

# The Formal Wilderness

## The Inexact Sciences

**Crispy Chicken**



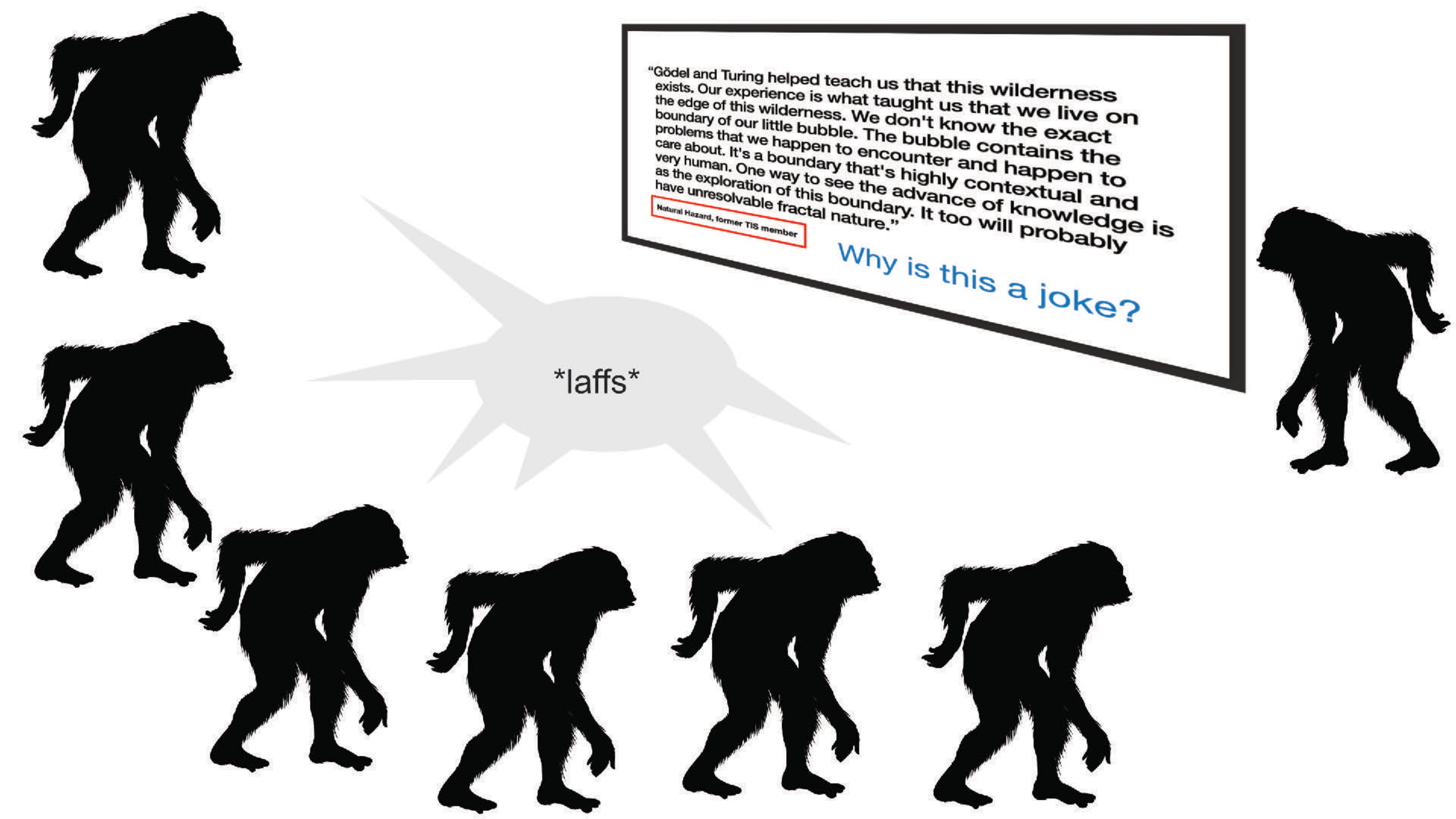


**“Gödel and Turing helped teach us that this wilderness exists. Our experience is what taught us that we live on the edge of this wilderness. We don't know the exact boundary of our little bubble. The bubble contains the problems that we happen to encounter and happen to care about. It's a boundary that's highly contextual and very human. One way to see the advance of knowledge is as the exploration of this boundary. It too will probably have unresolvable fractal nature.”**

