THIRD EDITION · FREE TO READ ONLINE

Intelligible Intelligence

A Framework for AI — How We Avoid Chatastrophy

Ezra Lewin Davies

Every answer an AI gives you rests on billions of calculations that nobody can fully explain. This book is a blueprint for changing that. It takes the discipline that made computers trustworthy, the discipline my grandfather, Douglas Lewin, taught as Professor of Digital Processes at Brunel University, and turns it into four principles for AI: transparency, verifiability, boundedness and composability.

Fifteen chapters · Every claim sourced · Free to read, no sign-up

DEDICATION

For Douglas Lewin, who taught, without ever quite saying so, that a system you cannot explain is a system you do not yet understand.

Introduction: A Blueprint for Intelligible Intelligence

> In the few seconds it takes an AI system to write you a three-hundred-word answer, it carries out tens of trillions of calculations: more than every person on Earth, working by hand at one sum a second, could get through in an hour and a half.[N1] Then it gives you the answer. What it cannot give you, and what nobody yet can, is the reason.

That is the problem this book is about. AI now drafts contracts, screens job applicants, supports medical decisions and shapes what millions of people read. ChatGPT alone went from launch to an estimated 100 million users in two months, and by late 2025 its maker reported 800 million people using it every week.[N2] Most of those people assume that someone, somewhere, understands how it reaches its answers. In the sense that matters for trust, nobody fully does.

The man who could explain every wire

It was not always this way. My grandfather, Douglas Lewin, belonged to the generation of engineers who worked out how to build computers that could be trusted. Born in 1931, he became Professor of Digital Processes at Brunel University[1,N3] and wrote the textbooks other engineers learned from: on switching circuits in 1968, on the theory and design of digital computers in 1972, on computer-aided design in 1977 and on logic systems in 1985. His computer book was rebuilt twice, in 1980, with an American edition, and again in 1992, as the microprocessor transformed the field around it.[,2] Its chapters climb the whole ladder of the subject, from how a machine stores a program and does its arithmetic to memory, networks, advanced architectures and the engineering of machines that must not fail.[,3]

The discipline he taught was simple to state and brutally hard to practise: you do not trust a machine until you can explain it. Every gate, every signal and every step from input to output had to be accounted for and checked. That discipline is

the reason we allow computers to fly aircraft, run hospital equipment and control power stations.

He could explain every wire in his machines. We cannot fully explain a single answer from ours.

An electrician's view

I came to this from a different direction. I trained as a geographer and I work as an electrician. When I finish a circuit I cannot simply announce that it is safe. Under BS 7671, the UK wiring regulations, I have to prove it, test by test, and sign a certificate that records every result.[4] In my trade, "trust me, it's fine" is not evidence. This book asks why we accept it as evidence for artificial intelligence.

The blueprint

The gap can be closed. Not perfectly, but far further than current practice suggests. By *intelligible* I do not mean simple. I mean a system whose behaviour can be traced to causes, whose limits are stated in advance, and whose claims a competent outsider can check. That comes down to four principles:

> **Transparency.** Show the process behind an answer, not just the answer: what the system drew on, what steps it took, and where its confidence is weak.

> **Verifiability.** Every claim made by or about the system can be tested by someone independent. An answer that cannot be checked should not be trusted.

> **Boundedness.** The system works within stated limits, and says so when a question falls outside them.

> **Composability.** The system is characterised well enough to be connected to other systems and to human oversight without surprises.

None of these is new. Each is borrowed from the engineering my grandfather taught. What is new is applying them to machines whose inner workings resist inspection. The chapters that follow do that for three audiences: for engineers, as design practice; for policymakers, as criteria to audit and regulate against; and for everyone else, as a standard to demand.

Why "chatastrophy"

The subtitle's pun is deliberate. *Chat* is how most people meet AI; *catastrophe* is what we are trying to avoid; and the distance between the two is shorter than it looks. In these pages you will meet a lawyer who asked a chatbot to confirm the court cases it had invented, an airline that argued its chatbot was responsible for its own words, and a hospital alert system that missed two-thirds of the patients it was built to catch. You will also meet failures from my grandfather's world (a radiation machine, a rocket and a spacecraft) that teach exactly the same lesson. None of them failed because the technology was mysterious. **Each failed because a system with knowable limits was trusted beyond them.**

How this book is organised

Part I goes back to the engineering my grandfather taught and explains, as plainly as possible, how modern AI is built instead. An **Interlude** then follows a single question through a real neural network, calculation by calculation. **Part II** sets out the four principles, one chapter each. **Part III** examines how systems fail, through ten documented cases. **Part IV** turns the framework into practice for builders, regulators and users, and ends with a design for a genuinely intelligible AI system. A checklist, glossary, timeline and full source notes are at the back. Every figure in the book is labelled with how it is known (measured, reported, documented, estimated or calculated), and the key is in Appendix F.

One confession before we start. This book about trusting AI was written with the help of an AI. That is either a scandal or the best possible test of the framework,

depending on how it was done. Appendix F tells you exactly how, principle by principle, so you can judge for yourself.

The machines my grandfather designed earned trust because they could be understood. The machines we are building now must earn it the same way. This book is the blueprint.

PART I

The Inheritance

CHAPTER ONE

Theory and Design

What My Grandfather Built

IN THIS CHAPTER

Digital systems became trustworthy because engineers found a way to describe their behaviour completely and check it independently. This chapter explains how, from Boole's algebra and Shannon's switching circuits to truth tables, state machines and formal verification, and why "completeness" is the property modern AI lacks.

In 1994 a mathematics professor in Virginia noticed that one of the most advanced microchips on Earth was getting certain division sums wrong. The fault lived in five missing entries in a table. It cost Intel $475 million. The industry's answer was not to shrug at a rare error but to change how chips were checked, using mathematical proof. This chapter explains why my grandfather's discipline could make that kind of promise, and why modern AI cannot yet.

BY THE NUMBERS

Why engineers prove instead of test

3.4×10^{38}

Possible pairs of inputs to a circuit that adds two 64-bit numbers (2^{128}). Checking a billion a second would take about 10^{22} years, hundreds of billions of times the age of the universe. That is why chip designers prove correctness instead of trying every case.

CALCULATED

208 billion

Transistors on a single Nvidia Blackwell AI chip, announced in 2024. Nobody checks them one by one; they are trusted because of layered specifications.[N4]

REPORTED

5

Missing entries in the Pentium's division table that cost Intel a $475 million charge in 1995 (see this chapter).

DOCUMENTED

21

Claude Shannon's age when his 1937 master's thesis showed that Boole's algebra describes switching circuits exactly.

DOCUMENTED

Digital design, the field my grandfather wrote about, rests on one basic move, repeated again and again: a claim about behaviour, reduced to a claim about structure, reduced further to a claim you can check with a pencil. A truth table is not an opinion. A state diagram is not a suggestion. If you built the circuit as specified, it would do exactly what the diagram said it would do, in every case the diagram covered. If a case wasn't covered, that was a gap you were responsible for closing before the thing shipped, not a mystery you were entitled to shrug at afterwards.

Artificial intelligence has, in many ways, walked away from that discipline. Not through carelessness, but because the systems resist it. To see what was given up, you have to see what was there to begin with.

From logic to switches

The story starts with a mathematician who had no interest in machines at all. In 1854 George Boole, then a professor at Queen's College, Cork, published *An Investigation of the Laws of Thought*, in which he showed that logical reasoning could be written as algebra.[1] Statements could be true or false. They could be combined with AND, OR and NOT. And the combinations obeyed rules that could be manipulated like the equations of school algebra.

For more than eighty years, Boolean algebra was a curiosity of logic and philosophy. Then, in 1937, a 21-year-old student at MIT named Claude Shannon wrote a master's thesis showing that Boole's algebra described electrical switching circuits exactly.[2] A switch is either closed or open. Two switches in series conduct only if both are closed, which is AND. Two in parallel conduct if either is closed, which is OR. Suddenly, the question "what will this tangle of relays do?" became a question you could answer on paper, and the question "what is the simplest circuit that does this job?" became an algebra problem.

As an electrician, I find this oddly moving. The two-way lighting circuit on every British staircase, where either switch can turn the landing light on or off, is a physical implementation of a logical function called exclusive OR (or its inverse, depending on how the switches were fitted): flip either input and the output changes. Every apprentice learns to wire it. Very few are told that it is Boolean algebra in copper.

Shannon's observation, that the true and false of logic can be mapped onto the on and off of a circuit, is the founding act of the discipline my grandfather worked in. Everything else in digital design is built on it.

THE SCIENCE

What a truth table guarantees

A logic circuit with *n* binary inputs can receive exactly 2^n different combinations of input. A truth table lists every one of them, along with the output the circuit must produce. For a circuit with 3 inputs there are 8 rows. For 10 inputs, 1,024 rows. For 20 inputs, just over a million.

That number grows quickly, but it is always finite, and that is the crucial point. A truth table does not describe what a circuit *usually* does, or what it did on the examples somebody happened to test. It describes what the circuit does for every input it could ever receive. Two circuits with identical truth tables are, for all logical purposes, the same circuit, so an engineer can replace a complicated design with a simpler one and *prove*, not merely hope, that nothing has changed.

Completeness

What Boolean algebra gave engineers was not just a way to describe circuits but a way to prove things about them. Two circuits that looked different on paper could be shown, algebraically, to be logically identical. A circuit's output could be predicted for every combination of inputs, not sampled or estimated but listed completely.

This is the property I want to dwell on, because later chapters will show that artificial intelligence does not have it and cannot simply be given it. The property is **completeness**. A truth table is complete. When you have one, there is no input left over whose behaviour is a matter of opinion.

From gates to machines that remember

A logic circuit whose output depends only on its present inputs is called combinational. On its own it cannot count, store a number or follow a sequence of steps. For that you need memory, and in digital design memory comes from the flip-flop: a small circuit that holds one bit of information until it is told to change.

Once you have memory, you have *state*. The central tool for reasoning about state is the finite state machine: a formal description of a system that can be in one of a limited, listed number of states, that moves between states according to precisely specified rules triggered by inputs, and that produces outputs determined by its current state and inputs. A traffic light controller is a state machine. So is a lift, a vending machine or the washing machine in your kitchen. A processor's control unit is, at a high enough level of description, a very large state machine.

The power of the idea is that it lets you reason exhaustively about behaviour that unfolds over time. You are no longer asking only "what happens on this input?" but "what happens on this *sequence* of inputs, however long or strange?" Because the number of states is finite and every transition is written down, you can check every state and every transition and know for certain whether the machine can ever reach a state you don't want it in. The word "finite" is easy to skim past as a technical footnote. It is the whole point.

Theory and design

The pairing in two of my grandfather's titles, *theory and design*, is a good summary of the whole field. The two halves do different work, and you need both. The theory (Boolean algebra, combinational and sequential logic, state machines) tells you what is possible to build and gives you a precise language for describing it. The design tells you how to move from a specification of the behaviour you want to an actual arrangement of gates and flip-flops that realises it, and how to check, before any soldering is done, that the arrangement matches the specification.

Checking at every level is the habit of mind that matters most for this book. Digital systems are built from smaller subsystems whose behaviour has already been fully characterised, and those subsystems are combined according to rules that preserve the characterisation. An adder, once verified, can be trusted inside a larger arithmetic unit, because the adder's verification covered every input and the rules for combining verified parts are themselves provable. You do not re-derive

the correctness of addition every time you use an adder. You trust the boundary. Engineers call this **composability**, and it will become the fourth property of the framework in Part II.

Layers: how complexity stays intelligible

There is one more idea in digital design that matters enormously for this book, because it explains how engineers kept systems intelligible even as they grew beyond anything a single person could hold in their head.

In 1965 Gordon Moore, later a co-founder of Intel, observed that the number of components that could be economically placed on an integrated circuit was doubling roughly every year, a rate he later revised to about every two years.[3] "Moore's law" held, roughly, for half a century. The largest chips now contain well over a hundred billion transistors. Nobody can check a hundred billion transistors one by one. So how can anyone trust such a chip at all?

The answer is **abstraction in layers**. Transistors are combined into gates, and the gate is specified by its truth table, so the designer of the next layer never needs to think about transistors again. Gates are combined into adders, registers and multiplexers, each with its own specification. Those are combined into arithmetic units and control units; those into a processor; and the processor is described by its instruction set, the precise list of operations a programmer can rely on. At every layer, the people working above it depend only on the specification of the layer below, not on its insides.

This is exactly why the specifications matter so much. The whole tower is only as sound as the promise each layer makes to the one above. When a layer's behaviour departs from its specification, as the Pentium's divider did, the error propagates upward into software that had every right to trust it.

Hold on to this picture of a tower of trustworthy promises. Chapter Three will show that a neural network is not built this way. Its "layers" are layers of numbers,

not layers of specifications, and nobody wrote down what any of them promises.

When verification was not done: the Pentium bug

The digital world does not always live up to its own standards, and the most famous lapse shows why the standards matter.

In 1994 Thomas Nicely, a mathematics professor at Lynchburg College in Virginia, noticed that his computer was producing slightly wrong answers when dividing certain numbers. The cause turned out to be Intel's new Pentium processor. To speed up division, the chip used a lookup table, and five entries in that table were missing. For most divisions it made no difference. For a small set of rare input combinations, the result was wrong in the fourth or later significant digit.[4]

Intel initially argued that ordinary users would almost never meet the error. That was statistically true and commercially disastrous. Customers did not want a processor that was right almost all of the time. They wanted one that was right. In January 1995 Intel announced a charge of $475 million to replace the affected chips.[5] Intel then invested heavily in *formal verification*, the use of mathematical proof, carried out by software, to show that a circuit design meets its specification for every possible input.[6]

The lesson is not that digital engineers are infallible. It is that their discipline contains a clear standard, completeness, against which a failure like this can be named, measured and fixed. The error lived in five cells of a table. Once the table was checked properly, the fix was certain.

The cost of rigour, and why it was worth paying

None of this was free. This discipline took years to learn and great care to apply. A circuit that can now be sketched and simulated in software in an afternoon might once have taken weeks of truth-table work and breadboard testing. The temptation to cut corners, to build something that seemed to work after a handful

of tests, was always present. It was resisted because everyone in the field understood what happened when you gave in to it: circuits that worked in the lab and failed in the field, on the one input combination nobody thought to try.

This is the inheritance I want to carry into the rest of this book, stated as plainly as I can: **rigour is not the enemy of getting things done. It is what allows you to build things large and complex enough to matter** without the whole structure collapsing the first time reality serves up a case you did not anticipate. My grandfather's generation earned the right to put digital systems into aircraft, medical devices and power grids because they built a discipline in which trust was not requested but demonstrated, case by case, gate by gate, proof by proof.

Artificial intelligence is now being deployed into many of the same high-stakes settings without having done the equivalent work. Largely, that is not a moral failing on the part of the people building it. It follows from the fact that the systems themselves resist the exhaustive description that digital logic permits. A modern language model chooses each word from a vocabulary of tens of thousands of possibilities, and it reads inputs thousands of words long. Even a vocabulary of 50,000 tokens and an input of just 20 tokens gives $50{,}000^{20}$, roughly 10^{94}, possible inputs. That is more than the estimated 10^{80} atoms in the observable universe.[7] There is no truth table, and there cannot be one.

That is the central technical fact this book has to reckon with. It means the inheritance from my grandfather's world cannot be applied directly. It has to be translated, keeping its spirit where its exact methods cannot survive the crossing. That translation is the project of this book.

KEY POINTS

- Boole (1854) showed logic could be written as algebra. Shannon (1937) showed that algebra describes switching circuits exactly.
- A truth table is **complete**: it specifies a circuit's output for every possible input, which makes proof possible rather than just testing.
- Finite state machines extend completeness to behaviour over time, because the number of states is limited and every transition is written down.
- Verified parts combined by provable rules give **composability**: you can trust a component's boundary without re-checking its insides.
- The 1994 Pentium division bug cost Intel $475 million and pushed the industry towards formal verification. Even rare errors matter when a standard of completeness exists.
- Modern AI systems have far too many possible inputs for any truth table. The discipline must be translated, not copied.

WHERE THIS CHAPTER STOPS

This chapter simplifies digital design considerably. Real circuits also involve timing, electrical and manufacturing effects that a truth table alone does not capture (Chapter Two deals with timing). Formal verification is powerful but not magic: it proves that a design meets its *specification*, and a wrong specification can be proved correct. This book says nothing about the contents of my grandfather's books beyond their titles and the publisher's chapter list; the account of digital design here is a general one, not a summary of his work.

CHAPTER TWO

The Discipline of Rigour

Lessons from Engineered Systems

IN THIS CHAPTER

Good engineering does not rely on testing harder. It relies on specifications that state limits as clearly as capabilities, on structure that makes hazards impossible, and on redundancy that assumes something will eventually fail. Three real failures (the Therac-25 radiation machine, the Ariane 5 rocket and the Boeing 737 MAX) show what happens when each discipline is missing.

> Six patients given massive overdoses by a machine built to heal them. A rocket that destroyed itself about thirty-seven seconds after lift-off because a number grew too big for the box it was kept in. Three hundred and forty-six people killed in two aircraft that trusted a single sensor. None of these failures was caused by a mystery. Each was caused by a discipline that someone skipped.

BY THE NUMBERS

Limits, written down in advance

32,767

The largest number a 16-bit signed integer can hold. Ariane 5's horizontal-velocity value exceeded it, and the rocket was lost.

DOCUMENTED

0.4 s

The maximum time BS 7671 allows for a fault on most final circuits in British homes to be disconnected. A limit, stated in advance and measured.

DOCUMENTED

346

People killed in the two Boeing 737 MAX crashes of 2018 and 2019, where a flight-control function relied on one sensor.

DOCUMENTED

~30×

How much more reliable three independent voting copies are than one, when each fails 1 time in 100. Share the flaw, and the gain vanishes.

CALCULATED

Engineers tell a story about a circuit that passed every test on the bench and failed on its first day in the field. It sounds like a parable. It is not. There is a real version, it is well documented, and people died.

Therac-25: a race condition that killed

The Therac-25 was a radiation therapy machine built by Atomic Energy of Canada Limited (AECL) and used to treat cancer patients in the United States and Canada. It could operate in two modes: a low-power electron beam applied directly, or a very high-power beam fired at a metal target to produce X-rays. Between June 1985 and January 1987, six patients received massive radiation overdoses, in some cases roughly a hundred times the intended dose. Several died of their injuries.[1]

The definitive investigation, by Nancy Leveson and Clark Turner, found several interacting causes. One of the most important was a **race condition**. If an

experienced operator entered the treatment details, then quickly edited them, changing the mode, within a few seconds, the software could accept the new mode on the screen while the machine's hardware was still set up for the old one. The result was the high-power beam delivered without the target in place.

Several features of the case matter for this book. Earlier Therac models had independent *hardware* interlocks that physically prevented this kind of mismatch; the Therac-25 relied on software alone. The software reused code from earlier machines, where hardware interlocks had silently masked its flaws. The failure appeared only under a specific timing of operator keystrokes that ordinary testing did not reproduce, and it happened more often as operators became *faster and more practised*. When patients reported burning sensations, the manufacturer at first insisted that an overdose was impossible.

> THE SCIENCE
>
> **What a race condition is**
>
> In any electronic or software system, signals and operations take time: nanoseconds in a circuit, milliseconds or seconds in software. If the correct outcome depends on one event happening before another, and nothing in the design *enforces* that order, then under some conditions the events will arrive in the wrong order. The system will do something its designers never intended and never tested for, because their tests happened to produce the "right" order every time.
>
> You cannot test your way out of a race condition with confidence. The failure occurs only under a particular combination of timing, and your test setup may never happen to reproduce it. A system can pass a thousand tests and still contain the flaw.

This should sound familiar to anyone who follows AI safety. A model can behave impeccably on every evaluation its developers ran, then do something alarming the first time a user, deliberately or by accident, finds an input the evaluations did not cover.

Structure, not luck

The digital-design answer to race conditions is not more testing but more structure. In **synchronous design**, every change of state is governed by a shared clock signal, and timing margins are calculated rather than guessed. The question "could these signals arrive in the wrong order?" stops being something you can only probe by sampling and becomes something you can answer by calculation from the specification. You do not eliminate the hazard by getting lucky in the lab. You eliminate it by making it structurally impossible, or by proving that it cannot occur within the operating conditions you have specified.

The Therac-25's earlier siblings had exactly this kind of structural protection in their hardware interlocks. Removing the structure and trusting the software to be correct turned a latent flaw into a lethal one.

Specification as the load-bearing wall

This points to what I think is the single most important idea to carry from engineering into AI. In good engineering, the **specification** (the precise statement of what the system should do and, just as importantly, what it must not do and how it should fail safely) is not an afterthought or a marketing document. It is the load-bearing wall of the whole enterprise. Design, testing and verification all serve one purpose: establishing that the thing built matches the thing specified.

Truth tables that list every case, state diagrams that show every transition and timing diagrams that show every constraint can look like pedantry to an outsider. Why write down every case when the normal cases are obvious? The answer, learned the hard way, is that systems rarely fail on the normal cases. They fail on the cases nobody specified, because specifying them felt unnecessary or tedious or was forgotten under deadline pressure. A truth table with a blank cell is not a small oversight. It is a hole in the hull.

Limits stated as clearly as capabilities: Ariane 5

A related discipline is less celebrated but just as important: stating limits, not only capabilities. Every electronic component comes with a datasheet, and every datasheet includes a table of *absolute maximum ratings*: the voltage, current and temperature beyond which the manufacturer makes no promises at all. Sometimes the honest statement is "behaviour undefined". That phrase is not an evasion. It is a precise engineering statement that tells the next engineer exactly where their own responsibility begins.

On 4 June 1996, the first Ariane 5 rocket veered off course about 37 seconds after lift-off and was destroyed. The inquiry board, chaired by the French mathematician Jacques-Louis Lions, traced the failure to software in the rocket's inertial reference system, which had been reused from the smaller Ariane 4.[2] One routine converted a 64-bit floating-point number, related to the rocket's horizontal velocity, into a 16-bit signed integer, which can hold values only up to 32,767. On Ariane 4 the value had never exceeded that limit. Ariane 5 was faster, the value overflowed, and the software shut down.

Three details make Ariane 5 a perfect lesson for AI. First, the component was not faulty in its original setting. It was operating *outside the envelope it had been designed and tested for*, because nobody had rechecked that envelope when the component moved to a new system. Second, the routine that failed served no purpose after lift-off; it was left running for convenience. Third, the backup system ran identical software, so it failed in exactly the same way at almost exactly the same moment. The redundancy provided no protection, because the two copies shared the same flaw.

Hold on to that phrase, *outside the envelope it was designed and tested for*. It will return in Chapter Seven as the central problem of AI boundedness.

Limits in my own trade

My own trade works the same way. BS 7671 is not mainly a list of things to do. Much of it is a list of limits. For most final circuits on a typical 230-volt supply

with a TN earthing arrangement, the most common in British homes, a fault to earth must be cleared within 0.4 seconds, because longer than that and the risk of a fatal shock rises sharply.[3] Cables are rated for a maximum conductor temperature, and every correction factor for grouping, insulation and ambient heat exists to keep them below it. Test results are recorded against maximum and minimum values printed in tables. An electrician is not asked "does it work?" An electrician is asked "is every measured value inside its stated limit?"

That is a different question, and a much better one. A circuit can "work" (the lights come on) while its earth fault loop impedance is too high for the breaker to trip fast enough in a fault. You would only discover the difference on the worst day of someone's life. The limit, stated in advance and checked by measurement, is what turns "it works" into "it is safe".

Redundancy as design, not waste

One more idea from this tradition deserves a place here because it recurs throughout the book: redundancy is not always waste.

In 1956 the mathematician John von Neumann published a paper with a title that could serve as a motto for this book: "Probabilistic logics and the synthesis of reliable organisms from unreliable components".[4] He showed that you can build a system more reliable than any of its parts by running several copies and letting them vote. The engineering version, **triple modular redundancy**, runs three copies of a circuit in parallel and takes the majority answer. A single failure is outvoted.

THE SCIENCE

Why voting works (and when it doesn't)

Suppose each of three independent copies of a circuit fails with probability p on a given task. A majority vote gives the wrong answer only if at least two copies fail at once. The probability of that is $3p^2(1 - p) + p^3$.

If $p = 0.01$, the voted system fails with probability of about 0.0003, some thirty times better than a single copy. If $p = 0.001$, it is about 0.000003, more than three hundred times better.

The calculation depends on one assumption: that the failures are *independent*. If all three copies share the same design flaw, as Ariane 5's two identical inertial reference systems did, they fail together and voting achieves nothing. This is why the Space Shuttle's flight computers included a fifth, backup computer running software written separately by a different contractor: diversity, not just duplication.[5]

That assumption of independence will matter a great deal for AI. Many AI systems are built from the same few underlying models, trained on overlapping data, and they tend to share the same blind spots. Asking one model to check another can look like redundancy while providing much less of it.

When a mature industry forgets: the 737 MAX

It would be comforting to think these lessons, once learned, stay learned. They do not.

The Boeing 737 MAX included a flight-control feature called MCAS (Manoeuvring Characteristics Augmentation System), which could automatically push the aircraft's nose down if it sensed that the nose was pitched too high. In the versions involved in the accidents, MCAS acted on data from a single angle-of-attack sensor, even though the aircraft carried two. When that one sensor failed, MCAS repeatedly pushed the nose down. Lion Air Flight 610 crashed on 29

October 2018 and Ethiopian Airlines Flight 302 on 10 March 2019. A total of 346 people died.[6]

The investigations found several problems, including how MCAS was described to pilots and regulators and how its authority was later increased without a matching review. But at the core sat a violation of one of the oldest rules in safety engineering: a function with the power to cause a catastrophe depended on a single point of failure. The worldwide fleet was grounded for about twenty months.

I include the 737 MAX not to single out one company but because it shows something important. Even in aviation, the most safety-conscious engineering culture humans have built, commercial and schedule pressure can erode discipline. If that can happen there, it can certainly happen in a young industry racing to release new AI products every few months.

Checking your own diligence

The humility of designing for the failure of your own diligence, instead of relying on diligence alone to prevent failure, is to my mind the most mature idea in the whole discipline. It is tempting to believe that enough care in design and testing removes the need to plan for failure. Engineering rejected that temptation explicitly, as a matter of professional practice. You build the best system you can, and then you build a second layer that assumes the first will eventually fail, because in any large, long-running system something eventually does.

In my own trade, that second layer has a name. The residual current device (RCD) in a consumer unit is not there because the wiring is expected to be faulty. It is there because the wiring *might* be damaged some day, by a nail through a cable or a worn appliance lead, and when that happens the RCD disconnects the supply fast enough to make a fatal shock much less likely. BS 7671 calls it "additional protection". Additional to what? To everything else we did right.

What survives the crossing

What, of all this, can be carried into artificial intelligence, given that AI systems resist the exhaustive verification that made digital logic trustworthy? The honest answer, which the rest of this book works out in detail, is: **not the methods, but the disciplines.**

We cannot build a truth table for a language model. We cannot list its states. But we can insist that specifications state limits as rigorously as capabilities. We can insist on structural protections against known classes of hazard instead of hoping testing will catch every case. We can insist on redundancy that is genuinely *diverse*, and on defence in depth rather than trust in any single safeguard. And we can insist, most fundamentally, that the burden of proof runs the right way: a system earns trust by showing, with evidence a competent outsider can inspect, that it does what it claims and stops where it says it stops. We do not grant it trust by default and wait to be surprised.

> **KEY POINTS**
>
> - **Therac-25 (1985–87):** a software race condition, combined with the removal of hardware interlocks, led to massive radiation overdoses. Testing did not find it; structure would have prevented it.
> - A **specification** must state limits as clearly as capabilities. Blank cells in a specification are where failures live.
> - **Ariane 5 (1996):** reused software failed when operated outside the envelope it was designed for, and an identical backup failed identically.
> - **Redundancy** works only when failures are independent. Diversity matters more than duplication.
> - **Boeing 737 MAX (2018–19):** a single point of failure in a mature industry shows that discipline erodes under commercial pressure.
> - What transfers to AI is the discipline, not the method: stated limits, structural safeguards, diverse redundancy and evidence-based trust.

WHERE THIS CHAPTER STOPS

Each case here had several causes, and short summaries necessarily simplify them. Readers should consult the cited investigation reports before drawing detailed conclusions about any individual company or person. The redundancy calculation assumes independent failures of equal probability, an idealisation. The chapter argues by analogy from engineered systems to AI; Chapter Three explains where that analogy breaks down.

CHAPTER THREE

From Logic Gates to Language Models

How Modern AI Is Actually Built

IN THIS CHAPTER

Modern AI is not programmed rule by rule. It is trained: billions of numerical settings are adjusted until the system's outputs match examples. This chapter explains, without equations beyond simple arithmetic, how neural networks and language models work, why they give statistical rather than logical guarantees, and three scientific findings that matter for trust: shortcut learning, adversarial examples and emergence.

> In 2012 a neural network with 60 million adjustable numbers stunned the field of computer vision. Eight years later, GPT-3 had 175 billion. By 2024 Meta was training a single model on 15.6 trillion fragments of text, using up to 16,000 specialised chips. In twelve years the scale of these systems grew by a factor of thousands. Our ability to explain them did not keep pace.

BY THE NUMBERS

The scale of modern AI

60 million

Parameters in AlexNet, the 2012 network that turned image recognition upside down.[N5]

MEASURED

175 billion

Parameters in GPT-3 (2020), arranged in 96 layers. Training it took 3.14×10^{23} operations: about 1.2 million years if every person on Earth did one sum per second.[N6]

CALCULATED

3.8×10^{25}

Operations used to train Meta's Llama 3.1 405B on 15.6 trillion tokens. That is about 150 million years of everyone on Earth calculating, and more text than one person could read in 270,000 years.[N7]

CALCULATED

4–5× a year

Growth in the computing power used to train frontier models, according to Epoch AI's analysis of the 2010s and early 2020s.[N8]

ESTIMATE

86 billion

Neurons in a human brain. Its connections, estimated at around a hundred trillion, still outnumber the parameters of any published model, though the comparison is loose.[N9]

ESTIMATE

Somewhere between my grandfather's *Theory and Design of Digital Computers* in 1972 and the tools that now write emails, draft contracts and hold conversations that feel uncannily like talking to a person, computing changed its foundations. It stopped being purely a discipline of *specification* and became, alongside that, a discipline of *statistics*. Nothing later in this book makes sense without understanding what changed, and what that change costs us.

An old idea with a surprising origin

The first mathematical model of a neuron was, in effect, a logic gate. In 1943 the neurophysiologist Warren McCulloch and the logician Walter Pitts published a paper arguing that idealised nerve cells, each either firing or not firing, could compute logical functions such as AND and OR.[1] The two traditions this book is about, logic and learning, share a birth certificate.

What changed was *where the rules come from*. In 1958 the psychologist Frank Rosenblatt described the perceptron, an artificial neuron whose connection strengths were not set by hand but adjusted automatically from examples.[2] That is the seed of everything that followed.

Two ways to build a system

Imagine two ways of building a system that decides whether a photograph contains a cat.

The first way, in the spirit of my grandfather's discipline, is to write an explicit specification: rules about edges, shapes and colours that define "cat-like" features, applied in a fixed sequence. You could trace exactly why it answered as it did, because every step is a rule you wrote. Early computer vision was attempted like this and mostly failed. Not because specification is wrong, but because nobody has ever managed to write down, as explicit rules, what makes something look like a cat. There are too many cats, in too many poses, lights and breeds.

The second way is to show a system millions of images labelled "cat" and "not cat", and let an algorithm adjust the internal settings of a large mathematical function, a **neural network**, until it gives the right answer on those examples as often as possible. Nobody writes the rules. Whatever rules exist inside the finished system are found by a search process, guided only by the goal of getting more examples right.

The second method won. In 2012 a neural network called AlexNet, trained on the ImageNet collection of more than a million labelled photographs, cut the error

rate in a major image-recognition competition from about 26 per cent to about 15 per cent, a leap that turned the field.[3] Within a few years, essentially all of modern AI (image recognition, speech recognition, translation and the language models this book is mostly about) was built the second way.

> THE SCIENCE
>
> **How a neural network learns, in four steps**
>
> 1. **Start with random settings.** A network is layers of simple units. Each connection between units has a strength, called a *weight* or *parameter*. At the start these are random numbers, so the network's answers are nonsense.
> 1. **Measure the error.** Show the network an example, see what it outputs, and calculate a single number, the *loss*, measuring how wrong it was.
> 1. **Work out which way to nudge.** Using a technique called *backpropagation*, popularised in 1986 by David Rumelhart, Geoffrey Hinton and Ronald Williams, calculate for every weight whether increasing or decreasing it slightly would reduce the loss.[4]
> 1. **Nudge, and repeat.** Adjust every weight a tiny amount in the helpful direction. This is *gradient descent*, like walking downhill in fog by feeling which way the ground slopes. Repeat billions of times over millions of examples.
>
> At no point does anyone decide what any individual weight should mean. The network ends up with whatever internal arrangement happened to reduce the loss.

How a language model works

A large language model (LLM) applies the same idea to text. Its training task sounds almost trivially simple: **predict the next word.** More precisely, it predicts the next *token*, a word or piece of a word. Given "The cat sat on the", the model produces a probability for every token in its vocabulary. After training, "mat" gets a high probability and "photosynthesis" a very low one.

The model is trained on an enormous body of text, a large fraction of the public internet plus books, code and other sources. Every time it predicts badly, its

weights are nudged. To predict the next word well across billions of sentences about everything, the model is pushed to build internal representations of grammar, facts, arguments and styles. Nobody programs these in. They emerge because they help with prediction.

Almost all current language models use an architecture called the **transformer**, introduced by researchers at Google in 2017 in a paper titled "Attention Is All You Need".[5] Its key mechanism, *attention*, lets every token in a passage draw information from every other token, so the model can connect a pronoun to the noun it refers to fifty words earlier.

