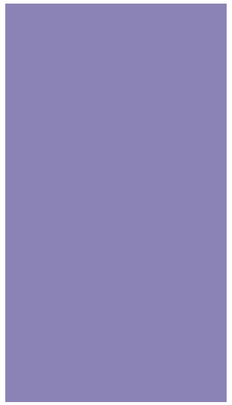
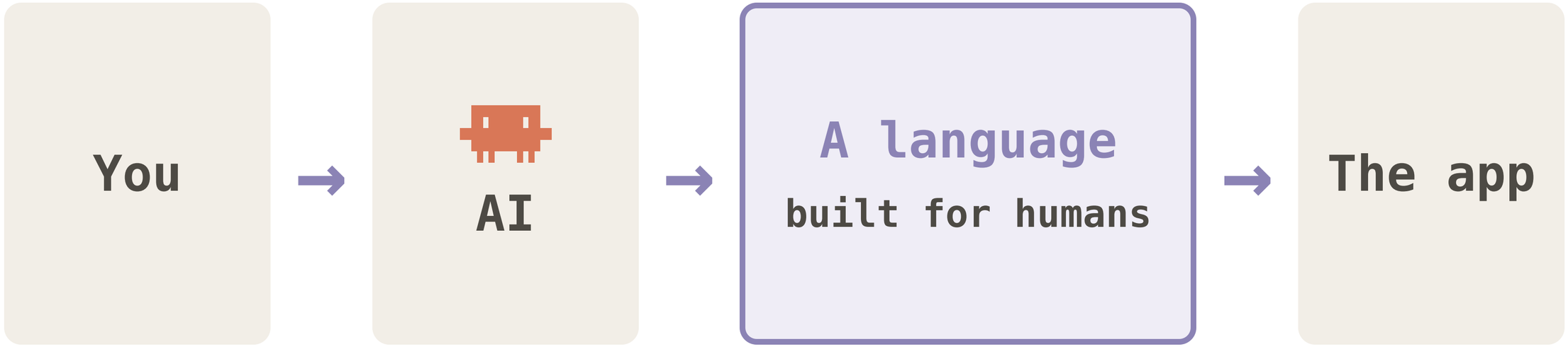
Bend



The last programming language

Every vibe-coded app passes through a language



How well AI codes depends on the language it writes.

Problem: languages weren't built for AI



Greg Brockman
President, OpenAI

rust is a perfect language for agents, given that if it compiles it's ~correct



Andrej Karpathy
Co-founder, OpenAI

[…] even Rust is nowhere near optimal for LLMs as a target language. What kind of language is optimal?

Some accidentally work better, but none intentionally.

Can a language make AI smarter?

JavaScript
More Errors
Dumber AI
(bugs everywhere)

<

Rust
Fewer Errors
Smarter AI
(reportedly great)