**Natural Hazard, former TIS member**

**Why is this a joke?**





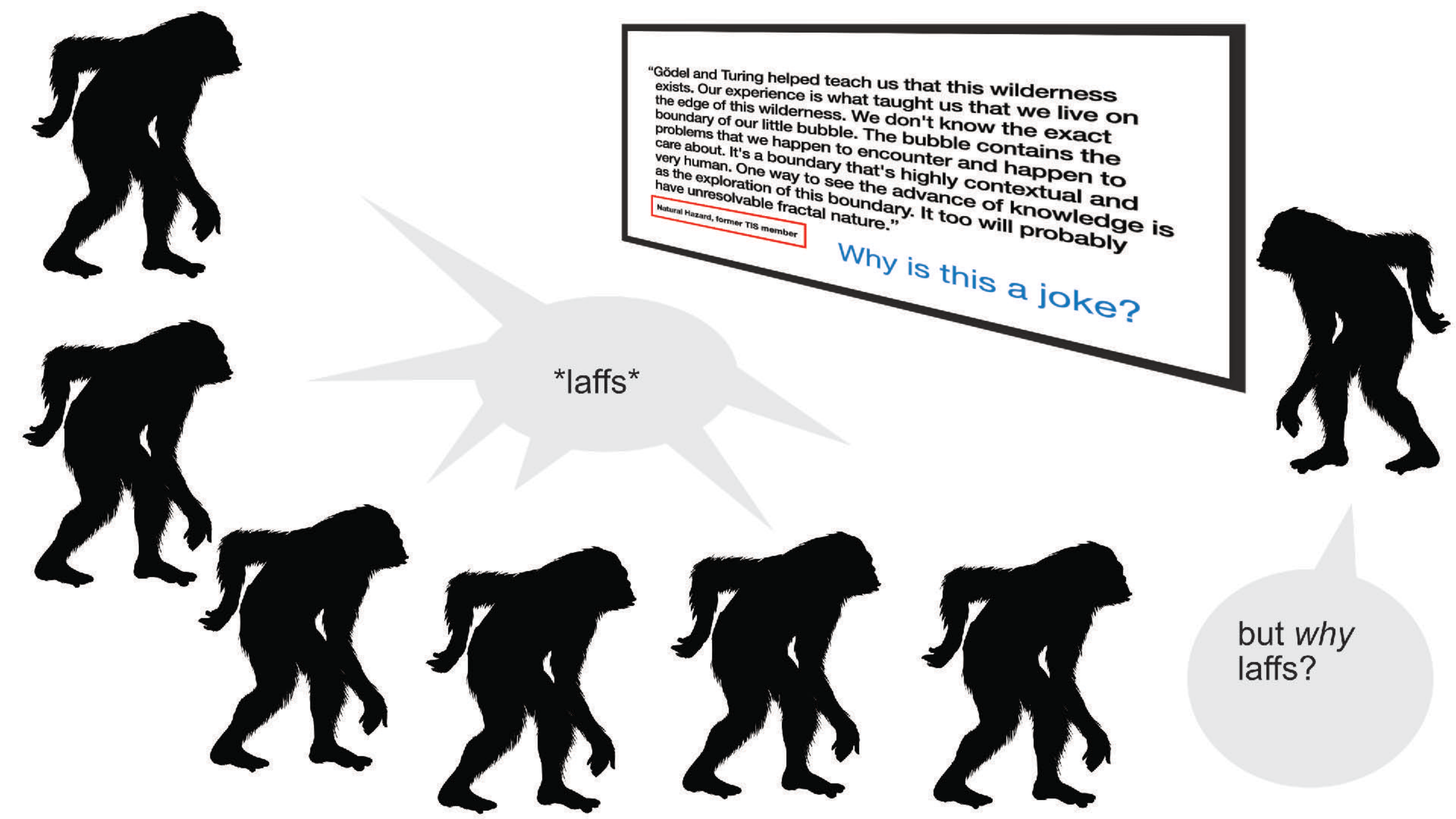
"Gödel and Turing helped teach us that this wilderness exists. Our experience is what taught us that we live on the edge of this wilderness. We don't know the exact boundary of our little bubble. The bubble contains the problems that we happen to encounter and happen to care about. It's a boundary that's highly contextual and very human. One way to see the advance of knowledge is as the exploration of this boundary. It too will probably have unresolvable fractal nature."

