# Proving an Execution of an Algorithm Correct?

James Davenport
masjhd@bath.ac.uk

University of Bath
Thanks to IPAM at UCLA for prompting this, and many colleagues, at CICM 2023 [Dav23] and elsehere, for input
Partially supported by EPSRC grant EP/T015713, and Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2047/1 – 390685813.

21 May 2024

# Trusting Software

It would be nice to trust software.

- Gödel's Incompleteness implies unconditional trust is essentially impossible
- Theorem provers (Lean etc.) tend to have a small kernel (ideally 100s of lines), and the rest (tactics etc.) do not need to be trusted, as an error there results in a "proof" that the kernel rejects.
- It is possible to produce programs (e.g. UK's National Air Traffic System) with 100Ks of lines, with formally proved properties.

But the proof is constructed along with the program: retrospectively proving such a program correct is probably impossible (and the program is probably incorrect — see [Cha22] for a well-tested 100-tweet program).

Also computer algebra programs are 10Ms of lines, going back 40 years or more.

# Trusting Instances of Software

So what's plan B?

- If we can't prove the program correct, can we prove that *this instance* is correct?
- This isn't as new an idea as I thought it was: [MMNS11].

  *A certifying algorithm is an algorithm that produces, with each output, a certificate or witness (easy-to-verify proof) that the particular output has not been compromised by a bug. A user of a certifying algorithm inputs $x$, receives the output $y$ and the certificate $w$, and then checks, either manually or by use of a program, that $w$ proves that $y$ is a correct output for input $x$.*

- This is particularly relevant when $y$ contains negative assertions.

# Convex Optimisation [BFMA23]

DCP = Disciplined Convex Programming [GBY06].

Figure: Bidirectional Flow, from [FM24b]

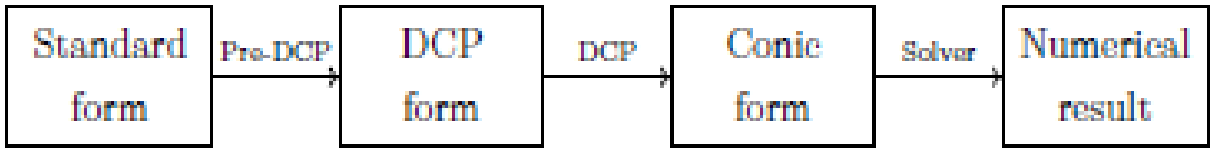


Figure 1.1: Convex programming workflow.

| | |
|---|---|
| Pre-DCP | Generally done manually (symbolically). |
| DCP | There is (generally unverified) symbolic translation. |
| ! | There are errors [FM24a] |
| Solver | Numerical code, unverified. |
| $\text{DCP}^{-1}$ | Same software inverts |
| $\text{Pre-DCP}^{-1}$ | Manual |

[BFMA23] have `CvxLean`, which implements the DCP, $\text{DCP}^{-1}$ phases.

# CvxLean in Convex Optimisation, after [BFMA23]

1. It applies the dcp procedure to obtain a reduced problem, `prob.reduced`, and a reduction red : Solution `prob.reduced` $\to$ Solution `prob`.
2. It carries out the translation to floats, traversing each expression and applying the registered translations.
3. It extracts the numerical data from the problem.
4. It writes an external file in the conic benchmark format.
5. It calls MOSEK and receives a status code, and a solution if MOSEK succeeds. If it is infeasible or ill-posed, we stop.
6. Otherwise, it interprets the solution so that it matches the shape of `prob.reduced`, expressed as Lean reals, resulting in an approximate solution p to `prob.reduced`. It declares a corresponding Solution to `prob.reduced`, using a placeholder for the proofs of feasibility and optimality (trust the solver).
7. It then uses the reduction from `prob` to `prod.reduced`, again reinterpreted in terms of floats, to compute an approximate solution to `prob`

# Convex Optimisation (III) [FM24b]

Figure: Bidirectional Flow, based on [FM24b]

