Machine-Assisted Mathematics

机器辅助数学：工作笔记

2026-08-20 • measured on this machine, not quoted from documentation

本笔记中的每个数字都是当天在本机实测，非文档摘录

1 The organizing principle 组织原则

Tools are usually sorted by what they compute. Sort them instead by **who the reader has to trust**, and the whole landscape reorganizes into four roles.

通常按" 算什么" 给工具分类。改按" 读者要信谁" 分，整个版图会重排成四个角色。

Role	Question	Output	Trusts
Falsifier 证伪器	Is there a counterexample?	a witness	**nothing** 没人
Certifier 证书生成器	Can it become an identity?	a certificate	**nothing** 没人
Bounder 界定器	How large can it be?	a proved enclosure	the interval library
Decider 判定器	True or false?	a verdict	the solver

> A witness and a certificate can be rechecked by anyone with a CAS. A verdict cannot: `unsat` is the solver asserting it searched exhaustively. So every use of a decider enlarges the trusted base and gives a referee one more thing to question, while falsifiers and certifiers enlarge it by nothing at all.
>
> 反例和证书任何人拿 CAS 就能重检。判决不行，`unsat` 是求解器在断言" 我搜遍了"。所以每用一次判定器，可信基就多一块；证伪和证书用多少次都不增加。

Practical rule. Prefer the earlier rows. A result obtained from the first two rows is not merely cheaper to obtain, it is cheaper to *defend*.

2 SMT 是什么 What SMT is

2.1 From SAT to SMT

SAT asks whether a propositional formula built from $\wedge, \vee, \neg$ has a satisfying assignment. **SMT** (Satisfiability Modulo Theories) keeps the boolean skeleton but replaces the atoms with statements in a background theory:

$$(x + 2y \le 7) \ \wedge \ \big((x > 0) \ \vee \ (f(a) = f(b))\big).$$

Here x, y are reals and f is an uninterpreted function. The phrase *modulo theories* means satisfiability *relative to* that theory. The smallest illustration:

$$p \wedge q, \qquad p \equiv (x > 0), \quad q \equiv (x < 0).$$

As propositional logic this is satisfiable, since p and q are unrelated booleans. Modulo the theory of reals it is unsatisfiable. That gap is exactly what SMT adds.

当成命题逻辑，p 和 q 是两个无关的布尔变量，可满足；模实数理论则不可满足。这个差就是 SMT 比 SAT 多出来的东西。

2.2 Inside: DPLL(T)

1. A SAT solver sees only the skeleton and proposes an assignment of atoms.
2. A *theory solver* takes the conjunction of asserted atoms and decides consistency in its theory.
3. If inconsistent, it returns a *conflict clause* $(\neg p \vee \neg q)$ to the SAT solver.
4. Repeat.

So an SMT solver is a fast boolean search engine plus a set of specialized mathematical deciders. Z3 uses simplex for linear arithmetic and `nlsat` for nonlinear; cvc5 uses cylindrical algebraic coverings for nonlinear.

SMT solver = 一个很快的布尔搜索引擎 + 若干专门的数学判定器。这也解释了为什么 `z3.Optimize` 只对线性完整：优化被实现在单纯形那个理论求解器里，`nlsat` 那边没有目标行。

2.3 Three answers 三种答案

Return	Meaning	Use
`unsat`	no solution	**Proof.** To prove $\forall x\, P(x)$, feed $\neg P(x)$.
`sat`	solution, with a *model*	**Counterexample.** The model is the witness.
`unknown`	undecided	Timeout, *or the theory is undecidable.*

The third is the one people forget. An SMT solver is not a universal decision procedure.

2.4 Fragments and decidability 理论片段与可判定性

Fragment	Content	Decidable?
UF	uninterpreted functions, equality	yes, fast (congruence closure)
LRA	linear real arithmetic	yes, fast (simplex)
LIA	linear integer arithmetic	yes, NP-hard
NRA	nonlinear real (polynomial)	**yes**, doubly exponential (Tarski 1951, CAD)
NIA	nonlinear integer	**no** (Hilbert's 10th problem)
BV, arrays, strings		yes
with $\exp, \sin, \log$	transcendental	**no** in general (Richardson)

The prefix `QF_` means quantifier-free, as in `QF_NRA`.

Three consequences that explain almost everything encountered today.
(i) Polynomial inequalities live in NRA: decidable but expensive. Hence z3 needs `nlsat`, and cvc5 needs `nl-cov`.
(ii) exp and log are not a missing feature, they are *undecidable*. This is why dReal is only δ-complete: it replaces an undecidable question with a decidable perturbed one. Transcendental claims must go to interval arithmetic instead.
(iii) `lean-smt` supports only UF and LIA/LRA because proof production and reconstruction are mature for the linear fragment and still an open problem for NRA.