Natural Hazard, former TIS member

Why is this a joke?

\*laffs\*





"Gödel and Turing helped teach us that this wilderness exists. Our experience is what taught us that we live on the edge of this wilderness. We don't know the exact boundary of our little bubble. The bubble contains the problems that we happen to encounter and happen to care about. It's a boundary that's highly contextual and very human. One way to see the advance of knowledge is as the exploration of this boundary. It too will probably have unresolvable fractal nature."

Natural Hazard, former TIS member

Why is this a joke?

\*laffs\*

but *why*  
laffs?



**“Most language games are about building new objects using previously legitimized moves. Most interesting ‘plays’ are about creating something using an unexpected base/move combination.”**

**Crispy Chicken, just now**

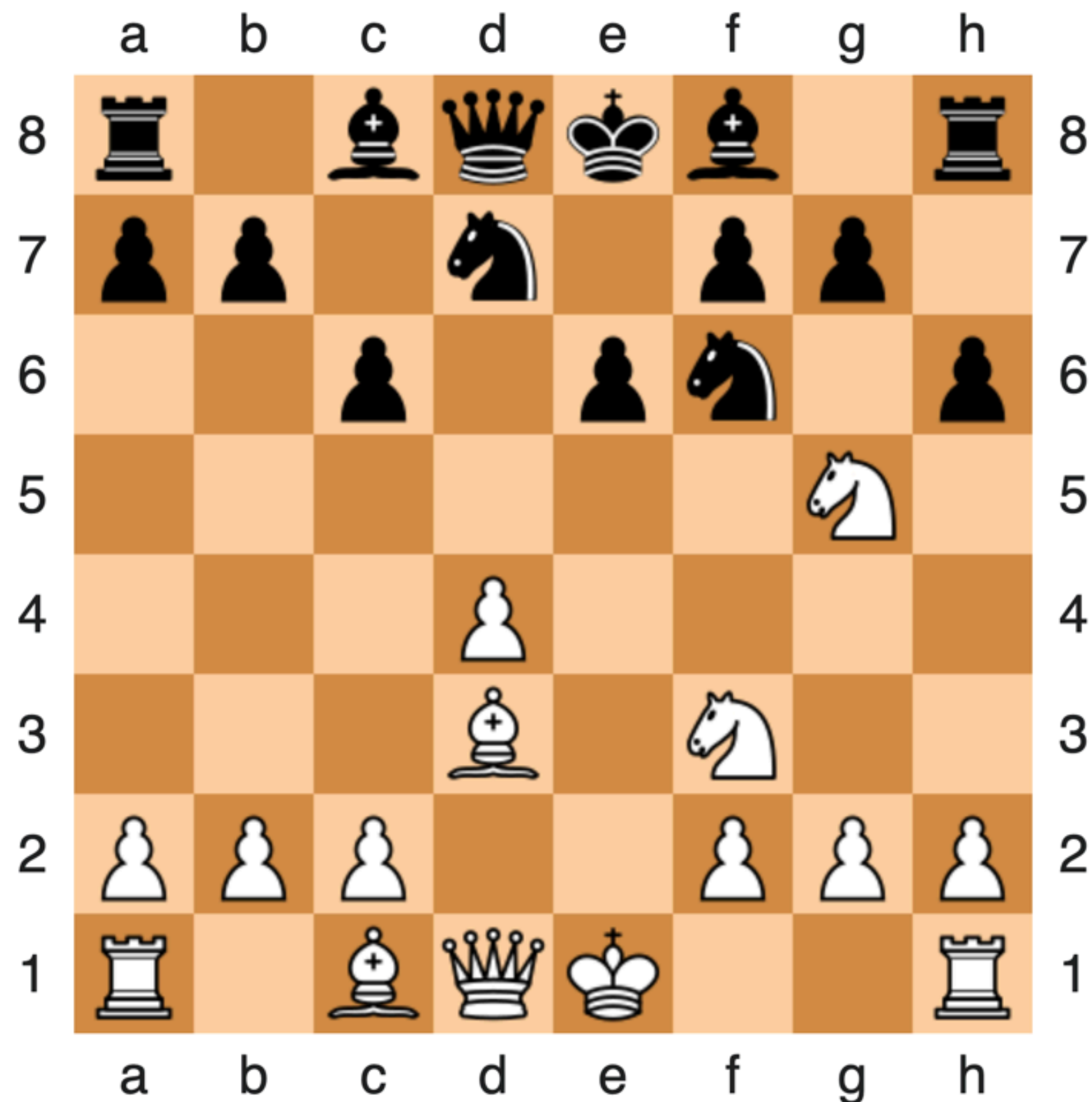








## Deep Blue–Kasparov, 1997, rd 6



Position after 7...h6

...Although the knight sacrifice is a well-known refutation, Kasparov reasoned that an engine wouldn't play the move without a concrete gain. It was later revealed that the Deep Blue team had added the variation into its opening database on the same day of the game. (Wikipedia)



**Math is a kind of language game.**



**“In mathematics, the aim is to increase one's confidence in the correctness of a theorem, and it's true that one of the devices mathematicians could in theory use to achieve this goal is a long chain of formal logic. But in fact they don't. What they use is a proof, a very different animal. Nor does the proof settle the matter; contrary to what its name suggests, a proof is only one step in the direction of confidence. We believe that, in the end, it is a