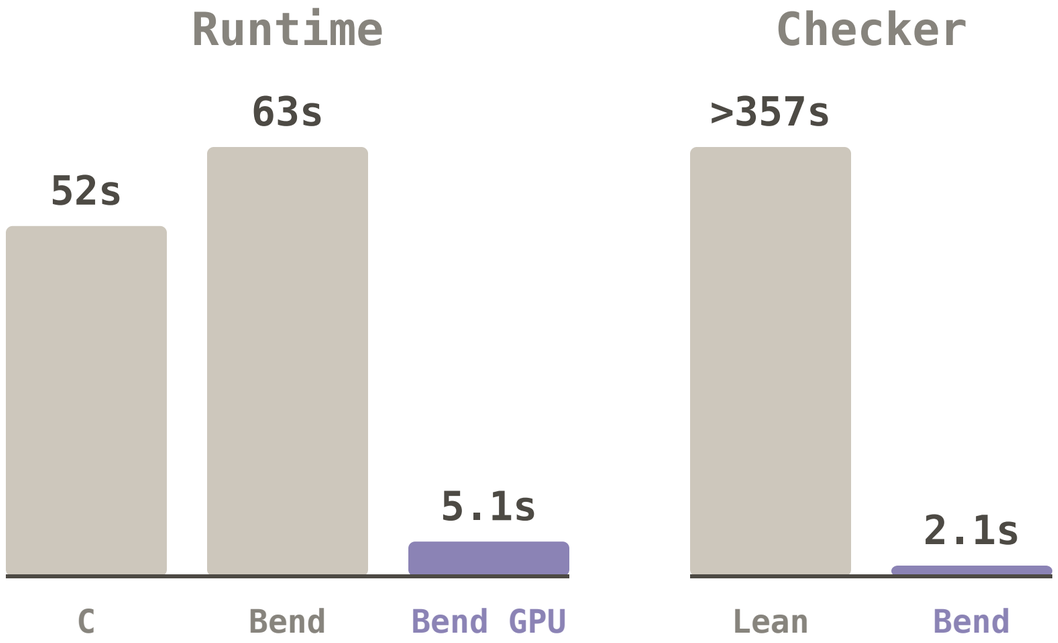
<

Lean
Zero Errors
Genius AI
(revolutionizing math)

Yes: stronger types are correlated with AI performance.
The strongest, Lean, is now **solving open math problems**!
But Lean is optimized for math, not for software.

Solution: Bend

A language for software



Time to run the full benchmark suite. Lower is better.

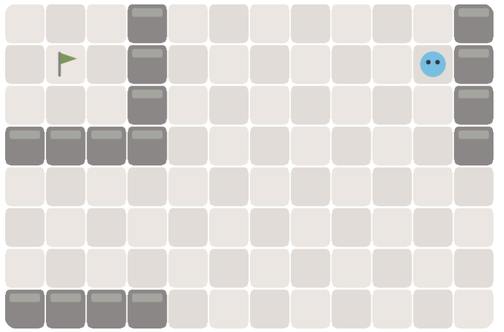
That makes AI smarter

```
theorem: the flag is unreachable
```

> Claude, make the board wrap around



Without proofs: it breaks

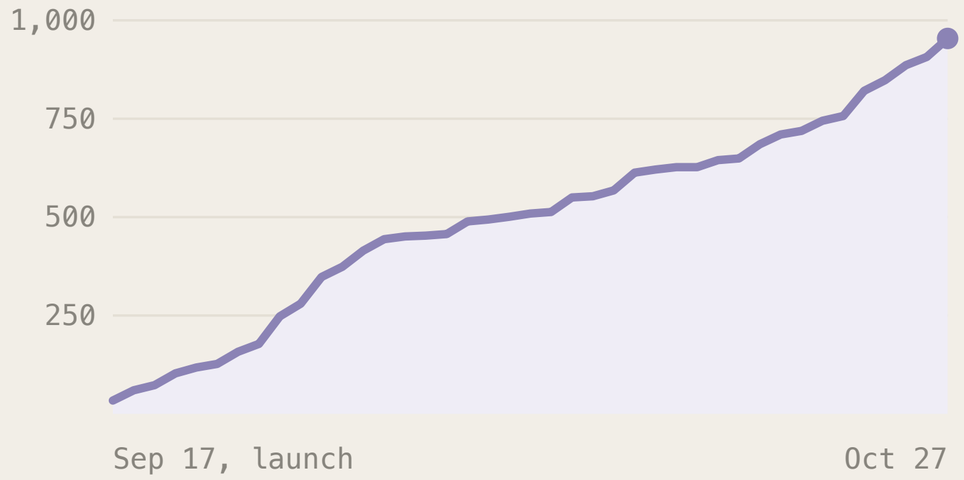


With proofs: it works!

