GO CHECK YOURSELF

Go to result 1–339 Go Random result

What "Lean-verified" means, and what it does not

READING LEVEL KIDS · EVERYONE · EXPERTS

Seven Lean demonstrations, each checked by Lean and leanchecker, and by a second kernel checker, nanoda, on a rebuild with Lean 4.35.0; not yet refereed.

Lean-checked 4 · computed 3 · open 1

Overview

A growing amount of mathematics, and of claims made by computers, carries a label saying it was checked in Lean. Lean is a program that checks every step of a proof against fixed rules. This book asks what that label covers, and whether a file Lean accepts can still say something false. It answers with seven small Lean files, built with Lean 4.34.1, and what Lean actually printed for each. Five of them make a clean-looking build say 2 + 2 = 5, and a different check catches each kind of trick. An added axiom or a sorry, a placeholder for a missing proof, shows in the axiom report. A switch that turns kernel checking off is caught by leanchecker, which replays every declaration through the kernel. A true theorem about a one-number system or a redefined plus sign passes every automatic check and is caught only by reading the statement. Four files are proved and checked in Lean as written, without any claim about what the statement means, and are accepted by leanchecker and by a separately written checker, nanoda. Two depend on an added or placeholder assumption, which nanoda refuses, and one is rejected by the re-check. A second separately written kernel, lean4lean, could not read the files here, so that check is open. The status is not yet refereed, and the check guide gives every command and its expected output.

CHECK IT YOURSELF

Cite: Sonny (2026). What "Lean-verified" means, and what it does not (preliminary). Go check yourself. https://gocheckyourself.org/read/lean-verified/everyone/

1. Start here
2. What the words mean
3. Try it yourself

Start here

Sonny

What this book shows

- My result. Seven small Lean 4.34.1 files (Lean has a small judge inside it, the kernel, that checks every step of a proof against a fixed list of rules), with what Lean actually printed. Five of them make a clean-looking build say 2 + 2 = 5, and a different check catches each kind of trick.
- What a clean build tells you: that Lean built the file without an error, given the definitions and axioms (starting assumptions that are not proved) in scope, and nothing more. It does not by itself show that the kernel checked every declaration: a switch can turn that check off and the build still passes. An added axiom or a sorry (a placeholder for a missing proof) shows in the axiom report (Lean's list of the assumptions a theorem rests on); a switch that turns kernel checking off is caught by leanchecker, which replays every declaration through the kernel; a true theorem about a one-number system or a redefined plus sign passes every automatic check and is caught by reading the statement, by no automatic check.
- Why it matters. More and more mathematics, and more claims made by computers, carry a "checked in Lean" label. Knowing which check catches which problem lets a reader ask the right question of a result.
- In Lean: all seven files. Four are proved in Lean, as written, and accepted by leanchecker and by a separately written kernel, nanoda (a second checker for Lean proofs, written by different people). Two depend on an added or placeholder assumption, which the axiom report shows and nanoda refuses. One is rejected by the re-check.
- Every theorem in this book is listed in the theorem library.

The question

When a mathematical result is called "Lean-verified", what has actually been checked, and can a file that Lean accepts still say something false?

Lean is a program for writing mathematics so exactly that a small piece of software, the kernel, can check every step. A "Lean-verified" label is a strong claim and is easy to over-read. The people who build Lean ask the same question: the Lean Language Reference has a chapter, "Validating a Lean Proof", on how to judge a result you did not write. In July 2026 the question stopped being only about readers' care: a user reported a bug in the kernel itself, in an issue titled "Kernel accepts wrong-structure projections, allowing an axiom-free proof of False" (issue 14576 in the Lean repository, 28 July 2026), and it was fixed the same day.

This book answers the question for one narrow case: what a clean build does and does not tell you, shown on seven small Lean files.

The short answer is that a clean build tells you that Lean built the file without an error, given the definitions and axioms in scope, and nothing more; a separate replay with leanchecker is what shows the kernel accepted every declaration. Five of the files below make a clean-looking build say 2 + 2 = 5, and a different check catches each kind of trick.

Why it matters

More and more mathematics, and more and more claims made by computers, are presented with a "checked in Lean" label. A reader who takes the label to mean "the English sentence I just read is proved" can be wrong in four ways: the Lean sentence may say something else, a definition may mean something else, part of the argument may sit outside Lean, or the checking program may have a bug. Knowing which check catches which problem lets a reader ask the right question of a result.