social process that determines whether mathematicians feel confident about a theorem--and we believe that, because no comparable social process can take place among program verifiers, program verification is bound to fail. We can't see how it's going to be able to affect anyone's confidence about programs.”**

**Richard De Millo et al.**





**Math is a kind of  
*social* language game.**



# Why?

- Math (as it is practiced) is not a directed search of axiom application,
- Math (as it is *verified*) attempts true formalization, but the tools are nascent



**What makes a theorem “cool”?**

(1) **Definition**—a precise and unambiguous description of the meaning of a mathematical term. It characterizes the meaning of a word by giving all the properties and only those properties that must be true.

(2) **Theorem**—a mathematical statement that is proved using rigorous mathematical reasoning. In a mathematical paper, the term theorem is often reserved for the most important results.

(3) **Lemma**—a minor result whose sole purpose is to help in proving a theorem. It is a stepping stone on the path to proving a theorem. Very occasionally lemmas can take on a life of their own (Zorn's lemma, Urysohn's lemma, Burnside's lemma, Sperner's lemma).

(4) **Corollary**—a result in which the (usually short) proof relies heavily on a given theorem (we often say that “this is a corollary of Theorem A”).

(5) **Proposition**—a proved and often interesting result, but generally less important than a theorem.



(6) **Conjecture**—a statement that is unproved, but is believed to be true (Collatz conjecture, Goldbach conjecture, twin prime conjecture).

(7) **Claim**—an assertion that is then proved. It is often used like an informal lemma.

