Creating Clarity around Reliable AI in Software.

Two ideas make modern AI simple: state and judgment.

state

paid	true
invoice	#4471
balance	12,400
status	shipped
owner	Trucking A
due	Oct 4
approved	false

You can point at it.

judgment

Is this urgent?
I think so.

Which account?
Account 3b. I'm 90% sure.

Why?
The contract said so, page 49.

Do we have to rebuild?
One expert says probably.
Another needs more info. She leans no.

State, once you trust it.

DANIEL ACKERMAN

dan@0to1.dev

System and product

system	product
what is true	what matters

> There's two sides. There's the system and then there's the product. The system is the state of the world. What data is being stored in the database? How do we count facts? The product is what you choose as state: what facts matter? How do we present them?
>
> Daniel Ackerman

The system is what is true. The product is what matters.

Amazon knows every box in every warehouse. That is the system. Amazon shows you one line: *arrives tomorrow*. That is the product. Someone decided that line matters to you. That someone was a person.

> The hardest part of any technical business is the product: what state should be tracked, when does that state matter, to whom, at what time. Even the best uses of AI struggle to reliably answer this. The answer may be different for every customer.
>
> Daniel Ackerman

State and judgment

state	judgment
what is written down	deciding what to write

> Text is *almost always* not state. Someone still has to decide what it means.
>
> Daniel Ackerman

Two kinds of memory

A database (Amazon's, yours, anyone's)

A giant list. Each line says one true thing. This box. This price. This address. Shipped: yes.

Anyone can open it. Everyone sees the same thing. Some databases remember every change. Some databases decide what is allowed to change.

This is state.

An AI's stream of thought

The AI does not keep a list. When you talk to it, your words become numbers. Then terabytes of weights do one thing: calculate the next token. Every single time. One token, then the next, then the next. Then the numbers become words again.

No one knows why it printed that token. Not the people who built it. Not the AI.

This is not state. It is judgment happening.

The engineering words for "words become numbers" are tokenization, encoding, embedding, projection. You do not need them. Just know: in, the words become numbers. Out, the numbers become words again.

Closed loop and open loop

Closed loop

Every step lands on something real. State changes. A rule runs. State changes again. When a judgment is needed, it sits in one small box, answers one small question, and the answer is written down as state. Then the loop closes.

AI never talks to AI without state in between.

This is every website you have ever used, with judgment added at the seams.

> How can we put this into a box where it makes a judgment, and then that goes to state?
>
> Daniel Ackerman

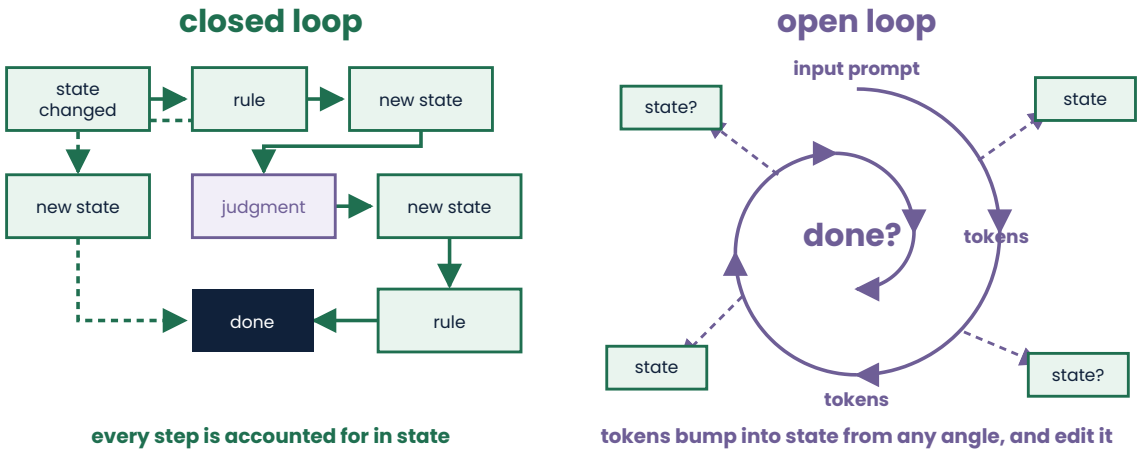
Open loop

The AI writes words. Those words go straight back into the AI. It writes more words. Around and around.

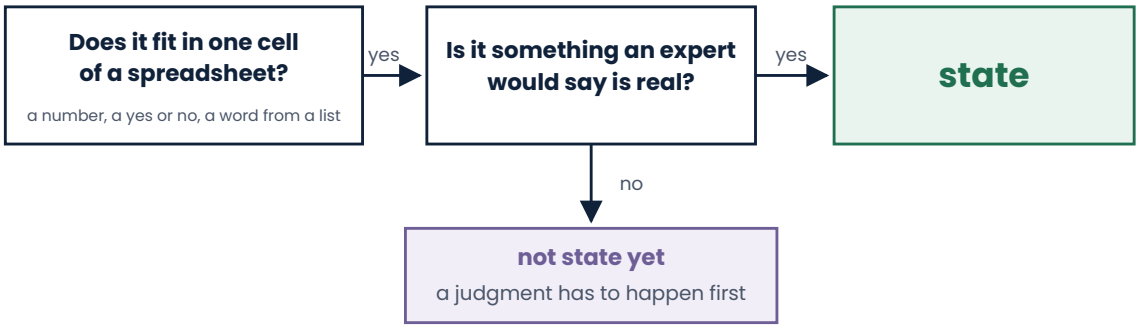
Nothing lands. No number, no yes or no, no label you could show your accountant. The judgments are made, hundreds of them, and none are pinned to state.