Reception

954

packages published on BendHub



4.0M

views on Bend posts

2.3k replies

24.5k

GitHub stars

4,800+ since launch

6,000

machines ran Bend

1,700 this week

170+

public Bend 2 repos

126 outside authors

A killer use case: Proof-driven AutoResearch



lauren
Grok team, SpaceXAI

very bullish on Bend [...] when you have a codebase that can formally verify itself, you can ship at an incredible pace. code review is solved



Giulio Rebuffo
Ex-Erigon (Ethereum)

Bend 2 is basically a software factory language. [...] YOU DONT NEED TO LOOK AT THE FINAL IMPL ANYMORE



evaan
Developer

bend is the language for ai development where you don't care about the implementation ai generates. and u don't have to trust it - you have a ~proof~ it works

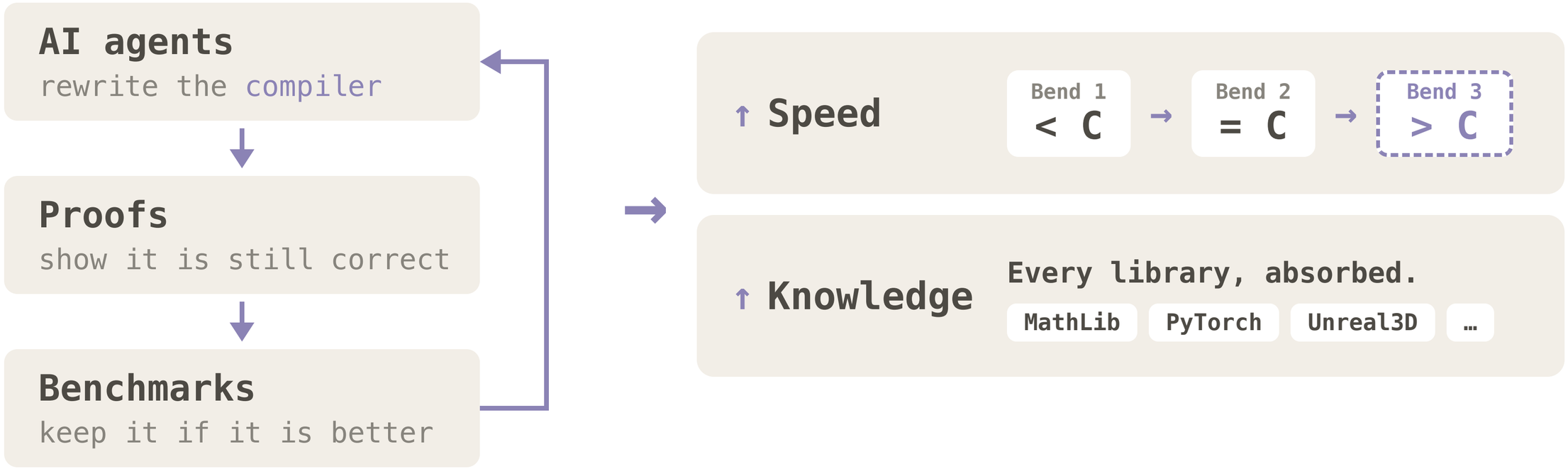
AI builders started using Bend to drive **AutoResearch**
(AI improving software on its own, in a nonstop loop)

If Proof-driven AutoResearch works so well…

why not use it on Bend itself?