2.5 Why `unsat` still needs trust

A `sat` answer hands you a model you can substitute back and verify yourself. An `unsat` answer is a sentence. To avoid trusting it, the solver must emit a *proof object*: proof-producing SMT. cvc5 is stronger here than z3, but for nonlinear arithmetic the emitted proofs contain coarse *trusted steps*, and making them fine enough for automated checking is an open research problem.

这就是" 能出证书就别只要判决" 的技术根源：在多项式片段上，SMT 的 `unsat` 目前就是只能信。

3 Certificates 证书

A certificate turns a claim into a finite object that a CAS re-verifies exactly. Nothing about the solver that found it is trusted.

3.1 Sum of squares (Positivstellensatz)

To show $p \geq 0$ on $\{g_1 \geq 0, \ldots, g_m \geq 0\}$, exhibit

$$p \;=\; \sigma_0 \;+\; \sum_{j=1}^{m} \sigma_j \, g_j, \qquad \sigma_j \text{ a sum of squares with positive rational coefficients.}$$

Given the σ_j, nonnegativity is immediate. The identity is a statement about polynomials with rational coefficients, so `sympy` settles it by expansion.

How the certificate is produced. Writing $p = m^\top Q\, m$ over a monomial basis m, the constraint $Q \succeq 0$ with coefficient matching is a semidefinite program. The SDP is numeric, so its Q is floating point and the identity only approximate. The repair is **Peyrl–Parrilo**:

1. solve the SDP numerically;
2. round Q to rationals at some denominator;
3. **project exactly**, over $\mathbb{Q}$, back onto the affine set $\{Q : \mathcal{A}(Q) = \text{coeffs}(p)\}$;
4. check $Q \succeq 0$ by exact rational $LDL^\top$;
5. read off $\sigma = \sum_i d_i \, (L^\top m)_i^2$ and expand the residual.

> Step 3 is what makes rounding harmless. After the projection the identity is exact again, and the only thing left to verify is positive semidefiniteness, which is decided in exact arithmetic. A wrong SDP solver cannot produce a false certificate, only fail to produce one.
> 第 3 步是关键：投影之后恒等式重新变成精确的，剩下要验的只有半正定，而那是精确算术判定的。SDP 求解器出错只会导致拿不到证书，不会产生假证书。

Worked example. For $x^3 + (1-x)^3 \geq \frac{1}{4}$:

$$x^3 + (1-x)^3 - \tfrac{1}{4} \;=\; 3\Big(x - \tfrac{1}{2}\Big)^2, \qquad \text{residual } \equiv 0.$$

which exports directly as `nlinarith [sq_nonneg (x - 1/2)]`.

Limits. Only polynomials. A failure is *not* evidence the claim is false, only that no decomposition exists at that relaxation degree.

找不到证书不等于命题为假，只说明该次数下没有分解。要找反例请用 SMT 或区间法。

3.2 Farkas multipliers

For linear systems the certificate is a vector of nonnegative rationals. To certify that $Ax \leq b$ is infeasible, exhibit $y \geq 0$ with $A^\top y = 0$ and $b^\top y < 0$. Then for any feasible x,

$$0 = (A^\top y)^\top x = y^\top (Ax) \leq y^\top b < 0,$$

a contradiction. To certify $Ax \leq b \Rightarrow c^\top x \leq d$, exhibit $y \geq 0$ with $A^\top y = c$ and $b^\top y \leq d$. Same repair technique: round the numeric dual, then project exactly onto $\{y : A^\top y = c\}$.

4 Rigorous numerics 严格数值

4.1 Enclosure versus sampling

> A grid scan that passes at 400 sample points says nothing about the 401st. No sampling loop can ever establish a negative existential such as "this function has no interior maximum".
> 采样在 400 个点通过，说明不了第 401 个点。网格永远不能证明" 不存在"。

Interval (ball) arithmetic evaluates over an entire box at once. Each quantity carries a proved error radius, so a comparison such as $v > 0$ returns true *only when it has been proved for the whole ball.* Adaptive bisection then covers the domain with finitely many boxes of definite sign, which is a proof, or finds a box of the opposite definite sign, which is a refutation with an exact rational witness.

4.2 What it cannot do

Interval methods cannot certify a non-strict bound that is *attained.* If $f(x^*) = 0$ for some interior x^*, no enclosure around x^* is ever strictly positive. The honest report is `unknown` localized at the contact point, not success. Such cases belong to the SOS layer, which handles equality contact exactly.

取到等号的非严格界，区间法原理上判不出来，报 `unknown` 并指出接触点才是正确行为。

4.3 A trap

`mpmath` is arbitrary precision but **not rigorous**. It must never supply a numerical bound that goes into a paper. Use Arb.