(8) **Axiom/Postulate**—a statement that is assumed to be true without proof. These are the basic building blocks from which all theorems are proved (Euclid's five postulates, Zermelo-Frankel axioms, Peano axioms).

(9) **Identity**—a mathematical expression giving the equality of two (often variable) quantities (trigonometric identities, Euler's identity).

(10) **Paradox**—a statement that can be shown, using a given set of axioms and definitions, to be both true and false. Paradoxes are often used to show the inconsistencies in a flawed theory (Russell's paradox). The term paradox is often used informally to describe a surprising or counterintuitive result that follows from a given set of rules (Banach-Tarski paradox, Alabama paradox, Gabriel's horn).



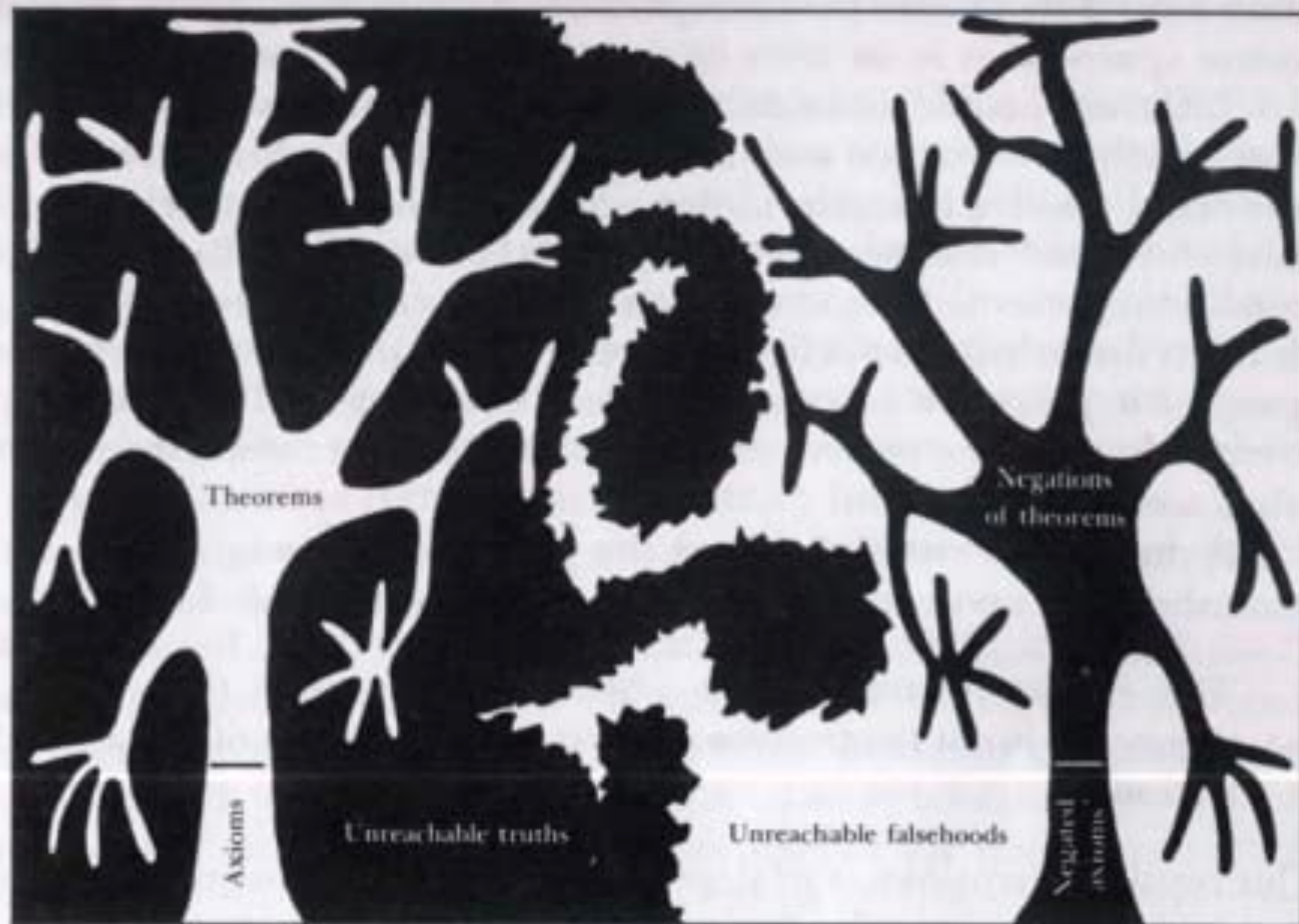
**If a theorem is cool, can we  
prove or disprove it?**