Next: Bend 2 builds Bend 3

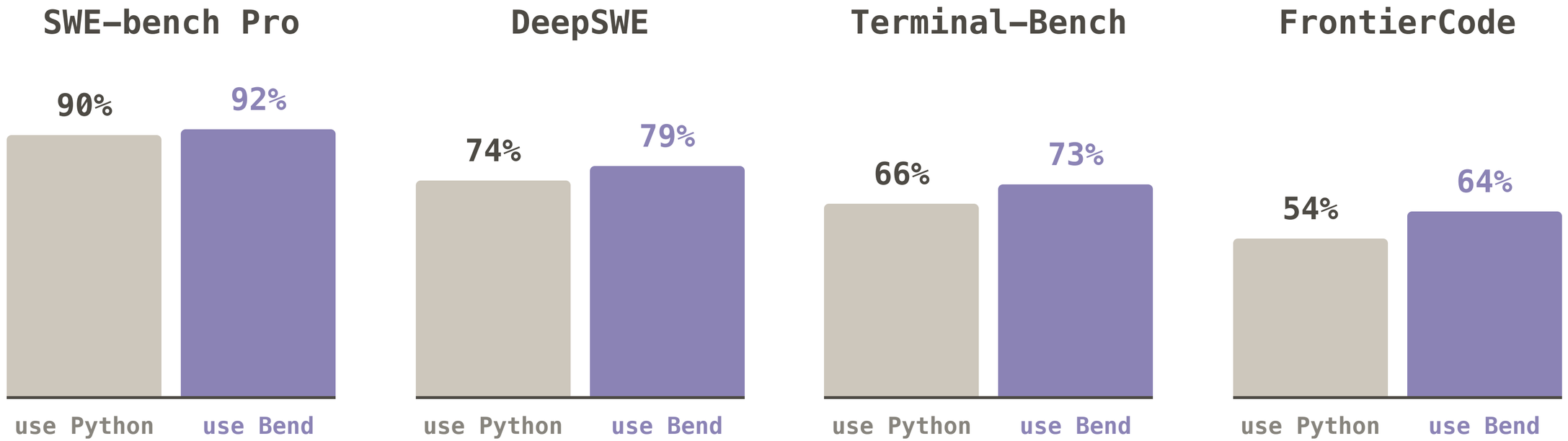
Mass compute, 24/7



Result: the compiler becomes intelligent.

Then: Bend 3 makes the AI smarter

Claude Opus 5.5



This is our main hypothesis.

Then: “use Bend” becomes the default

New apps



Rewrites



Using Bend gives you better results than other languages.

Business Model

- Bend 3 is **closed source.** We sell it **as a service.**
- Send us a **program:** we *optimize* it.
- Send us a **theorem:** we *prove* it.
- Send us a **prompt:** we *implement* it.

$5.5B+

a year paid for AI coding,
Claude Code and Cursor alone.

$2B+

a year on AI code review,
which proofs make obsolete.

A sufficiently smart compiler becomes a model.

Why now: the sky is falling

PHP2Shell Oct 20

70%

of all websites exposed by one bug an AI found.

Taken down: governments, 4chan, the Internet Archive.

anthropic.com/research/formal-verification

Anthropic Research · Oct 22

LLMs are making formal software verification feasible at a much larger scale

"The trajectory is clear: formal verification … will soon be as practical as fuzzing is today for **almost all production software**."



ThePrimeagen
@ThePrimeagen · Oct 22

Ok... I see why Bend is pretty radical

137k views · 1,957 likes

↓ Proof cost ↑ Exploit cost = **Formal verification explodes**

We spent 2 years building Bend for this. It's ready now.

Raising $40M to make “use Bend” the default

$30M

Bend 2
builds Bend 3

with mass compute

$5M

small core team ($2M/yr)

$5M

gunpowder: a flexible reserve

Most of it goes straight into the loop.

Five people and a fleet of AI agents built Bend

Victor Taelin

Founder, CEO

Created HVM, Bend and Kind.
Lead developer at the
Ethereum Foundation, 2017–19.

Vitor Chiarelli Neves
Cofounder, operations

Nicolas Abril
Engineer, Bend core

Lorenzo Battistela
Engineer, Bend core

Paulo Cavalcanti
Engineer, Bend core

Up to 200 AI agents write Bend's code at once.

In short,

Soon, AI writes all the code. A language only needs to be:

FAST
runs at full speed

`use C` `use CUDA`

SMART
reduces AI error

`use Lean`

↓ ↓

`use Bend` **The one language built for both.**

We will make Bend the default.

Appendix

What users say

@RealAstropulse had Claude port his Rust pixel-art tool to Bend

“Zero memory or concurrency bugs, which is the point of the language. […] I genuinely couldn't find a single false claim. It does what he says.”