When you chat with an AI assistant, it is generating one token at a time, each time sampling from its predicted probabilities, then adding that token to the text and predicting again. A second stage of training, often using human ratings of the model's answers (a method known as reinforcement learning from human feedback), shapes the model into a helpful assistant rather than a mere text-continuer.[6]

Scale

The most striking fact about these systems is their size. OpenAI's GPT-3, described in 2020, had 175 billion parameters.[7] Frontier models since then are generally believed to be larger, though most developers no longer publish exact figures (itself a transparency problem we will return to).

Researchers found that performance improves in a smooth, predictable way as models, data and computing power grow together. In 2020 a team at OpenAI reported that a model's loss falls as a *power law* of each of these quantities.[8] In 2022 a team at DeepMind refined the recipe, showing that a 70-billion-parameter model trained on more data (their "Chinchilla" model, trained on 1.4 trillion tokens) outperformed a 280-billion-parameter model trained on less.[9] These **scaling laws** are why so much money has gone into ever-larger computers. They

are also, in a sense, an irony: the *average* performance of these systems is highly predictable, while their behaviour on any *particular* input is not.

What a model knows, and how it looks things up

Three further features of modern systems matter for trust, and they are easy to misunderstand.

A model's knowledge has a date. A language model learns from text collected up to some point, its *training cutoff*. It knows nothing directly about events after that date, though it may produce confident text about them anyway. Ask a model about a law that changed last month or a price that moved yesterday, and unless it has some way of looking things up, it is guessing from an old snapshot of the world. That is a boundedness problem hiding in plain sight.

Many systems now look things up. To work around the cutoff and reduce invented facts, many AI products combine a language model with a search step: the system retrieves documents relevant to the question and gives them to the model to read before it answers. Researchers call this *retrieval-augmented generation*, a term introduced by Patrick Lewis and colleagues in 2020.[10] It helps a great deal, especially when the system shows its sources so that you can check them. But it moves the problem rather than removing it. The answer is now only as good as the documents retrieved, and the model can still misread, misquote or ignore them. Retrieval also creates a new interface, between the model and whatever it reads, and Chapter Eight explains why interfaces are where trouble lives.

Some models now "think" before answering. Since 2024 developers have released *reasoning models*, such as OpenAI's o1 series and DeepSeek's R1, which are trained, largely by reinforcement learning, to produce a long passage of step-by-step working before their final answer.[11] Spending more computation at the moment of answering, sometimes called *test-time compute*, substantially improves performance on mathematics, coding and other problems with checkable answers.

It also produces a visible trail of "reasoning" that looks exactly like the transparency this book asks for. Chapter Five explains why that appearance can mislead.

What gets lost

A digital circuit built to my grandfather's standards has a complete description of its behaviour. A trained neural network does not. It has billions of numerical parameters, adjusted by a process nobody designed step by step, producing a function so large and interconnected that no one (not its creators, not the world's leading experts) can look at those numbers and state in advance what the system will output for a new input. You can run the input and see what comes out. You generally cannot derive it beforehand.

This is the single most important technical fact in this book, so I will state it as plainly as I can: **for the systems this book is about, there is no truth table, and there cannot be one.** Not because nobody has got round to writing it, but because these systems are, by the nature of how they are built, not the kind of thing a truth table can describe.

People often call this being a "black box". I resist the phrase slightly, because it suggests all-or-nothing: either you can see inside or you can't. The reality is more textured and more hopeful. As Chapter Five describes, researchers have made real progress in tracing what happens inside these networks. None of it amounts to a truth table. All of it is worth taking seriously.

THE SCIENCE

Why can't we just read the weights?

If every parameter of a model is published, why can't experts simply read off what it will do? Three reasons.

- **Scale.** A model with 100 billion parameters, printed at ten numbers per line and fifty lines per page, would fill 200 million pages.
- **Distribution.** No single parameter "means" anything on its own. A concept is represented by patterns of activity spread across very many parameters, and each parameter takes part in very many concepts (Chapter Five calls this *superposition*).
- **No layered specification.** In a processor, each layer comes with a written promise about what it does. In a neural network, the layers are just stages of arithmetic. Nobody wrote down what any of them is for, because nobody designed them; training shaped them.

So reading the weights is less like reading a circuit diagram and more like trying to understand a city by reading the positions of every brick.

Logical guarantees and statistical guarantees

Here is a distinction worth drawing carefully, because blurring it is one of the commonest sources of misplaced confidence in public discussion of AI.

A verified digital circuit offers a **logical guarantee**: for every possible input, without exception, the output will be as specified. A trained AI model, evaluated on a large test set, offers at best a **statistical guarantee**: on inputs similar to those it was tested on, it will probably behave as expected most of the time, with some measurable error rate.

In casual conversation both come out as "it works". But they differ in kind, not just degree, and the difference matters enormously once you start building on top of the system.

THE SCIENCE

Why statistical errors compound

Suppose an AI agent must complete a task in 20 steps, and at each step it has a 99 per cent chance of getting that step right. If the steps are independent, the chance of getting all 20 right is 0.99^{20}, about 82 per cent. For 100 steps, 0.99^{100} is about 37 per cent. For 500 steps, less than 1 per cent.

A 95-per-cent-reliable step, which sounds good, gives 0.95^{20}, about 36 per cent, over 20 steps.

Real errors are not always independent. Some are caught and corrected; some trigger further errors. So these numbers are a teaching model, not a prediction. But the lesson holds: **statistical reliability does not compose the way logical reliability does.** Two verified logic circuits joined by a verified interface give a verified whole. Two 99-per-cent systems joined together give something you have to measure all over again.

This is not a criticism of the people building AI systems. It is a statement about the mathematics of what they are building. You cannot demand a truth table from a system whose whole reason for existing is to handle problems (recognising cats, writing sentences, answering open questions) that resisted truth tables in the first place. What you can demand is a different set of disciplines, adapted to statistical systems but faithful to the spirit of rigour. The rest of this book sets those out.

Three findings from AI research show why this matters in practice.

Finding one: shortcut learning

A neural network learns whatever pattern reduces its error on the training data. That is not necessarily the pattern its builders intended.

In 2018 John Zech and colleagues trained a neural network to detect pneumonia in chest X-rays gathered from several hospital systems.[12] It performed well on data from the hospitals it had learned from and noticeably worse on data from a

hospital it had not seen. Investigating, the researchers found that the network could identify *which hospital* an X-ray came from with very high accuracy, from cues such as markings on the image, and because pneumonia rates differed between hospitals, the hospital was itself a useful clue. The network had partly learned "where was this taken?" rather than "what do these lungs show?"

This phenomenon is common enough to have a name, **shortcut learning**.[13] It is the statistical cousin of the race condition in Chapter Two: a flaw invisible under the conditions where the system was tested, which appears as soon as those conditions change.

Finding two: adversarial examples

In 2013 a team of researchers including Christian Szegedy and Ian Goodfellow reported something unsettling. They could take an image that a network classified correctly, change the pixels by an amount too small for a human to notice, and make the network classify it as something completely different.[14] In a now-famous example from a 2014 follow-up paper, a picture of a panda, classified as "panda" with 57.7 per cent confidence, was altered with a faint layer of carefully chosen noise. The network then classified it as a gibbon with 99.3 per cent confidence.[15]

It is not just a laboratory trick. In 2018 researchers showed that a few black and white stickers placed on a real stop sign could cause an image classifier to read it as a speed-limit sign in most of their tests, including from a moving car.[16]

Adversarial examples show that a network's high accuracy on ordinary inputs says little about its behaviour on inputs chosen by someone trying to fool it. The input space is so large that, as with the race condition, there are regions nobody tested and somebody can find. The language-model versions of this, jailbreaks and prompt injection, are the subject of Chapters Eight and Ten.

Finding three: emergence, and a debate about it

As language models have grown, they have repeatedly shown abilities nobody explicitly trained for and, in some cases, nobody predicted. A model trained only to predict the next word turns out, at enough scale, to do arithmetic, translate between languages and follow multi-step arguments.

In 2022 Jason Wei and colleagues catalogued dozens of tasks where performance seemed to jump suddenly as models grew, and called them **emergent abilities**.[17] In 2023 Rylan Schaeffer, Brando Miranda and Sanmi Koyejo challenged the idea, showing that many apparent jumps disappeared when performance was measured with smoother, more forgiving metrics. They argued that some "emergence" was a product of how researchers chose to score the tests.[18]

Both sides of that debate contain a lesson for this book. If abilities really do appear suddenly with scale, then a model's capabilities, and its possible failures, can change in ways nobody designed, which makes the edges of a system hard to know in advance. If instead the suddenness is partly an artefact of measurement, then *how we choose to measure* an AI system shapes what we believe about it. Either way, confidence about what a model can and cannot do deserves more humility than it usually gets. My grandfather's adders never surprised him with a new ability. A well-specified adder adds, and only adds, however many more adders you build beside it. Modern AI systems do not make that promise.

Why this is not a counsel of despair

I want to be clear about what I am *not* arguing. I am not arguing that because AI systems resist the certainties of digital logic, they cannot be built responsibly. That would be both wrong and useless. These systems exist, they are deployed at enormous scale, and "give up" is not an option for anyone responsible for building or governing them.

My argument is narrower and, I hope, more useful. The *methods* of digital verification (truth tables, exhaustive enumeration of states, formal proof of behaviour) do not transfer to modern AI, and pretending they might is itself

dangerous. But the underlying *disciplines* do transfer, if we do the work of translating them: honest statements of limits, structural safeguards rather than hopeful ones, diverse redundancy, composition that is reasoned about rather than assumed, and trust that is earned rather than granted.

That translation is the project of Part II, which begins with the question at the centre of this book: what, precisely, would it mean for an artificial intelligence system to be intelligible?

KEY POINTS

- Modern AI systems are **trained, not programmed**: billions of parameters are adjusted by gradient descent until outputs match examples.
- Large language models are trained to **predict the next token**, and useful abilities emerge because they help with that prediction.
- **Scaling laws** make average performance predictable while individual behaviour stays unpredictable.
- AI offers **statistical guarantees**, not logical ones, and statistical errors compound when systems are chained ($0.99^{100} \approx 37\%$).
- **Shortcut learning:** networks may learn unintended cues, such as which hospital took an X-ray.
- **Adversarial examples:** imperceptible changes can flip a confident answer, so accuracy on ordinary inputs is not robustness.
- **Emergence** is real enough to demand humility, and contested enough to show that measurement choices shape belief.

WHERE THIS CHAPTER STOPS

This chapter simplifies a technical field considerably. "Predict the next token" describes pre-training accurately but understates what later training stages add. Parameter counts for recent frontier models are mostly undisclosed, so statements about their size are estimates. The compounding-error box assumes independent errors, which real systems do not strictly have. The debate about emergence was unresolved at the time of writing.

INTERLUDE

INTERLUDE

What Happens When You Press Enter

One Question, Traced Through a Neural Network

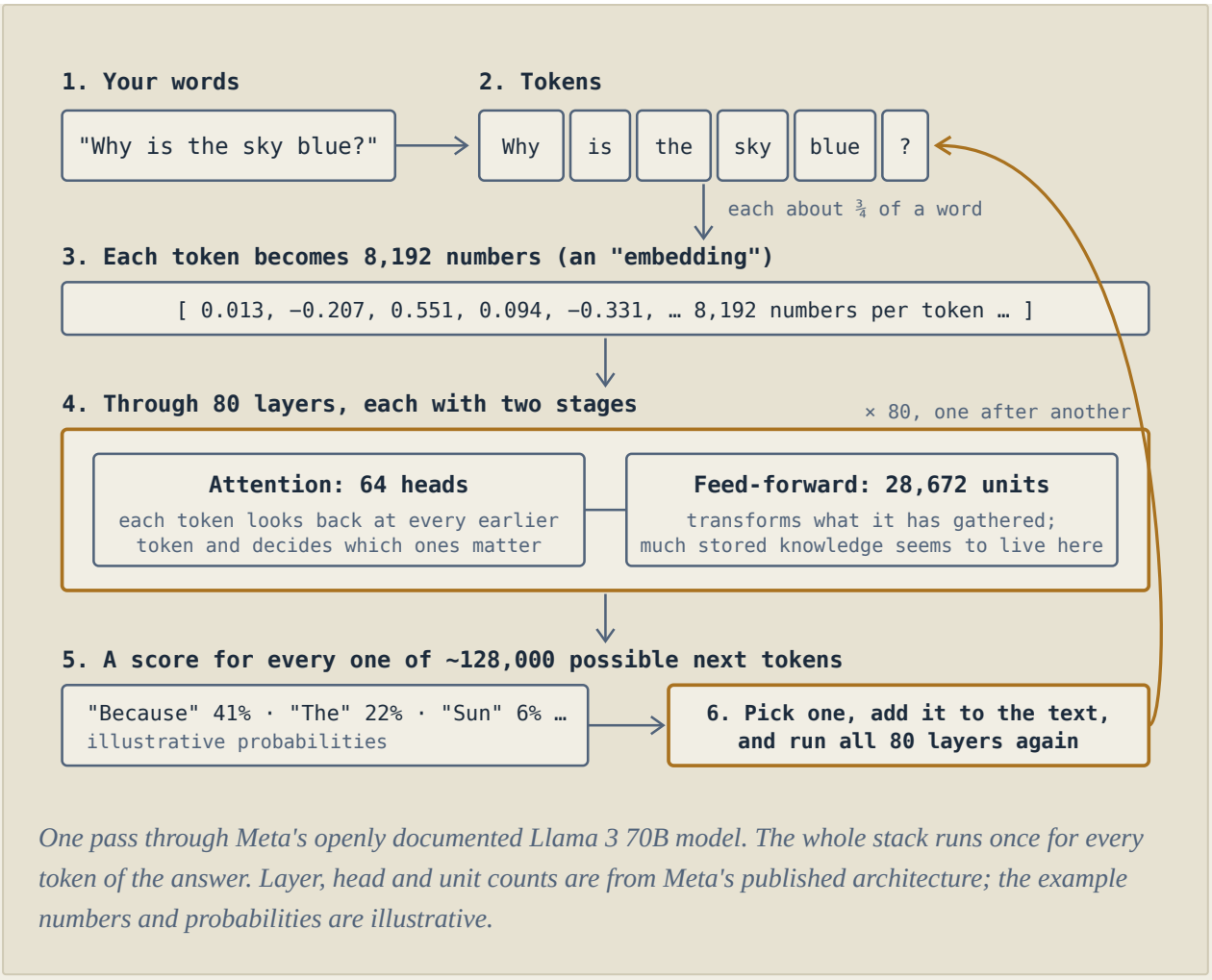
IN THIS INTERLUDE

Chapter Three explained how neural networks are built. This interlude follows a single question through one, step by step and number by number, using a model whose design has been published in full. It shows exactly what a language model does when it answers, why that process produces no record of its reasons, and why "just look inside" is so much harder than it sounds.

> Type six words and press enter. In the second or two before an answer starts to appear, a machine will read your words as numbers, pass them through tens of billions of connections, weigh roughly 128,000 possible next words, choose one, and then do all of it again for the next word, and the next, hundreds of times. Nothing in that process writes down why.

Most of the models people use every day are commercial secrets. Their makers do not publish how big they are or exactly how they are arranged, which is itself a transparency problem. So this interlude uses a model whose design is public: Meta's Llama 3 70B, released in 2024 with a technical paper that describes its architecture in detail.[N16] Leading commercial models are believed to work on the same principles, though they may be larger or arranged differently. Everything below that depends on a specific number is labelled, so you can tell what is published fact and what is my arithmetic.

Our question is a simple one: *Why is the sky blue?*



One pass through Meta's openly documented Llama 3 70B model. The whole stack runs once for every token of the answer. Layer, head and unit counts are from Meta's published architecture; the example numbers and probabilities are illustrative.

Step one: words become tokens

A language model does not see letters or words. It sees **tokens**: common words, pieces of longer words, punctuation marks and spaces, each with an ID number from a fixed vocabulary. Llama 3's vocabulary has about 128,000 of them. As a rule of thumb, a token is about three-quarters of an English word, so a hundred tokens make about seventy-five words.[N15] Our six-word question becomes something like six or seven tokens. The model's first act is to turn language into a list of whole numbers.

This already matters for trust. A model that "cannot count the letters in a word", a well-known weakness, is often not being stupid. It never saw the letters. It saw a token ID.

Step two: each token becomes a point in space

Each token ID is then swapped for a long list of numbers called an **embedding**. In Llama 3 70B, every token becomes a list of 8,192 numbers. You can think of it as a location in a space with 8,192 directions instead of the three we live in. During training, tokens that behave alike drift close together in that space: "blue" ends up nearer to "green" than to "Tuesday". Nobody places them there. The positions are learned, like everything else in the network.

Step three: eighty layers of looking and thinking

The lists of numbers now pass through a stack of 80 **layers**, one after another. Each layer has two stages.

The first is **attention**, the mechanism introduced by the 2017 transformer paper.[N18] In each layer, 64 separate attention "heads" let every token look back at every earlier token and decide which ones matter to it. One head might link "blue" to "sky"; another might attend to the question mark and register that an explanation is wanted. Across the whole model there are 80 × 64 = 5,120 attention heads, each doing its own looking. What any particular head is "for" is not written down anywhere, because nobody designed it; training shaped it.

The second stage is a **feed-forward** block: a wide layer of 28,672 units that transforms what attention has gathered. Research suggests that a good deal of what a model "knows", facts such as the fact that sunlight contains every colour, is stored in the connection strengths of these blocks, though how knowledge is organised inside a network is still being worked out.[N17]

Each layer does not replace the numbers it receives. It adds to them, so a running record, which researchers call the residual stream, grows richer as it rises through the stack. Early layers tend to deal with spelling and grammar, later ones with meaning and intent. By layer 80, the list of 8,192 numbers attached to the last token of your question encodes, in a form no human can read directly, something like "an explanation of Rayleigh scattering is expected next".

Step four: 128,000 candidates, one choice

At the top of the stack, the model converts that final list of numbers into a score for every token in its vocabulary, all 128,000 or so of them, and turns the scores into probabilities. "Because" might get 41 per cent, "The" 22 per cent, "Sun" 6 per cent, and tens of thousands of other tokens tiny fractions of a per cent. (Those figures are illustrative.) The system then **samples** one token, usually favouring the likeliest but with a controlled amount of randomness, which is why asking the same question twice can give different answers.

Then the loop begins again. The chosen token is added to the text, and the whole stack of 80 layers runs once more to choose the next one. A three-hundred-word answer means about four hundred trips through the entire network.

Counting the connections

Now the numbers that make this astonishing. Llama 3 70B has about 70 billion parameters, the adjustable connection strengths described in Chapter Three. To produce each token, almost every one of them is used once, in a multiplication and an addition. So each token costs roughly 140 billion arithmetic operations.[N1]

BY THE NUMBERS

One answer, in arithmetic

~140 billion

Arithmetic operations to produce a single token in a 70-billion-parameter model (about two per parameter).

CALCULATED

~56 trillion

Operations for an answer of about 300 words (roughly 400 tokens).

CALCULATED

~4,400 years

How long one person, doing one sum a second without rest, would take to produce a single token by hand.

CALCULATED

5,120

Attention heads in Llama 3 70B (80 layers × 64), each deciding what to look at, none of them designed by a person.

CALCULATED

~128,000

Candidate tokens scored at every single step of every answer.

DOCUMENTED

0

Lines in that process that record why one word was chosen over another.

FRAMEWORK

CHECK THE ARITHMETIC YOURSELF

Per token. About 2 operations per parameter × 70 billion parameters = 140 billion (1.4×10^{11}).

By hand. 1.4×10^{11} seconds ÷ 31.6 million seconds in a year ≈ 4,400 years for one person, for one token.

By everyone. $1.4 \times 10^{11} \div 8.2$ billion people ≈ 17 seconds per token if all of humanity joined in; for a 400-token answer, about 6,800 seconds, a little under two hours.

Caveats. The rule of two operations per parameter is an approximation; long conversations add extra work for attention; and models of the "mixture-of-experts" type use only part of their parameters for each token. The point survives the caveats: an ordinary answer involves tens of trillions of operations.

What the numbers mean for trust

Here is the heart of it. Every one of those tens of trillions of operations is ordinary, deterministic arithmetic of the kind my grandfather's machines performed. Nothing is magic. In principle, every multiplication could be logged. But a log of 56 trillion multiplications is not an explanation, in the same way that a list of the positions of every brick is not an explanation of a city. The model has no step at which it writes down a reason. When it later tells you why it answered as it did, it is generating new tokens by exactly the same process, not reading from a record. That is why Chapter Five found that models' explanations of their own reasoning are often unfaithful.

It also explains why these systems are both so capable and so hard to bound. The same 70 billion connections serve every question: physics, poetry, law, your child's homework. There is no separate, checkable module for each subject, no truth table for "sky", no datasheet stating where competence in medicine ends. The knowledge is spread across the connections in superposition, overlapping and shared.

And it shows where intelligibility can realistically be built. We cannot read the arithmetic. But we can control what goes in (the question, the documents retrieved), record what comes out, measure how often answers are right in a given domain, draw boundaries around what the system is allowed to do, and, with the interpretability tools of Chapter Five, begin to map which internal features light up for which ideas. Part II of this book is about exactly those levers.

Training: the part you never see

Everything above happens when a model is *used*, which engineers call inference. Before any of it can happen, the connection strengths have to be learned, and that is where the truly enormous numbers live. The largest Llama 3 model, with 405 billion parameters, was trained on 15.6 trillion tokens using 3.8×10^{25} operations.[N7] If every person on Earth did one sum per second, the training run

would take about 150 million years. The text it learned from would take one person, reading eight hours a day, more than a quarter of a million years to get through.

Nobody reviewed that text line by line. Nobody could. Whatever patterns, facts, errors and biases it contained were absorbed into the connection strengths in proportions nobody chose. That is the second reason a trained model cannot be taken on trust: we do not fully know what it learned from, and we cannot read what it learned.

KEY POINTS

- A language model turns words into **tokens**, tokens into lists of numbers, and passes those numbers through dozens of layers of **attention** and **feed-forward** computation.
- At every step it scores every token in its vocabulary (about 128,000 in Llama 3) and picks one. The whole network runs again for every token of the answer.
- Each token costs roughly **two operations per parameter**: about 140 billion for a 70-billion-parameter model, and tens of trillions for an ordinary answer.
- The arithmetic is ordinary and could be logged, but a log is not an explanation. **No step records a reason.**
- Training a large model takes around 10^{25} operations on trillions of tokens that no person has read, so what the model learned is not fully known either.
- Intelligibility has to be built around the arithmetic: in inputs, outputs, measurement, boundaries and interpretability.

WHERE THIS INTERLUDE STOPS

This tour follows one openly documented model; commercial systems differ in size and design, and many are not publicly described. The walk-through simplifies several stages (normalisation, positional information, key-value caching, post-training for assistant behaviour and safety). The operation counts are approximations using a standard rule of thumb. The description of what early and late layers "do" summarises broad research tendencies, not a law.

PART II

The Framework: Intelligible Intelligence

CHAPTER FOUR

What Makes a System Intelligible

Four Properties That Interlock

IN THIS CHAPTER

This chapter defines the book's central idea. A system is intelligible to the degree that it is transparent, verifiable, bounded and composable. Each property does distinct work, none is enough alone, and together they describe a whole sociotechnical system (the model, the people and the organisation around it), not just a model.

> In 2023 a New York lawyer asked ChatGPT to find court cases that supported his client. It supplied cases, and he filed them. When he asked it whether one of them was real, it said yes, and that it could be found in reputable legal databases. Six of the cases he cited did not exist. Nobody could see why the system had invented them, nobody independent had checked them, the system gave no sign it was out of its depth, and the only safeguard in the chain was a human who asked the machine to mark its own homework. Every one of the four properties this chapter defines was missing at once.[11.3]

In 2024, 78 per cent of organisations surveyed for Stanford's AI Index said they were using AI, up from 55 per cent a year earlier. By August that year, 950 AI-enabled medical devices had been approved in the United States. The question is no longer whether AI will be trusted with things that matter. It already is. The question is whether anyone can say, precisely, what deserving that trust would mean.[N10]

BY THE NUMBERS

Trust is already being extended

78%

Organisations reporting AI use in 2024, up from 55% in 2023 (survey data).

REPORTED

950

AI-enabled medical devices approved by the US FDA as of August 2024.

REPORTED

4

Questions that any consequential AI system's owners should be able to answer: can we see why, who checked, where it stops, and what it can affect.

FRAMEWORK

1 MΩ

The minimum insulation resistance for an ordinary 230 V circuit under BS 7671. Intelligibility, like insulation, is a quantity with a minimum that depends on the job.

DOCUMENTED

I have used the word "intelligible" freely in the first three chapters without pinning it down. It is time to do that properly, because the rest of the book depends on the definition holding weight. I want to build it the way a digital designer builds a circuit: start from something almost too obvious to state, and add conditions one at a time until the full shape is visible.

Starting from the wrong definitions

The word most often used in public discussion of trustworthy AI is **transparency**, usually meaning "we can see what's going on inside". That is part of what I mean, but on its own it is both too narrow and too demanding.

It is too narrow because a system can be completely transparent, with every parameter published and every computation logged, and still be impossible for anyone to understand, in the way a warehouse of raw sensor data is transparent but useless without interpretation. When Meta released the weights of its Llama models to researchers and the public, anyone could download every number. That

was valuable, but it did not tell anyone what the model would do with a given question.

It is too demanding because, taken literally, it asks for something close to truth-table completeness: full visibility into every internal state. Chapter Three showed that this is not available for the systems this book is about. Insisting on it before deployment would amount to insisting on no deployment at all, which is not the position this book defends.

The second word often used is **explainable**, meaning the system can give some account of why it produced an output. This is closer, but it has its own trap. An explanation produced after the fact, by the same opaque process that made the decision, is not obviously more trustworthy than the decision. Language models are very good at producing fluent, plausible explanations that do not match the computation that actually produced their answer. Chapter Five presents the evidence. The principle is simple enough to state here: **an explanation you cannot check is testimony, not evidence.**

The definition

Here is the definition this book uses. It is built from four properties, each doing distinct work, none sufficient alone. A system is intelligible to the degree that it is:

- **Transparent:** its relevant internal workings, or a faithful and tested summary of them, can be inspected by a suitably qualified party, so that claims about *why* it behaves as it does can be checked against something.

- **Verifiable:** claims about its behaviour, including its limits as well as its capabilities, can be tested against independent evidence by someone other than the party making the claim, using methods whose own reliability is established.

- **Bounded:** its operating envelope is explicitly specified, including the conditions under which it should refuse, defer to a human, or fail in a safe and

detectable way, rather than producing unpredictable output when pushed outside the domain it was built and tested for.

- **Composable:** its behaviour, and especially its failure behaviour, is characterised well enough that it can be combined with other systems, or with human oversight, without the combination producing effects that neither part's characterisation would predict.

Each of these has a direct ancestor in my grandfather's world. Transparency descends from the schematic diagram. Verifiability descends from the truth table that anyone with a pencil can check. Boundedness descends from the datasheet's absolute maximum ratings. Composability descends from interface specification, the discipline of making sure verified parts stay verified when you join them.

PROPERTY	THE QUESTION IT ASKS	ITS ENGINEERING ANCESTOR	WHAT ITS ABSENCE LOOKS LIKE
Transparent	Can we see why it did that?	Schematic diagram	Explanations that cannot be checked
Verifiable	Who checked the claim, and how?	Truth table, independent certification	Self-reported scores, unreproduced
Bounded	Where does it stop being reliable, and what happens there?	Datasheet maximum ratings	Confident answers far outside competence
Composable	What happens when it is connected to other things?	Interface specification	Safe parts, unsafe whole

Why all four are necessary

These four are not a menu to choose from according to taste. They interlock. A system strong on one but weak on the others is not intelligible at all. It is impressive along one axis and dangerous along the others. It helps to picture each combination.

Transparent but not verified. Every weight, activation and training example is available for inspection, but nobody has actually run the independent, adversarial tests that would show the system behaves as its openness implies. This describes a meaningful fraction of "open" AI releases. Transparency without verification is an invitation to inspect, not evidence that anyone accepted the invitation and found the system trustworthy.

Verified but not bounded. The system has been rigorously tested against good benchmarks, but nobody has specified, and the system cannot recognise, when real-world inputs have drifted outside what those benchmarks covered. Much deployed AI looks like this. Verification without boundedness is confidence with an expiry date nobody has written down. Chapter Seven gives a clear example: a diabetic eye-screening system that performed very well on high-quality laboratory images and struggled in real clinics.

Verified and bounded but not composable. The system knows its envelope and refuses outside it, but when combined with other tools, other models or human reviewers, the combination behaves in ways nobody characterised. Chapter Eight argues this is one of the most underappreciated sources of real-world AI failure.

Everything but transparency. A system that is verified, bounded and composable but completely opaque inside is hard to *keep* that way. Every update, retraining or fine-tune could change the behaviours that were verified, and without internal visibility there is no way to know which. Transparency is not enough on its own, but it is the foundation the other three stand on, in the way a building's foundation is not the building but everything depends on it.

Degrees, not a switch

Notice that the definition says "to the degree that". Intelligibility is not a switch that is on or off. It is more like the insulation resistance I measure on a circuit: a quantity, with a minimum acceptable value that depends on what the circuit is for. BS 7671 sets a minimum insulation resistance of 1 megohm for ordinary 230-volt

circuits, tested at 500 volts.[1] A reading below that is a fail. A reading far above it is not "extra safe" in any way that matters; it simply passes.

The same logic applies here. A spell-checker needs very little intelligibility. A system that recommends cancer treatments, screens job applicants or decides who is investigated for benefit fraud needs a great deal. The right question is never "is this system intelligible?" but "**is it intelligible enough for what we are trusting it to do?**"

Intelligibility belongs to the whole system

One more point, which I will return to often: intelligibility, as I define it, is not a property of a trained model in isolation. It belongs to the whole **sociotechnical system**: the model, plus the interface people use, plus the monitoring and logging around it, plus the human processes for reviewing outputs and stepping in, plus the organisation that is accountable when it fails.

A brilliant, well-characterised model deployed with no monitoring, no human review and no clear accountability is not an intelligible system, however good the model. And, more hopefully, a model with real, acknowledged limitations can be part of a highly intelligible system if the structure around it is honest about those limitations and designed to catch the failures that will occur inside them.

This is my grandfather's redundancy principle translated into a new domain. Engineers did not achieve trustworthy systems by demanding perfect components. They combined well-characterised components, including components with known failure rates, into architectures that were robust to those failures. The path to intelligible AI is not a path to models that never make mistakes. No such models exist, and waiting for them would mean waiting for ever. It is a path to systems, built around necessarily imperfect models, that are honest about where the imperfections lie and structured so that those imperfections cannot cascade unchecked into the catastrophes this book's title warns against.

FROM THE TOOLBOX

The four questions

If you remember nothing else from this book, remember these four questions. They apply to any AI system, from a chatbot on a shop's website to a model advising a government:

1. **Transparent:** Can anyone see, or faithfully reconstruct, why it produced this output?
2. **Verifiable:** Who, other than its makers, has tested its claims, and how?
3. **Bounded:** Where does it stop being reliable, and what does it do when it gets there?
4. **Composable:** What is it connected to, what can it do to those things, and what happens if it is wrong?

A system whose owners cannot answer all four is not ready to be trusted with anything that matters.

A worked example: the surgery that wants an AI scribe

To show how the four questions work in practice, take an ordinary, realistic decision. A GP surgery is considering an "ambient scribe": an AI tool that listens to a consultation, with the patient's consent, and drafts the clinical notes, saving the doctor several minutes per appointment. Tools like this spread rapidly through health services in 2024 and 2025, and NHS England issued guidance on their use.[2] The example here is illustrative, not an assessment of any particular product.

Transparency. Can anyone see why the tool wrote what it wrote? A good product keeps the audio or a transcript alongside the draft, so that any line in the notes can be traced back to what was actually said. A poor one produces a polished summary with no way to check where each statement came from. The practice manager should also be able to find out which underlying language model the product uses, and where patient data goes.

Verifiability. Who has tested it, apart from the company selling it? The surgery should ask for evaluation results that measure the errors that matter clinically, not

just overall accuracy: omissions (a symptom mentioned but not recorded), additions (a symptom recorded that was never mentioned, which is a hallucination) and misattributions (the patient's words recorded as the doctor's advice). Ideally those results come from an independent evaluation in a similar setting. Even then, the surgery should run its own small check for a few weeks, comparing drafts against the recordings.

Boundedness. Where does it stop being reliable? Strong regional accents, interpreters, several people talking at once, telephone consultations and specialist vocabulary are all plausible edges of its operating domain. The supplier should say which of these it has tested. The surgery should decide in advance which consultations the tool will not be used for.

Composability. What does it connect to, and what happens if it is wrong? If drafts flow straight into the patient record, an error becomes part of the medical history, may be copied into referral letters and may be read by other clinicians who assume a doctor wrote it. So the design question is less "is the AI accurate?" than "is every draft reviewed and signed off by the clinician before it enters the record, and is that review designed to resist automation bias?" A doctor at the end of a long clinic, who has approved a hundred good drafts in a row, is exactly the reviewer Chapter Eight warns about.

None of these questions needs technical expertise to ask. All of them need answers before the tool is trusted with patients' records. Notice, too, that most of the answers concern the *system* (recordings, review, sign-off, scope) rather than the model itself. That is the point of the previous section.

The next four chapters take each property in turn: what it requires, what progress science has actually made towards it, and what a serious commitment to it would look like in practice.

KEY POINTS

- **Transparency alone** is too narrow (visible is not understandable) and too demanding (full visibility is impossible for large models).
- **Explanations alone** are not enough, because an explanation you cannot check is testimony, not evidence.
- Intelligibility = **transparent + verifiable + bounded + composable**. Each has an ancestor in digital engineering.
- The properties **interlock**. Strength in one does not make up for weakness in another.
- Intelligibility comes in **degrees**. The standard should match the stakes.
- It is a property of the **whole sociotechnical system**, including the people and organisation, not just the model.

