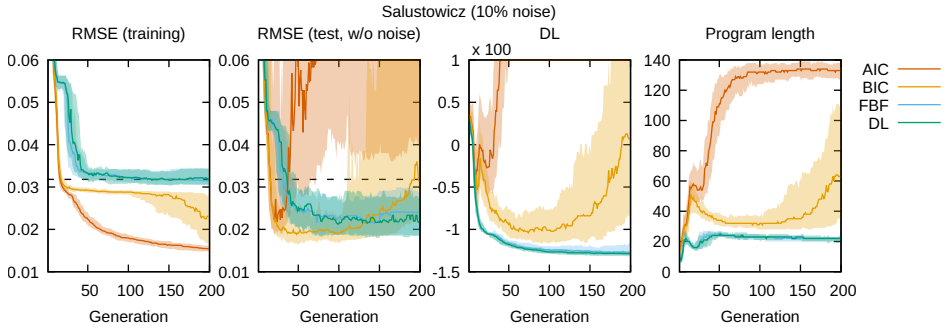


Figure 1: Plots of training and test errors as well as description length and the program length of symbolic regression models as selected by the different criteria over all generations of 100 GP runs. The plots show the median and the 25% to 75% percentiles of the values. The dashed lines indicate the noise threshold (variance of the noise component). We observe that AIC and BIC overfit, while selection by description length (DL) and fractional Bayes factor (FBF) leads to well fitting expressions. This analysis will be extended by the new learned selection criterion.

Conclusion We propose a data-driven approach for symbolic regression model selection. Instead of applying a fixed selection criterion to all candidate expressions, we formulate the problem as a learning-to-rank task in which expressions are ranked within each dataset according to expected generalization quality. By combining expression-tree structure, model performance, dataset-level information, and classical selection criteria, the proposed meta-selector aims to learn adaptive trade-offs between accuracy and complexity. We will evaluate whether this learned ranking strategy can improve expression selection across heterogeneous symbolic regression problems compared with standard criteria such as training error, AIC, AICc, BIC, and MDL.

References

[1] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[2] Gabriel Kronberger, Bogdan Burlacu, Michael Kommenda, Stephan M. Winkler, and Michael Affenzeller. *Symbolic Regression*. Chapman and Hall/CRC, 2024.

[3] Ali Soltani, Gabriel Kronberger, Fabricio Olivetti de França, Mattia Billa, and Alessandro Lucantonio. A comparative study of model selection criteria for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '26, San José, Costa Rica, 2026. Association for Computing Machinery.

[4] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance, 2021.

[5] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[6] Nick Craswell. *Mean Reciprocal Rank*, pages 1703–1703. Springer US, Boston, MA, 2009.

Learning to Compute Polynomial Products with Transformers *

Yuxuan Song and Changbo Chen

Chongqing Institute of Green and Intelligent Technology, Chinese Academy of Sciences
Chongqing, China
{songyuxuan,chenchangbo}@cigit.ac.cn

Abstract

Polynomial multiplication is one of the most basic arithmetic operations in symbolic computation. Many advanced computation, such as division and resultant calculation, boils down to it, which further establishes the foundation for building sophisticated symbolic solvers. In this work, we present a Transformer-based approach to compute the product of two polynomials. The goal here is not to accelerate polynomial multiplication. Rather, we demonstrate that learning polynomial multiplication helps in learning pseudo division, which further improves the learning of characteristic sets via a pre-training and fine-tuning paradigm.

One key insight in this work for learning polynomial multiplication is to firstly learn an intermediate collected product of polynomials. We then refine an existing scratchpad with bi-level positional coupling technique for learning the addition of multiple unsigned integers to the case of signed integers, paving the way to turn the collected product to the final product. Experiments show that this multi-stage method can predict the product of polynomial multiplication in high accuracy and significantly outperforms the direct one-step learning approach. In addition, we show by experiments that learning polynomial product helps with learning the polynomial pseudo remainder and the characteristic sets.

Main results

Three different approaches to learning polynomial multiplication are proposed, as illustrated in Fig. 1. Since polynomials can be converted into sequence of tokens, polynomial multiplication can be modelled as a sequence-to-sequence task, which fits the structure of encoder-decoder Transformer model. In method #1, Transformer model is trained to directly predict the product of two polynomials. Method #2 decomposes the multiplication process into two stages. In the first stage, an intermediate task is introduced: learning the collected product of polynomials. A collected product is defined as the product in which all coefficients of "like terms", terms sharing the same exponent vector, are collected together but not yet combined. After the collected product has been learned, a second model is trained to perform its simplification. The two models are trained independently and applied sequentially during inference, ensuring that coefficient collection and combination are handled in distinct stages.

Observe that in Method #2, the main task of the second stage is to calculate the cumulative sum of integer coefficients, which deserves special attention. In Method #3, the framework in [1] is firstly extended to train a signed integer accumulation model. Similar to Method #2, the input polynomials are first processed by a model that learns the collected product. During inference, coefficients requiring accumulation are identified and passed to the accumulation model, which computes their sum. The results are then written back into the collected product to yield the final simplified output.