@DaleCloudman rebuilt a Rust project in Bend 2

“Before it would keep having weird edge cases that wouldn't quite get fixed […] After, these are all gone.”

@andrewtjoslin ported app permission and subscription logic

“Allows tons of test removal and clean code.”

@diblacksmith asked for “something with application logic”

“Suddenly I got a working macOS binary (!!) with GUI.”

Fabio Akita planted a bug his tests missed; the proofs caught it

“That's Bend working as promised, down to the detail.”

What we're building

Soon AI will write almost all code, but it still writes it in languages designed for humans, and since no one can review millions of AI-written lines, bugs pile up.

The stricter a language is, the less the AI gets wrong, which is why AI is now solving open math problems in Lean, where every step must be proven, but Lean is too slow and limited for real software.

Bend brings Lean's proofs to a practical language that runs at C speed and on GPUs.

Since AI-written code comes with proof that it works, an AI can keep improving software with no human checking it, because any change that breaks it is rejected.

We're raising to run AutoResearch on Bend's own compiler at scale, making it the smartest compiler in the world, and to sell it as a service that turns any program into faster, proven-correct code.

FAQ

How does Bend work, in a nutshell?

A programming language, like TypeScript, whose types can state and check proofs, like Lean or Rocq. Unlike them, it does no inference, unification or typeclass search, so checking scales to huge codebases; linear types, as in Rust, let it run as fast as C.

Are apps built in Bend more secure by default?

No: you can write it like TypeScript. Proving your program's properties is what secures it, even against Anthropic's strongest models.

Who writes the laws?

Humans or AI; Bend only checks them. Experts write avionics laws; for a pet store, AI writes and proves them. AIs aren't great at laws yet, but writing them improves their code, and models keep improving. For “send us a prompt”, a standard LLM writes the laws; experts send their own.

Who writes the proofs?

AI models like Claude and GPT; humans can too. We provide the language, the fastest checker and the largest proven ecosystem, not the intelligence; our own symbolic provers may come later.

FAQ

Formal proofs are decades old. Why now?

Proofs were ~100× the code's size, too costly by hand. AI now writes them nearly free, and finds bugs: PHP2Shell hid in PHP, under ~70% of websites, for a decade.

Claude can prove C code now. Why a new language?

Proving old code is costly: Anthropic's proofs that libraries never crash took ~600k lines each and hours to check. In Bend, code ships with its proof, for any law.

Won't the world just port its code to Lean?

Proofs and rewrites are getting cheaper, toward free, while holes get costlier, as AIs do more harm through simpler ones. That pushes code to Rust, then, as it falls short, to proof assistants. But Lean doesn't scale, is slower than C, CPU-only and often inconsistent. Bend was built for this.

Laws can be wrong. Don't bugs just move into them?

They can: no proof knows what you meant, which is AI-complete.
But laws are tiny: once proven, “this room is unreachable” holds against every spell, glitch and spawn of a whole MMORPG, which no number of tests can promise. You read one line, not a million.

FAQ

Why not have AI write ever more tests instead?

Tests cover the cases someone thought of, and AI optimizing for weeks finds the gaps: fork GHC, with its thousands of tests, and AI breaks it on day three. One law, “the compiler equals the interpreter”, covers every input.

Don't real specs grow as big as the code?

Some must: nothing specifies the Mona Lisa better than its pixels. But most software needs few: Bend 3's spec, its interpreter plus a CUDA model, is under 1,000 lines, for a compiler that may reach millions. By 2030 we expect most code fully specified by laws, with specs shrinking to the smallest that capture the intent.

Does anything certify design?

Laws can state anything English can, like alignment, spacing or orphan lines, but not beauty. It matters little here: Bend targets math-heavy back ends, like banks, databases, compilers, physics engines and air traffic control.

How does a team work with laws day to day?

Code becomes like assembly: no one reviews it. The team edits prompts and laws, down to a locked color, and the AI regenerates the code and its proofs.