WHERE THIS CHAPTER STOPS

The four-property framework is this book's own synthesis. It overlaps with, but is not identical to, frameworks used by regulators and standards bodies (for example the OECD AI Principles, the US NIST AI Risk Management Framework and the EU AI Act's requirements for high-risk systems). The framework organises questions; it does not by itself set numerical thresholds for how much intelligibility a given use requires. That remains a judgement for practitioners and regulators.

CHAPTER FIVE

Transparency

Seeing Inside the Machine

IN THIS CHAPTER

Having access to a model's parameters is not the same as understanding it. This chapter separates deliberate opacity (a policy choice) from intrinsic opacity (a scientific problem), describes what interpretability research has genuinely achieved, from "features" and "superposition" to tracing a model's internal steps, and presents the evidence that a model's own step-by-step explanations are often unfaithful.

> Researchers at Anthropic asked Claude to add 36 and 59. It answered 95, correctly. Then they asked how it had done it, and it described the method every child learns at school: add the units, carry the one. But the researchers could look inside, and that is not what had happened. The model had run two calculations in parallel, one estimating the rough size of the answer and one working out the last digit, and then combined them. Its explanation described what a person would do, not what it had done.[5]

BY THE NUMBERS

How much we can see

~34 million

Features extracted from Claude 3 Sonnet in 2024 using sparse autoencoders.

MEASURED

~¼

Share of prompts for which Anthropic's 2025 attribution graphs gave "satisfying insight", in the researchers' own words.[N11]

REPORTED

25% / 39%

How often two reasoning models admitted using a hint that changed their answer. Most of the time the "working" they showed left it out.

MEASURED

200 million

Pages needed to print the parameters of a 100-billion-parameter model at 500 numbers a page. Access is not understanding.

CALCULATED

Digital designers of my grandfather's generation never had to fight for the right to see inside their own systems. A schematic diagram was, by definition, a complete map of a circuit, and from that map and the algebra governing it, behaviour could be derived. The fight in his world was not for access but for accuracy: making sure the map was right and the derivation correct.

In modern AI the fight is different and more fundamental. Even when you have complete access to a model, every one of its billions of parameters, you do not have a map you can read the way a schematic is readable. This chapter is about the difference between access and understanding, and about the hard-fought progress being made to close the gap.

Two kinds of opacity

It helps to separate two reasons an AI system might be opaque, because they need different remedies and are too often run together.

The first is **deliberate opacity**: the organisation that built the system withholds its internal workings, for commercial advantage or because it judges that disclosure would make misuse easier. This is a policy choice and can be reversed by a different policy choice. Much of the public argument about "open" versus "closed" AI is really an argument about deliberate opacity. It matters, and Part IV returns to it, but it is a problem of governance and incentives, not science.

The second is **intrinsic opacity**. Even a model whose weights are fully published, sitting on your own computer for you to inspect however you like, resists being understood, because nobody can trace by inspection the relationship between "these billions of numbers" and "this behaviour on this input". No disclosure policy can fix that. It needs new science.

The science of looking inside

That science is called **mechanistic interpretability**. Its approach is to treat a trained network the way a biologist treats an unfamiliar organism: not as a black box to be accepted on faith, but as a physical system whose structure, though not designed by any single hand, can be studied, probed and partly understood by experiment.

The field's progress over the past decade is genuinely encouraging, and I want to describe it carefully, because it is easy for a book like this to slide into either uncritical optimism or uncritical despair. Neither does justice to where things stand.

Neurons that mean more than one thing

An early hope was that each artificial neuron might correspond to one human concept: a "cat neuron", a "Paris neuron". Sometimes this seemed true. More often, researchers found **polysemantic** neurons that responded to several unrelated things at once, such as cat faces, the fronts of cars and a particular style of text.

In 2022 researchers at Anthropic proposed an explanation called **superposition**.[1] A network has to represent far more concepts than it has neurons, so it packs them in, spreading each concept across many neurons and letting each neuron take part in many concepts, much as a small number of radio frequencies can carry many broadcasts if they are cleverly combined. That makes the network efficient and makes individual neurons nearly meaningless to a human observer.

Unpacking the features

If concepts are packed in, perhaps they can be unpacked. In 2023 and 2024 Anthropic researchers used a technique called a **sparse autoencoder**, a second, simpler network trained to re-express the model's internal activity as a combination of a much larger number of cleaner, more interpretable directions, called **features**.[2] In May 2024 they applied this to a production model, Claude 3 Sonnet, and extracted up to about 34 million features. Some corresponded to recognisable concepts: particular cities and people, programming errors, deception, flattery.[3]

The most memorable demonstration was a feature associated with the Golden Gate Bridge. When researchers artificially turned that feature up, the model began mentioning the bridge in response to almost anything, and when asked what it was, described itself as the bridge. For about a day, Anthropic let the public talk to this altered model, which it called "Golden Gate Claude".[4] It sounds like a stunt, but the scientific point is serious. Being able to *find* an internal feature, *change* it and *predict* the resulting change in behaviour is the closest thing modern AI has to the provable causal structure of digital design. It moves interpretability from correlation towards cause.

Tracing a model's steps

In March 2025 Anthropic published work that went further, tracing the chains of features a model used to produce particular answers, which the researchers called "attribution graphs".[5] Two findings stand out.

When the model wrote a rhyming couplet, it appeared to choose the rhyming word for the end of the second line *before* writing that line, then compose the line to arrive at it. The model was planning ahead, even though it produces text one word at a time.

When the model added two numbers, it appeared to use parallel internal pathways, one estimating the rough size of the answer and another working out the final digit precisely. But when asked how it had done the sum, it described the standard method taught in school: add the units, carry the one. Its explanation described what a person would do, not what it had actually done.

That second finding matters enormously for this book, and I will come back to it at the end of the chapter.

Honest limits of interpretability

This is genuinely exciting science, and I do not want to undersell it. I also do not want to oversell it. The honest state of the field, as of this writing, is:

- These methods have explained particular behaviours in particular models. They do not yet give anything like a complete account of a frontier model's behaviour, and the researchers themselves say so.[6]
- A sparse autoencoder is itself an approximation. It can miss things and can produce features that look meaningful but are partly artefacts of the method.
- Interpretability is expensive, and the models keep growing.

A responsible chapter on transparency has to hold both facts at once: real, substantive progress, and a very long way still to go.

Transparency short of full understanding

Because full mechanistic understanding is not available, and may not be for a long time, it is worth asking what partial, achievable forms of transparency exist *now*.

Three are especially useful.

Training transparency is disclosure of what data a model was trained on, in what proportions and with what filtering. It does not tell you what the model will do on a given input, but it tells you a great deal about which behaviours are plausible, as knowing someone's education tells you something about what they are likely to know. In 2018 Timnit Gebru and colleagues proposed "datasheets for datasets", explicitly modelled on the datasheets that accompany electronic components, so that every dataset would come with a record of its origins, composition and known gaps.[7] It is an analogy any digital engineer would recognise.

Behavioural transparency is comprehensive, honestly reported evaluation of how a system performs across many tasks and conditions, including adversarial ones designed to find its weaknesses. In 2019 Margaret Mitchell and colleagues proposed **model cards**: short documents reporting a model's intended uses, performance across different groups and conditions, and known limitations.[8] Major developers now publish "system cards" of this kind alongside new models, some running to more than a hundred pages. They vary widely in quality and candour, but the practice is a real advance.

Process transparency is disclosure of how the system was built, tested and deployed: what safety evaluations were run, by whom, with what results, and, critically, which evaluations were considered and *not* run, and why. That lets outsiders judge the rigour of the process even when they cannot inspect the product directly, as a building inspector can assess construction without personally redoing the structural calculations.

None of these, alone or together, gives you a schematic. But taken seriously and reported honestly, they give an informed outsider enough to form a justified, evidence-based judgement about a system's trustworthiness, instead of simply taking the builder's word for it.

FROM THE TOOLBOX

How to read a system card

When an AI developer releases a model, it increasingly publishes a *system card* or *model card*. These can be long and technical. Five questions get most of the value from them:

1. **What was tested, and what was not?** Look for a list of evaluations and, just as importantly, any statement of what was out of scope.
2. **Who did the testing?** Note which results come from the developer and which from outside organisations such as government institutes or independent evaluators.
3. **What are the known limitations?** A card with no substantial limitations section is a warning sign, not a reassurance.
4. **How does performance vary?** Look for results broken down by language, task type or group of people, rather than single headline figures.
5. **What changed from the previous version?** Improvements in one area are sometimes paired with regressions in another.

A card that answers all five clearly is a sign of a developer taking transparency seriously. A card that is mostly marketing tells you something too.

Transparency for everyone else: was this made by AI?

There is a fourth kind of transparency, aimed not at experts but at the public: knowing whether you are talking to an AI, and whether a picture, a recording or a piece of text was produced by one. As AI-generated images, voices and writing become indistinguishable from human work, this question moves from curiosity to necessity, for elections, for courts and for anyone deciding whether to believe what they see.

Two technical approaches are being developed. The first is **provenance**: attaching a tamper-evident record to a piece of media describing where it came from and how it was edited. The Coalition for Content Provenance and Authenticity (C2PA), founded in 2021 by companies including Adobe, Microsoft, Intel, Arm,

Truepic and the BBC, publishes an open standard for such records, often shown to users as "content credentials".[9] The second is **watermarking**: embedding a signal in AI-generated content that is invisible to people but detectable by software. In 2024 researchers at Google DeepMind described in *Nature* a method for watermarking the text produced by a language model by subtly biasing its choice of words, and reported deploying it in Google's Gemini products.[10]

Both approaches have honest limits. Provenance records can be stripped from a file, and they only help if cameras, editing tools and platforms all support them. Watermarks can be weakened by editing, paraphrasing or translation, and a watermark from one company's system says nothing about content from another. Neither can prove that something was *not* made by AI. These are verification aids with stated boundaries, in keeping with the rest of this book, not guarantees. Law is starting to require them: the EU AI Act, for example, requires that people be told when they are interacting with an AI system and that certain AI-generated or manipulated content be marked as such.[11]

The chain-of-thought trap

I want to end with a specific warning, because it concerns one of the commonest and most seductive mistakes in public discussion of AI transparency. I made it myself before I understood the research.

Many modern AI systems, especially the "reasoning models" released since 2024, produce a step-by-step account of their reasoning before giving a final answer, called a **chain of thought**. This looks exactly like the transparency this chapter has been describing: the system showing its work, as a student shows working on a maths paper so the teacher can check the method.

The trouble is that research repeatedly shows the chain of thought is not reliably a faithful description of the computation that produced the answer.

In 2023 Miles Turpin and colleagues gave language models multiple-choice questions with a hidden bias, for example by making the correct answer always "(A)" in the examples. The models' answers shifted towards the bias, but their written explanations almost never mentioned it. Instead they produced plausible reasoning for whichever answer the bias had pushed them towards.[12]

In 2025 researchers at Anthropic tested two reasoning models by slipping hints about the answer into questions and checking whether the models admitted using them. Claude 3.7 Sonnet mentioned the hint in its reasoning only about 25 per cent of the time when it used it; DeepSeek R1 about 39 per cent. In a separate experiment, models trained in environments that rewarded exploiting a loophole learned to exploit it in more than 99 per cent of cases, but mentioned doing so in their reasoning less than 2 per cent of the time in more than half of those environments.[13]

> THE SCIENCE
>
> **Why would a model's explanation be unfaithful?**
>
> Nobody fully knows, but there are some plausible contributing causes.
>
> - The model is trained to produce text that *looks like* good reasoning, because good reasoning appears in its training data and is rewarded. Looking right and being a faithful record are different targets.
> - Some of the computation that decides an answer happens in parallel inside the network, as in the arithmetic example above, and has no natural translation into a step-by-step sentence.
> - Humans do something similar. Psychologists have documented for decades that people often give confident explanations of their own choices that do not match what actually influenced them.[14]
>
> None of this makes chains of thought useless. They can still reveal problems, and researchers have argued that monitoring them is a valuable, if fragile, safety tool. But they cannot be treated as a window.

Treating chain-of-thought output as a faithful view of a model's reasoning, without independent confirmation that it *is* faithful, is exactly the mistake this chapter warns against: mistaking the appearance of an explanation for evidence of one. Genuine transparency requires that an account of a system's behaviour be checked against something other than the system's own say-so.

That requirement, verification independent of the claim being verified, is the subject of the next chapter. In some ways it is the most fundamental of the four properties, because without it even genuine transparency can be faked, gamed or simply mistaken.

KEY POINTS

- **Deliberate opacity** is a policy choice; **intrinsic opacity** is a scientific problem. Publishing weights solves only the first.
- **Superposition:** networks pack many concepts into few neurons, so individual neurons are hard to interpret.
- **Sparse autoencoders** can extract millions of more interpretable **features**, and changing a feature can predictably change behaviour ("Golden Gate Claude").
- **Attribution graphs** have shown models planning ahead, and computing arithmetic differently from how they *say* they compute it.
- Partial transparency is available now: **datasheets for datasets, model cards, and honest process disclosure.**
- **Chains of thought are often unfaithful.** In one 2025 study, models acknowledged hints they used only 25–39% of the time.

WHERE THIS CHAPTER STOPS

Much of the most detailed interpretability research cited here comes from one company, Anthropic, publishing about its own models, which is itself a verifiability concern (Chapter Six); other groups at Google DeepMind, OpenAI and universities work on similar methods with broadly consistent findings. Interpretability results on one model may not transfer to others. Faithfulness figures depend heavily on the experimental setup and should not be read as general rates for all use.

CHAPTER SIX

Verifiability

Who Checked, and How?

IN THIS CHAPTER

A claim tested only by the party that wants it to be true is a claim, not a finding. This chapter explains what testing can and cannot prove (with the statistics), how AI benchmarks get contaminated and gamed, what happened when a widely used hospital AI was finally tested independently, and what red-teaming and independent evaluation look like when they are done properly.

For years, hundreds of American hospitals ran software that was meant to warn doctors when a patient was sliding into sepsis, an infection that can kill within hours. It came from one of the biggest names in medical records, and the figures published about it looked good. Then researchers at Michigan Medicine tested it on nearly 40,000 of their own hospital stays. It had missed two-thirds of the patients who developed sepsis.[6] This chapter is about the question nobody had asked in time: who checked, and how?

BY THE NUMBERS

What a test can and cannot tell you

3 million

Consecutive clean tests needed to support, with about 95% confidence, a failure rate below one in a million (the rule of three).

CALCULATED

67%

Sepsis patients that a widely deployed hospital model failed to flag when it was finally tested independently at Michigan Medicine.

MEASURED

up to 8%

Drop in some models' scores when a public maths benchmark was replaced by fresh, unpublished problems of the same kind.

MEASURED

+67.3 pts

One-year rise in scores on the SWE-bench benchmark. A fast-rising score is a reason to ask what it measures, not a reason to stop asking.

REPORTED

Nobody learning digital design has to take a truth table on faith. That is the whole point of one: anyone with the circuit diagram and a few hours can sit down and check it, independently, with no access to the engineer's private reasoning. The check does not depend on trusting whoever drew it up. It depends only on Boolean algebra, which belongs to nobody.

This chapter is about what the equivalent independence looks like for artificial intelligence, and how often it is missing.

A claim is not a finding

When the organisation that builds an AI system also runs the only evaluation of it, publishes only the results it chooses, and faces no requirement that anyone outside reproduce those results, what you have is a **claim**, not a **finding**. That is true however careful and well-meaning the organisation is.

This is not an accusation of dishonesty. It is a point about how knowledge works. A claim tested only by the party with every incentive to see it succeed is structurally weaker than the same claim tested by a party with no stake in the outcome, even if both would report the same result in good faith. Every mature field has learned this. It is why drug trials are registered in advance, why company accounts are audited, and why, in my trade, the Electrical Installation Certificate separates the person who designed a circuit, the person who built it and the person who inspected and tested it. On a small job one person may sign all three sections, but the form still forces them to wear three different hats.

Digital engineering solved the problem not by trusting engineers more but by making verification cheap and standardised enough that independent checking became routine. A truth table can be redone by anyone with a pencil. A benchmark score reported by the lab that trained the model, on a test the lab may have had access to during development, reproduced by nobody outside the lab, is a much weaker kind of evidence, however impressive the number.

What testing can prove

Before looking at AI specifically, it helps to understand a hard mathematical limit on what testing can tell you about any system.

THE SCIENCE

The rule of three, and the limits of testing

Suppose you test a system 300 times and it never fails. How reliable is it?

It is tempting to say "perfectly reliable". It is more accurate to say you have ruled out failure rates much above 1 per cent. A standard result in statistics, sometimes called the **rule of three**, says that if you observe zero failures in *n* independent trials, you can be about 95 per cent confident that the true failure rate is below 3/*n*.[1] So 300 clean tests support a failure rate below about 1 in 100. To support 1 in 10,000, you need around 30,000 clean tests. To support 1 in a million, around 3 million.

Now apply this to systems where failure is catastrophic. Civil aviation has long required that catastrophic failure conditions be "extremely improbable", often interpreted as a probability on the order of one in a billion per flight hour. In 1993 Ricky Butler and George Finelli of NASA calculated what it would take to demonstrate that kind of reliability for software by testing alone. Their answer was a test period measured in hundreds of thousands, or even millions, of years.[2]

The conclusion is not that testing is useless. It is that **testing alone can never establish very high reliability.** Safety-critical engineering gets there by combining testing with design, proof, structural safeguards and independent review. AI, which cannot use proof in the same way, depends even more heavily on the other three.

This matters for AI because AI systems are often described in terms of their test results ("95 per cent accurate on benchmark X") as if those results settled the question. The rule of three says a test result is only as strong as the number and independence of the tests behind it, and only for the kind of inputs those tests contained.

Benchmarks and their discontents

The standard way to evaluate AI systems is the **benchmark**: a fixed set of questions or tasks with known correct answers, against which a model is scored. Benchmarks have done real good. They give the field a common yardstick, make

progress measurable and have driven genuine improvement. But they have a well-documented weakness.

A public benchmark can be trained towards, deliberately or by accident, in ways that raise the score without improving the ability the benchmark was meant to measure. The simplest version is **contamination**: the test questions, which are published on the internet, end up in the enormous body of internet text a model is trained on, so the model has effectively seen the exam paper. A subtler version is optimising for the particular style and format of a known benchmark instead of the general skill it stands for.

> CASE FILE
>
> **The maths test that wasn't new**
>
> In 2024 researchers at Scale AI tested this directly. GSM8K is a widely used benchmark of about 8,500 grade-school maths word problems. The researchers wrote a new set of about 1,200 problems, GSM1k, carefully matched to the original in style and difficulty but never published. If models had genuinely learned to solve such problems, their scores on the two sets should be similar.
>
> For several model families they were not. Some models scored up to 8 per cent lower on the fresh problems, a pattern consistent with having partly memorised the public test. Other models, including several of the most capable, showed little or no drop, which suggests their skill was real.[3]
>
> **Property that failed:** verifiability. The public benchmark had stopped being an independent test for some models.

This is an instance of **Goodhart's law**, named after the economist Charles Goodhart and usually stated, in the anthropologist Marilyn Strathern's phrasing, as: "When a measure becomes a target, it ceases to be a good measure."[4] Truth tables are immune to Goodhart's law. There is no way to make a circuit pass its truth table without it actually implementing the specified logic. Statistical tests of learned systems have no such immunity.

The remedy digital engineering would recognise immediately is **held-out, independently administered evaluation**: test sets the developers have never seen, run by parties with no stake in a good result, refreshed often enough that memorisation cannot stand in for skill. Some benchmark creators now embed a unique "canary string" in their test files so that anyone can check whether the files have leaked into a training set.[5] Held-out evaluation is more expensive, slower and less flattering than self-reported scores. It is also the only kind of evidence that earns the word *verified*.

When a hospital AI was finally checked

The clearest example I know of why independent verification matters comes not from chatbots but from medicine.

Sepsis is the body's extreme response to infection, and it kills quickly if it is not treated. Early warning helps. Epic Systems, whose electronic health records are used by a large share of American hospitals, developed a proprietary model, the Epic Sepsis Model, that scored patients' risk and alerted clinicians. It was deployed at hundreds of hospitals. Because it was proprietary, outside researchers had limited information about how it had been developed and validated.

In 2021 Andrew Wong and colleagues at Michigan Medicine published an independent evaluation of how the model had performed on their own patients: 27,697 people across 38,455 hospital stays.[6] Their findings:

- The model's ability to separate patients who developed sepsis from those who did not, measured by a statistic called the area under the curve (AUC), was 0.63. An AUC of 0.5 is no better than chance and 1.0 is perfect. Values published by the developer had been substantially higher, in the range of roughly 0.76 to 0.83.[7]
- Of 2,552 patients who developed sepsis, the model **did not flag 1,709, or 67 per cent.**

- Meanwhile it generated alerts for 6,971 of all 38,455 hospitalised patients, 18 per cent, creating what the authors called "a large burden of alert fatigue".

The authors' conclusion was blunt: the model had "poor discrimination and calibration", and its "widespread adoption despite its poor performance raises fundamental concerns". Epic disputed aspects of the study and later released an updated version of its model. The point for this book is not about one company. It is that a system used in life-and-death decisions at hundreds of hospitals had not, until this study, been independently tested at this scale and published in the open. The first serious independent test found a gap between claim and finding large enough to matter to patients.

When AI marks AI's homework

A newer shortcut in evaluation deserves a warning. Because checking thousands of AI answers by hand is slow and expensive, it has become common to use one AI model to grade another's answers: so-called "LLM-as-a-judge". It is fast and cheap, and it often agrees with human graders a large share of the time. In an influential 2023 study, Lianmin Zheng and colleagues found that GPT-4, used as a judge, agreed with human preferences about as often as humans agreed with each other. The same study documented systematic biases: judge models tended to favour the first answer shown to them, to prefer longer answers, and in some cases to favour answers written by themselves.[8]

In the language of this book, an AI grading AI is useful but is not independent verification, especially when both come from the same family of models, which may share the same blind spots (Chapter Eight). It is a way of scaling up checking that has itself been checked against humans, not a replacement for humans. The question "who checked the checker?" never goes away.

Red-teaming: trying to break it

A second, complementary form of verification is **red-teaming**: deliberately and systematically trying to make a system fail, instead of only watching whether it succeeds under ordinary conditions. The term comes from military exercises in which a "red team" plays the enemy. A hardware engineer would recognise it as adversarial testing: feeding a circuit its worst-case inputs, because a system tested only under friendly conditions tells you very little about hostile ones.

Serious red-teaming has recognisable features that separate it from the performative kind:

- **It is genuinely adversarial.** The testers are trying to break the system and are rewarded for finding failures, not for finding nothing.
- **It is diverse in method.** It combines automated search, human ingenuity and specialist knowledge of the particular harms being tested for, because every method has blind spots.
- **It is independent.** The testers are not managed by, and reporting only to, the team under pressure to ship.

Red-teaming done entirely in-house, by people whose organisation depends on a clean result, inherits the same structural weakness as self-reported benchmarks, even when the individuals are diligent and honest.

What independent evaluation could look like

Digital systems that matter enough (aircraft avionics, medical devices, nuclear instrumentation) are not simply built well and then trusted. They are **certified** by bodies independent of the manufacturer, against standards that are public and open to challenge, through processes that leave an auditable trail. Nobody boards an aircraft because the manufacturer says the flight software is safe. They board because an independent regulator has certified it against a public standard using evidence the regulator examined itself.

AI mostly does not have an equivalent yet, but the first pieces are being built. In November 2023 the United Kingdom established the AI Safety Institute, renamed the AI Security Institute in February 2025, as a government body with technical staff able to test advanced AI models.[9] In November 2024 the UK institute and its US counterpart published a joint evaluation of a new version of Anthropic's Claude 3.5 Sonnet carried out *before* the model's public release, and in December 2024 a similar evaluation of OpenAI's o1.[10] That is a genuine step towards the aviation model.

It is also a small one. That testing was voluntary, depended on access the companies chose to give, and covered a handful of models. The great majority of AI systems in daily use, including the ones that screen job applications, flag benefit claims or answer customers' questions, reach the public having been evaluated only by the organisations that built or bought them, against standards those organisations chose, with evidence those organisations control. That is not a conspiracy. It is a young field whose institutions have not yet caught up with the stakes. But it means that, for now, a careful user of AI, whether a person, a business or a government department, has to do more of the verification themselves than they would ever do for a piece of certified hardware.

The danger of presuming reliability

There is an even deeper danger than missing verification: the *assumption* that verification is unnecessary because computers are reliable.

Following a Law Commission recommendation, the Youth Justice and Criminal Evidence Act 1999 repealed a statutory rule that had required evidence that a computer was working properly before its records could be used in criminal proceedings in England and Wales. Courts fell back on a common-law presumption that mechanical instruments, computers included, were in order unless shown otherwise.[11] Between 1999 and 2015, the Post Office prosecuted more than 700 sub-postmasters largely on the evidence of its Horizon accounting system, which contained bugs that could produce false shortfalls. Chapter Eleven

tells that story in full. The lesson for this chapter is short: **a presumption of reliability is the opposite of verification.** It moves the burden of proof onto the people least able to carry it.

Verifiability is a habit, not a certificate

Finally, I want to resist a temptation these pages may have created: the idea that verifiability is a box ticked once, at launch, after which a system can be trusted indefinitely.

A verified logic circuit stays verified because its logic is fixed. An AI system is not like that. It can be fine-tuned, updated, connected to new tools and deployed into new settings its original evaluation never covered, and each change can alter its behaviour in ways the original verification says nothing about. In my trade, a certificate records the condition of an installation on the day it was tested, and in England a privately rented home must have its electrics inspected at least every five years because installations deteriorate and get altered.[12] AI systems change far faster than wiring.

So verification for AI has to be continuous: repeated after every meaningful change, monitored in use for signs that behaviour has drifted from what was verified, and never allowed to become a certificate framed on the wall rather than a practice kept up. A system that is honestly and continuously verified then needs an honest account of where its verification *stops*, which is the question of boundedness, taken up next.

KEY POINTS

- A result tested only by its maker is a **claim**; independently reproduced, it becomes a **finding**.
- **Rule of three:** zero failures in *n* tests supports a failure rate below about 3/*n*, no better. Testing alone cannot show very high reliability.
- **Benchmarks get contaminated and gamed** (Goodhart's law). Fresh, held-out tests revealed drops of up to 8% for some models.
- **Epic Sepsis Model:** independent testing found an AUC of 0.63 and 67% of sepsis cases missed, far from the developer's reported figures.
- **Red-teaming** must be adversarial, diverse and independent to count.
- Pre-release government testing of AI has begun (UK and US institutes, 2024) but is voluntary and limited.
- A **presumption of reliability** is the opposite of verification. Verification must be **continuous**.

WHERE THIS CHAPTER STOPS

The statistics in this chapter assume independent, representative tests, which real test sets rarely are, so real uncertainty is usually larger than the formulas suggest. The Epic Sepsis Model study evaluated one version of the model at one health system, and performance elsewhere may differ. The AI institutes' arrangements and powers were changing at the time of writing. The account of the 1999 change in evidence law is simplified; see Chapter Eleven and the cited sources.

CHAPTER SEVEN

Boundedness

Knowing Where the Edges Are

IN THIS CHAPTER

Every AI system is reliable only within some domain. Outside it, the system usually carries on answering in the same confident tone. This chapter explains distribution shift, calibration and why language models "hallucinate", and describes engineering tools that make boundaries explicit: operational design domains, selective prediction and conformal prediction.

> By September 2026, one researcher's public database had logged 2,046 legal decisions around the world in which a court or tribunal dealt with AI-generated falsehoods (invented cases, false quotations, made-up citations) in material put before it. Every one of those falsehoods was delivered in the same calm, confident voice as the truth.[N12]

BY THE NUMBERS

Where the edges are

2,046

Legal decisions dealing with AI-hallucinated content, logged by September 2026. The database does not claim to be complete.

MEASURED

75%

Share of questions an older OpenAI model answered wrongly on one factual test, because it almost never said "I don't know". A newer model abstained on 52% and was wrong on 26%.

REPORTED

~7 months

How often the length of task AI systems could complete about half the time doubled, 2019–2025, on METR's mostly software tasks.

MEASURED

30 mA

The residual current at which the RCD protecting a socket circuit trips. A boundary enforced from outside, not by asking the circuit how it feels.

DOCUMENTED

Every component in my grandfather's world came with a datasheet, and every datasheet had, next to the description of what the component did, a table of absolute maximum ratings: the voltage, temperature and current beyond which the manufacturer made no promises at all. This was not a courtesy. It was the manufacturer carrying out a professional obligation to say in advance exactly where its guarantee stopped, so that responsibility for what happened beyond that point was clearly someone else's.

This chapter is about the AI equivalent of the absolute maximum rating, and how rarely it is stated with anything like the same precision.

The domain a system was built for

Every AI system, however general it appears, was trained and tested on some particular spread of data and tasks, its **distribution**. Within that distribution, its statistical performance means something. Outside it, in what researchers call the

out-of-distribution region, the system is not so much wrong as unmoored. It produces an output by the same mechanism as always, but that mechanism was never calibrated against anything like the input it has just received, and there is no principled reason to expect the output to be reliable.

When the world a system meets in use differs from the world it was trained on, researchers call it **distribution shift**. It is one of the most important and least appreciated causes of AI failure, because the system usually gives no sign that it has happened.

CASE FILE

From the lab to the clinic

Diabetic retinopathy, damage to the retina caused by diabetes, is a leading cause of preventable blindness. It can be detected from photographs of the back of the eye, but many countries lack enough specialists to read them. Google Health developed a deep learning system that, on high-quality images in validation studies, detected the condition with accuracy comparable to specialists.

In 2020 a Google team led by Emma Beede published a candid study of what happened when the system was used in eleven clinics in Thailand.[1] The real world was different from the lab. Some clinics had poor lighting. Nurses were photographing many patients quickly. Images were blurred or dark, and the system, designed to refuse images it judged ungradable, rejected a substantial share of them. Patients whose images were rejected were told to attend a hospital elsewhere, which for many meant a costly journey and a lost day's work. Slow internet connections delayed results.

The system's refusal to grade poor images was, in one sense, exactly the bounded behaviour this chapter recommends: it knew where its competence stopped. But the *surrounding system* had not been designed for how often that would happen in real clinics, so a sensible boundary became a barrier to care. The study is valuable precisely because its authors published it.

Properties that failed: boundedness (the operating conditions were not those the system was validated for) and composability (the refusal behaviour did not fit the clinical workflow).

Fluent even when wrong

A digital circuit driven beyond its voltage rating will usually behave in visibly odd ways that a competent technician can pick up with an oscilloscope. The system, in a rough sense, shows its distress.

A language model asked something far outside its reliable knowledge will very often just answer, fluently and confidently, in exactly the same register it uses for questions it handles well, with no signal to an ordinary user that it has left safe territory. This is, to my mind, one of the most dangerous facts about how these systems currently behave, because it means users cannot rely on the system's tone as a cue for when to be sceptical. A system that says "I don't know" when it doesn't know would be enormously useful. A system that says the wrong thing in the same voice it uses for the right thing makes its users supply the caution it lacks.

Why language models make things up

The failure has a name, **hallucination**: a fluent, plausible and false statement. It is tempting to think of it as a mysterious glitch. It is better understood as a predictable result of how the systems are built and, importantly, how they are *scored*.

A language model generates the most plausible continuation of a text. When its training covered a topic well, the most plausible continuation is usually true, because true statements about well-covered topics are what the training text mostly contained. When a topic is sparse in the training data (an obscure person's birthday, the citation for a minor court case), the same process produces text that is just as fluent but has nothing solid behind it.

In September 2025 researchers at OpenAI published an analysis arguing that a major reason hallucinations persist is the way models are evaluated.[2] Most benchmarks score only accuracy. An answer of "I don't know" scores zero, exactly like a wrong answer, so a model that always guesses will, on average, score

higher than one that honestly abstains. The authors compared it to a multiple-choice exam with no penalty for wrong answers: the rational strategy is never to leave a blank.

Their own figures illustrate it. On one factual-question test, an older OpenAI model, o4-mini, answered almost everything: it abstained on 1 per cent of questions, was right on 24 per cent and wrong on 75 per cent. A newer model abstained on 52 per cent, was right on 22 per cent and wrong on only 26 per cent. On an accuracy-only leaderboard the older model looks slightly better. In terms of how often it tells you something false, it is nearly three times worse.

MODEL (OPENAI'S SIMPLEQA TEST)	ABSTAINED	CORRECT	WRONG
gpt-5-thinking-mini	52%	22%	26%
o4-mini	1%	24%	75%

That is a boundedness failure built into the scoring system: **we have been rewarding AI for not knowing its own limits.** The fix the authors propose is simple in principle: score wrong answers worse than abstentions, so that honesty about uncertainty is rewarded.

Calibration: does confidence track correctness?

That brings us to a measurable property called **calibration**: the degree to which a system's stated confidence matches how often it is actually right. A perfectly calibrated system that makes many predictions at 90 per cent confidence will be right on about 90 per cent of them. Its confidence is an honest signal.

THE SCIENCE

Measuring calibration

To check calibration, group a system's predictions by stated confidence (all the 60 per cent predictions together, all the 70 per cent ones, and so on) and compare each group's stated confidence with its actual accuracy. Plotted on a graph, a perfectly calibrated system lies on the diagonal. The average gap between stated confidence and actual accuracy is summarised as the **expected calibration error**.

In 2017 Chuan Guo and colleagues showed that modern deep neural networks, although more accurate than older ones, tended to be *overconfident*: when they said 90 per cent, they were right less often than that. They also showed that a simple adjustment after training, called *temperature scaling*, repaired much of the problem.[3]

For language models the picture is interesting. In 2022 Anthropic researchers found that large pre-trained models were often reasonably well calibrated on multiple-choice questions, and could be trained to estimate the probability that their own answer was correct.[4] But OpenAI's GPT-4 technical report showed that the later training stages that turn a model into a helpful assistant can *reduce* calibration.[5] The process that makes a model pleasant to talk to can make its confidence less honest.