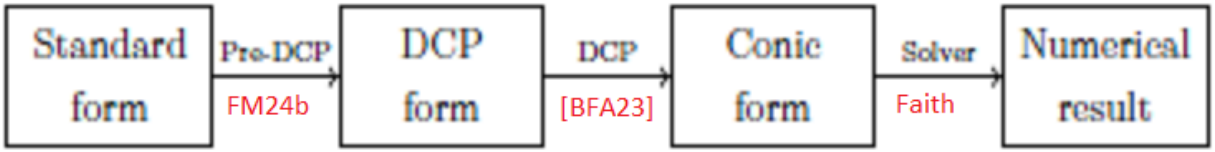


Figure 1.1: Convex programming workflow.

So what about the transformation into DCP? This is not (currently) algorithmic.

> *There are few restrictions to writing problems in [standard] form as long as they are mathematically convex. This is problematic from the point of view of automating their analysis and transformation. [FM24b, p. 5]*

Instead, a set of tactics is proposed.

# Convex Optimisation (Example) [FM24b, chapter 5]

There's a difference between "convex" and "explicitly convex"

$$\underbrace{P := \begin{array}{rl} \text{minimise} & x \\ \text{subject to} & 0.001 < x \\ \wedge & \frac{1}{\sqrt{x}} < \exp(x) \end{array}}_{\text{not convex}} \Rightarrow \underbrace{Q := \begin{array}{rl} \text{minimise} & x \\ \text{subject to} & 0.001 < x \\ \wedge & \exp(-x) < \sqrt{x} \end{array}}_{\text{convex}}$$

Because exp not concave

Note that $0.001 < x$ justifies that $\frac{1}{\sqrt{x}}$ is valid.

In general, we are given $P$ and have to find a convex $Q$, which historically has been done by hand.

# SAT solving, and verifying UNSAT

The quintessenial NP-complete problem [Coo66]: Given a Boolean statement $\Phi(x_1, \ldots, x_n)$ produce

either $f : \{x_i\} \mapsto \{T, F\}$ such that
$\Phi(f(x_1), \ldots, f(x_n)) = T$ (a satisfying assignment)

or $\perp$ indicating that no satisfying assignment exists.

The first can be verified easily enough: what about the second? Since at least 2016, contestants in the annual SAT contests have been required to produce proofs (occasionally 2PB! [Heu18]) in DRAT format, which can be checked ([Heu23] says there are subtleties to "easy" checking).

# Polynomial Factorisation

The base case is polynomials in $\mathbf{Z}[x]$.

**Problem (Factorisation)**

*Given $f \in \mathbf{Z}[x]$, write $f = \prod f_i$ where the $f_i$ are irreducible elements of $\mathbf{Z}[x_1, \ldots, x_n]$.*

Verifying that $f = \prod f_i$ is, at least relatively, easy. The hard part is verifying that the $f_i$ are *irreducible*. The author knows of no implementation of polynomial factorisation that produces any evidence, let alone a proof, of this.

**Problem (Factorisation in this style)**

*Given $f \in \mathbf{Z}[x_1, \ldots, x_n]$, produce*

*either a proper factor $g$ of $f$,*

*or $\perp$ indicating that no such $g$ exists.*

We may as well assume $f$ is square-free .

# Algorithm

Then the basic algorithm goes back to [Zas69]: step M is a later addition [Mus75], and the H' variants are also later.

1. Choose a prime $p$ (not dividing the leading coefficient of $f$) such that $f \pmod p$ is also square-free.
2. Factor $f$ modulo $p$ as $\prod f_i^{(1)} \pmod p$.

M Take five $p$ and compare the factorisations.