**“There exist formal systems whose negative space (set of nontheorems) is not the positive space (set of theorems) of any formal system.”**

**Douglas Hofstadter, GEB**





Well-formed formulas



**“I, just like the mathematicians of the early twentieth century, expected the world of formal systems and natural numbers to be more predictable than it is.”**

**Douglas Hofstadter, GEB**

**What kind of formal systems?**



# Peano Arithmetic

- 0 is a natural number.
- For every natural number  $x$ ,  $x = x$ .
- For all natural numbers  $x$  and  $y$ , if  $x = y$ , then  $y = x$ .
- For all natural numbers  $x$ ,  $y$  and  $z$ , if  $x = y$  and  $y = z$ , then  $x = z$ .
- For all  $a$  and  $b$ , if  $b$  is a natural number and  $a = b$ , then  $a$  is also a natural number.
- For every natural number  $n$ ,  $n+1$  is a natural number.
- For all natural numbers  $m$  and  $n$ ,  $m = n$  if and only if  $m+1 = n+1$ .
- For every natural number  $n$ ,  $n+1 = 0$  is false.

**...but also anything that allows for the  
Peano axioms to hold in a subset...**



# Reductions

- Define problem types A and B
- Show that problems of type A can be converted to problems of type B
- Rejoice, because:
  - All *universal* properties of type B problems now apply to type A problems.
  - Any properties that apply to a subset of type A problems, apply to at least some type B problems.

# Gödel Incompleteness Theorem at a Glance

- Very basic arithmetical systems include paradoxes
- Get fucked





**Wait but does this matter?**

# Crispy's Argument Towards Realism

- If we describe the universe with certain tools...
- And those tools accurately describe the universe...
- Then the subset of “problems” presented that the universe must “solve” by simply existing must either be “naturally computable”
- Actually, what does computable mean?



# Computable Numbers

- Think about the numbers between 0 and 1
- How many of them can a computer ever represent?
  - not many
  - ...but if you have a specific description of it you can always store that description

**If a problem is well-defined can  
we solve it?**



# Undecidability at a Glance

- There's a kind of problem type A
- There is no way to build a Turing Machine that will solve all problems of type A
- Get fucked



**Wait but does this matter?**



# Crispy's Decidability Strategy

- If problem  $A$  is undecidable, then it must be formally representable.
- We either have or haven't ever solved an instance of  $A$ .
  - If you have never solved an instance of  $A$ , it's probably just formally independent
  - If you have solved an instance of  $A$ , then you can probably find similar cases and decide on some subset of problem  $A$  that you understand how to deal with