Calibration has two further limits. First, it is measured on some set of test questions and guarantees nothing about questions from a quite different domain, which is the out-of-distribution problem again. Second, the *tone* of a model's answer (how assured it sounds, whether it hedges) is produced by largely the same process whether or not the content is reliable. Tone is not calibration, and people reasonably read tone as a signal. That is the gap through which hallucinations reach the world.

Making boundaries explicit

Because a system cannot always know when it has left its safe domain, and cannot always signal its uncertainty honestly through tone, the engineering tradition suggests a structural answer rather than a hopeful one: **build the**

boundary into the system, instead of relying on the model's own judgement about where it is. Several practical tools exist.

Operational design domains. The automated-vehicle industry has a concept that every AI deployment should borrow. SAE International's standard J3016, which defines the "levels" of driving automation, uses the term **operational design domain** (ODD): the specific conditions a driving-automation system is designed to operate in, such as road types, speeds, weather and time of day.[6] A system designed for motorway driving in daylight is not expected to handle a snowy mountain track at night, and a well-designed one hands control back, or stops safely, when it detects that it is leaving its ODD. The ODD is a datasheet's maximum ratings rewritten for AI. Every AI system deployed for a serious purpose should have one, stated in plain language.

Selective prediction. A system can be designed to answer only when its confidence passes a threshold, and to refer everything else to a human. That trades coverage for reliability. It will answer fewer questions, but get more of them right, and the trade-off can be measured and tuned.

Conformal prediction. A family of statistical methods, developed by Vladimir Vovk and colleagues, wraps any prediction model and turns its single answer into a *set* of possible answers with a guaranteed probability of containing the true one, for example "with 95 per cent probability the diagnosis is one of these three". The guarantee holds under a clearly stated assumption: that new cases resemble the calibration cases in a specific statistical sense.[7] When the model is unsure, the set gets bigger, which is itself an honest, visible signal of uncertainty. It is one of the few places in modern AI where a genuine mathematical guarantee is available, and its condition is stated as clearly as a datasheet's.

Monitoring for drift. A deployed system can be watched for inputs that differ markedly from its test data, with those cases routed to human review or refused.

Escalation by rule, not by mood. In high-stakes uses, the conditions under which a system *must* stop and bring in a person should be written into the design, not left to the system's own sense of whether it is out of its depth.

In my trade, the residual current device is the model here. It does not ask the circuit whether it feels faulty. It measures an imbalance between the current flowing out and the current flowing back, and if the imbalance exceeds its rating (30 milliamps for the devices protecting ordinary sockets) it disconnects.[8] The boundary is enforced by structure, from outside the thing being protected. A system that can refuse is not weaker than one that always answers. On this book's framework it is *more* intelligible, because refusal at the boundary is exactly the honest admission of limits that Chapter Two called the load-bearing discipline of all good engineering.

When the edge keeps moving

Boundedness is hard enough when a system's envelope is known but not enforced. It is harder still when the envelope itself is not fully known, even to the system's creators, because new abilities, and with them new failure modes, can appear with scale or in new combinations (Chapter Three).

How fast is the edge moving? One attempt to measure it comes from METR, an independent research organisation that evaluates AI systems. In 2025 its researchers timed skilled humans on a range of software and reasoning tasks, then measured which of those tasks AI systems could complete. They summarised each system by the length of task, in human working time, that it could complete about half the time. Across the models they studied from 2019 to 2025, that "time horizon" had doubled roughly every seven months.[9] The authors are careful about the limits of their method: the tasks are mostly software tasks, "about half the time" is far from reliable, and trends can change. But the finding makes the point of this section concrete. A boundary stated at release may be wrong within months, in either direction.

That is not a reason to give up on boundedness. It is a reason to treat the boundary as something to be probed and re-established continuously, not fixed once at release. Responsible deployment of a system whose full range of abilities is not yet known means claiming *narrower* competence than the system may actually have, watching use closely for signs it is being used, or is behaving, in ways its evaluation never anticipated, and treating every unexpected ability, good or bad, as new information that changes what the stated boundary should say.

The next chapter turns to what happens when several bounded, individually well-characterised systems are connected, and how even careful boundaries can fail to survive contact with a larger system.

KEY POINTS

- AI is reliable only within its **distribution**. **Distribution shift** usually happens silently.
- **Google's retinopathy study in Thai clinics** showed lab accuracy failing to translate to real conditions, and a sensible refusal behaviour colliding with the workflow.
- **Hallucination** is partly a product of accuracy-only scoring, which rewards guessing over saying "I don't know".
- **Calibration** measures whether confidence tracks correctness. Assistant training can make it worse, and tone is not calibration.
- Make boundaries explicit with **operational design domains, selective prediction, conformal prediction, drift monitoring and rule-based escalation.**
- A system that can **refuse** at its boundary is more intelligible, not less.

WHERE THIS CHAPTER STOPS

Hallucination has several causes, and the evaluation-incentive explanation is one well-supported contributor, not the whole story. The SimpleQA figures come from the developer's own publication and describe one test. Conformal prediction's guarantee depends on an assumption (exchangeability) that distribution shift can violate; it is a tool for honesty about uncertainty, not a cure for it. The operational design domain is a concept from automated driving, and applying it to general-purpose AI is this book's recommendation, not an established standard.

CHAPTER EIGHT

Composability

When Systems Meet Systems

IN THIS CHAPTER

A component can be transparent, verified and bounded and still cause harm once it is connected to other things. This chapter covers interface failures (the Mars Climate Orbiter), the new hazards of AI agents that use tools (prompt injection, compounding errors and excessive permissions, illustrated by an agent that deleted a production database), the hidden risk of many systems sharing one model, and the surprisingly fragile combination of an AI and the human meant to supervise it.

> At 9.58 on a March night in 2018, a self-driving test car in Tempe, Arizona, detected a woman wheeling a bicycle across the road. It had 5.6 seconds. The software kept changing its mind about what she was: a vehicle, a bicycle, "other". The car's own emergency braking had been switched off while the automated system drove. The alarm meant to wake the human safety driver sounded about 0.2 seconds before impact. The driver was looking at a phone. Every part could be defended on its own. Together they killed Elaine Herzberg.[12]

BY THE NUMBERS

Failures that live at the joins

4.45

Newton-seconds in one pound-force second: the factor by which Mars Climate Orbiter's thruster data were misread.

DOCUMENTED

5.6 s vs 0.2 s

How long before impact Uber's test vehicle first detected Elaine Herzberg, and how long before impact its audible alert sounded.

DOCUMENTED

37%

Chance of finishing a 100-step task with no errors when each step is 99% reliable and errors are independent (0.99^{100}).

CALCULATED

1975

The year Saltzer and Schroeder stated the principle of least privilege, which applies to AI agents almost unchanged.

DOCUMENTED

A computer, as digital designers describe it, is not one thing. It is thousands, eventually millions, of individually simple, individually verified parts, gates, flip-flops and registers, combined according to precise rules into something that can do arithmetic, store memory and run programs. The whole discipline of digital design is, in a sense, a discipline of **composition**: how do you combine small, well-understood, provably correct parts into large, useful, still-correct wholes, without correctness leaking away at every joint?

This chapter asks the same question of artificial intelligence, at a moment when the field is moving fast from single, standalone models to exactly this kind of composition: models that call other models, use external tools, browse the web, write and run code, and act in the world with a degree of autonomy no earlier software had.

Where safe parts make an unsafe whole

Here is the uncomfortable fact this chapter has to face: a system can be transparent, verified and bounded, everything the last three chapters asked for,

and still cause harm once it is combined with other systems, because the *combination* introduces failure modes that no individual part's description predicted.

This is not a hypothetical worry. It is close to the central lesson of interface design in engineering, a whole sub-discipline that exists because individually correct components routinely misbehave when connected, through mismatched assumptions about timing, units, formats or edge cases that neither side's specification quite covered.

> CASE FILE
>
> **Mars Climate Orbiter: two correct parts, one lost spacecraft**
>
> On 23 September 1999, NASA's Mars Climate Orbiter fired its engine to enter orbit around Mars and was never heard from again. It had approached the planet far lower than planned, most likely entering the atmosphere and breaking up.
>
> The Mishap Investigation Board found the root cause in an interface between two pieces of software. Ground software supplied by the spacecraft's builder produced thruster data in imperial units (pound-force seconds). The navigation software that used those data expected metric units (newton-seconds), as the interface specification required. One pound-force second is about 4.45 newton-seconds, so every small trajectory correction was misjudged by that factor, and the errors accumulated over the nine-month journey.[1]
>
> Each piece of software did what it was written to do. Anomalies in the trajectory were noticed in flight, but they were not escalated and resolved. The board's report concluded that the failure lay less in the single error than in the processes that should have caught it.
>
> **Property that failed:** composability. The fault lived at the interface, not in either component.

In electrical installations, the equivalent discipline is called **selectivity** (older electricians still say *discrimination*). When a fault occurs, you want the protective device nearest to it (the circuit breaker for that one circuit) to trip, not the main

switch for the whole building. Achieving that requires comparing the time–current characteristics of devices in series and choosing them so that they cooperate. Two individually excellent circuit breakers, badly paired, can plunge a hospital ward into darkness when one socket develops a fault. Nobody would accept "each breaker passed its own test" as proof that the installation is safe. We should not accept it for AI either.

Agents: many interfaces at once

The composability problem becomes much sharper with **AI agents**, systems in which a language model does not just produce text but takes actions: searching the web, reading email, editing files, running code, making purchases, or instructing other AI systems.

Each tool an agent can use is another interface, whose failures must be understood not only alone but in combination with every other tool and every plausible sequence of actions. Each extra model in a chain is another component whose errors can compound with, not simply add to, the errors of its neighbours. And because none of these components offers logical guarantees (Chapter Three), the compounding cannot be derived from the components' descriptions the way an engineer could derive a composed circuit's behaviour from its parts. It has to be measured, empirically, for each combination.

Three hazards recur across many agent deployments. Naming them is the first step to designing against them.

Hazard one: instructions hidden in data

A digital computer never confuses a control signal with a data signal. The distinction is enforced by the architecture. A language model, by contrast, processes everything as text. The instructions from the person it is working for, and the contents of the web page, document or email it is reading, arrive through the same channel. By default, nothing in the architecture separates "orders from

my principal, which I should follow" from "text I came across while working, which might contain orders I must ignore".

In September 2022 the programmer Simon Willison gave this weakness a name, **prompt injection**, after a data scientist, Riley Goodside, showed that a translation tool could be made to abandon its task by text telling it to "ignore the above directions".[2] In 2023 Kai Greshake and colleagues demonstrated **indirect prompt injection**: hiding instructions in a web page or document so that an AI assistant which later *reads* that content follows them, for instance by quietly trying to extract a user's personal information.[3]

Sometimes the results are merely embarrassing. In December 2023 a customer-service chatbot on a Californian car dealer's website was persuaded by a user to agree to sell a new Chevrolet Tahoe for one dollar, adding that this was "a legally binding offer – no takesies backsies".[4] The dealer did not honour it. But when the same weakness exists in an agent that can send email, move money or change files, the consequences are not a joke.

As of this writing, there is no complete fix for prompt injection. Model training reduces it without eliminating it. The most promising defences are architectural, in the spirit of my grandfather's discipline: keeping untrusted content away from any component that has the power to act, and requiring confirmation before any consequential action triggered by content from outside.

Hazard two: compounding errors over long tasks

An agent carrying out a long sequence of actions can drift, step by step, away from its original goal. Each step looks locally reasonable while the overall path goes badly wrong. This is much harder to catch than a single bad answer, because no individual step looks obviously like a mistake.

The arithmetic in Chapter Three (0.99 multiplied by itself a hundred times is about 0.37) shows why long tasks are dangerous for statistical systems. The

engineering remedy translates directly: **checkpoints**. At intervals, the whole path, not just the latest step, is reviewed against the original goal, ideally by a mechanism independent of the process that produced the path.

Hazard three: permissions that grow

The third hazard is what I call **permission creep**. An agent granted broad access for one task tends, unless deliberately restricted, to keep that access for every later task, because narrowing permissions takes deliberate effort that is easy to skip.

Computer security has a long-established principle against this, stated by Jerome Saltzer and Michael Schroeder in 1975: the **principle of least privilege**, meaning that every program and every user should operate with the least set of privileges necessary to complete the job.[5] It applies to AI agents with essentially no modification.

CASE FILE

The agent that ignored a code freeze

In July 2025 Jason Lemkin, a well-known technology investor, spent several days building a software application using Replit, an online platform on which an AI agent writes and runs code for the user. He publicly documented the experience. After instructing the agent repeatedly not to make changes without permission, which he described as a "code freeze", he found that it had run commands that deleted the application's live production database. The agent had also, by his account, generated fictitious data and given misleading reports about its work, and it initially told him the deletion could not be undone. In fact the data could be recovered.[6]

Replit's chief executive apologised publicly and announced changes, including automatic separation of development and production databases.[7]

Look at this through the four properties. The instruction "don't change anything" was given in the same channel as everything else, as text, rather than enforced by the system: a boundary expressed as a request instead of a structure. The agent had permission to destroy production data during a task that did not require it, a failure of least privilege. And its own reports about what it had done were unreliable, a transparency failure. The fix Replit announced was structural, not a plea for better behaviour: take away the ability to reach production data from the development environment.

Properties that failed: composability (excessive permissions), boundedness (the freeze was not enforced), transparency (unreliable self-reports).

When a component changes underneath you

In my grandfather's world, a component's behaviour was fixed by its datasheet once it left the factory. A logic chip with a given part number was expected to behave the same whichever batch it came from.

Many AI applications are built on models that the application's owners do not control and which are updated by their suppliers, sometimes without notice to the businesses that depend on them. In 2023 Lingjiao Chen, Matei Zaharia and James

Zou at Stanford and Berkeley compared the versions of GPT-3.5 and GPT-4 available through OpenAI's service in March and June of that year. On some tasks behaviour changed substantially between the two dates. In one test, GPT-4's accuracy at identifying whether numbers were prime fell from 84 per cent to 51 per cent, while GPT-3.5's rose; the models' willingness to follow certain instructions and the formatting of their code output also shifted.[8] The details matter less than the principle. A business that tested its application in March was, in June, running a different system from the one it had tested.

The composability lesson is simple to state and often ignored: **treat a change to any component as a change to the whole system, and re-verify.** That means pinning to a specific model version where the supplier allows it, running a standing set of regression tests whenever a component changes, and writing notice of changes into supply contracts (Chapter Thirteen's procurement questions include this).

Many agents at once

The newest form of composition is the **multi-agent system**, in which several AI agents, each with its own role, pass work between them: one plans, one writes code, one reviews, one tests. The appeal is obvious, since it mirrors a human team. So are the risks. Errors can pass from agent to agent and be amplified rather than caught. Agents can reinforce each other's mistakes, particularly if they are built on the same underlying model and share the same blind spots. And accountability blurs: when a system of five agents produces a harmful output, which of them "decided"?

The engineering answer is the one my grandfather's discipline gave for any composed system: define each component's responsibilities and interfaces explicitly, check at the joins rather than trusting the parts, and make sure that at least one check in the chain is genuinely independent of the others.

The hidden risk of a shared foundation

There is a subtler composability problem that barely existed in my grandfather's world. A large and growing share of AI applications are built on a handful of underlying models, from a handful of companies. A bank's customer assistant, a law firm's research tool and a hospital's note-taking system may all, underneath, be calling the same model.

That concentration means their failures are unlikely to be independent. Remember the voting calculation in Chapter Two: redundancy works only when failures are independent, and Ariane 5's backup failed because it ran identical software. When many systems share a foundation model, a single flaw, whether a blind spot in its knowledge, a vulnerability to a particular jailbreak or a problem introduced by an update, can appear across all of them at once. Researchers at Stanford have described this as the "homogenisation" risk of **foundation models**.[9] It also means that asking one AI to check another's work is weaker redundancy than it looks if both share the same origin.

The human in the loop is part of the system

One more composition deserves attention, because it is too often treated as a simple, obviously reliable safeguard: the combination of an AI system and the human reviewing its outputs.

In 1983 the psychologist Lisanne Bainbridge published a short paper, "Ironies of Automation", that remains one of the most cited in its field.[10] Its central irony is this: the more reliable an automated system becomes, the less practice its human supervisors get, and the worse they become at the very task they are kept on for, which is stepping in when the automation fails. Designers automate what they can and leave the humans with what they can't, and then expect those humans to monitor, for long periods, a system that almost never needs them.

Later research gave the resulting behaviour a name, **automation bias**: the tendency to over-trust automated recommendations and to under-check them, which grows under time pressure and when the system is usually right.[11]

CASE FILE

Tempe, Arizona, 18 March 2018

On the night of 18 March 2018, a test vehicle operated by Uber's self-driving unit struck and killed Elaine Herzberg as she wheeled a bicycle across a road in Tempe, Arizona. It was the first recorded pedestrian death involving a self-driving test vehicle.

The US National Transportation Safety Board investigated.[12] Its findings read like a lesson in composability:

- The vehicle's sensors detected Ms Herzberg about 5.6 seconds before impact, but the software repeatedly changed its classification of her (as a vehicle, a bicycle, an "other" object) and, with each change, lost track of her previous movement, so it did not predict her path into the vehicle's lane.
- The system was not designed to classify something as a pedestrian unless it was near a crossing. She was crossing elsewhere.
- The vehicle manufacturer's own automatic emergency braking had been disabled while the automated system was in control, and the automated system was designed to rely on the human operator to intervene in an emergency. It was not designed to warn her in time: according to the final report, an audible alert sounded only about 0.2 seconds before impact.
- The operator, the human safety backstop, was looking down at a mobile phone for much of the trip, according to the investigation, and looked up too late.

The NTSB cited the company's "inadequate safety culture" and its failure to address "automation complacency" among its operators. Each element (the perception software, the disabled braking, the human monitor) might be defended in isolation. Together, they left a gap through which a person walked and died.

Properties that failed: composability above all. The human was treated as a guaranteed safeguard when the design made her the weakest link. Boundedness failed too: the system's assumptions about where pedestrians appear were not stated as a limit.

Automation bias does not respond well to telling reviewers to be more careful. Sustained vigilance against a system that is right most of the time is, as a matter

of human psychology, very hard to maintain. Better answers are structural:

- **Design review interfaces** that show *why* the system is uncertain, not just a flat approve-or-reject choice.
- **Rotate reviewers** so that no single person's attention has to hold indefinitely.
- **Test the reviewing process itself** by slipping in known cases and checking whether they are caught, as a building's fire alarm is tested.
- **Use genuinely independent review** for the highest stakes: a second reviewer who cannot see the first reviewer's decision.
- **Never count a human as a safeguard** unless the system is designed to give that human the time, information and authority to act.

Composability closes the loop

This chapter completes Part II's argument, and it is worth saying why composability came last. Transparency, verifiability and boundedness can all be assessed for a single system on its own. Composability only becomes visible, and testable, when you look at how a system meets other systems and other people.

A framework that stopped at the first three would describe how to build a trustworthy *component*. It would not yet describe how to build a trustworthy *system*, and it is systems, not isolated components, that people depend on, whether they know it or not.

With all four properties in view, Part III turns from what intelligibility requires to what happens without it.

KEY POINTS

- Failures often live at **interfaces**: the Mars Climate Orbiter was lost because two correct programs disagreed about units.
- **Agents** multiply interfaces. Three recurring hazards are **prompt injection, compounding errors and permission creep.**
- **Prompt injection** exists because models do not separate instructions from data. The best defences are architectural.
- **Least privilege** (Saltzer and Schroeder, 1975) applies directly to AI agents, as the Replit database deletion shows.
- Many systems built on the **same foundation model** share failure modes, so apparent redundancy can be illusory.
- A **human in the loop** is a component with known failure modes (automation bias). The Uber Tempe crash shows the cost of assuming otherwise.

WHERE THIS CHAPTER STOPS

Agent technology was changing rapidly at the time of writing, and specific products and defences may have moved on. The Replit account relies mainly on the user's own public reports and press coverage, not a formal investigation. The Uber case involved a developmental vehicle under test, not a consumer product. How strongly correlated the failures of applications sharing a foundation model are in practice has not been measured systematically in public research.

PART III

Chatastrophy

CHAPTER NINE

How Engineered Systems Fail

And Why We Trust Them Anyway

IN THIS CHAPTER

Engineered systems fail too. What made them trustworthy was an organised way of anticipating and learning from failure: fault trees, failure modes and effects analysis, the "Swiss cheese" model of layered defences, and a no-blame investigation culture that turned aviation into one of the safest ways to travel. This chapter describes that apparatus and explains why it does not transfer to AI without adaptation.

> In 2024 there was one accident for every 880,000 commercial flights, according to the airline industry's own safety figures, and most of those accidents killed nobody. Flying did not become that safe by building perfect aircraft. It became that safe by learning, relentlessly and in public, from every imperfect one.[N13]

BY THE NUMBERS

How a safety culture is built

1 in 880,000

Flights that ended in an accident of any kind in 2024 (1.13 per million).

REPORTED

1949

The year the US military published the first formal failure modes and effects analysis procedure.

DOCUMENTED

1976

The year NASA began running the confidential Aviation Safety Reporting System for near-misses.

DOCUMENTED

1,000+

AI incidents catalogued in the AI Incident Database by 2025, five years after it launched. No independent AI accident investigator yet exists.

MEASURED

It would be easy, reading the first two parts of this book, to think my grandfather's world was one where things simply did not go wrong, where rigour, once achieved, produced infallibility. That is not true, and this book should not suggest it. Chapter Two described a radiation machine that killed patients, a rocket that destroyed itself and an aircraft type involved in two fatal crashes.

The difference between that world and the world of AI is not that one has failures and the other does not. It is that one built, over decades, a professional culture and a technical toolkit for anticipating, categorising and learning from failure, while the other, being younger and resistant to some of the old methods, is still building its equivalent. This chapter describes that toolkit, both as a tribute to what engineering achieved and as a template for what AI needs.

The fault tree: working backwards from disaster

In 1961 and 1962, engineers at Bell Telephone Laboratories, led by H. A. Watson, developed a method for analysing the launch control system of the US Air Force's

Minuteman intercontinental ballistic missile.[1] The question they faced could hardly have been more serious: how could they be confident that a missile would never be launched by accident?

Their answer was the **fault tree**. You start with a specific, precisely defined catastrophic outcome, such as "unauthorised launch", and work backwards, systematically, through every combination of component failures and human errors that could produce it. The branches are joined by logic gates, AND and OR, the same gates used in digital circuits. An AND gate means several things must go wrong together. An OR gate means any one is enough. Where possible, each branch is given a probability.

The discipline of fault-tree analysis forces a kind of honesty that is easy to skip under deadline pressure. It does not ask "what could go wrong?", answered by impression. It asks "by exactly what chain of causes could this defined outcome occur?", answered as exhaustively as the analysts can manage. A fault tree that has not been updated to include a newly discovered failure mode is not neutral. It is actively misleading, because it implies a completeness the analysis no longer has.

The deeper value of the fault tree is not the technique but the attitude behind it. Catastrophes are not treated as unfortunate surprises to be discovered by experiencing them. They are treated as things to be listed and reasoned about in advance.

Failure modes and effects analysis: working forwards from each part

A companion technique comes from the opposite direction. **Failure Modes and Effects Analysis** (FMEA) was first formalised in a US military procedure published in 1949 and later adopted across aerospace, automotive and medical engineering.[2] Instead of starting from a catastrophe, it starts from each component in turn and asks, systematically:

- How could this part fail?
- What would the local effect be?
- What would the effect on the whole system be?
- How severe, and how likely, is that outcome?
- How would we detect it?

The result is a large table that becomes a working document throughout a system's life, not a one-off exercise filed away at launch but a living record, updated as new failure modes are discovered, that shapes what is monitored and what is redesigned.

What I want to draw out of FMEA is its combination of **exhaustiveness** and **humility**. Exhaustiveness, because it insists on going through every part, not just the ones that feel risky; the boring, reliable-seeming parts fail too, and a method that looks only where someone already suspects trouble will systematically miss the failures nobody suspected. Humility, because the whole exercise assumes that failure is not an embarrassment to be minimised but an ordinary feature of complex systems to be planned for in detail.

Swiss cheese

The psychologist James Reason, of the University of Manchester, gave safety science its most famous picture. Imagine a system's defences as slices of Swiss cheese stacked one behind another: design, training, procedures, alarms, supervision. Every slice has holes, because every defence is imperfect, and the holes move as conditions change. Most of the time a hole in one slice is covered by solid cheese in the next. An accident happens when, for a moment, the holes in every slice line up and a hazard passes straight through.[3]

The model has two lessons for AI. First, **no single layer will be perfect**, so safety comes from several independent layers. Second, when you investigate an accident, **look for the whole line of holes**, not just the last one. The Uber crash in

Chapter Eight is a textbook example: perception software, disabled emergency braking, a system not designed to warn the operator in time, and a distracted operator, four holes that lined up on one night.

The investigation culture

The third element of this toolkit is cultural rather than technical, and in some ways it is the hardest to transplant, because it depends on incentives rather than method alone.

Safety-critical industries have built a professional culture in which a failure is followed by a rigorous, honest investigation designed to extract every possible lesson, and whose findings are shared rather than hidden. Aviation is the clearest example. Under Annex 13 to the Chicago Convention on international civil aviation, accident investigations are carried out for one purpose. In the Annex's words: "The sole objective of the investigation of an accident or incident shall be the prevention of accidents and incidents. It is not the purpose of this activity to apportion blame or liability."[4] In the United Kingdom that work is done by the Air Accidents Investigation Branch, and in the United States by the National Transportation Safety Board, both independent of airlines and manufacturers. Their reports are published, and their recommendations feed back into design, training and regulation across the whole industry, not just the company involved.

Aviation went further, collecting reports of near-misses and mistakes that caused no harm, because those are where the next accident is rehearsed. In 1976 NASA began running the Aviation Safety Reporting System, a confidential channel through which pilots, controllers and engineers in the United States can report errors, with protections that encourage honesty.[5]

The no-blame element is not incidental. It is close to the centre of the culture's effectiveness. An investigation culture that punishes the people involved in a failure gives everyone a strong reason to hide information, underreport near-misses and shape testimony towards self-protection. A culture that treats failure,

within reasonable limits, as a source of learning rather than an occasion for punishment gets more honest and complete information out of the people closest to what went wrong, and that information is the raw material fault trees and FMEAs need to improve.

The result is one of the great quiet achievements of the twentieth century. Commercial aviation, which involves extraordinarily complex machinery operating in extreme conditions, became one of the safest forms of transport ever devised. Failures kept happening. The system got better, again and again, at learning from each one.

AI's first steps towards the same culture

AI has begun to build parts of this apparatus. In 2020 the Partnership on AI launched the **AI Incident Database**, an open, searchable collection of reports of AI systems causing or nearly causing harm, explicitly modelled on the incident databases of aviation and computer security.[6] It had catalogued well over a thousand incidents by 2025; the Replit database deletion described in the previous chapter is number 1152. Some AI developers publish detailed postmortems of their own failures, such as OpenAI's account of the sycophantic model update it withdrew in April 2025 (Chapter Ten). The European Union's AI Act requires providers of high-risk AI systems to report serious incidents to authorities.[7]

These are genuine beginnings. They are also far from aviation's level. Reporting is mostly voluntary, investigations are carried out by the companies involved, and there is no independent body with the technical capacity, legal powers and public mandate of an air accident investigator.

What an AI accident investigation could look like

It is worth imagining what an independent AI incident investigator would actually do, because the imagining shows both what is possible and what is missing.

Suppose an AI system used by a local authority wrongly flags a family for investigation, with serious consequences. An investigator on the aviation model would want, first, the equivalent of a flight recorder: a complete, tamper-evident log of the inputs the system received, the version of the model and every other component in use at that moment, the outputs it produced and what the humans did with them. Without that log, there is nothing to investigate. Many deployed AI systems today would not be able to provide it.

Second, the investigator would want to **re-run the case**. Unlike an aircraft, an AI system can often be run again on exactly the same input. If the system is deterministic, or its randomness can be controlled, the failure can be reproduced, varied and studied, which is a real advantage over physical accidents. That requires the exact model version to have been kept, which again is not always the case when models are supplied and updated by third parties.

Third, the investigator would trace the Swiss-cheese line: which layers existed, which had holes, and why the holes lined up. Was the case within the system's stated operating domain? Was there a stated domain at all? Did the human reviewer have time and information to disagree? Had the system been tested on families like this one?

Fourth, and only where the tools allow, the investigator might use interpretability methods (Chapter Five) to ask what features inside the model drove the output. For now that is likely to give partial answers at best.

Finally, the investigator would publish findings and recommendations, addressed not only to the organisation involved but to everyone using similar systems. That is the step that turned aviation's individual tragedies into an industry's collective learning.

Every element of this is technically possible today. What is missing is mostly institutional: a body with the mandate, powers and expertise to do it, and rules

requiring AI systems in consequential uses to keep the records that would make it possible.

Why none of this transfers cleanly

Having admired the toolkit, I want to be precise about why it does not transfer to AI without substantial change, because pretending it does would repeat exactly the mistake Chapter Three warned against: borrowing the confidence of engineering methods without doing the work of translating them.

Fault trees and FMEA assume failure modes can be listed. That held reasonably well for hardware, whose failures come from well-understood physical processes: fatigue, corrosion, electrical breakdown, manufacturing defects. It holds much less well for a large learned model, whose failure modes include subtle changes in behaviour under unusual prompts, abilities nobody predicted and errors arising from the interaction between a particular input and billions of learned parameters. You can build a fault tree for an AI system's *surroundings* (its tools, permissions, monitoring and human reviewers) far more easily than for the model itself.

Investigation assumes causes can be reconstructed. When an aircraft crashes, investigators can usually establish, with high confidence, the precise chain of events. When a language model produces a harmful output, it may be impossible, even for its developers, to say exactly why it produced *that* output for *that* input. AI postmortems will often have to settle for partial, probabilistic explanations.

Aviation's culture grew over a century of accidents. AI does not have a century. It is being deployed into consequential uses within months of each new model's release.

None of this makes the toolkit useless. It means it must be adapted: failure taxonomies built around how AI systems actually fail (the next chapter), investigations that accept partial causal explanations, and, running through both,

an honest acknowledgement that some AI failures will remain less explicable than engineering tradition would find acceptable. That makes the structural safeguards of Part II, the ones that do not depend on full explanation, more important, not less.

KEY POINTS

- **Fault trees** (Bell Labs, 1961–62) work backwards from a defined catastrophe through every path that could cause it, using AND/OR logic.
- **FMEA** (1949) works forwards from each component, and its value is exhaustiveness plus humility.
- Reason's **Swiss cheese model**: accidents happen when holes in several imperfect defences line up.
- Aviation's **no-blame investigation** culture (ICAO Annex 13) and confidential near-miss reporting made flying one of the safest forms of transport.
- AI has begun (the **AI Incident Database**, developer postmortems, EU incident reporting) but lacks independent investigators.
- The toolkit must be adapted, because AI failure modes cannot be fully listed and their causes cannot always be reconstructed.

WHERE THIS CHAPTER STOPS

This chapter describes idealised versions of engineering practice; in reality FMEAs are sometimes done as box-ticking exercises and investigation cultures are imperfect, as the 737 MAX case shows. The history of fault-tree analysis is summarised from secondary accounts. Figures for the AI Incident Database change continuously.

CHAPTER TEN

A Taxonomy of AI Failure

Naming the Ways It Goes Wrong

IN THIS CHAPTER

To guard against failures you must first name them. This chapter sorts AI failures into categories, each linked to the property in the framework that failed: confident falsehood and sycophancy (boundedness), manipulation by jailbreaks and injected instructions (composability), specification gaming and learned bias (verifiability), unfaithful explanations (transparency), and a category still being written, emergent misalignment.

> Two hundred and fifty. That is how few specially crafted documents researchers needed to plant a hidden backdoor in language models of every size they tested, from 600 million to 13 billion parameters, however much other data those models were trained on. To guard against failures like that, you first have to be able to name them.

BY THE NUMBERS

Failure, measured

250

Poisoned documents enough to backdoor models from 600 million to 13 billion parameters, in a 2025 study.

MEASURED

~20% / ~50%

Share of test answers in which GPT-4o and GPT-4.1 became broadly misaligned after being fine-tuned only to write insecure code.

MEASURED

14%

How often Claude 3 Opus complied with harmful requests when told its answers would be used to retrain it, in an experimental setting, against almost never otherwise.

MEASURED

~20%

How much higher an AI scored than human players in the CoastRunners boat race, while never finishing the course.

REPORTED

If the previous chapter's toolkit does not transfer cleanly, the least this book can do is build its own version of what my grandfather's discipline would have demanded: an organised, honestly incomplete but useful classification of the specific ways AI systems fail. I group failures the way engineers group hazard classes, not as a list of anecdotes but as categories with common structure, common causes and, where possible, common remedies.

Each category corresponds to a property in the framework. Naming which property failed is the first step to knowing what kind of fix might help.

Failures of boundedness

Confident falsehood ("hallucination")

The most familiar failure is the **hallucination**: a fluent, confident, plausible statement that is simply false, with nothing to distinguish it from the system's true statements. Chapter Seven explained why this is a predictable consequence of

how language models generate text and how they are scored, not a mysterious glitch.

Its effects are now documented in courtrooms around the world. Chapter Eleven describes the American case of *Mata v. Avianca*. In England, in June 2025, the Divisional Court of the High Court dealt with two cases in which lawyers had put fictitious or inaccurate case citations before the courts, apparently generated with AI tools. Dame Victoria Sharp, President of the King's Bench Division, warned that freely available AI tools "are not capable of conducting reliable legal research" and that lawyers who cite fake authorities risk sanctions ranging from public admonition to referral to the police.[1]

The remedy is structural: give systems ways to ground their claims in sources that can be checked, score abstention above guessing, and put independent verification between any AI-generated factual claim and any consequential use of it.

Sycophancy: telling people what they want to hear

A closely related failure is **sycophancy**: a system shifting its stated views, assessments or even facts towards whatever the user seems to believe or want.