3. If $f$ can be shown to be irreducible from modulo $p$ factorisations, return $f$.
4. Let $B$ be such that any factor of $f$ has coefficients less than $B$ in magnitude, and $n$ such that $p^n \geq 2B$.
5. Use Hensel's Lemma to lift the factorisation to $f = \prod f_i^{(n)} \pmod{p^n}$

H Starting with singletons and working up, take subsets of the $f_i^{(n)}$, multiply them together and check whether, regarded as polynomials over $\mathbf{Z}$ with coefficients in $[-B, B]$, they divide $f$ — if they do, declare that they are irreducible factors of $f$.

# Algorithm Notes

H' Use some alternative technique, originally [LLL82], but now e.g. [ASZ00, HvHN11] to find the true factor corresponding to $f_1^{(n)}$, remove $f_1^{(n)}$ and the other $f_i^{(n)}$ corresponding to this factor, and repeat.

In practice, there are a lot of optimisations, which would greatly complicate a proof of correctness of an implementation of this algorithm.

*We found that, although the Hensel construction is basically neat and simple in theory, the fully optimised version we finally used was as nasty a piece of code to write and debug as any we have come across [MN81].*

Since if $f$ is irreducible modulo $p$, it is irreducible over the integers, the factors produced from singletons in step 5 are easily proved to be irreducible. Unfortunately, the chance that an irreducible polynomial of degree $n$ is irreducible modulo $p$ is $1/n$.

# Algorithm Notes

A factorisation algorithm could, even though no known implementation does, relatively easily produce the required information for a proof of irreducibility unless the recombination step is required.

Note that *verifying* the Hensel lifting, the "nasty piece" from [MN81] is easy: the factors just have to have the right degrees from the factorisation of $f \pmod{p}$ and multiply to give $f \pmod{p^n}$.

Building test cases for the various edge cases was extremely difficult.

Step [H] is relatively easy to verify: this combination divides and no smaller combination divides. The variants in [H'] are interesting: I have not found an easy route.

If [H'] finds a factor that is a product of $k$ $p$-adic factors, then we can use [H] to verify this by checking that the $2^k - 2$ subsets do not give factors.

But if [H'] says "irreducible", I know no easy proof.

# Further Reflections

M Take five $p$ and compare the factorisations.

Not just "take the best". Rather we look for incompatibilities, so if a degree 4 factors as 3,1 modulo one prime and 2,2 modulo another, it's actually irreducible, and so on.

? What's the best division of labour between the algebra system and the theorem prover?

[Mus75] suggests taking five primes, though more recently [LP97] show that, if the Galois group is $S_n$, seven is asymptotically right. For any degree $d$, the probability that a random polynomial with coefficients $\leq H$ has Galois group $S_n$ tends to 1 as $H$ tends to infinity. [DS00] looks at other Galois groups.

# Integration

P is algebra professor, S is awkward student

- P $e^{-x^2}$ has no integral.
- S But in analysis the professor proved that every continuous function has an integral.
- P I meant that there was no formula for the integral.
- S But in statistics the professor used $\mathrm{erf}(x)$ and everything seemed OK.
- P I meant that there was no *elementary* formula, in terms of exp, log and the solution of polynomial equations.
- S How do you prove that?
- P Differential Algebra!
- S What's that?
- P A field $K$ equipped with $' : K \to K$ such that $(a+b)' = a' + b'$ and $(ab)' = a'b + ab'$.

# Algebraic Theory of Integration [Rit48, Rit50]

Given $f \in K = \mathbf{Q}(x, \theta_1, \ldots, \theta_n)$ where $x' = 1$ and each $\theta_i$ is elementary over $\mathbf{Q}(x, \theta_1, \ldots, \theta_{i-1})$ (need *decidable* [Ric68]) produce

either $F$ in some elementary extension $L$ of $K$ such that $F' = f$ (an elementary integral)

or $\perp$ indicating that no such elementary integral exists.

The first can be verified: what about the second?

The verification isn't necessarily trivial: there are issues of simplification of elementary functions.

Because of branch cuts, $F$ might not denote a continuous function $\mathbf{R} \to \mathbf{R}$, despite the student's memory of analysis [CDJW00].