What answering it would help with

- A reader of a "Lean-verified" claim may use the same short list of checks (the axiom report, a re-check through the kernel, a second kernel, reading the statement) to see how far the label can be trusted.
- Authors may be able to say which of these checks a result has passed, in the same words every time, so that labels become comparable.
- Teachers may use the seven files as small, runnable examples of what a formal check does and does not cover.

What our research shows

What this is

A SHORT BOOK ON WHAT A "LEAN-VERIFIED" LABEL COVERS, WITH SEVEN SMALL LEAN 4.34.1 FILES THAT SHOW HOW A CLEAN-LOOKING BUILD CAN STILL SAY "2 + 2 = 5" (SONNY, 2026).

Still open

WAYS A LEAN RESULT CAN MISLEAD BEYOND THE FOUR THIS BOOK COVERS. THE BOOK EXPLAINS THE LABEL; IT REVIEWS NO ONE'S PROOFS.

Checked by

EACH FILE WAS BUILT WITH LEAN 4.34.1 AND RE-CHECKED WITH LEANCHECKER; THE SAME FILES BUILT WITH LEAN 4.35.0 WERE ALSO GIVEN TO A SEPARATELY WRITTEN KERNEL, NANODA. SUPPORTING FILES ARE AVAILABLE ON REQUEST.

<!--BOX-->

What's in this book

Counts are computed from the book's statement file (its table of seven demonstrations), except the open item, which comes from chapter 4 and the check guide.

Status	Count
Proved and checked in Lean (the Lean statement as written, accepted by the kernel; the meaning of the statement is not claimed)	4 (the honest file, the one-number system, the different plus sign, the faulty definition)
Accepted by the build but dependent on an added or placeholder assumption, visible in the axiom report	2
Stored with kernel checking turned off, rejected by the re-check	1
Open	1 (a run with lean4lean, which could not read the compiled files here)

Item	Count
Chapters	4
Statement files	2 (a table and its machine-readable form)
Written check recipes	7, one per demonstration, plus an optional second-kernel recipe and the fingerprint check
Lean files	7
Fingerprint list	1 (SHA-256 of the seven Lean files and the build files)

<!--/BOX-->

Yesterday, today, tomorrow

Yesterday. Leonardo de Moura and the Lean community built Lean. Kim Morrison and Sebastian Ullrich wrote leanchecker (earlier lean4checker), which replays a compiled proof through the kernel to detect tampering. Mario Carneiro wrote lean4lean and ammkrn wrote nanoda, two checkers that re-implement the kernel separately. The Lean reference manual has a chapter, "Validating a Lean Proof", on how to judge a result you did not write. In July 2026 a bug in the kernel itself was reported and fixed.

Today. I ran seven small demonstrations on Lean 4.34.1, on one Windows PC, and pasted what Lean actually printed. An added axiom builds and is shown by the axiom report. A sorry builds with a warning and is shown by the axiom report. A true theorem about a one-number system, or about a redefined plus sign, passes every automatic check and is caught only by reading the statement. A switch that turns kernel checking off lets a false statement build with "no axioms", and leanchecker rejects it. The separate kernel, nanoda, accepted the files that only a reader can catch, and did not accept the one with checking switched off.

Tomorrow. The same checks can be run by anyone on these files in a few minutes. lean4lean, a second separately written kernel, could not read the compiled files here, so only nanoda was used; running lean4lean too would be a natural next step. Other readers may know more ways a Lean result can mislead, and those can be added here with the same format: the file, the real output, and the check that catches it.

Check it yourself

In this book: the project file that builds the seven Lean files, a SHA-256 fingerprint (a 64-character code computed from a file; change one bit and the code changes) for each of those files, and a written recipe for each demonstration in the check guide. The seven Lean files are in the demos folder, and chapter 3 prints each one in full. A recipe gives the exact command, the exact output to expect, and the standard tool that produces it: Lean's build tool, the axiom report, leanchecker, and optionally nanoda. To try one, install Lean through elan (Lean's installer), go to the lean folder and run:

```
lake build Demo.Honest
lake env leanchecker Demo.Honest
```

The build prints that the theorem does not depend on any axioms, and leanchecker accepts silently.

How I direct this work, and my own tools, are not part of this book. Everything you need to check the results is public: the results, their certificates and fingerprints, the Lean proofs, and a written recipe for each check. In this book the results are the seven

Lean files and what each check printed; there are no certificate files beyond the fingerprint list of those files.