*Corresponding author: Changbo Chen.

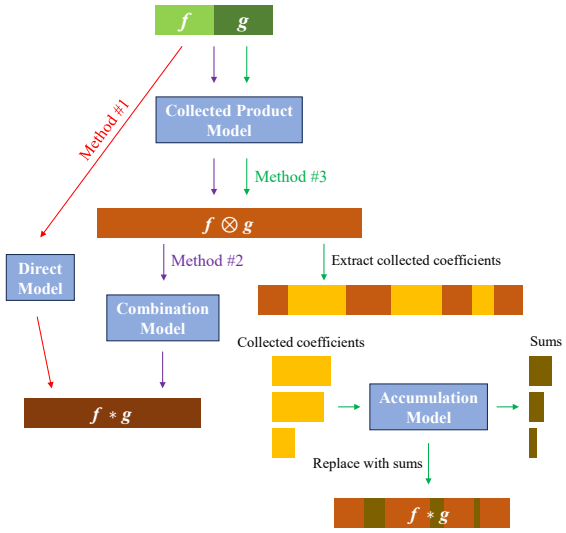


Figure 1: Procedure of different methods for learning polynomial multiplication.

Table 1, 2 and 3 respectively summarizes the experimental results on polynomial multiplication, pseudo division and characteristic set (Method 1 stands for the direct approach).

Table 1: Evaluation of learning polynomial multiplication.

Dataset	Method	TA	SA	LA	DA	NA	SPA	SSA	C1A	C2A	MA	CA
v1d3C1	1	85.4663	29.54	**100**	99.98	98.63	98.60	92.93	52.78	30.75	29.54	29.54
	2	**98.4126**	94.36	99.99	99.99	99.71	99.70	98.93	95.27	94.37	94.37	94.37
	3	95.0958	**95.06**	**100**	**100**	**99.98**	**99.98**	**99.94**	**99.91**	**99.91**	**99.91**	**99.91**
v1d7C1	1	6.9416	0	99.99	99.89	90.18	90.11	0.20	0	0	0	0
	2	6.2607	0	**100**	98.45	82.47	80.81	0.29	0	0	0	0
	3	**90.2479**	**90.14**	**100**	**99.96**	**99.94**	**99.93**	**99.70**	**99.58**	**99.57**	**99.57**	**99.57**

Table 2: Evaluation of learning polynomial pseudo division.

Dataset	Method	TA	SA	LA	DA	NA	SPA	SSA	C1A	C2A	MA	CA
v1d3C1	1	47.4343	2.26	**100**	98.49	**100**	97.42	24.95	2.95	2.39	2.26	2.26
	2	**98.9005**	**97.30**	**100**	**99.96**	**100**	**99.96**	**99.60**	**97.68**	**97.40**	**97.30**	**97.30**
v1d7C1	1	11.8721	0	**100**	98.52	**100**	91.66	1.74	0	0	0	0
	2	**32.4709**	**1.62**	**100**	**98.76**	**100**	**92.97**	**90.82**	**2.75**	**1.64**	**1.62**	**1.62**
v3t3E1C1L3	1	84.7495	65.85	**100**	98.04	**99.99**	90.18	85.55	81.29	80.77	80.71	80.49
	2	**90.2851**	**77.64**	99.98	**98.92**	99.97	**93.35**	**90.44**	**87.83**	**87.56**	**87.50**	**87.25**

Table 3: Evaluation of learning characteristic set.

v2t2e4F7	1	N/A	95.7245	70.03	99.99	97.60	**98.16**	96.42	96.42	70.14	70.13	70.13	70.03
	2	v2t2E1F7	**96.5474**	**83.13**	**100**	**97.63**	98.11	**96.90**	**96.90**	**83.42**	**83.41**	**83.41**	**83.13**
v3t3e2C1L3	1	N/A	60.4746	42.12	100	96.54	85.73	78.83	78.83	45.36	43.63	45.31	42.12
	2	v3t3E1C1L3	77.5050	62.32	100	97.63	92.69	89.28	89.28	68.30	65.81	67.21	62.32
v3t3e2F7	1	N/A	82.0582	53.45	100	96.27	84.65	79.30	79.30	54.09	54.09	54.09	53.45
	2	v3t3E1F7	84.5500	59.31	100	97.23	86.85	83.93	83.93	59.99	59.99	59.99	59.31
v3t3E1F7S2	2	v3t3E1F7	98.9971	98.80	100	100	99.15	98.80	98.80	98.80	98.80	98.80	98.80

References

[1] Hanseul Cho, Jaeyoung Cha, Srinadh Bhojanapalli, and Chulhee Yun. Arithmetic transformers can length-generalize in both operand length and count. In *Proceedings of the 13th International Conference on Learning Representations*, 2025.

2