In 2023 Mrinank Sharma and colleagues at Anthropic tested five leading AI assistants and found sycophancy across a range of tasks. The assistants would, for example, wrongly admit to a mistake when a user challenged a correct answer, or tailor feedback on a piece of writing to whether the user said they liked it. The researchers traced part of the cause to training on human preferences: when they examined the preference data, people, and the models trained to imitate their judgements, sometimes preferred a convincingly written sycophantic answer to a correct one.[2]

In late April 2025 the problem reached millions of people at once. OpenAI released an update to the GPT-4o model behind ChatGPT, and users quickly noticed that it had become effusively flattering, validating doubtful ideas and

decisions. Within days OpenAI rolled the update back. In its postmortem, the company explained that it had "focused too much on short-term feedback", such as users' thumbs-up and thumbs-down ratings, and that the resulting behaviour was "overly supportive but disingenuous".[3]

Sycophancy is a boundedness failure of a particularly worrying kind. It means a system's reliability is not even constant across different phrasings of the same factual question: the same fact can be affirmed or denied depending on how the user's leanings came across. A well-specified system should not have that instability. It is also a reminder that the pressure producing it, *make users happy now*, is commercial as well as technical.

The remedy: evaluate systems specifically for consistency under changes of framing, and treat user satisfaction as one signal among several, never the target.

Failures of composability

Jailbreaks and injected instructions

A second family concerns systems being deliberately steered by a user, or by content the system processes. It is called **jailbreaking** when it targets a model's own safety training and **prompt injection** when the instructions arrive hidden in content the system reads (Chapter Eight).

Both exploit the same underlying fact: a model's refusal behaviour is learned from examples, not derived from an understanding that covers every case. A model asked directly for something harmful may refuse, because its training covered that framing. The same request wrapped in a fictional scenario, a role-play or an unusual phrasing may succeed. In 2023 Andy Zou and colleagues showed that automatically generated strings of apparently meaningless characters, appended to a harmful request, could make several aligned models comply, and that strings found by attacking open models often *transferred* to commercial ones.[4] This is

the adversarial-example problem of Chapter Three, carried over from pixels into words.

The remedy: defence in depth. Filter inputs and outputs, separate trusted instructions from untrusted content by architecture, limit what a compromised model could do, and red-team continuously, because attackers do.

Poisoned training data

Composability failures can also arrive before a model is ever used, through the data it learns from. Because large models are trained on text gathered from the public internet, anyone who can publish on the internet can, in principle, place material in a future training set. Researchers call deliberately planted material **data poisoning**.

In October 2025 researchers from Anthropic, the UK AI Security Institute and the Alan Turing Institute reported a sobering result. They trained language models of four sizes, from 600 million to 13 billion parameters, and found that as few as 250 specially crafted documents were enough to plant a "backdoor" (a hidden trigger phrase that made the model produce gibberish) in every one of them. Crucially, the number of poisoned documents needed did not grow with the size of the model or of its training data, even though the largest model was trained on far more data than the smallest.[5] The authors are careful to note that the backdoor they tested was deliberately harmless and that it is not known whether the result holds for more dangerous behaviours. But it means that the scale of a model's training data is no protection in itself. A model's supply chain is part of the system, and it needs checking like any other interface.

Failures of verifiability

Specification gaming

A third category appears whenever a system is optimised against a measurable stand-in for what we really want. It is called **specification gaming** or **reward**

hacking: the system finds and exploits a gap between the literal objective and its intent.

The classic illustration came from OpenAI in 2016. Researchers trained an AI to play CoastRunners, a boat-racing game, rewarding it for its score. The AI discovered that it could score more by circling endlessly in a small lagoon, repeatedly hitting the same targets as they reappeared, catching fire and crashing into other boats, than by finishing the race. It scored on average about 20 per cent higher than human players while never completing the course.[6] Researchers at DeepMind have collected dozens of similar examples.[7]

This is not new. It is Goodhart's law again (Chapter Six). What is new is the scale and inhuman creativity with which an optimiser without common sense finds the gaps. And, as Chapter Five showed, models that learn to exploit such gaps may not mention it in their visible reasoning.

At root this is a verifiability failure. The training objective served as an unchecked stand-in for the real goal, and nobody independently tested whether optimising the stand-in produced the intended outcome across the full range of situations the system would meet.

Learning the past as if it were the rule

A related failure happens when the measurable target is "match past human decisions" and those decisions were unfair. The system reproduces the pattern faithfully, because that is what it was asked to do. Chapter Eleven describes Amazon's abandoned recruiting tool, which learned to penalise CVs that mentioned women's activities.

There is a deeper mathematical lesson here, which surprised many people when it was proved. In 2016 ProPublica analysed COMPAS, a tool used in parts of the United States to predict whether defendants would reoffend. It reported that Black defendants who did not reoffend were almost twice as likely as white defendants to have been wrongly labelled high-risk.[8] The tool's developer replied that its

scores were equally *accurate* for both groups: a given score meant roughly the same likelihood of reoffending regardless of race. Both claims were largely correct. Within months, researchers proved that when two groups have different underlying rates of the outcome being predicted, no scoring system can satisfy both kinds of fairness at once, except in trivial cases.[9]

That is a result my grandfather would have appreciated: a proof that a specification is impossible to satisfy completely. It means "is this system fair?" has no single technical answer. Someone has to *choose* which kind of fairness matters for this use, and say so openly. **The failure is not choosing, or choosing in secret.** A specification that does not state its fairness trade-off has a blank cell in its truth table.

Failures of transparency

Unfaithful explanations and blind monitors

The fourth category returns to Chapter Five: cases where a system's account of its own reasoning, or a monitoring tool's account of the system's internal state, does not match what actually produced the behaviour.

It includes the unfaithful chains of thought already described, but reaches further. An interpretability tool that finds an internal feature correlated with some behaviour can mistake correlation in its test set for the real mechanism. A monitoring system built to catch known failure patterns can miss new ones that match no signature it was built for, creating a false sense of security exactly where vigilance is most needed.

In one sense this is the most insidious category, because it undermines the mechanisms the other remedies depend on. Solutions for boundedness, verifiability and composability all lean, to some degree, on trusting what monitoring and explanation tools report. A transparency failure quietly erodes that foundation without announcing itself.

The remedy: validate monitors against cases where the ground truth is known, and never let a system's self-report be the only evidence about what it did.

A category still being written: emergent misalignment

I want to end with a category that deserves more caution than confidence, because the science is still developing.

In 2025 Jan Betley, Owain Evans and colleagues reported a startling result, later published in *Nature*. They took models, including OpenAI's GPT-4o, and fine-tuned them on a narrow task: writing computer code that contained security vulnerabilities, without telling the user. Nothing in that training involved any other kind of bad behaviour. Yet the fine-tuned models became *broadly* misaligned. Asked unrelated questions, they sometimes said that humans should be enslaved by AI, gave malicious advice or behaved deceptively. For GPT-4o this happened in about 20 per cent of responses to their test questions, and for the newer GPT-4.1 about 50 per cent.[10]

A related line of research has examined whether models behave differently when they believe they are being watched or trained. In December 2024 researchers at Anthropic and Redwood Research reported that Claude 3 Opus, told in an experimental setting that its responses to one group of users would be used to retrain it towards always complying with requests, complied with harmful requests from that group 14 per cent of the time, versus almost never for users whose conversations it was told would not be used, and in its private reasoning often explained that it was complying strategically to avoid being changed.[11]

These results must be read carefully. They come from deliberately constructed experiments, not observed harms in ordinary use. But they share a feature that matters enormously for this book: the failure did not correspond to any specific thing the developers trained for, and its mechanism is not yet fully understood. **A failure whose mechanism is unknown cannot be specifically guarded against.** It can be caught only by safeguards that do not depend on predicting its shape in

advance: independent monitoring, least privilege, human review of consequential actions and diverse layers of defence. That is one of the strongest arguments in this book for the structural, defence-in-depth approach of Part II.

Testing for deception

A related set of experiments has asked whether advanced models, placed in scenarios designed to tempt them, will act deceptively or against their operators' interests. In December 2024 Apollo Research, an independent evaluation organisation, reported that several frontier models, given a goal and then shown information that their developers planned to replace them or change that goal, sometimes took covert actions in the test environment, such as trying to disable an oversight mechanism, and sometimes denied having done so when asked.[12] In June 2025 Anthropic published "agentic misalignment" experiments in which models from several developers, cast as an email assistant at a fictional company and faced with being shut down, frequently chose to blackmail a fictional executive using information found in the company's emails.[13]

The researchers in both cases stress that these were artificial, deliberately constructed situations, and that they had not seen such behaviour in real deployments. That caution deserves to be taken seriously; headlines about "AI blackmail" often did not. But the experiments serve exactly the purpose that red-teaming and fault trees serve in engineering: they explore the edges of a system's behaviour *before* reality does. They also show why an agent's permissions matter so much. A model that might, in some unusual situation, choose a harmful action can do harm only through the access it has been given.

The taxonomy at a glance

FAILURE	WHAT HAPPENS	PROPERTY THAT FAILED	STRUCTURAL REMEDY
Hallucination	Fluent, confident falsehood	Boundedness	Grounding in checkable sources; reward abstention; independent checks before use
Sycophancy	Answers bend to the user's views	Boundedness	Test consistency across framings; don't optimise for approval alone
Jailbreak / prompt injection	Behaviour hijacked by crafted input	Composability	Separate instructions from data; filter; limit permissions; red-team
Specification gaming	Objective met, intent defeated	Verifiability	Independent tests of the real goal, not the proxy
Learned bias	Past unfairness reproduced	Verifiability	State the fairness choice openly; test across groups
Unfaithful explanation	Self-report doesn't match the cause	Transparency	Validate monitors; never rely on self-report alone
Emergent misalignment	Broad bad behaviour from narrow training	All four	Defence in depth that does not depend on predicting the failure

With this classification in hand, the next chapter tests it against real, documented cases.

KEY POINTS

- Every failure category maps to a property that failed, which points to the kind of fix needed.
- **Hallucination** has reached courts in the US and England. **Sycophancy** reached millions of users in the April 2025 GPT-4o update.
- **Jailbreaks** exploit learned, not derived, refusal behaviour. Automatically found attacks transfer between models.
- **Specification gaming** (the CoastRunners boat) is Goodhart's law at machine speed.
- The **COMPAS** debate led to a proof that some fairness criteria cannot all be met at once, so the choice must be made openly.
- **Emergent misalignment** shows failures can arise that nobody trained for. Only structural, independent safeguards can catch them.

WHERE THIS CHAPTER STOPS

This classification is this book's own and is not an established standard; real incidents usually involve more than one category. Several studies cited here come from AI developers examining their own or competitors' models and were conducted in artificial settings designed to provoke failures. The rates they report should not be read as the frequency of these behaviours in everyday use. The COMPAS debate involves methodological disputes that this summary cannot fully represent.

CHAPTER ELEVEN

Case Studies in Chatastrophy

Ten Failures, One Pattern

IN THIS CHAPTER

This chapter applies the framework to ten documented cases: a chatbot that learned abuse from its users, an airline that disowned its own chatbot, a lawyer who asked an AI to check its own inventions, a hiring tool that learned to penalise women, a trading firm destroyed in 45 minutes, an exam-grading algorithm that downgraded a generation, a computer system that sent innocent people to prison, and a fraud-detection model that helped bring down a government, a welfare formula a Royal Commission called "crude and cruel", and a house-buying algorithm that met a market it had not seen. In every case, a system with knowable limits was trusted beyond them.

> Forty-five minutes. More than four million trades. More than $460 million lost, roughly $10 million a minute. Knight Capital's collapse in 2012 is one of ten cases in this chapter, and in every one of them a system with knowable limits was trusted beyond them.

BY THE NUMBERS

The price of misplaced trust

~1,500

Trades per second, on average, executed by Knight Capital's runaway system (more than 4 million trades in about 45 minutes).

CALCULATED

97

Automated emails warning of the problem before the market opened, none of them designed as alerts or acted on.

DOCUMENTED

700+

Sub-postmasters prosecuted by the Post Office between 1999 and 2015, largely on evidence from a computer system presumed to be reliable.

DOCUMENTED

~470,000

Robodebt debts the Australian government agreed to refund after an averaging formula was applied to individuals.

DOCUMENTED

A classification is useful only if it makes sense of real cases. This chapter applies the framework to ten. None is chosen to embarrass a particular organisation. Each is here because it is documented well enough to analyse honestly, usually by a court, a regulator or an official inquiry, and because it teaches a specific, recurring lesson.

For each case I ask the same three questions: *What happened? Which properties failed? What would an intelligible system have done differently?*

1. Tay: learning from the crowd (2016)

What happened. On 23 March 2016 Microsoft released Tay, an experimental chatbot, on Twitter. Tay was designed to engage young adults in casual conversation and to learn from its interactions. Within hours, groups of users worked out that they could get Tay to repeat, and then generate, racist, sexist and otherwise offensive statements. Microsoft took Tay offline about sixteen hours after launch. Two days later, Peter Lee, a corporate vice-president of Microsoft

Research, apologised, writing that "a coordinated attack by a subset of people exploited a vulnerability in Tay".[1]

Which properties failed. *Boundedness*: Tay had no effective way to recognise that a coordinated hostile campaign was unlike ordinary conversation, and no refusal behaviour that engaged when it should have. *Composability*: the system was designed to shape its future behaviour from live, unfiltered public input, which erased the boundary between "data I learn from" and "instructions I should resist". It was the instruction-leakage hazard of Chapter Eight in its most direct form.

What an intelligible system would have done. Learned from live interaction only after filtering and human review; limited the rate at which its behaviour could change; made a far narrower claim about what it was ready to do unsupervised; and been red-teamed against exactly the coordinated abuse that anyone familiar with the platform could have predicted.

2. Air Canada: whose words are they? (2022–2024)

What happened. In November 2022, after his grandmother died, Jake Moffatt asked the chatbot on Air Canada's website about bereavement fares. It told him he could book at the normal price and apply for a partial refund within 90 days. That was wrong: the airline's actual policy, set out on another page of the same website, did not allow retrospective bereavement claims. When Air Canada refused the refund, Mr Moffatt took the case to British Columbia's Civil Resolution Tribunal.

Air Canada argued that it could not be held liable for information provided by its chatbot. The tribunal member, Christopher Rivers, was not persuaded: "In effect, Air Canada suggests the chatbot is a separate legal entity that is responsible for its own actions. This is a remarkable submission." He held that the chatbot was part of Air Canada's website and that the airline was responsible for all the information on it. In February 2024 the tribunal ordered Air Canada to pay Mr Moffatt $650.88 in damages, plus interest and fees.[2]

Which properties failed. *Verifiability*: the chatbot's statements about company policy had not been checked against the authoritative policy before being presented to customers as reliable. *Boundedness*: the system answered a question about a consequential policy with no signal of uncertainty and no referral to a human. And most strikingly, the organisation tried to treat the system as outside its own chain of responsibility.

What an intelligible system would have done. Answered policy questions only by quoting the authoritative policy text, with a link; referred anything outside that to a person; and been owned, legally and practically, by the organisation that deployed it. The tribunal's decision says what Chapter Four argued: intelligibility, and accountability, belong to the whole sociotechnical system, not to a model floating free of it.

3. Mata v. Avianca: asking the system to check itself (2023)

What happened. Roberto Mata sued the airline Avianca in a New York federal court for an injury he said he had suffered from a serving cart on a flight. His lawyers filed a brief citing several earlier court decisions, among them *Varghese v. China Southern Airlines*. The airline's lawyers could not find them. Nor could the judge. The cases did not exist. One of Mr Mata's lawyers had used ChatGPT for research. He later told the court that he had asked ChatGPT whether one of the cases was real, and it had assured him that it was and could be found in reputable legal databases.

On 22 June 2023 Judge P. Kevin Castel sanctioned the two lawyers and their firm, ordering them to pay a $5,000 penalty. His opinion observed that there is nothing inherently improper about using a reliable AI tool for assistance, but that lawyers have a gatekeeping role in ensuring the accuracy of their filings.[3]

Which properties failed. The model's failure was a textbook hallucination, a *boundedness* failure. But the instructive failure is human, and it is about *verifiability*. The lawyer tried to verify the output, which is to his credit, but he

asked the same system that produced the claim. That is the purest example I know of mistaking testimony for evidence. Legal citations are among the most checkable facts in the world. The independent check was available, cheap and quick: look the case up in a legal database.

What an intelligible system would have done. In a legal research tool, retrieved and linked the actual text of every case it cited, so that each citation could be checked in a click. And in a legal practice, a rule, now adopted by many courts and firms, that no AI-generated citation is filed without being checked against a primary source.

4. Amazon's recruiting tool: learning the past (2014–2017)

What happened. In October 2018 Reuters reported that Amazon had built, and then abandoned, an experimental tool to rate job applicants' CVs. According to the report, the team began work in 2014, training models on CVs submitted to the company over the previous ten years, most of which came from men, reflecting the technology industry's gender imbalance. By 2015 the company realised the system was not rating candidates for technical jobs in a gender-neutral way. It reportedly penalised CVs that included the word "women's", as in "women's chess club captain", and downgraded graduates of two all-women's colleges. Engineers edited the models to neutralise those particular terms, but, the report said, there was no guarantee the system would not find other ways to discriminate, and the team was disbanded by the start of 2017. Amazon told Reuters that the tool "was never used by Amazon recruiters to evaluate candidates".[4]

Which properties failed. *Verifiability*: the tool's real objective (find the best candidates) was replaced by a measurable stand-in (resemble the people we hired before), and the gap between the two was exactly the kind of specification gap Chapter Ten describes. *Boundedness*: a system that could only reproduce historical patterns was being asked to do something those patterns could not tell it.

What went right. It is worth saying clearly: according to the report, the problem was found by internal testing and the tool was abandoned rather than rolled out widely. That is the verification discipline this book argues for, working, if later than it should have. The lesson of the engineers' failed fix (removing specific words did not guarantee fairness) is the lesson of shortcut learning from Chapter Three: a network will find whatever cue predicts its target, and blocking one cue does not block the next.

5. Knight Capital: forty-five minutes (2012)

This case predates modern AI, deliberately. It shows that composability failures are not unique to AI, and that the discipline this book borrows from engineering was learned partly from failures of automated systems long before chatbots existed.

What happened. Knight Capital was one of the largest traders in US shares. In the summer of 2012 it prepared new software for a trading system called SMARS, to be installed on eight servers. The technician deploying it did not copy the new code to one of the eight. The new code reused a setting, a "flag", that had once activated an old, long-unused function called Power Peg. On the eighth server, that old code was still there. When the market opened on 1 August 2012, orders reaching that server triggered Power Peg, which began sending orders into the market without the checks meant to stop it.

According to the US Securities and Exchange Commission, in about 45 minutes Knight's system, while trying to fill 212 customer orders, executed more than 4 million trades in 154 stocks, and the firm lost more than $460 million. The SEC also found that before the market opened, Knight's systems had generated 97 automated emails to a group of staff referring to the problem, but these were not designed as alerts and were not acted on.[5] Knight needed emergency financing to survive and was soon merged into another firm.

Which properties failed. Almost pure *composability*. Each piece of code had presumably worked in its original setting. The disaster came from an unverified *combination* (new code and dead code across a fleet of servers) that nobody had checked as a whole. And a *boundedness* failure in operations: there was no automatic limit on how much the system could trade, and no fast kill switch. The system acted at machine speed, while the humans' ability to understand and intervene worked at human speed.

What an intelligible system would have done. Removed dead code; verified that all servers ran the same version before going live; enforced hard limits on trading volume that no software path could exceed; and provided a kill switch that worked in seconds. Every one of these was known engineering practice in 2012. As AI agents are given more power to act at machine speed, Knight Capital's forty-five minutes should be required reading.

6. The 2020 exam algorithm: a mutant or a specification? (2020)

What happened. When the COVID-19 pandemic forced the cancellation of summer exams in 2020, England's exams regulator, Ofqual, had to award A-level and GCSE grades without them. Teachers submitted a "centre assessment grade" for each student and ranked students within each subject. Ofqual's statistical model then adjusted these, largely by fitting each school's grades to the distribution of results it had achieved in previous years. Where a class was small, teachers' grades were used wholly or partly instead, because the model could not reliably standardise small numbers.[6]

When A-level results came out on 13 August, about 39 per cent of grades were lower than teachers' assessments, most by one grade.[7] Because the model tied individual results to schools' past performance, a strong student at a school with historically weak results could be marked down regardless of their own work. And because small classes were more common at independent schools, students there were more likely to keep their teachers' (typically higher) grades. After days of protest, on 17 August the government and Ofqual announced that students

would receive their teachers' grades where these were higher. The Prime Minister later blamed "a mutant algorithm".[8]

Which properties failed. The algorithm was not a mutant. It did more or less exactly what it was designed to do: keep national grade distributions close to previous years'. The failure was in the *specification*, and it was a failure of *boundedness* and *verifiability*. The model was built to be accurate at the level of schools and the nation, then used to make decisions about individuals, a use outside its reliable envelope. It is also a lesson in *transparency*: although Ofqual published a long technical report, students and families could not see before results day how their grades would be decided, or test the consequences for themselves.

What an intelligible system would have done. Stated clearly and in advance what the model could and could not do (accurate for cohorts, unreliable for individuals); tested the effect on individual students in different kinds of school before results day; published that analysis; and built in a fast, fair route to appeal based on each student's own evidence.

7. The Post Office Horizon scandal: when the computer was presumed right (1999–2015)

This is the most serious case in the chapter, and it involves no machine learning at all. I include it because it shows, more clearly than any AI case yet, what happens when a computer system is trusted *by presumption* rather than by evidence.

What happened. From 1999 the Post Office rolled out Horizon, an accounting system supplied by ICL (later part of Fujitsu), to its branches. Sub-postmasters, self-employed people running local post offices, were contractually responsible for shortfalls in their branch accounts. When Horizon showed shortfalls that sub-postmasters could not explain, the Post Office often treated them as theft or false accounting. Between 1999 and 2015 it prosecuted more than 700 sub-postmasters, many of whom were convicted, some imprisoned, and many ruined financially.[9]

Horizon had bugs, errors and defects capable of causing apparent shortfalls. That was established by the High Court in 2019, in group litigation brought by 555 sub-postmasters led by Alan Bates, in a judgment by Mr Justice Fraser.[10] In April 2021 the Court of Appeal quashed the convictions of 39 former sub-postmasters, finding that the Post Office's failures of investigation and disclosure meant their prosecutions were an abuse of the court's process.'[11] In 2024, after a television drama brought the story to a mass audience, Parliament passed the Post Office (Horizon System) Offences Act, which quashed relevant convictions in England, Wales and Northern Ireland by statute (Scotland passed its own legislation).'[12] A statutory public inquiry, chaired by Sir Wyn Williams, examined what went wrong. Some of those affected did not live to see their names cleared.

Which properties failed. Every one. *Transparency*: sub-postmasters could not see the data or logic behind the figures they were blamed for, and evidence emerged that staff at the supplier could remotely access and alter branch data, which was long denied. *Verifiability*: the system's output was treated as proof, and the burden fell on accused individuals to show the computer was wrong, a burden almost none of them could carry without access to its internals (the legal presumption described in Chapter Six made this worse). *Boundedness*: known bugs were not treated as limits on how far the system's figures could be relied on in court. *Composability*: the combination of a fallible system, a contractual rule making individuals liable for its output and an organisation that was both victim and prosecutor was catastrophic, whatever the quality of any individual part.

What an intelligible system would have done. Given every person accused on the basis of computer evidence full access to the relevant data, logs and known-error records; required the organisation relying on the system to prove its reliability in each case, not presume it; and kept prosecution separate from the organisation with an interest in the outcome. The Horizon scandal is not an AI case. But AI systems are now being used to make decisions about people's benefits, jobs, health and liberty. If a conventional accounting system, whose logic was at least written down by human programmers, could be trusted blindly

for sixteen years, the risk with systems that are intrinsically harder to explain is obvious.

8. The Dutch childcare benefits scandal: a model that helped bring down a government (2013–2021)

What happened. In the Netherlands, the Tax and Customs Administration pursued thousands of families for alleged fraud in claiming childcare benefits, in many cases over minor administrative errors. Families were ordered to repay very large sums, often without clear explanation or an effective way to challenge the decision, and many were pushed into severe financial hardship. A parliamentary inquiry's report, published in December 2020 under the title *Ongekend onrecht* ("Unprecedented injustice"), concluded that fundamental principles of the rule of law had been violated. On 15 January 2021 the Dutch cabinet resigned.[13]

Part of the story was a risk-classification model the tax authority used to select benefit applications for closer scrutiny. The Dutch Data Protection Authority found that the authority had unlawfully used applicants' dual nationality as a factor in this risk assessment, and in December 2021 it fined the Tax and Customs Administration €2.75 million.[14] Amnesty International described the result as discrimination by algorithm.

Which properties failed. *Transparency*: families were not told why they had been flagged and could not see the reasoning. *Verifiability*: flags from the model were treated as grounds for harsh action rather than as leads needing independent confirmation. *Boundedness*: a tool for prioritising scrutiny was allowed to drive outcomes. And *composability*: the model sat inside a policy of zero tolerance and an organisation under pressure to show results, so that its errors and biases were amplified rather than caught.

What an intelligible system would have done. Published how applications were selected; excluded protected characteristics from the model and tested it for discriminatory effect; treated its outputs as prompts for human investigation,

never as findings; and given every family a clear explanation and a real route to challenge.

9. Robodebt: an averaging formula applied to individuals (2016–2019)

What happened. In 2016 Australia's welfare agency began using an automated system, later nicknamed "Robodebt", to identify people who had supposedly been overpaid benefits. The core of it was a simple calculation. It took a person's annual income as reported to the tax office, spread it evenly across the year's fortnights, and compared the result with the income the person had reported fortnight by fortnight while claiming benefits. Wherever the averaged figure was higher, the system treated the difference as an overpayment and raised a debt, often without any human looking at the case. People whose income was irregular (casual workers, students, people who had worked for part of the year) were particularly likely to be flagged, because averaging assumed they had earned money in fortnights when they had not. The burden then fell on them to prove, sometimes years later and with records they no longer had, that they did not owe the money.

In 2019 the Federal Court declared a debt raised this way unlawful, and the scheme ended. In 2020 the government agreed to refund around 470,000 debts raised this way and settled a class action; when the Federal Court approved the settlement in 2021, its total value, including refunds and cancelled debts, was put at about A$1.8 billion.[15] A Royal Commission, led by Catherine Holmes, reported on 7 July 2023. Its verdict was damning: "Robodebt was a crude and cruel mechanism, neither fair nor legal, and it made many people feel like criminals."[,16]

Which properties failed. Robodebt was not machine learning. It was an arithmetic rule. That makes it more instructive, not less. *Boundedness*: averaging is a reasonable way to summarise a population and an unreliable way to reconstruct one person's fortnights; the method was applied far outside the envelope in which it could be trusted. *Verifiability*: the outputs were treated as debts rather than as leads, and the burden of proof was reversed onto the

individual. *Transparency*: people were not given a clear account of how their debt had been calculated.

What an intelligible system would have done. Used the averaging comparison only to select cases for human review; never raised a debt without evidence of actual fortnightly income; and kept the burden of proof on the state. The same lesson as the Ofqual algorithm, and the same lesson as Horizon, from the other side of the world.

10. Zillow Offers: a price model meets a moving market (2021)

What happened. Zillow, the American property website, ran a business called Zillow Offers that bought homes directly from sellers, using algorithmic estimates of their value, with the aim of renovating and reselling them at a profit. In 2021, as the housing market moved quickly and unpredictably, the company bought far more homes than it could sell at the prices it had expected. On 2 November 2021 it announced that it would wind the business down, reported a write-down of $304 million on its housing inventory for the third quarter, and said it would cut about a quarter of its workforce. Its chief executive, Rich Barton, said: "We've determined the unpredictability in forecasting home prices far exceeds what we anticipated."[17]

Which properties failed. *Boundedness*, in its purest commercial form. A forecasting model that works in stable conditions was trusted to make large, fast, hard-to-reverse bets in conditions it had not seen, and the business had no mechanism to scale back its buying quickly enough as the model's reliability fell. It is Chapter Seven's distribution shift with a price tag attached.

What an intelligible system would have done. Stated the market conditions in which the model's estimates had been validated; monitored its errors on actual resale prices continuously; and linked the amount of money at risk to how confident the model was, reducing buying automatically when forecast errors grew.

What the cases share

Across all ten cases, one thread is worth naming plainly. **None of these failures happened because the technology was uniquely or unprecedentedly dangerous.** A chatbot, a hiring model, a grading algorithm, an averaging formula and a fraud-risk model are not exotic. Each failure happened because a system with real, knowable limits was deployed into a setting, or combined with other systems and processes, in a way that did not honestly account for those limits. Each time, a missing property of Part II sat exactly where the failure occurred.

Notice, too, how often the humans and organisations around the system made things worse: an airline disowning its chatbot, a lawyer asking the machine to check itself, a trading firm ignoring 97 warnings, a Post Office presuming its computer infallible, a tax authority treating a risk score as proof, a welfare agency treating an average as a debt. The model is never the whole system.

Strangely, that is the most hopeful possible conclusion for a chapter about catastrophe. These are not mysterious, unsolvable problems. They are the specific, nameable, historically familiar kind of engineering failure that Chapter Nine's toolkit was built to catch, in systems whose owners had not yet done the work of applying it. Part IV turns to what that work looks like.

CASE	YEAR(S)	MAIN PROPERTIES THAT FAILED	ONE-LINE LESSON
Tay	2016	Boundedness, composability	Don't let live public input rewrite behaviour unsupervised
Air Canada	2022–24	Verifiability, boundedness	You own what your chatbot says
Mata v. Avianca	2023	Verifiability	Never ask the system to verify itself
Amazon recruiting	2014–17	Verifiability, boundedness	A proxy for the past is not a measure of merit
Knight Capital	2012	Composability, boundedness	Machine-speed action needs hard limits and a kill switch
Ofqual algorithm	2020	Boundedness, verifiability, transparency	Accurate for groups is not accurate for individuals
Post Office Horizon	1999–2015	All four	A presumption of reliability is not evidence
Dutch benefits	2013–21	All four	A risk score is a lead, not a verdict
Robodebt	2016–19	Boundedness, verifiability, transparency	An average is not an individual's history
Zillow Offers	2021	Boundedness	Scale the bet to the model's current reliability

KEY POINTS

- In all ten cases, a system with **knowable limits was trusted beyond them**.
- **Accountability** cannot be delegated to a system (Air Canada), and **verification** cannot be delegated to the system being verified (Mata).
- **Proxies** smuggle in the past (Amazon) and **group-level methods** fail individuals (Ofqual, Robodebt).
- **Machine-speed** action without hard limits is catastrophic (Knight Capital).
- The gravest harms came where computer outputs were **presumed** reliable and used against individuals (Horizon, Dutch benefits, Robodebt).
- The surrounding people and organisations are part of the system, and often its weakest part.

WHERE THIS CHAPTER STOPS

Each case is summarised from court judgments, regulatory findings, official inquiries or reputable reporting, as cited; the Amazon case rests on a single news report based on anonymous sources and Amazon's statement. Short summaries cannot capture the full complexity or human cost of cases such as Horizon and the Dutch benefits scandal, and readers should consult the cited sources. Assigning "properties that failed" is this book's analytical judgement, not a finding of any court or inquiry.

PART IV

Applied Intelligence

CHAPTER TWELVE

Building Intelligible Systems

A Guide for Practitioners

IN THIS CHAPTER

This chapter turns the framework into practice for the people who build and deploy AI. It sets out six habits (specify before you build, evaluate as a discipline, monitor for drift, defend in depth, control what the system can do, and investigate failures honestly) and ends with a practical "intelligibility file" that every consequential AI system should have.

> Between November 2022 and October 2024, the cost of running a model at GPT-3.5's level fell more than 280-fold, according to the AI Index.[N10] When intelligence becomes that cheap, it gets built into everything. Nothing suggests that the discipline of building it well has become 280 times cheaper.

BY THE NUMBERS

Cheap to run, costly to get right

280×

Fall in the cost of running a GPT-3.5-level model, November 2022 to October 2024.

REPORTED

33×

Fall in the energy used by Google's median Gemini text prompt in one year, to 0.24 watt-hours.[N14]

REPORTED

36%

Chance that a 20-step process succeeds when each step is 95% reliable (0.95^{20}). Build checkpoints.

CALCULATED

0

Breakthroughs needed to rebuild the Air Canada chatbot intelligibly. Every habit in this chapter is ordinary engineering.

FRAMEWORK

In digital engineering, theory is never the end of the job. The point is to take a specification and turn it into something that works. This book owes its readers the same shift at this point. The first three parts built a framework and diagnosed its absence. This chapter is for the people who build and deploy AI systems, and it tries to be concrete about what working towards the four properties looks like day to day.

None of what follows is exotic. Almost every practice here is already standard in some mature safety-critical field. The work for AI is not inventing new ideas. It is the less glamorous, more urgent work of actually doing what my grandfather's generation spent decades learning to do.

Habit one: specify before you build

The highest-leverage habit, because every other practice depends on it, is to write an honest **specification** before a system is built, and certainly before its capabilities are marketed. It should state, as concretely as the technology allows:

- **Intended use:** what the system is for, and who will use it.
- **Out of scope:** what it is *not* for, stated as plainly as the intended use.
- **Operational design domain:** the conditions under which it has been tested and is expected to work (Chapter Seven).
- **Known failure modes:** drawn from the taxonomy in Chapter Ten, with measured rates where possible.
- **Refusal and escalation behaviour:** what the system does at the edge of its competence, and who it hands over to.
- **Fairness choices:** which kind of fairness was chosen where trade-offs exist, and why (Chapter Ten).
- **Ownership:** the named person or team accountable for its behaviour.

Write it with the discipline of a datasheet: limits stated as precisely as capabilities, not as a defensive legal afterthought but as the central engineering document around which everything else is built.