About this research: https://gocheckyourself.org/about/#how

The chapters

- What "Lean-verified" means: the guarantee, step by step.
- What "Lean-verified" does not mean: four things the label leaves out.
- Five ways to get 2 + 2 = 5 past a build: the seven files, with real output.
- Kernel bugs, and why a second checker helps: the case where the check itself is wrong.

The statement of the seven demonstrations, the check guide, the plain words page, the children's version and the credits page complete the book.

We stand on the shoulders of giants: Lean, its checkers and the people who found and fixed kernel bugs are other people's work, credited on the credits page.

Where to go next

- Which one of the seven files would you have missed if you only read the build log? See Five ways to get 2 + 2 = 5 past a build.
- If the kernel itself can be wrong, how much does a second kernel help? See Kernel bugs, and why a second checker helps.

Go check yourself.

Have a question about this? Ask it

What the words mean

What the words mean

The question: what do the words in a "Lean-verified" claim mean, in plain words and for experts? The answer in brief: each is one link in a chain, from the kernel that judges a proof, to the axiom report, to a second look, to the meaning of the sentence itself.

"Lean-verified" is a short label for a long chain of steps. Each idea below has a plain version and a version for experts.

The kernel

Lean has a small judge inside it. You hand it a statement and a proof, and it checks every step against a fixed list of rules. If it says yes, the proof is correct for the statement as written, using the definitions and starting assumptions that are in scope. It does not decide whether the statement is the one you meant.

Axioms and the axiom report

An axiom is a starting assumption that is not proved. Lean uses three standard ones. You can ask Lean to list the assumptions a theorem rests on. If the list is empty or holds only the three standard names, nothing extra was assumed. If it shows the special placeholder axiom, someone left a gap. If it shows a name nobody recognises, someone added an assumption, and the theorem is true only if that assumption is. The list reports the stored assumptions; a separate kernel replay is another check.

A second look

There are two kinds. One is Lean looking again at its own stored work, with the same judge, which catches work that was stored without being judged. The other is a different judge, written by different people in a different language, which is given the proof as plain text. If both say yes, a bug in one is unlikely to be in the other. It is still not a guarantee.

A true sentence that means something else

A proof is a proof of the exact sentence written in Lean, with the exact meanings of the symbols in scope. If someone redefines the plus sign, or uses a number system with a single number, or leaves a condition out of a definition, then Lean will happily confirm a sentence that looks like something else. The only defence is for a person to read what the sentence and its definitions really say.

The switch that turns checking off

Lean has a setting, meant for experts debugging Lean itself, that tells it to store a declaration without sending it to the judge. A file that sets it can store a false statement, and the build and the assumption list both look clean. Looking again with leanchecker sends the stored statement to the judge, which refuses it. The setting can be placed in the build file, so searching the build file matters as much as searching the Lean files.

Where to go next

- Which of these words are shown at work, with real Lean output? See Five ways to get 2 + 2 = 5 past a build.
- What does the label leave unsaid? See What "Lean-verified" does not mean.

Experts: the files and checks are here: check guide.

Yesterday, today, tomorrow

Yesterday: Lean was built so that a very small program, the kernel, is the judge of every proof, and so that a reader need not trust the person who wrote it. Its authors also wrote tools to look again at a finished proof, and two other groups wrote separate kernels that can re-check a proof exported from Lean.

Today: I ran small examples to show what the label covers. An extra axiom, an unfinished proof, a redefined plus sign, a number system with one number and a switch that turns checking off each give a Lean file that "proves" 2 + 2 = 5 (or a related falsehood). In each case the book shows what Lean printed and which check catches it.

Tomorrow: The same questions can be asked of any Lean result. Readers who know other ways a Lean result can mislead are asked to say so.

We hope it helps others find more results and keeps their curiosity going. We'd love to collaborate with anyone.

Try it yourself

The figures of this chapter are the interactive checks on the tools page (they need JavaScript).

Each result on its own page

- Demo 1: the honest file, 2 + 2 = 4
- Demo 2: the added axiom, 2 + 2 = 5 from a declared axiom
- Demo 3: the unfinished proof, 2 + 2 = 5 by a placeholder
- Demo 4: the one-number system, 2 + 2 = 5 below 1
- Demo 5: the different plus sign, 2 + 2 = 5
- Demo 6: the faulty definition, 1 is prime
- Demo 7: the kernel-check-off file, False stored unchecked
- Open question: A second-kernel run with lean4lean

© 2026 Sonny. The books are under CC BY 4.0 (Creative Commons Attribution 4.0 International): reuse them with credit and a link. Each book’s licence file lists the files that keep their own licences.