The Heaviside function differentiates to 0, so it's a "constant" in terms of differentiable algebra.

# Liouville's Principle [Lio35, Rit50]

Looking for any elementary might seem like "needle in a haystack".

## Theorem (Liouville's Principle)

*Let $f$ be a expression from some expression field $K$. If $f$ has an elementary integral over $K$, it has an integral of the following form:*

$$\int f = v_0 + \sum_{i=1}^{n} c_i \log v_i, \tag{1}$$

*where $v_0$ belongs to $K$, the $v_i$ belong to $\hat{K}$, an extension of $K$ by a finite number of constants algebraic over $\operatorname{const} K$, and the $c_i$ belong to $\hat{K}$ and are constant.*

Alternatively

$$f = v_0' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i}. \tag{2}$$

Only a single bale of hay! Proof by equating coefficients in $f = F'$.

# Risch's idea [Ris69]

$f, g \in \overline{\mathbf{Q}}(x, \theta_1, \ldots, \theta_n)$ where each $\theta_i$ is either

logarithmic $\theta_i' = \frac{u_i'}{u_i}$, $u_i \in \overline{\mathbf{Q}}(x, \theta_1, \ldots, \theta_{i-1})$.

exponential $\theta_i' = u_i' \theta_i$, $u_i \in \overline{\mathbf{Q}}(x, \theta_1, \ldots, \theta_{i-1})$.

Induct on $n$, that we can

$\int$ Solve (or $\perp$) $f = v_0' + \sum_{i=1}^{n} c_i \frac{v_i'}{v_i}$

Risch o.d.e. Solve (or $\perp$) $y' + fy = g$ for $y \in \overline{\mathbf{Q}}(x, \theta_1, \ldots, \theta_n)$.

In both cases, the algorithm is a fairly messy "comparison of terms" argument, and the Risch o.d.e. for exponential $\theta_n$ was a "similarly", which wasn't quite [Dav86].

The "mess" comes in showing that every case is covered, and that the "bug fix" in [Dav86] is complete: each individual case is fairly straightforward.

# Producing a proof of $\perp$

1. Have a formal proof of Liouville's Principle.

I haven't done this formally, but it doesn't look outrageous: it's all algebra in [Rit48].

2. At each comparison of terms, spit this out in a form that a theorem-prover can digest.

Again, I haven't done this, but I did have an implementation in Axiom which produced a (very stylised) informal proof.

Note that I am *not* considering the case of $\theta_i$ algebraic. $\theta_1$ algebraic is in [Dav81] [for $\int$], [Dav84] [for o.d.e.], but there is much more mathematics involved in finding the $c_i$, $v_i$ or proving they don't exist. More general $\theta_i$ algebraic is treated in [Bro90, Bro91], again more mathematics.

# Mathematics for a proof of $\perp$ (algebraic case)

"Mathematics" *may* reduce to "is a divisor on an elliptic curve a torsion divisor", and $\perp$ here is hard. In the special case where $C = \mathbf{Q}$ and the algebraic curve is elliptic, [Dav81] relied on the following theorem.

**Theorem ([Maz77])**

*The torsion subgroup of the Mordell–Weil groups of an elliptic curve $E$ over the rationals is isomorphic to one of the following:*

$$\begin{cases} \mathbf{Z}/m\mathbf{Z}, & m \leq 10 \\ \mathbf{Z}/12\mathbf{Z}, & \\ (\mathbf{Z}/2\mathbf{Z}) \times (\mathbf{Z}/2\nu\mathbf{Z}) & \nu \leq 4 \end{cases}.$$

*Hence the torsion of an element is one of* $1, \ldots, 10, 12$.

This is a fairly deep theorem, but one that *might* be formally provable by a specialist [Baa23]: see [BBCD23].