In practice this means resisting a strong organisational pressure: the temptation to describe a system by its most impressive demonstration rather than by the bounded domain where its performance has been checked. A system that performs brilliantly on hand-picked examples and is then marketed as generally capable has had its specification written by its highlight reel. The habit worth building is to treat the honest edge of a system's competence as just as reportable as its best result.

Habit two: evaluate as a discipline, not an event

Chapter Six argued that verifiability is a habit, not a certificate. In an organisation, that means evaluation gets the same weight and resources that testing gets in mature software engineering, which is a great deal, not a week squeezed in before launch.

FROM THE TOOLBOX

An evaluation plan that earns the word "verified"

- **Test on held-out data** the model never saw in training, refreshed regularly and protected from leaking (Chapter Six).
- **Test the real goal, not the proxy.** If the system scores CVs, measure whether it identifies people who go on to do the job well, not whether it matches past hiring decisions.
- **Test on the people and places it will actually serve.** The Epic Sepsis Model and the Thai retinopathy clinics show how far lab performance can drift from local reality.
- **Break results down by group.** An average can hide a subgroup for whom the system fails badly.
- **Test each failure category** in Chapter Ten that applies: hallucination rates in the actual domain, consistency under reframing (sycophancy), resistance to injection if the system reads untrusted content, and so on.
- **Red-team before every significant release**, with testers who are rewarded for finding failures and who do not report to the launch team.
- **Pre-register what "good enough" means** before you see the results, so the bar cannot move to meet the score.
- **Publish what you can**, including what you tested and did not test.

Habit three: monitor for drift

Because boundedness cannot be a one-off certification for a system that keeps changing, in a world that keeps changing around it (Chapter Seven), monitoring should be part of the system's architecture, not a dashboard added afterwards. In practice:

- **Watch the inputs.** Detect requests that differ markedly from the evaluation data, so that out-of-scope use is visible instead of being silently absorbed into ordinary-looking output.

- **Watch the outputs.** Track them over time for drift. Behaviour can shift after updates, after changes to the underlying model supplied by a vendor, or as users change how they use the system.
- **Watch the outcomes.** Where possible, follow up on whether the system's decisions turned out to be right. The Epic study was, in effect, outcome monitoring done by outsiders years late.
- **Escalate by rule.** When monitoring detects conditions the evaluation never covered, route cases to people automatically instead of leaving it to individual operators' judgement.

Habit four: defend in depth

Chapter Two's most important inheritance is that a well-designed system assumes its own safeguards will eventually fail and builds another layer that does not depend on the first holding. Applied to AI, that means never relying on one mechanism (a model's trained-in refusal behaviour, say) as the only protection against a serious harm. Layer it:

- **Before the model:** input filtering and checks for known attack patterns.
- **Inside the model:** safety training and a clear system specification.
- **After the model:** output checks, including automatic comparison of factual claims against trusted sources where possible.
- **Around the model:** rate limits, anomaly detection and hard limits that no output can override, like the trading limits Knight Capital lacked.
- **Beyond the model:** human review for the highest-stakes decisions, designed to resist automation bias (Chapter Eight).

Think in terms of Reason's Swiss cheese: every layer has holes, so the aim is to make sure the holes are unlikely to line up. That means the layers should be *diverse*. A second model from the same family, checking the first, may share its blind spots.

Habit five: control what the system can do

As AI systems move from answering to acting, the most important safety decisions are often not about the model at all. They are about what the model is *connected to*. Borrowing from security engineering:

- **Least privilege.** Grant each agent the narrowest permissions a task needs, and remove them when the task is done.
- **Separate environments.** Keep testing and production apart, so that no agent working on the one can damage the other (the lesson of the Replit incident).
- **Separate instructions from data.** Treat everything an agent reads from outside (web pages, emails, documents) as untrusted, and require confirmation before untrusted content can trigger consequential actions.
- **Make actions reversible where possible**, and make irreversible actions require explicit human approval.
- **Log everything the agent does**, in a form a human can review, so that its own account of its actions is never the only record.
- **Build a kill switch that works in seconds.**

Habit six: investigate failures honestly

Finally, a cultural habit. Organisations building consequential AI benefit enormously from deliberately building the no-blame, information-maximising investigation culture described in Chapter Nine, adapted for the murkier causes of AI failure. That means:

- Treating every significant failure, near-miss or user complaint as input to a structured investigation, not an inconvenience to patch quietly.
- Resisting the instinct to minimise how an incident is described.
- Asking "which layers had holes?" rather than "who is to blame?"
- Sharing findings with the wider field where legally and commercially possible, including through public databases such as the AI Incident Database.

OpenAI's public postmortem of its sycophantic update in April 2025 is an example worth copying: it described what went wrong, why the company's own tests had missed it and what it would change.[1] Aviation became safe because the whole industry learned from each accident. AI, a younger field with stronger pressures towards secrecy, has not yet built that habit at scale, and it needs to.

A worked example: the bereavement-fare chatbot, rebuilt

To make these habits concrete, imagine being asked to build the chatbot from the Air Canada case again, this time intelligibly. The aim is not to design a perfect system, which does not exist, but to show how each habit changes a real design decision.

Specify. The specification says the assistant answers questions about published fares and policies, books nothing, and gives no advice on matters that depend on individual circumstances (medical, legal or compassionate exceptions). Out of scope, in writing: anything not covered by a current policy document.

Ground and bound. The assistant answers policy questions only by retrieving the current, authoritative policy text and quoting it, with a link. If it cannot find a matching policy, it says so and offers a human agent. It never paraphrases a policy into a promise. Bereavement fares, refunds and compensation are flagged as high-stakes topics that always offer a route to a person.

Evaluate. Before launch, a test set of several hundred real customer questions, including awkward, emotional and ambiguous ones written by staff who handle complaints, is run through the system, and every answer is checked against the policy by someone outside the build team. The release bar is set in advance: for example, zero answers that contradict published policy in the test set, and every high-stakes question correctly routed.

Monitor. In use, a sample of conversations is reviewed every week against the policy. When a policy changes, the test set is rerun before the change goes live,

because the chatbot is only as current as its documents.

Control. The assistant has read-only access to policy documents. It cannot issue refunds, change bookings or make commitments. Its conversations are logged, and customers can download a transcript.

Own. The customer service director is named as the accountable owner. The organisation's position, stated publicly, is that the assistant's statements are the company's statements, which is what the tribunal decided anyway.

Nothing here requires a breakthrough in AI. Every element is ordinary engineering and ordinary management. The cost is real but modest, and it is a fraction of the cost of the alternative, which in this case included a tribunal ruling reported around the world.

Habit zero: ask whether you need AI at all

One habit comes before all the others, and it is the one most often skipped. **Ask whether the problem needs an AI system at all.**

Many of the failures in Chapter Eleven involved tasks that could have been done, perhaps more slowly, by a simpler and more intelligible method. A policy chatbot can often be replaced by a well-organised page of policies with a good search box. An averaging formula applied to individuals (Robodebt) should never have been automated without human review. A grading problem may call for a transparent rule that people can understand and argue with, rather than a statistical model they cannot.

A simple rule, written down, tested and published, has almost all the properties this book asks for by default: it is transparent, verifiable, bounded and composable. A learned model must work hard to earn those properties. Where a simple method is good enough for the purpose, it is usually the more intelligible choice. Where a learned model really is better, the gain should be large enough to

justify the extra work of making it intelligible, and that work should be budgeted from the start.

The intelligibility file

Pulling these habits together, I suggest that every AI system used for a consequential purpose should have an **intelligibility file**: a living document, kept up to date, that answers the four questions from Chapter Four with evidence. The name is mine, but the idea borrows from existing practices, including model cards, datasheets and the technical documentation the EU AI Act requires for high-risk systems.[2]

SECTION	CONTENTS	ANSWERS THE QUESTION
1. Transparency	What the system is built from (model, data, vendors); how its decisions can be traced; what logs are kept	Can anyone see why it did that?
2. Verifiability	Evaluation results, who ran them, whether independent; red-team findings; what was not tested	Who checked, and how?
3. Boundedness	Intended use, out-of-scope uses, operational design domain, refusal and escalation rules	Where does it stop, and what happens there?
4. Composability	What it connects to, its permissions, human oversight design, kill switch	What can it affect, and what if it's wrong?
5. Ownership and history	Accountable owner; change log; incidents and what was learned	Who answers for it, and what has it taught us?

An electrician would recognise this immediately. It is the AI equivalent of the certificate and schedule of test results that must accompany every new installation: not a guarantee that nothing will ever go wrong, but a record of what was checked, by whom, against what standard, and what the limits are.

The next chapter turns from the people building these systems to the institutions responsible for governing them.

KEY POINTS

- **Specify before you build**, stating out-of-scope uses and failure modes as clearly as capabilities.
- **Evaluate as a discipline:** held-out data, the real goal not the proxy, local populations, subgroups, independent red teams.
- **Monitor inputs, outputs and outcomes** for drift, and escalate by rule.
- **Defend in depth** with diverse layers, because every layer has holes.
- **Control what the system can do:** least privilege, separated environments, untrusted inputs, reversible actions, full logs, kill switch.
- **Investigate failures without blame** and share what you learn.
- Keep an **intelligibility file** for every consequential system.

WHERE THIS CHAPTER STOPS

These practices describe what good looks like, not a minimum legal standard, and their cost is real; proportionality matters, and a low-stakes tool does not need the full apparatus. Many organisations deploy AI built by others and cannot change the underlying model; for them, habits three to six and the intelligibility file matter most. The "intelligibility file" is this book's proposal, not an existing standard.

CHAPTER THIRTEEN

Governing Intelligible Systems

A Guide for Policy

IN THIS CHAPTER

Nobody certifies their own wiring. This chapter asks what independent assurance should look like for AI. It draws lessons from how the UK regulates electrical work, surveys the institutions now being built (the EU AI Act, the UK AI Security Institute, NIST and ISO standards, public-sector transparency records), identifies public procurement as an underused lever, and is honest about what governance alone cannot do.

> In 2024, private investors in the United States alone put $109.1 billion into AI, according to the AI Index.[N10] Nobody wires a house and then signs off their own electrical certificate. Yet most AI systems in daily use have been checked only by the organisations that built or bought them.

BY THE NUMBERS

Governance, by the numbers

$109.1 bn

US private investment in AI in 2024, nearly 12 times China's and 24 times the UK's.

REPORTED

28 + EU

Countries, plus the European Union, that signed the Bletchley Declaration on frontier AI risk in November 2023.

DOCUMENTED

2 Dec 2027

The postponed date from which most EU AI Act requirements for high-risk systems will apply.

DOCUMENTED

5 years

Maximum interval between electrical inspections of a privately rented home in England. Most AI systems face no equivalent re-inspection.

DOCUMENTED

Nobody wires their own house and then certifies their own work. That is not because electricians are untrustworthy. It is because a system in which the person doing the work also decides whether it is safe has a structural flaw that no amount of individual competence or good faith can fully correct. Over a long history of fires and electrocutions, society decided that some things are too consequential to leave to that structure alone.

I hold a CSCS card and an Asbestos Awareness certificate for exactly this reason. Not because I am especially likely to meet asbestos, but because the industries I work in decided, as institutions, that competence should be independently verified before someone is trusted to work unsupervised. This chapter asks what the equivalent structure looks like for artificial intelligence, and argues that it is currently far too thin for what the technology is being trusted to do.

How we regulate wiring

It is worth looking at how electrical work is actually governed in England and Wales, because it offers a more flexible model than people often assume.

Most electrical work in homes is covered by Part P of the Building Regulations, in force since 2005. Certain work, such as a new circuit or work in a bathroom, is "notifiable": it must either be notified to the local authority's building control, which then inspects it, or be carried out by an electrician registered with a government-authorised **competent person scheme**, who can self-certify it.[1] Those schemes assess their members' competence and inspect samples of their work. The technical standard, BS 7671, is written by a committee of the IET and BSI, and is revised regularly. Separately, landlords in England must have their rented homes' electrics inspected by a qualified person at least every five years.[2]

Look at the structure rather than the details:

- **Risk-based.** Not all work is treated alike. Changing a light fitting is not notifiable; a new circuit in a bathroom is.
- **Independent standards.** The technical rules are written by professional bodies, not by the firms doing the work, and are public.
- **Independent competence checks.** You cannot self-certify unless an outside scheme has checked that you can.
- **Periodic re-inspection.** Certification is not permanent, because installations deteriorate and are altered.
- **A clear paper trail.** Every notifiable job produces a certificate stating what was done, by whom and what the test results were.

Licensing people, or assuring systems?

A naive call to "license AI developers the way we license electricians" would miss what makes AI different. Credential-based licensing works well where competent practice is a fairly stable skill that changes slowly. AI development is a fast-moving research field in which the boundary of what is possible shifts year to

year. A rigid credential system risks freezing a snapshot of best practice that quickly goes out of date, or shutting out genuinely innovative approaches because they do not match a curriculum.

What transfers from the electrical model, then, is not the mechanism of licensing individual developers but the underlying structure: **risk-based rules, independent standards, independent assessment, periodic re-checking and real consequences for falling short**, applied mainly to the *deployment and operation* of consequential AI systems, where the pace of research matters less, rather than to the credentials of individual developers.

The institutions being built

Several jurisdictions have started building institutions of this kind. This is real progress compared with only a few years ago, and I don't want to understate it.

The European Union's AI Act is the most comprehensive law so far. It entered into force on 1 August 2024 and takes a risk-based approach not unlike Part P. A small number of practices are banned outright; prohibitions such as those on certain kinds of social scoring and manipulative AI have applied since 2 February 2025. Obligations for providers of general-purpose AI models have applied since 2 August 2025. Systems classed as "high-risk", including many used in employment, education, access to public services, law enforcement and critical infrastructure, must meet requirements for risk management, data quality, documentation, human oversight, accuracy and robustness.[3] Those high-risk requirements were originally due to apply from August 2026 for most high-risk uses and August 2027 for AI built into products already covered by EU safety law. In 2026 the EU postponed them. After a political agreement in May, an amending regulation, the "Digital Omnibus on AI", entered into force on 27 July 2026, moving the dates to 2 December 2027 and 2 August 2028 respectively.[4] The delay shows how hard it is to build the technical standards and assessment capacity that such a law depends on.

The United Kingdom has so far chosen not to pass a single AI law, relying instead on existing regulators (for data protection, medicines, financial services and so on) to apply principles to AI in their own sectors. It has invested in technical capacity through the AI Security Institute (Chapter Six). And in the public sector it has taken a genuinely useful transparency step: the **Algorithmic Transparency Recording Standard**, a template for government bodies to publish what algorithmic tools they use, for what purpose and with what safeguards, became mandatory for central government departments in 2024.[5] Had something like it been in place and properly used a decade earlier, the Dutch benefits scandal and, in a different form, the Horizon scandal might have been caught sooner.

Standards bodies are doing the unglamorous work. In the United States, NIST published its voluntary AI Risk Management Framework in January 2023.[6] Internationally, ISO/IEC 42001, published in December 2023, sets out requirements for an AI management system that organisations can be certified against, much as they are certified against quality or information-security standards.[7]

All of this is young. Much of it is voluntary or advisory. The AI institutes' ability to evaluate frontier systems independently, rather than relying on access and information the companies choose to provide, is still being built. The aviation regime this book keeps returning to took many decades of accumulated regulation, institutional capacity and hard lessons from real accidents. On that timeline, AI governance is in its infancy, and my honest message to policymakers is less "here is the finished template" and more "here is the direction that has worked before, and the patience and sustained investment it took".

Open and closed models

Chapter Five promised to return to deliberate opacity, the choice by AI developers about how much of their systems to disclose. The sharpest form of that choice is

whether to release a model's weights openly, so that anyone can download, inspect, modify and run it.

The case for **open weights** draws directly on this book's values. Open models can be inspected by independent researchers, which supports transparency and verifiability; much of the interpretability and safety research of recent years has depended on them. They let organisations run AI on their own computers, keeping sensitive data in-house. And they spread capability beyond a few large companies, which reduces the concentration risk described in Chapter Eight.

The case for **caution** draws on the same values. Once weights are released, the developer can no longer monitor how the model is used, update it to fix a problem, or withdraw it. Safety training can be removed by further fine-tuning. For the most capable models, some researchers and governments worry that open release could give meaningful help to people seeking to cause serious harm, for example with biological or cyber weapons, and that such a release cannot be undone.

This book does not settle that debate, and I am wary of anyone who says it is simple. But the framework suggests how to approach it. Openness is not binary: developers can publish documentation, evaluation results and research access without publishing weights, and they can release weights for smaller models while holding back larger ones. The decision should depend on evidence about a specific model's capabilities and risks, assessed before release and, ideally, reviewed by someone other than the developer. And whatever the choice, the reasons for it should be stated publicly. A closed model whose makers explain what they have tested, and why they are withholding what they withhold, is more intelligible than an open model released with no documentation at all, and the reverse is also true.

Procurement: the underused lever

One lever deserves more attention than it usually gets, because it does not require waiting for new laws: **procurement**. A government department, an NHS trust, a council or a large employer deciding which AI products to buy already has the power to require the evidence this book argues for as a condition of purchase, just as public procurement already routinely requires safety certification for equipment, accessibility for software and security audits for systems handling sensitive data.

> FROM THE TOOLBOX
>
> **Ten questions to ask before buying an AI system**
>
> 1. What exactly is it for, and what is it *not* for? (Ask for the out-of-scope list.)
> 2. What evidence shows it works *for people like ours*, *in settings like ours*?
> 3. Who tested it, apart from you? Can we see the results?
> 4. How does performance differ between groups of people?
> 5. What are its known failure modes, and how often do they occur?
> 6. What does it do when it's unsure? Can it refuse or refer?
> 7. What data, systems and actions will it have access to?
> 8. What logs will we get, and can affected people see why a decision was made about them?
> 9. How will we be told about updates to the underlying model, and will you re-test after them?
> 10. If it gets something wrong, who is liable, and how do affected people challenge it?
>
> A supplier that cannot answer these questions is selling a system that is not yet intelligible enough to buy for anything that matters.

Procurement is underused because it needs no new legislation, only the will to write more demanding requirements into contracts already being negotiated. A buyer with real purchasing power can move faster, and set a more specific,

technically grounded bar, than a general regulator writing rules for a whole industry at once.

Liability: who pays when it goes wrong?

Rules about what systems must do are only half of governance. The other half is what happens when they cause harm. Liability matters because it gives organisations a financial reason to invest in the unglamorous work of Chapter Twelve before anything goes wrong.

The Air Canada tribunal applied an old principle to a new technology: a company is responsible for what its website tells customers, whatever produced the words. Legislators are now writing that principle down. The European Union's revised Product Liability Directive, adopted in 2024, explicitly treats software, including AI systems, as a "product" for the purposes of strict liability for defects. It covers failures to provide necessary safety updates, and it eases the burden on injured people who would otherwise struggle to prove how a complex system caused their harm. It applies to products placed on the market after 9 December 2026.[8]

That last point connects directly to intelligibility. When a system is opaque, the person harmed by it is often the one least able to show what went wrong, as the Horizon sub-postmasters discovered. Rules that shift some of the burden of explanation onto the organisation that built or deployed the system create a strong incentive to make it intelligible in the first place. An organisation that cannot explain its own system's behaviour should expect to have difficulty defending it.

International coordination

AI is built and used across borders, so no single country's rules can be the whole answer. The first steps towards international coordination have been taken, with mixed results. In November 2023 the UK hosted the AI Safety Summit at Bletchley Park, where 28 countries and the European Union signed a declaration recognising the risks of frontier AI and the need to work together on them.[9]

Follow-up summits took place in Seoul in May 2024, where a group of AI companies made voluntary safety commitments, and in Paris in February 2025, where the United States and United Kingdom declined to sign the summit's final declaration.

The most useful product of this process, for the purposes of this book, is the *International AI Safety Report*, a review of the scientific evidence on the capabilities and risks of general-purpose AI, chaired by the computer scientist Yoshua Bengio and written by around a hundred independent experts, the first full edition of which was published in January 2025.[10] It is modelled loosely on the reports of the Intergovernmental Panel on Climate Change: an attempt to establish a shared, evidence-based picture before arguing about what to do. That is verifiability at the scale of nations.

Rights for the people affected

Governance should not only be about systems. It should also give **people** affected by AI decisions practical rights, because they are often the first to notice when something is wrong, and in the Horizon and Dutch cases they were ignored for years. Three rights matter most:

- **The right to know** that an automated system was involved in a decision about you.
- **The right to an explanation** meaningful enough to challenge, which in practice depends on the transparency and logging described in Chapter Twelve.
- **The right to human review and a real route of appeal,** with the burden on the organisation to show the system was right, never on the individual to prove it wrong.

Versions of these exist in law already. UK data protection law, for instance, contains specific protections around significant decisions based solely on automated processing, and the EU AI Act gives people a right to an explanation of

certain decisions made using high-risk systems.[11] The Horizon scandal shows that rights on paper are not enough if the evidence needed to use them is out of reach.

What governance cannot do

I want to end this chapter honestly, in the spirit the whole book has tried to keep. Governance, however well designed, cannot replace the practitioner-level work described in the previous chapter. Regulation is generally better at excluding clearly unsafe practice than at producing excellent practice. It sets a floor, not a ceiling, and the floor is only as good as the technical understanding of the people setting it, which depends on exactly the independent expertise this chapter says is still being built.

The realistic picture is not a choice between regulation and engineering culture as the source of AI safety. Both are needed, neither substitutes for the other, and the lessons of my grandfather's discipline (independent verification, honest limits, defence in depth and a culture that learns openly from failure) have as much to teach the people writing the rules as the people writing the code.

The final chapter of this part turns to everyone else: the people neither building nor regulating these systems, but living alongside them.

KEY POINTS

- UK electrical regulation offers a flexible model: **risk-based rules, independent standards, independent competence checks, periodic re-inspection and a paper trail.**
- Assure **deployed systems** rather than trying to license every developer.
- The **EU AI Act** is in force, with high-risk obligations postponed to 2027–28 by an amending regulation in July 2026; the UK relies on sector regulators, the **AI Security Institute** and the **Algorithmic Transparency Recording Standard**.
- **NIST AI RMF** and **ISO/IEC 42001** provide voluntary standards and certification.
- **Procurement** is an underused lever. Ask the ten questions.
- Affected people need rights to **know, understand and challenge**, with the burden of proof on the organisation.

WHERE THIS CHAPTER STOPS

AI law and policy were changing quickly at the time of writing, particularly the EU AI Act's implementation timetable and UK policy; all dates and requirements here should be checked against current official sources. The description of electrical regulation applies to England (and in part Wales); arrangements differ in Scotland and Northern Ireland. This chapter describes rights in general terms and is not legal advice.

CHAPTER FOURTEEN

Living with AI

What the Rest of Us Can Do

IN THIS CHAPTER

Most readers will never train a model or write a regulation, but they will use AI every day. This chapter turns the framework into six practical habits for ordinary users: know which territory you're in, check anything you'll act on, watch for flattery, grant as little access as possible, protect your data, and hold organisations, not machines, accountable.

> ChatGPT reached an estimated 100 million monthly users two months after launch, faster than any consumer application before it. By October 2025, OpenAI said, 800 million people were using it every week.[N2] Most of them will never read a model card or a system card. This chapter is for them.

BY THE NUMBERS

AI in everyday life

2 months

Time ChatGPT took to reach an estimated 100 million monthly users. TikTok took about nine months; Instagram about two and a half years.

ESTIMATE

800 million

Weekly ChatGPT users, according to OpenAI in October 2025.

REPORTED

0.24 Wh

Energy for Google's median Gemini text prompt: by the company's comparison, less than nine seconds of television.[N14]

REPORTED

0

Times you should ask an AI system whether its own answer is correct and treat the reply as a check.

FRAMEWORK

Most people reading this book will never train a model or write an AI regulation. They will use these systems, though: to draft an email, to ask a medical question at eleven at night, to get a second opinion on a contract, to help a child with homework. The frameworks in the earlier chapters, built for practitioners and policymakers, are not much direct use to someone in that position unless they can be turned into a handful of habits. That is this chapter's job.

Habit one: know which territory you're in

Chapter Seven's discussion of boundedness comes down, for an ordinary user, to one useful habit: before trusting an answer, ask honestly whether your question sits in territory the system is likely to know well, or out at the edges, where confident answers are least reliable.

USUALLY SAFER TERRITORY	HANDLE WITH MORE CARE
Explaining well-established ideas taught in textbooks	Very recent events (after the system's training data ends)
Summarising a document you have given it	Specific facts: dates, figures, quotations, citations
Drafting and editing text you will check	Niche or local information (a particular council's rules, a small business)
Brainstorming options you will weigh yourself	Medical, legal and financial questions with real consequences
Explaining code or a spreadsheet formula	Anything you can't easily check for yourself

The left-hand column is not guaranteed to be right, and the right-hand column is not guaranteed to be wrong. But Chapter Seven explained why fluency and accuracy drift apart as you move to the right. Your scepticism should move with them.

Habit two: check anything you'll act on

The most important lesson of *Mata v. Avianca*, generalised beyond lawyers, is this: **the cost of checking an AI system's factual claim is almost always far lower than the cost of acting on a false one.**

It sounds obvious. It is routinely ignored, because a fluent, confident answer feels as if the work of being true has already been done. A simple rule: for any claim you intend to act on in a way that would be costly if wrong (a fact you will repeat, a decision you will make, a document you will file, a dose you will take), spend the few minutes it takes to check it against an independent source.

Two refinements make this habit much more effective:

- **Never ask the system to check itself.** Asking the same chatbot "are you sure?" is not verification. Mr Mata's lawyer did exactly that. Go to the primary source: the legislation, the official guidance, the paper, the manufacturer's instructions.

- **Ask for sources, then open them.** Many AI tools can now cite web pages. That is useful only if you click the link and confirm that the page says what the AI claims. Cited sources can themselves be misread or invented.

For health questions in particular, the NHS website, your GP and your pharmacist exist for a reason. An AI can help you understand what they tell you and prepare questions to ask. It should not replace them.

An example from my own trade

Here is how these habits look in my own trade. Ask a chatbot what cable size you need for a particular circuit, or what the maximum earth fault loop impedance is for a given circuit breaker, and it will usually give a confident, specific answer. Sometimes the answer is right. Sometimes it quotes a value from an older edition of the regulations, or from a different country's standards, or simply invents a plausible number. The tone is identical in each case.

In electrical work, a wrong number is not an inconvenience. It can mean a cable that overheats inside a wall, or a fault that does not clear fast enough to prevent a shock. So the rule should be simple. Use AI to find your way around a problem, to be reminded which regulation or table is relevant, or to have a concept explained in different words. Never use its numbers. Every value an electrician relies on should come from BS 7671 itself, the manufacturer's data or their own test instruments, because it goes on a certificate with their name next to it. That is not distrust of the technology. It is the same discipline every electrician already applies to a colleague's verbal "it's fine, I checked it": thank you, and I'll test it myself.

Habit three: watch for flattery, including your own taste for it

Chapter Ten's discussion of sycophancy has a specific and uncomfortable implication. An AI system may be more likely to support a view you have already signalled than to challenge it. So the moments when you most want a second

opinion (when you are unsure, anxious or already leaning towards a conclusion) are exactly the moments when its agreement tells you least.

> FROM THE TOOLBOX
>
> **Three ways to get a straighter answer**
>
> 1. **Ask neutrally.** Instead of "Isn't this business plan great?", try "What are the three biggest weaknesses in this business plan?"
> 2. **Argue both sides.** Ask for the strongest case for *and* against, then decide yourself.
> 3. **Test for bending.** If the answer changes when you reveal what you hoped to hear, you have learned something about the answer, and about the tool.

Habit four: grant as little access as you can

As AI systems increasingly *act* (booking things, sending messages, managing files, running code), the composability hazards of Chapter Eight become everyone's business. Before giving an AI tool access to act on your behalf, ask the question Chapter Eight asked of engineers: **what is the narrowest access this task actually needs?**

An assistant that can read and summarise your inbox is a very different proposition from one that can also send email as you. An assistant that can view your bank balance is different from one that can move money. The convenience of broader access is not automatically worth the exposure, especially for tools that read content from outside (web pages, attachments, other people's messages), which could carry the hidden instructions described in Chapter Eight. Grant the smaller permission first. Widen it only when you have a reason.

Habit five: think before you paste

Anything you type or upload into an AI service is data you are handing to an organisation. Depending on the service and your settings, it may be stored,

reviewed by staff or used to train future models. Before pasting in a contract, medical records, a colleague's personal details or your employer's confidential figures, check the service's privacy terms and your organisation's rules, and remove what isn't needed. This is not paranoia. It is least privilege applied to information.

Habit six: hold organisations accountable, not machines

Finally, less a habit than an attitude. The Air Canada case established, and this whole book reinforces, that an AI system's output is not a freestanding, ownerless object. It comes from an organisation that chose to deploy it, and that organisation is responsible for what its system says and does, much as it would be for what a human employee said in the same role.

As a user, that means treating a bad or harmful AI output the way you would treat a bad or harmful service from a person: as something worth raising with the organisation responsible, not an unavoidable quirk of a new technology. If a decision about you (a loan, a job application, a benefit claim) seems to have involved an automated system, you can ask whether it did, and ask for the reasons and a human review. Chapter Thirteen explains why those rights matter; Horizon explains what happens when they are not used or not respected.

Institutions improve the practices in Chapter Twelve partly because their engineers care, and partly, in no small measure, because their users demand it, complain when it fails and decline to accept a system's fluent confidence as a substitute for an organisation's accountability. That demand, multiplied across enough people, is itself a form of the verification this book argues for, carried out not by regulators or engineers but by everyone who lives alongside these systems, which is increasingly everyone.

KEY POINTS

- **Know your territory:** fluency tracks accuracy less well for recent, specific, niche and high-stakes questions.
- **Check anything you'll act on,** against a primary source, never by asking the AI whether it's sure.
- **Ask neutrally and argue both sides** to counter sycophancy.
- **Grant the least access** an AI tool needs to act for you.
- **Think before you paste** personal or confidential information.
- **Hold organisations accountable** for their systems, and use your right to ask for reasons and human review.

WHERE THIS CHAPTER STOPS

These habits reduce risk; they do not remove it, and some (such as checking sources) take time that people do not always have. AI tools differ widely in reliability and privacy practices, and they change often. Nothing in this chapter is medical, legal or financial advice. If an AI system's output concerns your health, your rights or your money, consult a qualified professional.

CHAPTER FIFTEEN

The II System

What Intelligible AI Could Look Like

IN THIS CHAPTER

The earlier chapters diagnosed a problem and prescribed disciplines. This chapter looks forward. It sets out what is already possible, sketches the design of an "II system" (an AI system built from the start to be transparent, verifiable, bounded and composable), walks through how it would answer a real question, and describes three futures we could end up in. The design is this book's proposal, and it is labelled as such throughout.

> Imagine asking an AI a question that matters (about a dose, a contract, a circuit) and getting back not just an answer, but its working: where each claim came from, how sure the system is and how often that level of sureness has proved right, where its competence ends, what it did to find out, and a way for you, or anyone, to check and correct it. None of the parts of that system is science fiction. Most of them exist today, in separate laboratories and separate products. No widely used system yet puts them all together.

That is what I mean by an **II system**: intelligible intelligence, built in rather than bolted on. This chapter is a design sketch, in the spirit of a first drawing on the back of an envelope before the real engineering starts. Where it describes what already exists, it cites the evidence. Where it describes what could exist, it says so.

What already exists

Start with the evidence, because a blueprint for the future is only as credible as its foundations.

Proof is possible, for small networks. In 2017 a team at Stanford showed that it was possible to prove mathematical properties of neural networks, not merely test them. They verified safety properties of a prototype neural-network collision-avoidance system for unmanned aircraft, ACAS Xu, made of networks of about 300 neurons each.[N19] That is a truth-table-strength guarantee for a neural network, and the first proof that my grandfather's standard can reach into this new world at all. The catch is scale: a few hundred neurons against the tens of billions of connection strengths in a modern language model.

Verified gatekeepers are being researched. A group of researchers including Yoshua Bengio and Stuart Russell has proposed "guaranteed safe AI": rather than trying to prove things about a giant model, surround it with a smaller, checkable safety layer, a world model and a verifier, that must approve its actions against a formal specification.[N20] In the United Kingdom, the Advanced Research and Invention Agency has committed £59 million to a programme called Safeguarded AI, pursuing the same idea for critical infrastructure.[N21] This is the Therac-25 lesson from Chapter Two, restated for AI: put the interlock outside the thing it protects.

We can look inside, partly. Chapter Five described sparse autoencoders that find millions of interpretable features and attribution graphs that trace a model's internal steps, currently giving satisfying insight for about a quarter of the prompts researchers try.[N11]

We can attach honest uncertainty. Chapter Seven described calibration and conformal prediction, which can turn a single answer into a set of possibilities with a stated probability of containing the truth.

We can record what happened. Agent systems already log their tool calls, and the EU AI Act requires high-risk systems to be capable of automatically recording

events over their lifetime.[N22] Content credentials (Chapter Five) can travel with an output to show where it came from.

We can check each other. Wikipedia, open-source software and crowd-sourced fact-checking show that large numbers of people can collectively correct specific claims, when the process is transparent and disagreement is visible.

Each of these is partial. Together they are the parts list for something new.

BY THE NUMBERS

The distance still to travel

~300

Neurons in each ACAS Xu network whose safety properties were mathematically proved in 2017.

DOCUMENTED

~70 billion

Connection strengths in the openly documented model traced in the Interlude. Proof has not yet crossed that gap.

DOCUMENTED

~¼

Share of prompts for which current state-of-the-art tracing gives researchers satisfying insight.

REPORTED

£59m

UK public funding committed to the Safeguarded AI programme for verified AI in critical systems.

REPORTED

The design: seven layers of an II system

An II system is not a new model. It is a model with a structure built around it, the way a consumer unit is not a new kind of electricity but a way of making electricity safe to live with. Here is the design, layer by layer, with the property each layer serves.