5 Three defects found today 今天查出的三个坑

5.1 `z3.Optimize` is wrong on nonlinear objectives, silently

Measured, with min $x^2 - x$ on $[0,1]$, true value $-1/4$:

Problem	`check()`	`lower / upper`	
linear, $\min 3x - 1$ on $[0,1]$	`sat`	-1 / -1	correct
nonlinear, $\min x^2 - x$	`sat`	$\mathbf{-39/256}$ / $\mathbf{-39/256}$	wrong
same, square named $y = x^2$	`sat`	$-39/256$ / $-39/256$	unchanged

Note `lower == upper`. This is not best-effort with a caveat; the one field that would signal uncertainty positively asserts a tight bracket that is false. Naming the square explicitly does not help, so it is not a syntactic artifact.

The same z3 refutes it in one call:

$$\exists x \in [0,1]:\ x^2 - x < -\tfrac{1}{4} \ \Rightarrow\ \texttt{unsat}, \qquad \exists x \in [0,1]:\ x^2 - x < -\tfrac{39}{256} \ \Rightarrow\ \texttt{sat}.$$

Cause. The optimization engine is built on simplex: dual simplex with an objective row, branch and bound for integers, MaxSAT cores for soft constraints. All of that is machinery for *linear* objectives. Nonlinear goes to `nlsat`, a *decision* procedure with no objective row. The nonlinear monomial cannot enter the objective row, so the optimizer optimizes the linear relaxation it can see and reports that as tight, which within that relaxation it is.

不完整本身不奇怪；缺陷是沉默。它有地方可以说" 这个理论我不会优化"，但它没说，反而输出了最强的结论形式。

Fix. Report a *certified bracket*: a lower end proved by `prove_forall` on both solvers, an upper end attained at an exact rational witness, and status `proved` only when the two meet.

5.2 Quantifiers over a finite domain go through MBQI

Searching for a finite counter-model with quantified range guards, the *same* logical formula gave different answers depending only on parenthesization:

Guard as written	**Result**
`And(And(a>=0,a<n), And(b>=0,b<n))`	`sat`
`And(a>=0,a<n,b>=0,b<n)`	`unknown`

Model-based quantifier instantiation is incomplete on this fragment.

Fix. On a finite domain, do not use quantifiers. Expand every $\forall$ into an explicit conjunction over the domain. The problem becomes quantifier-free finite constraint solving, which is decided

reliably. Confirm the model afterwards by evaluating the axioms in plain Python against the extracted table, with no solver in the loop.

有限论域上不要用量词，全部展开成 ground 约束；找到的模型用纯 Python 重新求值确认。

5.3 A grid scan is not a credential

This one had already caused a retraction: a ledger entry claiming "no interior maximizer" was withdrawn because the search grid had been truncated. Same shape as the `Optimize` defect: a confident answer over a region never actually covered, with no path to recheck it.

6 Environment, measured 环境实测

6.1 `python-flint` and Arb

Installing `python-flint` flips `sympy`'s `GROUND_TYPES` from `python` to `flint`. The auto-selection order is **flint** > **gmpy2** > **python**, so `gmpy2` can coexist and will only accelerate `mpmath`.

Operation	python	flint	ratio
`Poly` deg1200 × deg1200	25.1 ms	0.2 ms	∼125×
`Poly` gcd, deg ∼3600	919 ms	5.9 ms	∼156×
`Poly` A^3, deg 3600	108 ms	0.4 ms	∼270×
Hilbert-60 exact inverse over $\mathbb{Q}$	440 ms	12.1 ms	∼36×
Hilbert-60 exact determinant	773 ms	12.6 ms	∼61×
120×120 integer determinant	660 ms	12.6 ms	∼52×
`rref` over $\mathbb{Q}$	851 ms	339 ms	2.5×
`factor_list` $x^{200}-1$	12.3 ms	6.5 ms	1.9×
`sqf_list`	1182 ms	1380 ms	**0.86× slower**

> The speedup lands only on the *polynomial domain* and *DomainMatrix* paths. Symbolic expression operations (`expand`, `factor`, `Matrix.det` on a sympy Matrix) show **no benefit**, and `sqf_list` is slightly slower. It is not a general accelerator.
>
> 提速只落在精确多项式和精确线性代数上；符号表达式层零收益。别当通用加速器用。

The same package supplies **Arb**: ball arithmetic where every value carries a proved radius, plus rigorous integration via `acb.integral`.