When it is wrong, you cannot find where. When it is right, you cannot say why.

> The chain of judgments is right 90% of the time, 90% of the time. When the chain is wrong, it is expensive to inspect.
>
> Daniel Ackerman



How do you know when something is state?



More examples

Simple state

The light is on. Or off.

Complex state

Do I owe more tax this year on line 57b? The grey area is already written into law. Your accountant makes a judgment. After the IRS audit, it becomes state.

What state looks like. What judgment looks like.

state	judgment
recording what is true	interpreting and acting upon state

State

What happened	In English	What it looks like in code
A customer signs up.	A new line in the list.	`INSERT INTO users`
A payment lands.	Mark it: paid.	`UPDATE invoice SET status='paid'`
We check a website page for the word refund.	Found it, or did not.	`found = 'refund' in page_text`
A price changes.	Change the number. Keep the old one.	`UPDATE price; INSERT audit_log`

All of those are state changes. Precise. Repeatable. Anyone can check them.

Judgment

Judgment is different. Judgment takes the messy things: numbers, words, documents, pictures, sounds, big data. It reads them on its own, with no person watching. That is the whole point. It is the smallest place where AI fits. Then it writes down one small, real thing.

What do judgments look like

A yes or a no

Is this email about a delivery date?

Yes.

One from a short list

Which account does this payment belong to?

Trucking A.

A few labeled fields

Who was this receipt from, for how much, and when?

Acme Supply. $412. March 3.

Judgments come in sizes

Ask a small question and almost everyone gives the same answer. Ask a big one and two experts will disagree.

Small judgments can be made at companies without training. Big judgments require experts. The bigger the judgment, the more the answers drift apart, and the less you can promise about the box.

Bigger judgments are possible. Self-driving cars prove it. They also show the bill: billions in investment, years of testing, and insurers who measured every mile before they would underwrite it. That is what a big judgment in a box costs. Plan for that bill, or keep the box small.

Four ways to use judgment

closed loop

judgment in a box

open loop

judgment everywhere

Everyone using AI today is in one of these four columns.

Judgment in a box	AI in a harness	The magic box	Not yet
A system is designed to intelligently use judgment Who: Daniel Ackerman. Old school AI experts. "The magic box" skeptics.	**AI use of judgment, inside of a state-based system** Who: serious builders. Graph-based agent systems. Coding agents with a harness. AI labs.	**AI use of judgment** Who: most of the AI labs (Anthropic, OpenAI, etc.) have, will, or still represent this idea. "You type one thing and magic happens."	**Human use of judgment** Who: careful operators. Accountants. Regulated businesses. Anyone burned once. People who run on Standard Operating Procedures, which is the same system as judgment in a box.
A closed loop is written in code, like a Standard Operating Procedure. We decide ahead of time where a judgment is needed, human or AI. Each one is precise, measurable, trackable. Every step is known in advance.	Best case: an expert defines the goal with care. A harness keeps some real state, and a person watches it change. Underneath, it is still tokens bumping into state and editing it.	Best case: you describe the job well, the model is strong, it does the job. You read the result. It is right. It was fast. The loop lived in the tokens the whole time.	People do it by hand. Judgments are written on paper or in a sheet. Humans can make bigger judgments than any box. The trade is opex.

Judgment in a box	AI in a harness	The magic box	Not yet
Good • Same answer every time. • Poor judgments can be tracked and improved over time. • Cheaper to run. Less AI cost. • Safe to build on top of.	**Good** • The healthiest open loop. • Some real state in the harness. • An expert may be able to catch the worst. • Handles surprises, sometimes.	**Good** • Fast to try. • Feels like magic. • Sometimes it works.	**Good** • Right, usually. • Someone is accountable. • Same shape as judgment in a box. Easiest camp to move from. • Some judgments cannot be written as a Standard Operating Procedure. Taste. Art. Intuition.
Who controls when and where judgments happen?			
We do. Only where we put them. You decide those spots, and the state around them, ahead of time.	The harness controls a few. The agent controls the rest, countless judgments, one on top of another.	The agent. Every step is a judgment. Nobody knows which ones were made, or when.	A person. Big judgments, and many of them.
How, and who, changes the state? (The rows. The spreadsheet.)			
Code does. Every change is written in advance. Slower to build the first time.	The AI does. It edits a database, calls a tool, uses a connector. Each touch can go wrong.	The AI does, with nothing between it and your data.	A person does. They type it in.
How do we know when a bad judgment happened?			
Yes. Each one is written down. Point at it. Fix it. Rerun.	Tricky, sometimes impossible. You read tokens to check tokens.	No. Nobody can say why. Judgments build on each other, so one bad one spreads to the rest.	A person rechecks a person, every single time. Or you catch it after the consequences.