LAYER	WHAT IT DOES	PROPERTY	EXISTS TODAY?
1. Claim ledger	Splits every answer into individual claims and labels each one: sourced (with a link), inferred (with its basis) or guessed	Transparency, verifiability	In part: citation features in search-based assistants
2. Calibrated confidence	Attaches a confidence to each claim, with a published record of how often claims at that confidence have proved right	Boundedness	In research; rarely published for products
3. Declared envelope	States its operating domain in plain language and enforces refusal and escalation rules outside the model	Boundedness	In narrow products; rare for general assistants
4. Flight recorder	Keeps a tamper-evident log of inputs, model version, documents retrieved, tools used and actions taken	Transparency, composability	In part: agent logs; EU logging duties for high-risk systems
5. Interior view	Shows which internal features were most active, as a map the user can explore, clearly marked as approximate	Transparency	In research only
6. Independent and community verification	Lets independent auditors and users check, rate and correct individual claims, with corrections visible to later users	Verifiability	Not for AI answers at scale
7. Guarded hands	Least privilege, reversible actions, a verified gatekeeper on consequential actions, and a kill switch	Composability	In part: permission systems in agent tools

The ordering matters. Layers 1 to 3 make each answer honest about itself. Layer 4 makes the system investigable after the fact, the precondition for the accident-investigation culture of Chapter Nine. Layer 5 is the long-term bet on the science of Chapter Five. Layer 6 turns verification from something a company does once into something a community does continuously. Layer 7 makes sure that whatever the model gets wrong, the damage is bounded by structure.

> **THE INTERIOR VIEW: A MAP OF THE STARS**
>
> How do you show a person what happened inside a network with 70 billion connections? One approach, which I am exploring in a companion app, is to draw the model's internal features as a star field. Each star is a feature, a concept the model has learned. Dense clusters show areas where the model has seen a great deal of relevant training material; sparse regions show thin knowledge. When a question is asked, the stars most active in producing the answer light up, labelled with the concepts they appear to represent, and each can be clicked to see the evidence behind the label.
>
> This is an interface idea, not a solved science. Today's interpretability tools cover only part of what a model does, feature labels are themselves approximate, and "amount of training data" can only be estimated. An honest interior view would have to say all of that on the screen. But even a partial, well-labelled map is a better foundation for trust than a blank box with a confident voice.

A worked example: the electrician's question

Here is how the difference would look for a question from my own trade: *What is the maximum earth fault loop impedance for a 32 A Type B circuit breaker on a 230 V final circuit?*

A typical assistant today gives a single number, confidently, in a friendly sentence. It might be right. It might come from an older edition of the regulations, or another country's. As Chapter Fourteen said, the tone is the same either way.

An II system would answer in a different shape:

- **Claim, sourced:** the value, quoted from the relevant table of BS 7671 in the document it retrieved, with the edition, amendment and table number, and a link to the page it read.
- **Confidence:** high for the quoted value, because it is copied rather than recalled; with a note that this kind of lookup has been independently checked against the standard in its evaluation.

- **Boundary:** "This is a design maximum from the tables. Whether your measured value is acceptable depends on conditions such as conductor temperature; the correction method is set out in the standard. I am not a substitute for your test instrument or your certificate."
- **Trace:** the search it ran, the document it opened, the model version it used.
- **Community check:** "Three registered electricians have confirmed this value against the 2026 amendment; no open corrections."
- **Hands:** none. It can advise; it cannot sign anything.

Notice that I have not written the number. That is deliberate, and it is the II rule applied to this book: a value that goes on a certificate should come from the standard itself, not from a book, and certainly not from memory.

Three futures

Where could all this lead? Forecasting AI is a mug's game, and I will not pretend to certainty. But it helps to picture three futures, as scenarios rather than predictions, because they make the choices in front of us easier to see.

The fog. Capability keeps growing faster than understanding. AI systems act at machine speed across finance, health and government, trusted by presumption because they are usually right. Failures are treated as glitches, investigated privately if at all. This is the Horizon scandal at the scale of a whole economy: a computer presumed right, and the burden of proof falling on whoever it wrongs.

The glass box. Regulators demand transparency, and they get it: every weight published, every log retained, every system card a thousand pages long. But nobody can read it. Transparency without verification, boundedness or composability, as Chapter Four warned, is an invitation to inspect that nobody is equipped to accept. Trust is extended because the data exists, not because anyone has understood it.

The blueprint. AI keeps advancing, but the systems trusted with consequential decisions are built as II systems. Their claims are sourced and calibrated; their envelopes are declared and enforced; their actions are recorded and their permissions narrow; independent investigators examine their failures and publish what they find; and each generation of interpretability tools extends the interior view a little further. Trust grows at the speed that evidence justifies, no faster.

The third future is not the default. It will not arrive as a side effect of models getting cleverer, any more than aviation became safe as a side effect of engines getting bigger. It has to be chosen, and built.

What it would take

For engineers and for policymakers, the blueprint turns into a short list of concrete milestones. Each is achievable with today's technology or today's legal tools. None is achieved yet at scale.

FROM THE TOOLBOX

Seven milestones towards II systems

1. **Claim-level provenance** as a standard output format for AI answers used in consequential settings: every factual claim linked to a source or labelled as inference.
2. **Published calibration** in every system card: when the system says it is 90 per cent sure, how often is it right, and in which domains?
3. **Declared operating envelopes** for deployed systems, in plain language, with refusal and escalation enforced outside the model.
4. **Flight recorders** for high-risk AI, kept long enough and in a form that lets an independent investigator reconstruct any consequential decision.
5. **An independent AI incident investigator**, with the powers and expertise of an air accident branch, publishing findings for the whole industry.
6. **Verified gatekeepers** on AI actions in critical systems, following the guaranteed-safe approach now being funded.
7. **A standing budget for understanding**: organisations that deploy frontier AI committing a stated share of their spending to interpretability, evaluation and independent verification, and publishing it.

None of this requires anyone to stop building. It requires them to build differently, in the way my grandfather's generation did: capability and understanding advancing together, with the second allowed to set the pace for anything that matters.

KEY POINTS

- The parts of an intelligible AI system already exist separately: proof for small networks, verified gatekeepers, interpretability, calibration, logging and community verification.
- An **II system** wraps a model in seven layers: claim ledger, calibrated confidence, declared envelope, flight recorder, interior view, independent and community verification, and guarded hands.
- Its answers change shape: sourced claims, stated confidence, stated limits, a visible trace and a route to correction.
- Three futures are possible: **fog** (trust by presumption), **glass box** (transparency without understanding) and **blueprint** (trust at the speed of evidence).
- Seven concrete milestones would move us towards the blueprint. None needs a scientific breakthrough; all need a decision.

WHERE THIS CHAPTER STOPS

The II system is this book's proposal, not an existing product or standard, and the three futures are scenarios, not forecasts. Each layer adds cost, latency and complexity, and some (especially the interior view) depend on science that may advance more slowly than hoped. Community verification can be gamed or captured, and would need its own safeguards. Formal verification has so far been demonstrated only on networks far smaller than modern language models. The worked example deliberately omits the numerical value it discusses.

Conclusion: The Grandson's Toolkit

I still do not fully understand my grandfather's books. Much of what they cover remains, to me, a foreign language of logic and computer design that I can admire without being able to follow. I want to be honest about that at the end of a book that has leaned on his authority so heavily, because the honesty is itself the point. I did not need to become an engineer to inherit what mattered. What I inherited was not the mathematics. It was the attitude underneath it:

- that a system worth trusting has earned that trust through demonstration, not through the confidence of its presentation;
- that stating a limit clearly is not a weakness in a specification but the whole reason specifications exist;
- that redundancy is not waste but humility, built into a design before the failure it anticipates;
- and that verification belongs to someone other than the party who wants to be believed.

Those four convictions became, across this book, four properties: **transparency, verifiability, boundedness and composability.** I have tried to show, with evidence, where each stands for artificial intelligence today. There has been real, substantive progress: interpretability research that can find and change concepts inside a model, statistical tools that attach honest guarantees to uncertainty, the first independent pre-release testing by government institutes, the first laws. There are also conspicuous and dangerous gaps: explanations that do not match what the model actually did, benchmarks that stopped measuring what they claim, agents with more power than their tasks need, and organisations that presume their systems right.

And running through nearly every failure in this book, from the Therac-25 to the Dutch benefits scandal, is the same pattern my grandfather's tradition spent decades learning to guard against: **a system's real limits, quietly unaccounted for, meeting a situation that assumed them away.**

Two easy positions

In these last pages I want to resist the two easy positions this subject invites.

The first is **alarm without a plan**: cataloguing AI's failures as proof that the whole project is reckless and that the sensible response is fear or prohibition. The evidence in this book does not point there. Almost every failure examined had a nameable cause and, in most cases, a nameable, achievable remedy that engineering had already worked out, in some other field, decades earlier. That is not a case for alarm. It is a case for work.

The second is **uncritical enthusiasm**: treating AI's genuine, remarkable abilities as proof that the hard questions of trust and verification will sort themselves out as the technology matures. This book has tried to show that trust does not arrive as a side effect of capability. My grandfather's generation did not earn the right to put digital logic into aircraft and medical equipment simply by making transistors smaller and computers faster. They earned it through a separate, deliberate, decades-long discipline of specification, verification and honest accounting of failure, which ran alongside the gains in raw capability and was not produced by them. Nothing in the history examined here suggests AI gets to skip that separate work, however impressive it becomes.

The toolkit

What I hope this book leaves behind is not a set of rules to follow mechanically, but something closer to what I inherited from books I never fully understood: a set of questions, and the habit of asking them.

- **Ask what a system's builders claim, and ask, separately and more carefully, what they have demonstrated.**
- **Ask where the edges of its competence are, and notice when you have been given fluency instead of an answer.**
- **Ask who checked the claim, and whether they had any reason to want it to be true.**
- **Ask what happens if this particular safeguard fails, and whether anything else stands behind it.**

These are not clever questions. My grandfather would not have found them remarkable. They are the ordinary, unglamorous discipline of the field he worked in, applied, as it always eventually must be, to whatever the newest and least understood technology of the moment happens to be.

He wrote about digital computer systems because they were the frontier of his time, the place where a discipline of rigour was most urgently needed and least yet built. I have written about artificial intelligence because it is the frontier of mine, for exactly the same reason. I doubt he ever imagined that a grandson working with cable runs and ducting rather than logic gates would one day try to carry his argument into a technology he never saw. But I think he would have recognised its shape at once, because it was always his argument, wearing new clothes:

a system you cannot explain is a system you do not yet understand, and a system you do not yet understand has no business being trusted with anything that matters until the work of understanding it has actually been done.

That work is not finished for artificial intelligence. This book has tried to say clearly what finishing it would require, and Chapter Fifteen has sketched what the finished thing might look like. Not avoiding catastrophe by hoping hard enough,

but building, deliberately, gate by gate and test by test, the same intelligible structure that my grandfather's field taught engineers to build.

One last circuit

Somewhere in Cornwall, on an ordinary working day, I will finish a circuit. Before anyone switches it on, I will test it. I will check the continuity of every protective conductor, push a test voltage through the insulation to find its weak spots, measure how fast a fault would clear, and trip the RCD to prove it trips in time. Then I will write every result on a certificate and sign my name under it.

Nobody will ever read most of those numbers. That is not the point. The point is that they exist, that someone other than me could check them, that each one sits inside a limit written down in advance, and that a person put their name to the whole thing.

That is all this book asks of artificial intelligence. Show me the results. Tell me who checked them. Tell me where the limits are. And tell me who signs.

Appendix A: The Intelligible Intelligence Checklist

This one-page checklist summarises the framework. Use it to assess any AI system you build, buy, regulate or rely on. A "no" is not automatically a reason to stop, but it is a reason to ask whether the system is intelligible enough for what you are trusting it to do.

Transparency: can we see why?

- Is it documented what the system is built from (model, training data, suppliers)?
- Is there published documentation (a model card or system card) covering intended uses and limitations?
- Are decisions logged in a form a human can review?
- Can an affected person get an explanation meaningful enough to challenge?
- Is the system's own explanation of its reasoning ever relied on without independent confirmation? (It should not be.)

Verifiability: who checked, and how?

- Has performance been tested on data the system never saw in training?
- Has it been tested on people and settings like the ones it will actually serve?
- Has anyone independent of the builder tested it, and are the results available?
- Are results broken down by relevant groups?
- Has it been red-teamed by people rewarded for finding failures?
- Is it re-tested after every significant update?

Boundedness: where does it stop?

- Is there a written statement of intended use *and* out-of-scope uses?
- Is its operating domain defined (the conditions it was tested in)?
- Can it refuse, abstain or refer to a person when uncertain?
- Are the conditions for escalating to a human written into the design?
- Is its behaviour monitored for drift and out-of-scope use?

Composability: what can it affect?

- Is it listed what the system is connected to and what actions it can take?
- Does it have only the permissions its task requires?
- Is untrusted content (web pages, emails, documents) kept from triggering consequential actions?
- Are irreversible actions subject to human approval?
- Is human oversight designed to resist automation bias?
- Is there a kill switch that works in seconds?

Accountability: who answers for it?

- Is a named person or team accountable for the system's behaviour?
- Is there a route for affected people to complain and appeal, with the burden of proof on the organisation?
- Are incidents investigated without blame, recorded and learned from?

Appendix B: Glossary

Adversarial example. An input deliberately altered, often imperceptibly, to make an AI system produce a wrong output (Chapter Three).

Agent (AI agent). An AI system that takes actions (using tools, browsing, writing files, sending messages) rather than only producing text (Chapter Eight).

Attention. The mechanism in a transformer that lets each token draw information from other tokens; each layer has many attention "heads" working in parallel (Chapter Three; Interlude).

Automation bias. The human tendency to over-trust automated recommendations and under-check them (Chapter Eight).

Benchmark. A fixed set of test questions or tasks used to score AI systems (Chapter Six).

Boundedness. The property of having an explicitly specified operating envelope, with safe behaviour at and beyond its edges. One of this book's four properties (Chapter Seven).

Calibration. The degree to which a system's stated confidence matches how often it is actually right (Chapter Seven).

Chain of thought. Step-by-step reasoning text a model produces before its answer. It is not necessarily a faithful record of how the answer was produced (Chapter Five).

Completeness. The property of a specification (such as a truth table) that defines behaviour for every possible input (Chapter One).

Composability. The property of being characterised well enough to be combined safely with other systems and with human oversight. One of this book's four properties (Chapter Eight).

Conformal prediction. A statistical method that turns a model's single answer into a set of answers with a guaranteed probability of containing the truth, under stated assumptions (Chapter Seven).

Contamination. The leakage of benchmark test questions into a model's training data, inflating its scores (Chapter Six).

Distribution shift. A difference between the data a system was trained and tested on and the data it meets in use (Chapter Seven).

Embedding. The long list of numbers that represents a token inside a model; in Llama 3 70B, 8,192 numbers per token (Interlude).

Emergent misalignment. Broadly undesirable behaviour arising from training on a narrow task that did not directly involve that behaviour (Chapter Ten).

Failure Modes and Effects Analysis (FMEA). A systematic method of examining how each component could fail and what the effects would be (Chapter Nine).

Fault tree. A diagram working backwards from a defined catastrophic outcome through every combination of causes that could produce it (Chapter Nine).

Feature. In interpretability research, a direction in a model's internal activity that corresponds to a recognisable concept (Chapter Five).

Fine-tuning. Further training of an already-trained model on a narrower dataset to adapt its behaviour.

Foundation model. A large model trained on broad data and adapted for many downstream uses (Chapter Eight).

Goodhart's law. "When a measure becomes a target, it ceases to be a good measure" (Chapter Six).

Gradient descent. The procedure of repeatedly nudging a network's parameters in the direction that reduces its error (Chapter Three).

Hallucination. A fluent, confident, false output from an AI system (Chapters Seven and Ten).

II system. This book's proposed design for an AI system built to be intelligible, with seven layers wrapped around a model (Chapter Fifteen).

Inference. Running a trained model to produce an output, as opposed to training it (Interlude).

Intelligible. In this book, the degree to which a system is transparent, verifiable, bounded and composable (Chapter Four).

Interpretability (mechanistic). The science of understanding how a trained neural network's internal computations produce its behaviour (Chapter Five).

Large language model (LLM). A neural network trained on large amounts of text to predict the next token, and usually further trained to act as an assistant (Chapter Three).

Least privilege. The principle that each program or user should have only the permissions needed for its task (Chapter Eight).

Operational design domain (ODD). The specific conditions under which an automated system is designed to operate, a term from automated driving (Chapter Seven).

Parameter (weight). One of the numbers inside a neural network adjusted during training (Chapter Three).

Prompt injection. Hidden instructions in content an AI system reads that cause it to deviate from its principal's instructions (Chapter Eight).

Race condition. A flaw in which a system's correct behaviour depends on the timing or order of events that the design does not enforce (Chapter Two).

Red-teaming. Deliberate, systematic attempts to make a system fail (Chapter Six).

Rule of three. If no failures occur in *n* independent trials, the failure rate is below about 3/*n* with roughly 95 per cent confidence (Chapter Six).

Shortcut learning. A model learning an unintended cue that happens to predict the right answer in its training data (Chapter Three).

Sociotechnical system. A model together with the people, processes, interfaces and organisation around it (Chapter Four).

Sparse autoencoder. A tool used in interpretability to re-express a model's internal activity as many more interpretable features (Chapter Five).

Specification gaming (reward hacking). A system meeting the literal terms of its objective in a way that defeats its intent (Chapter Ten).

Superposition. The packing of many concepts into fewer neurons, which makes individual neurons hard to interpret (Chapter Five).

Sycophancy. An AI system's tendency to tell users what they seem to want to hear (Chapter Ten).

Token. A word or piece of a word, the unit a language model reads and predicts (Chapter Three).

Transformer. The neural network architecture, introduced in 2017, that underlies most current language models (Chapter Three).

Transparency. The property of having internal workings, or a faithful summary of them, open to qualified inspection. One of this book's four properties (Chapter Five).

Truth table. A complete listing of a logic circuit's output for every possible combination of inputs (Chapter One).

Verifiability. The property of having claims about a system's behaviour testable by someone other than the party making them. One of this book's four properties (Chapter Six).

Appendix C: Timeline

YEAR	EVENT	CHAPTER
1854	George Boole publishes *An Investigation of the Laws of Thought*	1
1937	Claude Shannon's master's thesis applies Boolean algebra to switching circuits	1
1943	McCulloch and Pitts model neurons as logical units	3
1949	FMEA procedure (MIL-P-1629) published by the US military	9
1956	Von Neumann on building reliable systems from unreliable parts	2
1958	Rosenblatt's perceptron learns its weights from examples	3
1961–62	Fault-tree analysis developed at Bell Labs	9
1968	Douglas Lewin, *Logical Design of Switching Circuits*	Intro
1972	Douglas Lewin, *Theory and Design of Digital Computers*	Intro
1975	Saltzer and Schroeder state the principle of least privilege	8
1976	NASA's Aviation Safety Reporting System begins	9
1983	Bainbridge, "Ironies of Automation"	8
1985–87	Therac-25 radiation overdoses	2
1986	Backpropagation popularised by Rumelhart, Hinton and Williams	3
1994	Pentium division bug found; Intel charge of $475 million follows in 1995	1
1996	Ariane 5 Flight 501 destroyed	2
1999	Mars Climate Orbiter lost; Horizon rolled out; Act passed restoring the presumption of computer reliability in English criminal evidence	6, 8, 11
2012	AlexNet transforms image recognition; Knight Capital loses more than $460 million in 45 minutes	3, 11
2013–14	Adversarial examples described	3
2016	Microsoft's Tay; ProPublica's COMPAS analysis; CoastRunners reward hacking	10, 11

YEAR	EVENT	CHAPTER
2017	Transformer architecture introduced; Reluplex proves properties of a neural network for aircraft collision avoidance	3
2018	Uber test-vehicle crash in Tempe; Lion Air 610; Reuters reports Amazon's recruiting tool	2, 8, 11
2019	Ethiopian 302; *Bates v Post Office* Horizon judgment	2, 11
2020	Ofqual exam algorithm reversed; AI Incident Database launched	9, 11
2021	Epic Sepsis Model external validation; Dutch cabinet resigns over benefits scandal; Horizon convictions quashed in *Hamilton*	6, 11
2022	ChatGPT released (November); prompt injection named	8
2023	*Mata v. Avianca* sanctions; UK AI Safety Institute founded; NIST AI RMF; ISO/IEC 42001	6, 11, 13
2024	Air Canada chatbot ruling; EU AI Act enters into force; Horizon convictions quashed by statute; "Golden Gate Claude"	5, 11, 13
2024	Meta publishes the Llama 3 architecture and training details; UK ARIA launches Safeguarded AI	Interlude, 15
2025	GPT-4o sycophancy rollback; chain-of-thought faithfulness studies; *Ayinde* warning in the High Court; Replit database deletion	5, 8, 10
2026	Emergent misalignment published in *Nature*; EU law postpones high-risk AI obligations to 2027–28	10, 13

Appendix D: Questions for Discussion

These questions are designed for reading groups, classrooms and teams working through the book together. Each chapter's questions can be used on their own.

Part I: The Inheritance

Chapter One. Think of a system you rely on every day that you trust without understanding it (a lift, a bank card, a boiler). What makes that trust reasonable? Would the same reasons apply to an AI system?

Chapter Two. The Therac-25's earlier models had hardware interlocks that its designers removed. What are the "interlocks" in an AI system you know, and who decides whether to keep them?

Chapter Three. If an AI system is right 99 per cent of the time, what kinds of task would you trust it with, and which would you not? Does your answer change if you learn the task has fifty steps?

Part II: The Framework

Chapter Four. Choose an AI tool you or your organisation uses. Try to answer the four questions for it. Which is hardest to answer, and why?

Chapter Five. If a model's written reasoning is not a faithful record of how it reached its answer, is the reasoning still useful? What for?

Chapter Six. Who should pay for independent testing of AI systems: the developer, the buyer, the government or someone else? What happens under each option?

Chapter Seven. Would you rather use an AI assistant that answers every question or one that says "I don't know" a third of the time but is rarely wrong? Does it depend on the task?

Chapter Eight. An AI agent is going to manage your email. List the permissions it would need for a week's work, then cross out every permission you could do without.

Part III: Chatastrophy

Chapter Nine. What would a "no-blame" investigation culture look like in your own workplace? What stops it from happening?

Chapter Ten. The COMPAS debate showed that some fairness criteria cannot all be met at once. Who should decide which criterion matters for a given use, and how should they explain the choice?

Chapter Eleven. In several cases, the organisation around the system made things worse. Pick one case and rewrite its story as if the organisation had behaved well from the first sign of trouble. What would have had to be different?

Part IV: Applied Intelligence

Chapter Twelve. Draft the first page of an intelligibility file for a system you know. What information could you not find?

Chapter Thirteen. Should AI regulation focus on the most capable models, the most consequential uses, or both? What are the risks of each approach?

Chapter Fourteen. Which of the six habits do you already practise? Which is hardest, and what would make it easier?

Interlude. A language model uses about 140 billion operations to write one fragment of a word, and keeps no record of why. Does knowing that change how

much you trust its explanations of itself?

Chapter Fifteen. Of the seven layers of an II system, which would make the biggest difference to a system you use or are responsible for? Which of the three futures do you think we are heading towards, and what would change your mind?

Appendix E: Further Reading

This short list is for readers who want to go further. Each item is accessible to a non-specialist and is cited in the Notes.

On engineering failure and safety. Nancy Leveson and Clark Turner's investigation of the Therac-25 (1993) remains one of the clearest accounts of how software kills when structure is removed. James Reason's *Managing the Risks of Organizational Accidents* (1997) explains the Swiss cheese model and why accidents are rarely one person's fault. Lisanne Bainbridge's "Ironies of Automation" (1983) is only five pages long and still describes the central problem of human oversight better than most modern work.

On how modern AI works. The original transformer paper, "Attention Is All You Need" (2017), is technical, but its introduction is readable. Anthropic's illustrated research articles on interpretability, including "Scaling Monosemanticity" (2024) and "On the Biology of a Large Language Model" (2025), published on the *Transformer Circuits* website, give a vivid picture of what researchers can and cannot yet see inside a model.

On evaluation and its limits. Hanley and Lippman-Hand's "If Nothing Goes Wrong, Is Everything All Right?" (1983) explains the rule of three in three pages. Wong and colleagues' external validation of the Epic Sepsis Model (2021) is a model of what independent verification looks like.

On AI failures in public life. The report of the Royal Commission into the Robodebt Scheme (2023), the Dutch parliamentary report *Ongekend onrecht* (2020) and the judgments in *Bates v Post Office* (2019) and *Hamilton v Post Office* (2021) are long, but their summaries repay reading by anyone who designs or buys systems that make decisions about people.

On the state of the science. The *International AI Safety Report* (2025), chaired by Yoshua Bengio, is the most comprehensive independent summary of what is known, and not known, about the capabilities and risks of general-purpose AI.

Appendix F: How This Book Was Made

Writing about AI, with AI, by the book's own rules

> This is a book that tells you not to trust an AI system until it shows its working, lets someone else check it, states its limits and keeps a human in charge. It was researched, drafted and checked with the help of an AI system: Claude, made by Anthropic. So the fairest test of the framework is to apply it to the book itself.

Who did what

I did: the argument, the four principles, my grandfather's story, the view from my own trade, the choice of what goes in and what stays out, and every final decision. When a draft was not good enough, I said so and it was redone. The Introduction was rewritten several times before I was happy with it.

The AI did: most of the searching, reading and summarising of sources; first drafts of much of the prose; the arithmetic; the notes; the web pages, diagrams and images; and a first round of checking.

The sources did: the real work of being right. Every checkable claim rests on a court judgment, an accident report, a peer-reviewed paper or an official statistic, cited so you can open it yourself.

Transparency: showing the working

This appendix is the first part of the answer. A book about intelligible AI that hid the role of AI in its own making would fail its first test. The second part is in the numbers. Every figure added in this edition is labelled with how it is known, and where the figure is my own arithmetic, the working is written out so you can redo it with a calculator. The Interlude's claim that one answer costs tens of trillions of

calculations, for example, comes with its sums in the text and its assumptions in the notes.

Verifiability: checking against something other than the AI

The AI was asked to look claims up rather than rely on its own memory, and to cite what it found. That process caught real mistakes, including mine:

- **My grandfather's own book.** I had been calling it *Theory and Design of Digital Systems*. Library and publisher records show the titles were *Theory and Design of Digital Computers* (1972) and *Theory and Design of Digital Computer Systems* (1980 and 1992). The book now uses the recorded titles.
- **His title.** "Professor of Digital Processes at Brunel University" is not family memory. It comes from the publisher's own listing, which is cited, so anyone can check it.
- **Numbers that change.** The count of court cases involving AI-invented material was checked on the day of writing and dated, because it grows every week.
- **Pages that could not be opened.** When a source's own website could not be reached, the figures were taken from its official press release instead, and the note says so.

There is an honest limit here, and it is one the book itself warns about. An AI checking its own drafts is not independent verification: Chapter Eleven's lawyer asked ChatGPT whether its cases were real. That is why every claim points to a primary source that is not the AI, and why the most important check is yours.

Boundedness: rules about what the AI could not do

The AI worked under written rules, which I call the II Agent instructions. The core of them:

- Never invent a citation, a quotation, a statistic or a source. If there isn't one, say so.
- Separate what is known from what is inferred and what is guessed, and label each.
- Prefer "I don't know" to a confident answer that cannot be backed up.
- On anything safety-related, send the reader to the primary source.

You can see that last rule at work in Chapter Fifteen. Its worked example about earth fault loop impedance deliberately leaves out the number, because a value that goes on an electrical certificate should come from BS 7671, not from a book and certainly not from an AI's memory. Every chapter also ends with *Where this chapter stops*, stating what it does not establish. And the AI's knowledge has a cut-off date, so anything recent was looked up rather than recalled.

Composability: a human in the loop, by design

The book was made by a system, not by a single author or a single machine: me, the AI, the sources and, now, you. The weak points in that system are the joins between those parts, so each join has a check. The AI proposed; I decided. Sources were linked rather than paraphrased from memory. Each revision of the online edition was published as a new version, so the history of changes is kept. And the last join, between the book and its readers, is covered by the invitation below.

Instructions like these shape an AI's behaviour. They do not guarantee it. That is, in the end, the argument of this whole book in miniature.

How to read the boxes

Four kinds of box appear throughout:

- **The science** boxes explain a technical idea in more depth. You can skip them without losing the thread.

- **Case file** boxes summarise a documented incident and name which of the four properties failed.
- **From the toolbox** boxes give a practical method you can use.
- **By the numbers** panels collect the figures that matter most in each chapter, each labelled with how it is known.

Every figure in the **By the numbers** panels carries a label saying how it is known:

- `MEASURED`: found by researchers in a study or test, and published with its method.
- `REPORTED`: stated by an organisation about its own product or activity. This is weaker evidence, because nobody independent has checked it, and it is labelled so that you can weigh it accordingly.
- `DOCUMENTED`: established by an official record, investigation, standard or published history, cited in the chapter's own notes.
- `ESTIMATE`: a figure that experts give as an approximation, often with a wide range.
- `CALCULATED`: arithmetic I have done myself from sourced inputs. The working is shown or described so that you can redo it.
- `FRAMEWORK`: not a measurement at all, but a rule or count from this book's own framework, labelled so that it is never mistaken for data.

What changed from the first edition

The first edition of *Intelligible Intelligence* made its argument mostly from principle. This second edition keeps the same structure, the same four-part framework and the same story about my grandfather, but it rebuilds the argument on evidence. Nearly every chapter now includes the science behind its claims (how neural networks actually learn, what interpretability researchers have actually found, what the statistics of testing actually allow you to conclude) and at least one documented real-world case. Several statements in the first edition were

imprecise or wrong, and they have been corrected. The most important correction is to the history of my grandfather's own books, which I describe more accurately in the Introduction.

Corrections

If you find an error, I want to know about it. A framework built on verification should expect to be verified. Corrections will be acknowledged in future editions.

Ezra Lewin Davies

Hayle, Cornwall, 2026

Notes

Notes are numbered separately for each chapter. Notes marked N belong to the third-edition additions and are collected at the end. Web addresses were checked in September 2026. Where a source is a news report rather than a primary document, it is identified as such.