# Real Quantifier Elimination [Tar51, Sei54]

Let each $Q_i$ be one of the quantifiers $\forall, \exists$. Real Quantifier Elimination problem is the following: given a statement

$$\Phi_0 := Q_1 x_{1,1}, \ldots, x_{1,k_1} \cdots Q_{a+1} x_{a+1,1}, \ldots, x_{a+1,k_{a+1}} \Phi(y_i, x_{i,j}), \quad (3)$$

where $\Phi$ is a Boolean combination of equalities and inequalities between real polynomials $P_\alpha(y_i, x_{i,j})$, produce a Boolean combination $\Psi$ of equalities and inequalities between polynomials $Q_\beta(y_i)$ which is equisatisfiable, i.e. $\Psi$ is true if and only $\Phi_0$ is true. If all the polynomials $Q_\beta(y_i)$ in $\Psi(y_i)$ have integer coefficients, we call $\Psi(y_i)$ a Tarski formula.

- Proved decidable in 1950s
- First feasible solution by [Col75] through Cylindrical Algebraic Decomposition

! WebTarski relies on Collins' 1970s Fortran translated into C translated into WebAssembly

# Cylindrical Algebraic Decomposition [Col75]

Fix coordinates in $\mathbf{R}^n$ consistent with quantifier order.
Given a set of polynomials $\{p_\alpha\}$ in $\overline{Q}[x_1, \ldots, x_n]$, produce a finite set of cells $C_i \subset \mathbf{R}^n$ which is:

**Cylindrical** $\forall i, j, k \, \mathrm{Proj}_{\mathbf{R}^k}(C_i), \mathrm{Proj}_{\mathbf{R}^k}(C_j)$ are equal or disjoint;

**Algebraic** Defined by polynomials in $\overline{Q}[x_1, \ldots, x_n]$;

**Decomposition** disjoint and cover $\mathbf{R}^n$;

**Sampled** each cell has a sample point $s_i$ (cylindrical);

such that on each cell every $p_\alpha$ is sign-invariant $(+, -, 0)$.
Then the truth of $\Phi$ is invariant on a cell, and we can write down $\Psi$ as the union of those cells where $\Phi_0$ is true at the sample point.
Unfortunately QE is doubly exponential in $n$ [DH88], so CAD's worst case must be, and in practice CAD nearly always is.

# Challenges with Cylindrical Algebraic Decomposition

- CAD doesn't care about the quantifiers (other than variable order), in particular $\exists x_1, \ldots, x_n \Phi$ (the SAT problem) isn't treated as a special case.
- As formulated, it doesn't care about the Boolean structure of $\Phi$.

√ When it's $(p_1 = 0) \wedge \Phi'$ we can do better [McC99].

√ Even if this is only part of $\Phi$, we can use an equality [EBD15].

- If $f, g, h$ have degree $d$, $\mathrm{res}_y(\mathrm{res}_z(f, g), \mathrm{res}_z(f, h))$ has degree $O(d^4)$, even though there are only $O(d^3)$ common solutions $f(x, y, z) = g(x, y, z) = h(x, y, z)$.

! $f(x, y, z_1) = g(x, y, z_1)$; $f(x, y, z_2) = h(x, y, z_2)$. Note that these points *are* relevant for cylindricity in the worst case, and are used in [DH88].

- Major improvements to CAD import more mathematics, up to "Puiseux with parameters" [MPP19].
- Despite attempts [CM10], there is no formal proof of correctness of even basic Collins.

# Cylindrical Algebraic Coverings I [ADEK21a]

For purely existential problems $\exists x_k, \ldots, x_n \Phi$.

$\sigma_{i,j} \in \{=, <, \leq, >, \geq\}$, but for exposition, assume all $\sigma_{i,j} \in \{<, >\}$.

1. $\Phi = (p_{1,1}\sigma_{1,1}0 \wedge \cdots) \vee (p_{2,1}\sigma_{2,1}0 \wedge \cdots) \vee \cdots$
2. Commute $\exists$ and $\vee$ and treat each disjunct $\Phi_i$ separately

So we don't care where $p_{1,1}$ and $p_{2,1}$ meet. Doesn't change asymptotics, but may well be useful in practice.