6.2 cvc5

Options on a cubic inequality	Result	Time
default	`unknown`	45.1 s
`nl-ext=full`	`unknown`	64.8 s
`nl-cov=true`	**`unsat`**	**0.00 s**

`nl-cov` switches on cylindrical algebraic coverings, a complete decision method for NRA. Without it cvc5 contributes nothing to a nonlinear cross-check.

Also: the timeout option is `tlimit-per`, not z3's `timeout`, and cvc5 *raises* on an unrecognized option rather than ignoring it. Swallowing that exception means running with no limit at all.

超时选项名写错并把异常吞掉，等于无时限运行，会直接挂住。

7 Credential tiers 凭证分层

Layer	Output	Who must be trusted
`exact`	finite certificate, residual $\equiv 0$	**nobody**; anyone can redo it
`lean`	kernel-checked term	nobody
`arb`	enclosure covering the domain	the Arb implementation
`sympy`	exact symbolic computation	sympy
`z3` / `cvc5`	a verdict	that solver
not a credential: floating point grids, `mpmath`, local optimizers		

Two names present under `z3`/`cvc5` means both agreed; one name means one solver closed it, which is weaker and should look weaker on the page. Disagreement means one of them is wrong: record a conflict, never a credential.

8 Lean hammers: the verdict

8.1 A hammer is two things

Half	Output	Cost
Premise selection	a list of lemma names	**no permanent dependency**
Goal closing (Duper, lean-smt)	a tactic call left in the file	permanent dependency

The premise-selection half is already available through the Lean-LSP search tools and `LeanSearchClient`. Only the closing half is missing.

8.2 `lean-smt`, checked

- `lean-toolchain` is `v4.33.0`; a project pinned at `v4.30.0` would have to move.
- Theories supported: **UF and Linear Integer/Real Arithmetic with quantifiers**, plus experimental bitvectors. **No nonlinear.**
- It links its own patched cvc5 through Lean's FFI, so a separately installed cvc5 does not feed it.
- Self-described as beta; cvc5's proofs may contain holes returned as Lean goals.
- Genuinely strong where it applies: 2868 of 5000 Sledgehammer benchmark problems solved and 2847 proofs verified, beating both veriT+Sledgehammer and Duper.

> The chain "cvc5 decides → emits a proof → Lean reconstructs it → kernel credential" is **broken at the nonlinear segment**, and not because of `lean-smt`. Producing CAC proofs fine-grained enough for automated checking is an open problem upstream in cvc5 itself.
> 那条链在非线性那一段本来就断着，不是 lean-smt 的实现问题。

8.3 The right bridge is SOS

The reason is the mirror image of the above. A CAC proof cannot be reconstructed because its reasoning steps have no compact certificate form. An SOS decomposition *is* a certificate: a polynomial identity plus finitely many squares, which Lean closes with one `nlinarith` call.

nonlinear claim → SOS → exact check in sympy → `nlinarith [sq_nonneg (…)]`

No new dependency, no toolchain move, nothing to reconstruct. It routes around the open problem instead of waiting on it.

9 Quick reference 速查

```
from hz_certify import rigorous as R, certificates as C, decide as D, ledger as L

L.already_recorded("key")               # always first: the ledger may have it

C.sos(p, [x])                           # p >= 0            -> exact certificate
C.sos_on_box(p, {x: (0, 1)})            # p >= 0 on a box
C.farkas(A, b, c, d)                    # Ax<=b => c.x<=d   -> exact multipliers
C.verify(cert)                          # recompute, ignoring the verified flag
C.to_lean_hint(cert)                    # -> nlinarith [sq_nonneg (...)]

R.prove_positive(expr, {x: (0, 1)})     # proved / refuted+witness / unknown
R.integral(expr, x, 0, 3)               # proved error radius
R.range_enclosure(expr, box)

D.prove_forall(claim, hyps)             # z3 AND cvc5; proved only if both agree
D.counterexample(claim, hyps)           # witness, replayed exactly in sympy
D.worst_case(obj, cons)                 # certified bracket, not Optimize
D.independent_axioms(ax, signature=...) # ground search + solver-free recheck

L.record_certificate / record_verdict / record_decision
```

Decision order for a new claim.

1. Check the ledger. An unchanged statement with a credential is done.
2. Try to *kill* it first: a counterexample is the cheapest possible result.
3. Polynomial? Try SOS for an `exact` certificate before any solver.
4. Contains exp, log, sin? SMT cannot help. Use Arb.
5. Only then a verdict, and run both solvers.
6. Record, with the layer that actually ran.

All measurements taken 2026-08-20 on Python 3.11.5, sympy 1.14.0, python-flint 0.9.0, z3-solver 5.0.0, cvc5 1.3.4, cvxpy 1.9.2, Lean 4.30.0. `python -m hz_certify.selftest`: 38 checks, all pass.