Judgment in a box	AI in a harness	The magic box	Not yet
How is it made reliable, and how reliable is it?			
Each box is tested on your real documents before it goes in. Same answer every time. Cheap to run. Banks and regulators understand it, because it is how they already work.	An area of active research, with countless proposals. Getting better. Not yet what a business needs for money, contracts, or compliance. And expensive: tokens on every run, an expert on every check, and a harness to build and maintain.	It is not made reliable. Every run costs tokens. Insurers, banks, and regulators do not trust systems that live only in this space.	Either a Standard Operating Procedure, or exclusively human judgment, or a mix of both. Expensive. Requires expertise or training.

> A lot of people will sell you on the dream of, oh, the AI can just figure it all out, "no mistakes." The simple truth is: no. That's not reliable. It's not at the level where I'd put my reputation or money on the line for it. Human judgment, or careful judgment in a box, is necessary for reliable systems.
>
> Daniel Ackerman

The same job, four ways

A $4,000 payment lands. Where does it go in the books? Should anyone be told?

state judgment person

Judgment in a box

1. **state** Payment lands.
2. **judgment** Which account? `3b, 90%`.
3. **person** Optional: approve before it is written.
4. **state** Row written. P&L updates.
5. **judgment** Tell the owner? no.
6. **state** Done. Every step written down.

AI in a harness

1. **person** Expert writes the playbook.
2. **judgment** AI reads payment, books, playbook.
3. **state** Harness logs step one.
4. **judgment** AI edits the books.
5. **state** Harness logs step two.
6. **judgment** AI decides whether to email.
7. **person** Expert reads twenty pages. Looks right.

The magic box

1. **person** "Handle new payments. Tell me if anything matters."
2. **judgment** AI reads.
3. **judgment** AI decides.
4. **judgment** AI edits the books.
5. **judgment** AI emails someone.
6. **state** "Done." Books changed. Nobody can say why.

Not yet

1. **person** Opens the statement.
2. **person** Decides the account.
3. **state** Types the row.
4. **person** Checks the P&L. Decides whether to email.
5. **state** Done. Time spent on expensive judgment.

Look at the left column. Every judgment was one small question. Every answer landed as state. If step 2 was wrong, you can find it, fix it, and rerun. Nothing else moved on a guess.

> We know that it made this judgment call. And if that judgment call's wrong, then we need to go inform the next five steps: hey, this was a bad call. Let's fix this.
>
> Daniel Ackerman

This is not new. Before LLMs, this was the only way AI was used. "Build a neural network to judge if this loan gets money." That is a judgment in a box, using big data and careful training to achieve accuracy.

Your company's Standard Operating Procedures work the same way, with the judgments spelled out. So does the way you revise them.

What is new: AI makes it easier to build bigger systems, with more state changes, one after another. And LLMs let you build more kinds of judgment, easily and reliably, as long as each one stays in a box.

Building it

state	judgment
causes other changes	tested, layered, undoable

Why coded systems exist

Certain state changes cause other changes. A package is marked delivered, so a text goes out. A payment is marked paid, so the balance drops. That is the reason software exists at all. These changes are precise. They happen the same way every time.

Train each box

Give a box a thousand real documents and the right answer for each. Run it. Count the misses. Fix. Run again. When it is right 99.99% of the time on your documents, it goes in.

> What judgment can we 99.99% of the time believe that AI can do today and tomorrow?
>
> Daniel Ackerman

Layer them

Put a small box at every seam where a messy thing needs to become a real thing. Many boxes. Each one small. Each one tested on its own.

Let people undo

Because every judgment becomes state, a person can point at it and say: that one was wrong. Two settings, chosen per box.

Small stakes

Write now. Keep a log. One click to undo. Everything downstream reruns from the fixed state.

Big stakes

Write as pending. Nothing downstream fires until a person approves.

In Conclusion

state	**judgment**
grows more valuable every day	stays in its box

Systems built this way will, in the longest run, be far more valuable than a self-grown agent that runs around and jumps between this and that.

> You'd rather have a less intelligent lawyer that is reliably correct than a brilliant lawyer who makes mostly genius and, once in a blue moon, critical errors in their chain of judgments.
>
> Daniel Ackerman

> Once you have it all in state, you can reliably build extremely powerful software.
>
> Daniel Ackerman

state	**judgment**
what is written down	deciding what to write

1. A database holds state. An AI makes judgments.
2. Put each judgment in a small box. Test it. Write the answer down as state.
3. State changes cause other changes. That is software.
4. Big judgments stay with experts. Small ones go in boxes.
5. If AI must talk to AI, put state in between.