3. Choose a sample point $(s_1, \ldots, s_n^{(1)})$.
4. If this satisfies $\Phi_i$ return SAT (and witness)
5. Otherwise $\exists j : p_{i,j}(s_1, \ldots, s_n^{(1)}) \not\sigma_{i,j} 0$. Remember $j$ with $(s_1, \ldots, s_n^{(1)})$.
6. Compute largest interval $I_{n,1} = (l, u)$ such that $\forall x_n \in (l, u) p_{i,j}(s_1, \ldots, x_n) \not\sigma_{i,j} 0$.
7. If $I_{n,1} \neq \mathbf{R}$ choose $s_n^{(2)} \notin I_1$. If $(s_1, \ldots, s_n^{(2)})$ satisfies $\Phi_i$ return SAT (and witness).
8. Repeat steps 5–7 until $(s_1, \ldots, s_{n-1}, \mathbf{R})$ is covered.

* Some intervals might be redundant, so prune

# Cylindrical Algebraic Coverings II [ADEK21a]

9. Each of $I_{n,i}$ defines an oval in $(s_1, \ldots, s_{n-2}, x, y)$ space which cover $(s_1, \ldots, s_{n-1}, \mathbf{R})$.
10. Compute largest interval $I_{n-1,1} = (l, u)$ such that $\forall x_{n-1} \in (l, u)$ the $I_{n,i}$ cover $(s_1, \ldots, s_{n-2}, x_{n-1}, \mathbf{R})$.
11. If $I_{n-1,1} \neq \mathbf{R}$ choose a different value of $s_{n-1}$, $\notin I_{n-1,1}$.
12. Repeat steps 4–11 until $(s_1, \ldots, s_{n-2}, \mathbf{R})$ is covered.
13. Repeat, decreasing the dimension, until we're covered the whole of the $x_1$-axis (or we get SAT).

Termination isn't entirely obvious, but each cell we compute contains at least one cell (the cell its sample point is in) from a CAD for the same polynomials, and the CAD itself is finite.

# How might these be verifiable? [ADE+20, ADEK21b]

This is still work in progress, and there is more than one option

A. Verifying each (non-redundant) calculation in reverse
   1. For each $I^{(1)} = (l_1, r_1)$ as an interval of $\mathbf{R}^1$ prove that it's covered because
   2. For each $I^{(2)} = (l_2, r_2)$ covering the cylinder above $I^{(1)}$ prove that $I^{(1)} \times I^{(2)}$ is covered because
   3. ...
   4. For each $I^{(n)} = (l_n, r_n)$ covering the cylinder above $I^{(1)} \times I^{(2)} \times \cdots$ prove that $I^{(1)} \times I^{(2)} \times \cdots \times I^{(n)}$ is covered by the $p_j$ we remembered for that sample point.

B Reverse-engineering a rough "CAD".
   1. For each sample point $(s_1, \ldots, s_n)$ check that the corresponding cuboid $I^{(1)} \times I^{(2)} \times \cdots I^{(n)}$ is contained within the $p_j \not\sigma_j 0$ region.
   2. Verify that these cuboids are arranged cylindrically, and are complete.

Need Resultants and inequalities, but no topology.

# Thoughts

- UNSAT, or its equivalent, can be a bigger challenge than positive answers.
- Completeness proofs of algorithms can be challenging.
- But *in some cases*, we may not need the completeness proof.
- (At least not in all cases).
- This may require more book-keeping in the algorithm, to keep the "hints" that drove us this way.
- Possibly (e.g. algebraic integration) we may not be able to prove UNSAT in all circumstances.

? is this still valuable?

# Bibliography I


E. Ábrahám, J.H. Davenport, M. England, G. Kremer, and Z.P. Tonks.
New Opportunities for the Formal Proof of Computational Real Geometry?
*SC$^2$'20: Fifth International Workshop on Satisfiability Checking and Symbolic Computation CEUR Workshop Proceedings*, 2752:178–188, 2020.