FAQ

Who will buy Bend 3 first, and how will they find it?

AI labs running AutoResearch, some already using Bend 2's laws. Bend 2's guide will tell AIs about Bend 3, so the AI itself offers the signup.

Doesn't a closed Bend 3 fight adoption?

No. The language, the checker and a C-speed compiler are free and open. Only Bend 3, served like an AI model, is paid.

Compilers rarely sell. Why would Bend 3?

Neither did language models in the GPT-2 days. At scale, Bend 3 becomes a new thing: a model made of code. People will pay for it as they pay for AI models.

Why pay for Bend 3 if Bend 2 is free?

Speed is money. Spark is free, yet Databricks charges about 2× for Photon, its closed engine. Chip-design compilers earned Synopsys and Cadence $12B in 2025.

How do investors price bets like this?

Axiom raised $200M at $1.6B and Harmonic $120M at $1.45B, both proving in Lean. Ricursive, whose AI evolves chip designs: $300M at $4B.

FAQ

What does the $30M buy?

Building Bend 3 until it beats C, then CUDA, and serving it to companies.

What goes into Bend 3, and what comes out?

A prompt or a codebase goes in; a far faster program, proven correct, comes out. Unlike LLMs tuning CUDA kernels, it may rewrite the algorithm itself, since its output is proven equal to its input.

Won't someone just buy the company?

Then they pay for our head start. Google paid $2.4B to license Windsurf's tech and hire its leaders; Qualcomm agreed to buy Modular (Mojo) for $3.9B.

Can't the labs copy this, with more compute or a fork of Lean?

No. The loop needs speed and laws for every behavior, including the compiler's, which few can write. Rust has no proofs. Lean is slow, not fully formalized and CPU-only; fixing it means reinventing Bend, as compiling the full λ-calculus at C speed is unsolved after decades. Bend 3 is closed, and our lead compounds every month the loop runs.

FAQ

Has the loop ever worked?

Yes: AutoResearch is how Bend 2 reached C speed. Without laws, bugs grow with scale; with laws, the same loop can take far more compute. We're now ready to scale.

Bend 1 faded. Why would Bend 2 stick?

It won't. Bend 1 was a GPU research prototype, still 100× slower than C on one core. Bend 2 matches C, but doesn't bring astronomical value yet. Bend 3 will.

Models have seen little Bend code. Can they write it?

Yes, better than many languages, like Ruby. Anthropic already trains on our code, so Claude knows Bend; its syntax is Python with types, so Opus makes almost no syntax errors from a few examples; and its proofs are like Lean's, which models are deeply RL-trained on. With Bend's checker, Opus proved 20 of 20 tasks; without it, 0 of 20. Models do miss tricks for *fast* Bend; new models trained on Bend 2 will learn them, and Bend 3's loop makes their usual patterns fast.

Can AI prove real software, not just toys?

Yes. From laws alone, Opus built and proved an SQL query optimizer, an HTTP/1.1 parser and a JSON codec, for $1 to $5 each.

FAQ

Will an AI-built ecosystem compound?

Yes. Reusing 36 proven libraries, Opus proves a new one 1.8× faster and 1.5× cheaper.

Is AI-written Bend code reliable?

Yes. In billing code, AI's proofs caught 42 of 42 bugs; tests caught 27. From the RFC alone, Opus wrote a JSON validator that passes all 283 hidden tests; cJSON fails 17.

Can the AI fake a proof? Who checks the checker?

No: the checker rejects any wrong proof, so at worst the AI fails to find one. Its kernel is proven correct in 3,895 lines of Lean; a $10k bounty to break it is unclaimed.

Why GPUs?

Every company owns GPU clusters, every gamer a big GPU, but GPU languages are so low-level that they run little beyond shaders and LLMs. Bend is the first high-level one.

I'll not invest in a language without an anime intro.

Say no more. x.com/VictorTaelin/status/2104568082169749982