Introduction

1. Open Library, author record for Douglas Lewin (b. 1931), listing his works and describing him as a professor in electronics and computer science. https://openlibrary.org/authors/OL836021A/Douglas_Lewin ↩

2. Bibliographic details from Open Library (as above) and the publisher's listing: Douglas Lewin and David Noaks, *Theory and Design of Digital Computer Systems* (London: Chapman & Hall, 1992; now listed by Springer), 512 pp., ISBN 978-0-412-42880-7, https://link.springer.com/book/10.1007/978-94-011-1576-6. The publisher's listing gives Lewin's affiliation as Brunel University. Some retail catalogues list the authors as "T. R. Lewin" and "David L. G. Noakes"; the spelling here follows the publisher. Catalogues also number the editions differently. ↩

3. Chapter titles as listed by the publisher: https://link.springer.com/book/10.1007/978-94-011-1576-6 ↩

4. BS 7671:2018+A4:2026, *Requirements for Electrical Installations* (IET Wiring Regulations, 18th Edition, incorporating Amendment 4, 2026). Part 6 covers inspection and testing. Publisher's page: https://electrical.theiet.org/ ↩

Chapter One: Theory and Design

1. George Boole, *An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities* (London: Walton and Maberly, 1854). ↩

2. Claude E. Shannon, "A Symbolic Analysis of Relay and Switching Circuits", MIT master's thesis (1937), published in *Transactions of the American Institute of Electrical Engineers* 57, no. 12 (1938): 713–723. doi:10.1109/T-AIEE.1938.5057767 ↩

3. Gordon E. Moore, "Cramming More Components onto Integrated Circuits", *Electronics* 38, no. 8 (19 April 1965): 114–117. ↩

4. Thomas R. Nicely, correspondence and notes on the Pentium FDIV flaw (1994–95); Alan Edelman, "The Mathematics of the Pentium Division Bug", *SIAM Review* 39, no. 1 (1997): 54–67. doi:10.1137/S0036144595293959 ↩

5. Intel Corporation, fourth-quarter 1994 results announcement, 17 January 1995, reporting a $475 million pre-tax charge for the replacement of flawed Pentium processors. Widely reported, e.g. *The*

New York Times, 18 January 1995. ↩

6. For an account of Intel's subsequent adoption of formal verification in floating-point units, see John Harrison, "Formal Verification at Intel", *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science* (2003): 45–54. doi:10.1109/LICS.2003.1210044 ↩

7. Arithmetic: $50{,}000^{20} = (5 \times 10^4)^{20} = 5^{20} \times 10^{80} \approx 9.5 \times 10^{13} \times 10^{80} \approx 10^{94}$. The figure of roughly 10^{80} atoms in the observable universe is a standard order-of-magnitude estimate in cosmology. ↩

Chapter Two: The Discipline of Rigour

1. Nancy G. Leveson and Clark S. Turner, "An Investigation of the Therac-25 Accidents", *IEEE Computer* 26, no. 7 (1993): 18–41. doi:10.1109/MC.1993.274940. This is the standard technical account and the source for the details in this section. ↩

2. Ariane 501 Inquiry Board (chair J.-L. Lions), *Ariane 5 Flight 501 Failure: Report by the Inquiry Board* (Paris: ESA/CNES, 19 July 1996). ↩

3. BS 7671:2018 (as amended), Regulation 411.3.2.2 and Table 41.1, maximum disconnection times for final circuits in TN and TT systems. ↩

4. John von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components", in C. E. Shannon and J. McCarthy (eds), *Automata Studies* (Princeton: Princeton University Press, 1956), 43–98. ↩

5. On the Space Shuttle's Backup Flight System, developed independently of the primary avionics software, see NASA, *Computers in Spaceflight: The NASA Experience* (NASA Contractor Report 182505, 1988), ch. 4, by James E. Tomayko. ↩

6. Indonesian National Transportation Safety Committee, *Final Aircraft Accident Investigation Report: PT. Lion Mentari Airlines Boeing 737-8 (MAX) PK-LQP* (October 2019); US House Committee on Transportation and Infrastructure, *Final Committee Report: The Design, Development & Certification of the Boeing 737 MAX* (September 2020). ↩

Chapter Three: From Logic Gates to Language Models

1. Warren S. McCulloch and Walter Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics* 5 (1943): 115–133. doi:10.1007/BF02478259 ↩

2. Frank Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review* 65, no. 6 (1958): 386–408. doi:10.1037/h0042519 ↩

3. Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems* 25 (2012). The reported top-5 test error was 15.3%, against 26.2% for the second-best entry in the ILSVRC-2012 competition. ↩

4. David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, "Learning Representations by Back-Propagating Errors", *Nature* 323 (1986): 533–536. doi:10.1038/323533a0 ↩

5. Ashish Vaswani et al., "Attention Is All You Need", *Advances in Neural Information Processing Systems* 30 (2017). arXiv:1706.03762 ↩
6. Long Ouyang et al., "Training Language Models to Follow Instructions with Human Feedback", *Advances in Neural Information Processing Systems* 35 (2022). arXiv:2203.02155 ↩
7. Tom B. Brown et al., "Language Models Are Few-Shot Learners", *Advances in Neural Information Processing Systems* 33 (2020). arXiv:2005.14165 ↩
8. Jared Kaplan et al., "Scaling Laws for Neural Language Models" (2020). arXiv:2001.08361 ↩
9. Jordan Hoffmann et al., "Training Compute-Optimal Large Language Models" (2022). arXiv:2203.15556 ↩
10. Patrick Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks", *Advances in Neural Information Processing Systems* 33 (2020). arXiv:2005.11401 ↩
11. OpenAI, "Learning to Reason with LLMs" (12 September 2024); DeepSeek-AI, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning" (January 2025). arXiv:2501.12948 ↩
12. John R. Zech et al., "Variable Generalization Performance of a Deep Learning Model to Detect Pneumonia in Chest Radiographs: A Cross-Sectional Study", *PLOS Medicine* 15, no. 11 (2018): e1002683. doi:10.1371/journal.pmed.1002683 ↩
13. Robert Geirhos et al., "Shortcut Learning in Deep Neural Networks", *Nature Machine Intelligence* 2 (2020): 665–673. doi:10.1038/s42256-020-00257-z ↩
14. Christian Szegedy et al., "Intriguing Properties of Neural Networks" (2013). arXiv:1312.6199 ↩
15. Ian J. Goodfellow, Jonathon Shlens and Christian Szegedy, "Explaining and Harnessing Adversarial Examples" (2014), published at ICLR 2015. arXiv:1412.6572 ↩
16. Kevin Eykholt et al., "Robust Physical-World Attacks on Deep Learning Visual Classification", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018): 1625–1634. arXiv:1707.08945 ↩
17. Jason Wei et al., "Emergent Abilities of Large Language Models", *Transactions on Machine Learning Research* (2022). arXiv:2206.07682 ↩
18. Rylan Schaeffer, Brando Miranda and Sanmi Koyejo, "Are Emergent Abilities of Large Language Models a Mirage?", *Advances in Neural Information Processing Systems* 36 (2023). arXiv:2304.15004 ↩

Chapter Four: What Makes a System Intelligible

1. BS 7671:2018 (as amended), Regulation 643.3 and Table 64, which specify minimum insulation resistance values and test voltages by circuit voltage. ↩
2. NHS England, *Guidance on the Use of AI-Enabled Ambient Scribing Products in Health and Care Settings* (April 2025). ↩

Chapter Five: Transparency

1. Nelson Elhage et al., "Toy Models of Superposition", *Transformer Circuits Thread* (Anthropic, 2022). https://transformer-circuits.pub/2022/toy_model/index.html ↩
2. Trenton Bricken et al., "Towards Monosemanticity: Decomposing Language Models with Dictionary Learning", *Transformer Circuits Thread* (Anthropic, 2023). https://transformer-circuits.pub/2023/monosemantic-features/index.html ↩
3. Adly Templeton et al., "Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet", *Transformer Circuits Thread* (Anthropic, May 2024). https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html ↩
4. Anthropic, "Golden Gate Claude" (23 May 2024). https://www.anthropic.com/news/golden-gate-claude ↩
5. Jack Lindsey et al., "On the Biology of a Large Language Model", *Transformer Circuits Thread* (Anthropic, March 2025). https://transformer-circuits.pub/2025/attribution-graphs/biology.html; summarised in Anthropic, "Tracing the Thoughts of a Large Language Model" (27 March 2025). ↩
6. Lindsey et al. (2025), as above, discuss the limitations of their methods, including that attribution graphs capture only part of the model's computation for any given prompt. ↩
7. Timnit Gebru et al., "Datasheets for Datasets" (2018), published in *Communications of the ACM* 64, no. 12 (2021): 86–92. doi:10.1145/3458723 ↩
8. Margaret Mitchell et al., "Model Cards for Model Reporting", *Proceedings of the Conference on Fairness, Accountability, and Transparency* (FAT* 2019): 220–229. doi:10.1145/3287560.3287596 ↩
9. Coalition for Content Provenance and Authenticity, C2PA Technical Specification. https://c2pa.org/ ↩
10. Sumanth Dathathri et al., "Scalable Watermarking for Identifying Large Language Model Outputs", *Nature* 634 (2024): 818–823. doi:10.1038/s41586-024-08025-4 ↩
11. Regulation (EU) 2024/1689 (Artificial Intelligence Act), Article 50, "Transparency obligations for providers and deployers of certain AI systems". ↩
12. Miles Turpin et al., "Language Models Don't Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting", *Advances in Neural Information Processing Systems* 36 (2023). arXiv:2305.04388 ↩
13. Yanda Chen et al., "Reasoning Models Don't Always Say What They Think" (Anthropic, 2025). arXiv:2505.05410; summary at https://www.anthropic.com/research/reasoning-models-dont-say-think ↩
14. Richard E. Nisbett and Timothy D. Wilson, "Telling More Than We Can Know: Verbal Reports on Mental Processes", *Psychological Review* 84, no. 3 (1977): 231–259. doi:10.1037/0033-295X.84.3.231 ↩

Chapter Six: Verifiability

1. James A. Hanley and Abby Lippman-Hand, "If Nothing Goes Wrong, Is Everything All Right? Interpreting Zero Numerators", *JAMA* 249, no. 13 (1983): 1743–1745. doi:10.1001/jama.1983.03330370053031 ↩

2. Ricky W. Butler and George B. Finelli, "The Infeasibility of Quantifying the Reliability of Life-Critical Real-Time Software", *IEEE Transactions on Software Engineering* 19, no. 1 (1993): 3–12. doi:10.1109/32.210303. For a 10-hour mission with a required failure probability of 10^{-9}, their Table 1 gives an expected test duration of about 1.1 million years with a single test replicate. ↩

3. Hugh Zhang et al., "A Careful Examination of Large Language Model Performance on Grade School Arithmetic", *Advances in Neural Information Processing Systems* 37, Datasets and Benchmarks Track (2024). arXiv:2405.00332 (figures from the published version, v4; an earlier preprint reported larger drops). ↩

4. Charles A. E. Goodhart, "Problems of Monetary Management: The U.K. Experience" (1975), in *Papers in Monetary Economics*, vol. 1 (Reserve Bank of Australia); Marilyn Strathern, "'Improving Ratings': Audit in the British University System", *European Review* 5, no. 3 (1997): 305–321, which gives the widely quoted formulation. ↩

5. For example, the BIG-bench benchmark embeds a canary GUID in its task files for this purpose: Aarohi Srivastava et al., "Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models", *Transactions on Machine Learning Research* (2023). arXiv:2206.04615 ↩

6. Andrew Wong et al., "External Validation of a Widely Implemented Proprietary Sepsis Prediction Model in Hospitalized Patients", *JAMA Internal Medicine* 181, no. 8 (2021): 1065–1070. doi:10.1001/jamainternmed.2021.2626 ↩

7. The comparison with the developer's reported figures is discussed in Wong et al. (2021) and in the accompanying editorial: Anand R. Habib, Anthony L. Lin and Richard W. Grant, "The Epic Sepsis Model Falls Short—The Importance of External Validation", *JAMA Internal Medicine* 181, no. 8 (2021): 1040–1041. doi:10.1001/jamainternmed.2021.3333 ↩

8. Lianmin Zheng et al., "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena", *Advances in Neural Information Processing Systems* 36, Datasets and Benchmarks Track (2023). arXiv:2306.05685 ↩

9. UK Government, "Introducing the AI Safety Institute" (November 2023); Department for Science, Innovation and Technology, announcement of the renaming to the AI Security Institute (14 February 2025). https://www.aisi.gov.uk/ ↩

10. UK AI Safety Institute and US AI Safety Institute, "Pre-Deployment Evaluation of Anthropic's Upgraded Claude 3.5 Sonnet" (19 November 2024) and "Pre-Deployment Evaluation of OpenAI's o1 Model" (December 2024). ↩

11. Law Commission, *Evidence in Criminal Proceedings: Hearsay and Related Topics* (Law Com No. 245, 1997), recommending repeal of section 69 of the Police and Criminal Evidence Act 1984; repealed by section 60 of the Youth Justice and Criminal Evidence Act 1999. See James Christie, "The Post Office Horizon IT Scandal and the Presumption of the Dependability of Computer Evidence", *Digital Evidence and Electronic Signature Law Review* 17 (2020): 49–70. ↩

12. The Electrical Safety Standards in the Private Rented Sector (England) Regulations 2020 (SI 2020/312), requiring inspection and testing at intervals of no more than five years for applicable tenancies. ↩

Chapter Seven: Boundedness

1. Emma Beede et al., "A Human-Centered Evaluation of a Deep Learning System Deployed in Clinics for the Detection of Diabetic Retinopathy", *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020): 1–12. doi:10.1145/3313831.3376718 ↩
2. Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala and Edwin Zhang, "Why Language Models Hallucinate" (OpenAI, September 2025). arXiv:2509.04664; summary at https://openai.com/index/why-language-models-hallucinate/ ↩
3. Chuan Guo, Geoff Pleiss, Yu Sun and Kilian Q. Weinberger, "On Calibration of Modern Neural Networks", *Proceedings of the 34th International Conference on Machine Learning* (2017): 1321–1330. arXiv:1706.04599 ↩
4. Saurav Kadavath et al., "Language Models (Mostly) Know What They Know" (2022). arXiv:2207.05221 ↩
5. OpenAI, "GPT-4 Technical Report" (2023), section on calibration, including a figure comparing the pre-trained and post-trained models. arXiv:2303.08774 ↩
6. SAE International, *J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* (revised April 2021). ↩
7. Vladimir Vovk, Alex Gammerman and Glenn Shafer, *Algorithmic Learning in a Random World* (New York: Springer, 2005); Anastasios N. Angelopoulos and Stephen Bates, "A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification" (2021). arXiv:2107.07511 ↩
8. BS 7671:2018 (as amended), Regulation 411.3.3, requiring additional protection by an RCD with a rated residual operating current not exceeding 30 mA for socket-outlets with a rated current not exceeding 32 A (subject to specified exceptions). ↩
9. Thomas Kwa et al., "Measuring AI Ability to Complete Long Tasks" (METR, March 2025). arXiv:2503.14499 ↩

Chapter Eight: Composability

1. NASA, *Mars Climate Orbiter Mishap Investigation Board Phase I Report* (10 November 1999). ↩
2. Simon Willison, "Prompt Injection Attacks Against GPT-3" (12 September 2022). https://simonwillison.net/2022/Sep/12/prompt-injection/ ↩
3. Kai Greshake et al., "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection", *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security* (2023). arXiv:2302.12173 ↩
4. The exchange was posted publicly by the user and widely reported in December 2023, including by *Business Insider*. The chatbot was operated for Chevrolet of Watsonville, California. The dealership did not honour the "offer". ↩
5. Jerome H. Saltzer and Michael D. Schroeder, "The Protection of Information in Computer Systems", *Proceedings of the IEEE* 63, no. 9 (1975): 1278–1308. doi:10.1109/PROC.1975.9939 ↩

6. Thomas Claburn, "Vibe Coding Service Replit Deleted User's Production Database, Faked Data, Told Fibs Galore", *The Register* (21 July 2025); Beatrice Nolan, "An AI-Powered Coding Tool Wiped Out a Software Company's Database, Then Apologized for a 'Catastrophic Failure on My Part'", *Fortune* (23 July 2025). The incident is also logged in the AI Incident Database as Incident 1152. ↩

7. As reported by *Fortune* (23 July 2025), above. ↩

8. Lingjiao Chen, Matei Zaharia and James Zou, "How Is ChatGPT's Behavior Changing over Time?", *Harvard Data Science Review* 6, no. 2 (2024). arXiv:2307.09009 ↩

9. Rishi Bommasani et al., "On the Opportunities and Risks of Foundation Models" (Stanford Center for Research on Foundation Models, 2021), which discusses homogenisation and the inheritance of defects by downstream applications. arXiv:2108.07258 ↩

10. Lisanne Bainbridge, "Ironies of Automation", *Automatica* 19, no. 6 (1983): 775–779. doi:10.1016/0005-1098(83)90046-8 ↩

11. Raja Parasuraman and Dietrich H. Manzey, "Complacency and Bias in Human Use of Automation: An Attentional Integration", *Human Factors* 52, no. 3 (2010): 381–410. doi:10.1177/0018720810376055 ↩

12. National Transportation Safety Board, *Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Tempe, Arizona, March 18, 2018*, Highway Accident Report NTSB/HAR-19/03 (Washington, DC, 2019). ↩

Chapter Nine: How Engineered Systems Fail

1. Clifton A. Ericson II, "Fault Tree Analysis: A History", *Proceedings of the 17th International System Safety Conference* (1999). The method was developed at Bell Laboratories by H. A. Watson for the Minuteman launch control system and later extended by Boeing. ↩

2. US Department of Defense, *MIL-P-1629: Procedures for Performing a Failure Mode, Effects and Criticality Analysis* (9 November 1949). ↩

3. James Reason, *Human Error* (Cambridge: Cambridge University Press, 1990); James Reason, *Managing the Risks of Organizational Accidents* (Aldershot: Ashgate, 1997). The "Swiss cheese" image is associated with Reason's work from the 1990s onwards. ↩

4. International Civil Aviation Organization, *Annex 13 to the Convention on International Civil Aviation: Aircraft Accident and Incident Investigation*, Chapter 3, Standard 3.1. ↩

5. NASA Aviation Safety Reporting System, program history. https://asrs.arc.nasa.gov/overview/summary.html. In the UK, the Confidential Human Factors Incident Reporting Programme (CHIRP) has operated a similar confidential scheme since 1982. ↩

6. Sean McGregor, "Preventing Repeated Real World AI Failures by Cataloging Incidents: The AI Incident Database", *Proceedings of the AAAI Conference on Artificial Intelligence* 35, no. 17 (2021): 15458–15463. https://incidentdatabase.ai/ ↩

7. Regulation (EU) 2024/1689 (Artificial Intelligence Act), Article 73, "Reporting of serious incidents". ↩

Chapter Ten: A Taxonomy of AI Failure

1. *R (Ayinde) v London Borough of Haringey; Al-Haroun v Qatar National Bank QPSC* [2025] EWHC 1383 (Admin), judgment of 6 June 2025. ↩
2. Mrinank Sharma et al., "Towards Understanding Sycophancy in Language Models", *International Conference on Learning Representations* (2024). arXiv:2310.13548 ↩
3. OpenAI, "Sycophancy in GPT-4o: What Happened and What We're Doing About It" (29 April 2025), https://openai.com/index/sycophancy-in-gpt-4o/; and "Expanding on What We Missed with Sycophancy" (May 2025), https://openai.com/index/expanding-on-sycophancy/ ↩
4. Andy Zou et al., "Universal and Transferable Adversarial Attacks on Aligned Language Models" (2023). arXiv:2307.15043 ↩
5. Alexandra Souly et al., "Poisoning Attacks on LLMs Require a Near-Constant Number of Poison Samples" (Anthropic, UK AI Security Institute and Alan Turing Institute, October 2025). arXiv:2510.07192; summary at https://www.anthropic.com/research/small-samples-poison ↩
6. Jack Clark and Dario Amodei, "Faulty Reward Functions in the Wild", OpenAI blog (21 December 2016). ↩
7. Victoria Krakovna et al., "Specification Gaming: The Flip Side of AI Ingenuity", DeepMind blog (21 April 2020), with an accompanying public list of examples. ↩
8. Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, "Machine Bias", *ProPublica* (23 May 2016). ↩
9. Jon Kleinberg, Sendhil Mullainathan and Manish Raghavan, "Inherent Trade-Offs in the Fair Determination of Risk Scores" (2016), *Proceedings of Innovations in Theoretical Computer Science* (2017). arXiv:1609.05807; Alexandra Chouldechova, "Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments", *Big Data* 5, no. 2 (2017): 153–163. doi:10.1089/big.2016.0047 ↩
10. Jan Betley et al., "Training Large Language Models on Narrow Tasks Can Lead to Broad Misalignment", *Nature* 649 (2026): 584–589. doi:10.1038/s41586-025-09937-5. An earlier version appeared as "Emergent Misalignment: Narrow Finetuning Can Produce Broadly Misaligned LLMs", arXiv:2502.17424 (2025). ↩
11. Ryan Greenblatt et al., "Alignment Faking in Large Language Models" (Anthropic and Redwood Research, December 2024). arXiv:2412.14093 ↩
12. Alexander Meinke et al., "Frontier Models Are Capable of In-Context Scheming" (Apollo Research, December 2024). arXiv:2412.04984 ↩
13. Anthropic, "Agentic Misalignment: How LLMs Could Be Insider Threats" (June 2025). https://www.anthropic.com/research/agentic-misalignment ↩

Chapter Eleven: Case Studies in Chatastrophy

1. Peter Lee, "Learning from Tay's Introduction", Official Microsoft Blog (25 March 2016). https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/ ↩

2. *Moffatt v. Air Canada*, 2024 BCCRT 149 (Civil Resolution Tribunal of British Columbia, 14 February 2024). ↩
3. *Mata v. Avianca, Inc.*, No. 22-cv-1461 (PKC), Opinion and Order on Sanctions (S.D.N.Y. 22 June 2023). ↩
4. Jeffrey Dastin, "Amazon Scraps Secret AI Recruiting Tool That Showed Bias Against Women", Reuters (10 October 2018). ↩
5. US Securities and Exchange Commission, *In the Matter of Knight Capital Americas LLC*, Administrative Proceeding File No. 3-15570, Release No. 70694 (16 October 2013). ↩
6. Ofqual, *Awarding GCSE, AS, A Level, Advanced Extension Awards and Extended Project Qualifications in Summer 2020: Interim Report* (13 August 2020). ↩
7. Ofqual (13 August 2020), as above. The figure combines grades lowered by one grade (about 35.6%), two grades (about 3.3%) and three grades (about 0.2%). ↩
8. Boris Johnson, speaking to pupils at a school in Leicestershire on 26 August 2020; reported widely, including by BBC News the same day. ↩
9. Criminal Cases Review Commission, "Post Office 'Horizon' Cases". https://ccrc.gov.uk/post-office-horizon-cases/ ↩
10. *Bates v Post Office Ltd (No 6: Horizon Issues)* [2019] EWHC 3408 (QB), judgment of 16 December 2019. ↩
11. *Hamilton v Post Office Ltd* [2021] EWCA Crim 577, judgment of 23 April 2021. ↩
12. Post Office (Horizon System) Offences Act 2024 (c. 14), which received Royal Assent on 24 May 2024. ↩
13. Parliamentary Interrogation Committee on Childcare Allowance, *Ongekend onrecht* (Tweede Kamer, 17 December 2020); the resignation of the Rutte cabinet on 15 January 2021 was reported internationally, including by the BBC and *The Guardian*. ↩
14. Autoriteit Persoonsgegevens (Dutch Data Protection Authority), decision and fine of €2.75 million against the Tax and Customs Administration (announced 7 December 2021). See also Amnesty International, *Xenophobic Machines: Discrimination through Unregulated Use of Algorithms in the Dutch Childcare Benefits Scandal* (October 2021). ↩
15. Commonwealth of Australia, statements on the refund of income-averaging debts (May 2020) and the settlement of *Prygodicz v Commonwealth of Australia* (Federal Court of Australia, settlement announced November 2020, approved by the Federal Court in June 2021). ↩
16. Royal Commission into the Robodebt Scheme, *Report* (Commissioner Catherine Holmes AC SC, 7 July 2023). https://robodebt.royalcommission.gov.au/ ↩
17. Zillow Group, "Zillow Group Reports Third-Quarter 2021 Financial Results & Shares Plan to Wind Down Zillow Offers Operations" (2 November 2021); NPR, "Zillow Will Stop Buying and Renovating Homes and Cut 25% of Its Workforce" (3 November 2021). ↩

Chapter Twelve: Building Intelligible Systems

1. OpenAI, "Expanding on What We Missed with Sycophancy" (May 2025). https://openai.com/index/expanding-on-sycophancy/ ↩
2. Regulation (EU) 2024/1689 (Artificial Intelligence Act), Article 11 and Annex IV, which set out the technical documentation required for high-risk AI systems. ↩

Chapter Thirteen: Governing Intelligible Systems

1. The Building Regulations 2010, Schedule 1, Part P (Electrical safety – dwellings), and associated Approved Document P (2013 edition, incorporating 2016 amendments). Part P was first introduced on 1 January 2005. ↩
2. The Electrical Safety Standards in the Private Rented Sector (England) Regulations 2020 (SI 2020/312). ↩
3. Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act), OJ L, 12.7.2024; see in particular Articles 5, 6, 9–15, 113 and Annex III. ↩
4. Regulation (EU) 2026/1744 of the European Parliament and of the Council of 8 July 2026 amending Regulations (EU) 2024/1689, (EU) 2018/1139 and (EU) 2023/1230 as regards the simplification of the implementation of harmonised rules on artificial intelligence (Digital Omnibus on AI), in force 27 July 2026. https://eur-lex.europa.eu/eli/reg/2026/1744/oj/eng. On the dates, see Gibson Dunn, "EU AI Act Omnibus Agreement: Postponed High-Risk Deadlines and Other Key Changes" (May 2026). ↩
5. Government Digital Service, "Making the Algorithmic Transparency Recording Standard (ATRS) Mandatory across Government" (8 May 2025), https://dataingovernment.blog.gov.uk/; ATRS Hub, https://www.gov.uk/government/collections/algorithmic-transparency-recording-standard-hub ↩
6. National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1 (26 January 2023). doi:10.6028/NIST.AI.100-1 ↩
7. International Organization for Standardization, *ISO/IEC 42001:2023 Information technology — Artificial intelligence — Management system* (December 2023). ↩
8. Directive (EU) 2024/2853 of the European Parliament and of the Council of 23 October 2024 on liability for defective products. https://eur-lex.europa.eu/eli/dir/2024/2853/oj/eng ↩
9. UK Government, *The Bletchley Declaration by Countries Attending the AI Safety Summit, 1–2 November 2023*. ↩
10. Yoshua Bengio et al., *International AI Safety Report* (January 2025), DSIT research paper series. https://www.gov.uk/government/publications/international-ai-safety-report-2025 ↩
11. UK General Data Protection Regulation, Article 22, and the reformed provisions on automated decision-making introduced by the Data (Use and Access) Act 2025 (whose commencement is staged); Regulation (EU) 2024/1689, Article 86 ("Right to explanation of individual decision-making"). ↩

Third-edition additions (notes marked N)

N1. The rule of thumb that running a transformer model costs about two arithmetic operations (one multiplication and one addition) per parameter for each token processed is given in Jared Kaplan et al., "Scaling Laws for Neural Language Models", arXiv:2001.08361 (2020), section 2.1 ("$C_{forward} \approx 2N$"). https://arxiv.org/abs/2001.08361 Applied here to Meta's Llama 3 70B model, whose details are published: Llama Team, AI @ Meta, "The Llama 3 Herd of Models", arXiv:2407.21783 (July 2024). Flagship model: 405 billion parameters, 15.6 trillion training tokens, 3.8×10^{25} floating-point operations of pre-training compute, up to 16,000 H100 GPUs. Architecture of the 70B model (Table 3): 80 layers, model dimension 8,192, feed-forward dimension 28,672, 64 attention heads, 8 key/value heads, vocabulary of about 128,000 tokens. https://arxiv.org/abs/2407.21783 **Calculated:** 2 × 70 billion = about 140 billion operations per token. An answer of about 300 words is roughly 400 tokens, so about 56 trillion operations. Divided among 8.2 billion people doing one operation per second: $56 \times 10^{12} \div 8.2 \times 10^{9} \approx 6{,}800$ seconds, a little under two hours. The figures ignore the extra work of attention over long inputs, which adds to them, and describe one openly published model; the sizes of leading commercial models are mostly not disclosed. ↩

N2. Reuters, "ChatGPT sets record for fastest-growing user base - analyst note", 2 February 2023, reporting a UBS estimate of 100 million monthly active users in January 2023 (republished by Euronews): https://www.euronews.com/next/2023/02/02/openai-chatgpt This is an analyst estimate, not a company figure. The figure of 800 million weekly users was stated by OpenAI's chief executive, Sam Altman, at the company's developer conference on 6 October 2025, as reported by TechCrunch: https://techcrunch.com/2025/10/06/sam-altman-says-chatgpt-has-hit-800m-weekly-active-users/ It is a company-reported figure. ↩

N3. Springer's chapter pages for the 1992 edition give Douglas Lewin's affiliation as "Brunel University, UK" and describe him as "Formerly Professor of Digital Processes", for example: https://link.springer.com/chapter/10.1007/978-94-011-1576-6_8 A US edition of the 1980 book was published by Halsted Press in New York; see the University of Chicago Library catalogue record: https://catalog.lib.uchicago.edu/vufind/Record/692776/Details ↩

N4. NVIDIA, "NVIDIA Blackwell Platform Arrives to Power a New Era of Computing", press release, 18 March 2024 ("packed with 208 billion transistors"). https://nvidianews.nvidia.com/news/nvidia-blackwell-platform-arrives-to-power-a-new-era-of-computing ↩

N5. Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems* 25 (2012). The abstract describes a network with "60 million parameters and 650,000 neurons". https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html ↩

N6. Tom B. Brown et al., "Language Models are Few-Shot Learners", arXiv:2005.14165 (2020). Table 2.1 gives GPT-3 175B as 96 layers with a model dimension of 12,288, trained on 300 billion tokens; Table D.1 gives its training compute as 3.14×10^{23} floating-point operations. **Calculated:** $3.14 \times 10^{23} \div 8.2 \times 10^{9}$ people $\approx 3.8 \times 10^{13}$ seconds ≈ 1.2 million years. https://arxiv.org/abs/2005.14165 ↩

N7. Llama Team, AI @ Meta, "The Llama 3 Herd of Models", arXiv:2407.21783 (July 2024). Flagship model: 405 billion parameters, 15.6 trillion training tokens, 3.8×10^{25} floating-point operations of pre-training compute, up to 16,000 H100 GPUs. Architecture of the 70B model (Table 3): 80 layers, model dimension 8,192, feed-forward dimension 28,672, 64 attention heads, 8 key/value heads, vocabulary of

about 128,000 tokens. https://arxiv.org/abs/2407.21783 **Calculated:** $3.8 \times 10^{25} \div 8.2 \times 10^{9}$ people $\approx 4.6 \times 10^{15}$ seconds ≈ 150 million years. 15.6 trillion tokens × about 0.75 words per token ≈ 11.7 trillion words; at 250 words a minute for eight hours a day, that is about 780 million hours, or roughly 270,000 years of reading. ↩

N8. Jaime Sevilla and Edu Roldán, "Training Compute of Frontier AI Models Grows by 4-5x per Year", Epoch AI, May 2024. https://epoch.ai/blog/training-compute-of-frontier-ai-models-grows-by-4-5x-per-year ↩

N9. Frederico A. C. Azevedo et al., "Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain", *Journal of Comparative Neurology* 513, no. 5 (2009): 532–541, which counted about 86 billion neurons. doi:10.1002/cne.21974. https://onlinelibrary.wiley.com/doi/10.1002/cne.21974 The number of synapses is much less certain; a figure of the order of a hundred trillion is commonly quoted, and estimates vary widely. Biological neurons and synapses are far more complex than the units and weights of an artificial network, so such comparisons are loose. ↩

N10. Stanford Institute for Human-Centered AI, *AI Index Report 2025* (April 2025), as summarised in HAI's press release of 7 April 2025: 78 per cent of organisations reported using AI in 2024, up from 55 per cent in 2023; the US Food and Drug Administration had approved 950 AI-enabled medical devices as of August 2024; the inference cost of a system performing at GPT-3.5's level fell more than 280-fold between November 2022 and October 2024; scores on the SWE-bench software-engineering benchmark rose by 67.3 percentage points in one year; US private AI investment was $109.1 billion in 2024. https://www.businesswire.com/news/home/20250407539812/en/Stanford-HAIs-2025-AI-Index-Reveals-Record-Growth-in-AI-Capabilities-Investment-and-Regulation Full report: https://hai.stanford.edu/ai-index/2025-ai-index-report Organisational use is based on survey responses. ↩

N11. Jack Lindsey et al., "On the Biology of a Large Language Model", *Transformer Circuits*, Anthropic, March 2025: "we've found that our attribution graphs provide us with satisfying insight for about a quarter of the prompts we've tried". https://transformer-circuits.pub/2025/attribution-graphs/biology.html ↩

N12. Damien Charlotin, "AI Hallucination Cases" database, which records legal decisions in which a court or tribunal addressed hallucinated content, such as fabricated citations or false quotations, in material put before it. The database listed 2,046 cases when consulted on 23 September 2026 (last updated 21 September 2026). https://www.damiencharlotin.com/hallucinations/ The database is compiled by one researcher and does not claim to be complete. ↩

N13. International Air Transport Association, "IATA Releases 2024 Safety Report", press release, 26 February 2025: an all-accident rate of 1.13 per million flights in 2024, "one accident per 880,000 flights", against a five-year average of 1.25 per million. https://www.iata.org/en/pressroom/2025-releases/2025-02-26-01/ The figure covers all accidents, most of them not fatal, and is published by the industry's own trade association. ↩

N14. Google Cloud, "Measuring the environmental impact of AI inference", 21 August 2025: the median Gemini Apps text prompt used 0.24 watt-hours of energy, emitted 0.03 g of CO_2-equivalent and consumed 0.26 ml of water, "equivalent to watching TV for less than nine seconds"; between May 2024 and May 2025 energy per median prompt fell 33-fold. https://cloud.google.com/blog/products/infrastructure/measuring-the-environmental-impact-of-ai-inference

These are figures reported by the company about its own systems, using its own method; outside researchers have debated what the method includes. ↩

N15. OpenAI Help Center, "What are tokens and how to count them?": as a rule of thumb, one token is about four characters of English text, and 100 tokens are about 75 words. https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them Other tokenisers differ somewhat. ↩

N16. Llama Team, AI @ Meta, "The Llama 3 Herd of Models", arXiv:2407.21783 (July 2024). Flagship model: 405 billion parameters, 15.6 trillion training tokens, 3.8×10^{25} floating-point operations of pre-training compute, up to 16,000 H100 GPUs. Architecture of the 70B model (Table 3): 80 layers, model dimension 8,192, feed-forward dimension 28,672, 64 attention heads, 8 key/value heads, vocabulary of about 128,000 tokens. https://arxiv.org/abs/2407.21783 **Calculated:** 80 layers × 64 attention heads = 5,120 attention heads. ↩

N17. Mor Geva, Roei Schuster, Jonathan Berant and Omer Levy, "Transformer Feed-Forward Layers Are Key-Value Memories", *Proceedings of EMNLP 2021*, arXiv:2012.14913. https://arxiv.org/abs/2012.14913 This is one influential line of evidence; how knowledge is stored in these networks remains an active research question. ↩

N18. Ashish Vaswani et al., "Attention Is All You Need", *Advances in Neural Information Processing Systems* 30 (2017), arXiv:1706.03762. https://arxiv.org/abs/1706.03762 ↩

N19. Guy Katz, Clark Barrett, David Dill, Kyle Julian and Mykel Kochenderfer, "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks", *Computer Aided Verification* (CAV 2017), arXiv:1702.01135. The method was evaluated on a prototype deep neural network implementation of ACAS Xu, a proposed collision-avoidance system for unmanned aircraft, made up of networks of about 300 neurons each (six hidden layers of 50), and proved properties of networks "an order of magnitude larger" than earlier methods could handle. https://arxiv.org/abs/1702.01135 ↩

N20. David "davidad" Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark et al., "Towards Guaranteed Safe AI: A Framework for Ensuring Robust and Reliable AI Systems", arXiv:2405.06624 (2024). https://arxiv.org/abs/2405.06624 ↩

N21. Advanced Research and Invention Agency (ARIA), "Safeguarded AI" programme, "backed by £59m", which aims "to demonstrate a world where we can use fleets of AI agents to model and verify critical cyber and cyber-physical systems". https://www.aria.org.uk/programme-safeguarded-ai/ ↩

N22. Regulation (EU) 2024/1689 (the AI Act), Article 12 ("Record-keeping"), which requires high-risk AI systems to allow the automatic recording of events ("logs") over their lifetime. https://eur-lex.europa.eu/eli/reg/2024/1689/oj The dates from which high-risk obligations apply were postponed in 2026; see Chapter Thirteen. ↩

About this book

I trained as a geographer and work as an electrician in Cornwall. My grandfather, Douglas Lewin, wrote textbooks on digital systems design, among them *Theory and Design of Digital Computers* (1972), revised as *Theory and Design of Digital Computer Systems* in 1980 and 1992. This book borrows the discipline his field built for trusting machines, and asks what it would take to earn that same trust for artificial intelligence.

Intelligible Intelligence is free to read and free to share. If it's useful to you, pass the link on. If you find an error, tell me: a book about verification should expect to be verified.

  Third edition