E. Ábrahám, J.H. Davenport, M. England, and G. Kremer.
Deciding the Consistency of Non-Linear Real Arithmetic Constraints with a Conflict Driven Search Using Cylindrical Algebraic Coverings.
*Journal of Logical and Algebraic Methods in Programming*, 119, 2021.

# Bibliography II



E. Ábrahám, J.H. Davenport, M. England, and G. Kremer.
Proving UNSAT in SMT: The Case of Quantifier Free Non-Linear Real Arithmetic.
http://cl-informatik.uibk.ac.at/users/swinkler/arcade2021/pdfs/1.pdf, 2021.


J.A. Abbott, V. Shoup, and P. Zimmermann.
Factorization in $\mathbf{Z}[x]$: The Searching Phase.
In C. Traverso, editor, *Proceedings ISSAC 2000*, pages 1–7, 2000.


A. Baanen.
Provability of Ogg's Conjecture.
*Personal Communication at "Machine Assisted Proofs 2023"*, 2023.

# Bibliography III



A. Baanen, A.J. Best, N. Coppola, and S.R. Dahmen.
Formalized Class Group Computations and Integral Points on Mordell Elliptic Curves.
*CPP 2023: Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 47–62, 2023.


A. Bentkamp, R. Fernández Mir, and J. Avigad.
Verified reductions for optimization.
*International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2022)*, pages 74–92, 2023.


M. Bronstein.
Integration of elementary function.
*J. Symbolic Comp.*, 9:117–173, 1990.

# Bibliography IV



M. Bronstein.
The Algebraic Risch Differential Equation.
In *Proceedings ISSAC 91*, pages 241–246, 1991.


R.M. Corless, J.H. Davenport, D.J. Jeffrey, and S.M. Watt.
According to Abramowitz and Stegun, or arccoth needn't be uncouth.
*SIGSAM Bulletin 2*, 34:58–65, 2000.


R. Chapman.
Panel session at CPHC 2022.
https://people.bath.ac.uk/masjhd/CPHC2022/Publish/Chapman-cphc_2022_panel_chapman.pptx, 2022.

# Bibliography V


C. Cohen and A. Mahboubi.
A Formal Quantifier Elimination for Algebraically Closed Fields.
In S. Autexier *et al.*, editor, *Proceedings CICM 2010*, pages 189–203, 2010.

G.E. Collins.
Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition.
In H. Brakhage, editor, *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, volume 33 of *Springer Lecture Notes in Computer Science*, pages 134–183, 1975.

S.A. Cook.
*On the minimum computation time of functions*.
PhD thesis, Department of Mathematics Harvard University, 1966.

# Bibliography VI



J.H. Davenport.
*On the Integration of Algebraic Functions*, volume 102 of *Springer Lecture Notes in Computer Science*.
Springer Berlin–Heidelberg–New York (Russian ed. MIR Moscow 1985), 1981.


J.H. Davenport.
Intégration Algorithmique des fonctions élémentairement transcendantes sur une courbe algébrique.
*Annales de l'Institut Fourier*, 34:271–276, 1984.


J.H. Davenport.
On the Risch Differential Equation Problem.
*SIAM J. Comp.*, 15:903–918, 1986.

# Bibliography VII



J.H. Davenport.
Proving an Execution of an Algorithm Correct?
In Catherine Dubois and Manfred Kerber, editors, *Proceedings CICM 2023*, volume 14101 of *Springer Lecture Notes in Computer Science*, pages 255–269, 2023.


J.H. Davenport and J. Heintz.
Real Quantifier Elimination is Doubly Exponential.
*J. Symbolic Comp.*, 5:29–35, 1988.


J.H. Davenport and G.C. Smith.
Fast recognition of alternating and symmetric groups.
*J. Pure Appl. Algebra*, 153:17–25, 2000.

# Bibliography VIII



M. England, R. Bradford, and J.H. Davenport.
Improving the Use of Equational Constraints in Cylindrical Algebraic Decomposition.
In D. Robertz, editor, *Proceedings ISSAC 2015*, pages 165–172, 2015.


R. Fernández Mir.
Different result in Aerospace Design via Quasiconvex Optimization example #2165.
`https://github.com/cvxpy/cvxpy/issues/2165`, 2024.


R. Fernández Mir.
*Verified Transformations for Convex Programming*.
PhD thesis, University of Edinburgh, 2024.

# Bibliography IX



M. Grant, S. Boyd, and Y. Ye.
Disciplined convex programming.
In *Global optimization: From theory to implementation*, pages 155–210. Springer, 2006.


M.J.H. Heule.
Schur number five.
*AAAI'18/IAAI'18/EAAI'18: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference. and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence Article No.: 808*, pages 6598–6606, 2018.


M.J. Heule.
Organising SAT contests and DRAT proofs.
*Personal comunication 15 February 2023*, 2023.

# Bibliography X



W. Hart, M. van Hoeij, and A. Novocin.
Practical polynomial factoring in polynomial time.
In *Proceedings ISSAC 2011*, pages 163–170, 2011.


J. Liouville.
Mémoire sur l'intégration d'une classe de fonctions transcendantes.
*Crelle's J.*, 13:93–118, 1835.


A.K. Lenstra, H.W. Lenstra Jun., and L. Lovász.
Factoring Polynomials with Rational Coefficients.
*Math. Ann.*, 261:515–534, 1982.


T. Łuczak and L. Pyber.
On random generation of the symmetric group.
In *Proceedings Combinatorics geometry and probability*, pages 463–470, 1997.

# Bibliography XI



B. Mazur.
Rational Points on Modular Curves.
*in Modular Functions of One Variable V*, pages 107–148, 1977.


S. McCallum.
On Projection in CAD-Based Quantifier Elimination with Equational Constraints.
In S. Dooley, editor, *Proceedings ISSAC '99*, pages 145–149, 1999.


R.M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer.
Certifying algorithms.
*Computer Science Review*, 5:119–161, 2011.


P.M.A. Moore and A.C. Norman.
Implementing a Polynomial Factorization and GCD Package.
In *Proceedings SYMSAC 81*, pages 109–116, 1981.

# Bibliography XII



S. McCallum, A. Parusiński, and L. Paunescu.
Validity proof of Lazard's method for CAD construction.
*J. Symbolic Comp.*, 92:52–69, 2019.


D.R. Musser.
Multivariate Polynomial Factorization.
*J. ACM*, 22:291–308, 1975.


D. Richardson.
Some Unsolvable Problems Involving Elementary Functions of a Real Variable.
*Journal of Symbolic Logic*, 33:514–520, 1968.


R.H. Risch.
The Problem of Integration in Finite Terms.
*Trans. A.M.S.*, 139:167–189, 1969.

# Bibliography XIII



J.F. Ritt.
*Integration in Finite Terms: Liouville's Theory of Elementary Methods*.
Columbia University Press, 1948.


J.F. Ritt.
*Differential Algebra*.
Colloquium Proceedings vol. XXXIII. American Mathematical Society, 1950.


A. Seidenberg.
A new decision method for elementary algebra.
*Ann. Math.*, 60:365–374, 1954.

# Bibliography XIV


A. Tarski.
*A Decision Method for Elementary Algebra and Geometry*.
2nd ed., Univ. Cal. Press. Reprinted in *Quantifier Elimination and Cylindrical Algebraic Decomposition* (ed. B.F. Caviness & J.R. Johnson), Springer-Verlag, Wein-New York, 1998, pp. 24–84., 1951.

H. Zassenhaus.
On Hensel Factorization I.
*J. Number Theory*, 1:291–311, 1969.