**If a problem is decidable can we  
solve it?**

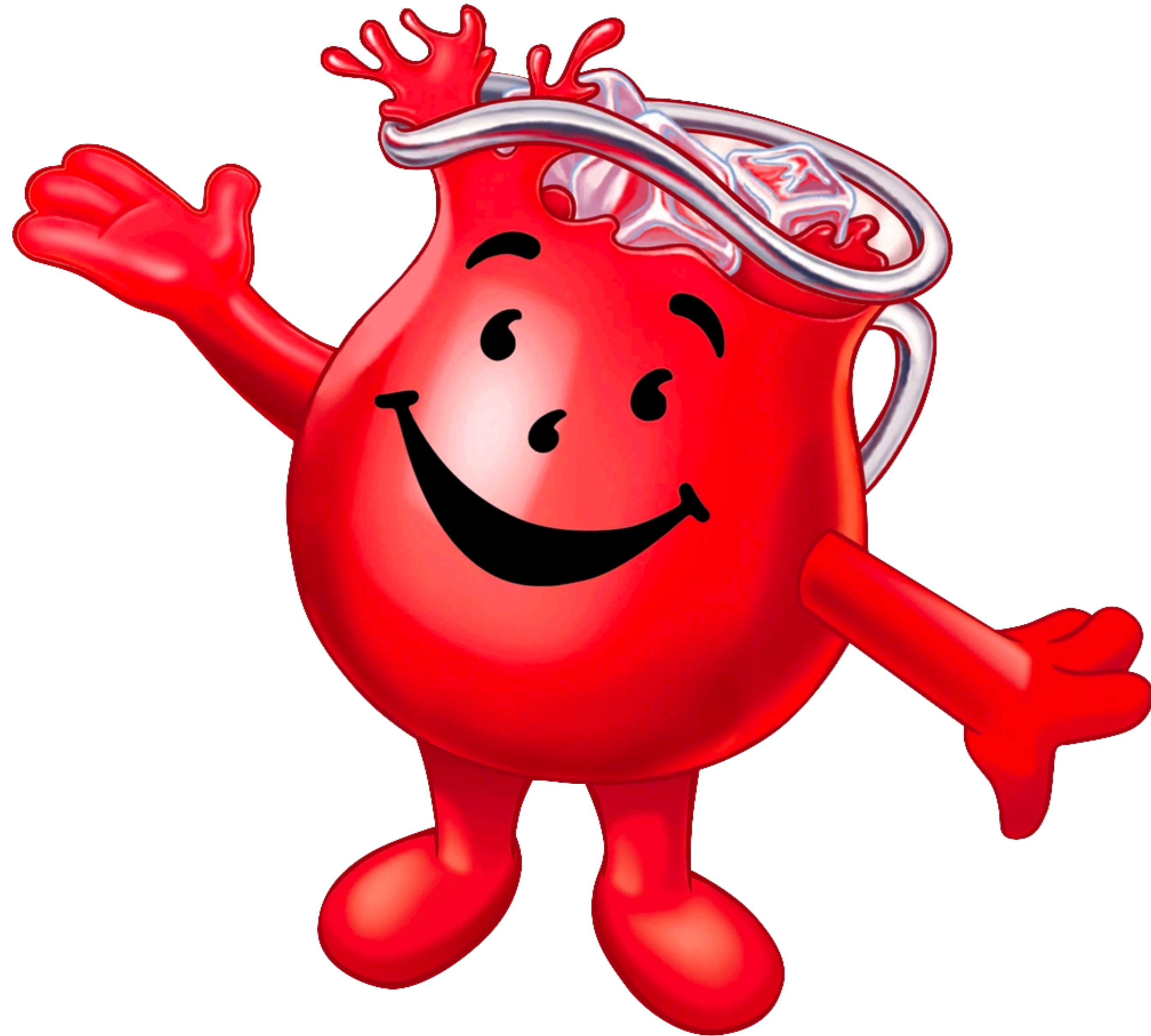


# Formal Independence at a Glance

- There's a kind of problem type A
- The tools that you've used to describe and analyze problem type A provably can't solve it
- Get fucked

**Wait but does this matter?**

oh yeah





# Axiom of Constructibility

- Are all sets describable as a combination of significantly simpler sets?
  - e.g. this is true for integers
- But this does not follow from our usual foundations, known as ZFC
- ...which means we're doing math with objects we couldn't construct from basic computational building blocks.

**If a problem is solvable, can we eventually find an efficient solution?**

# Obvious Counter-Example

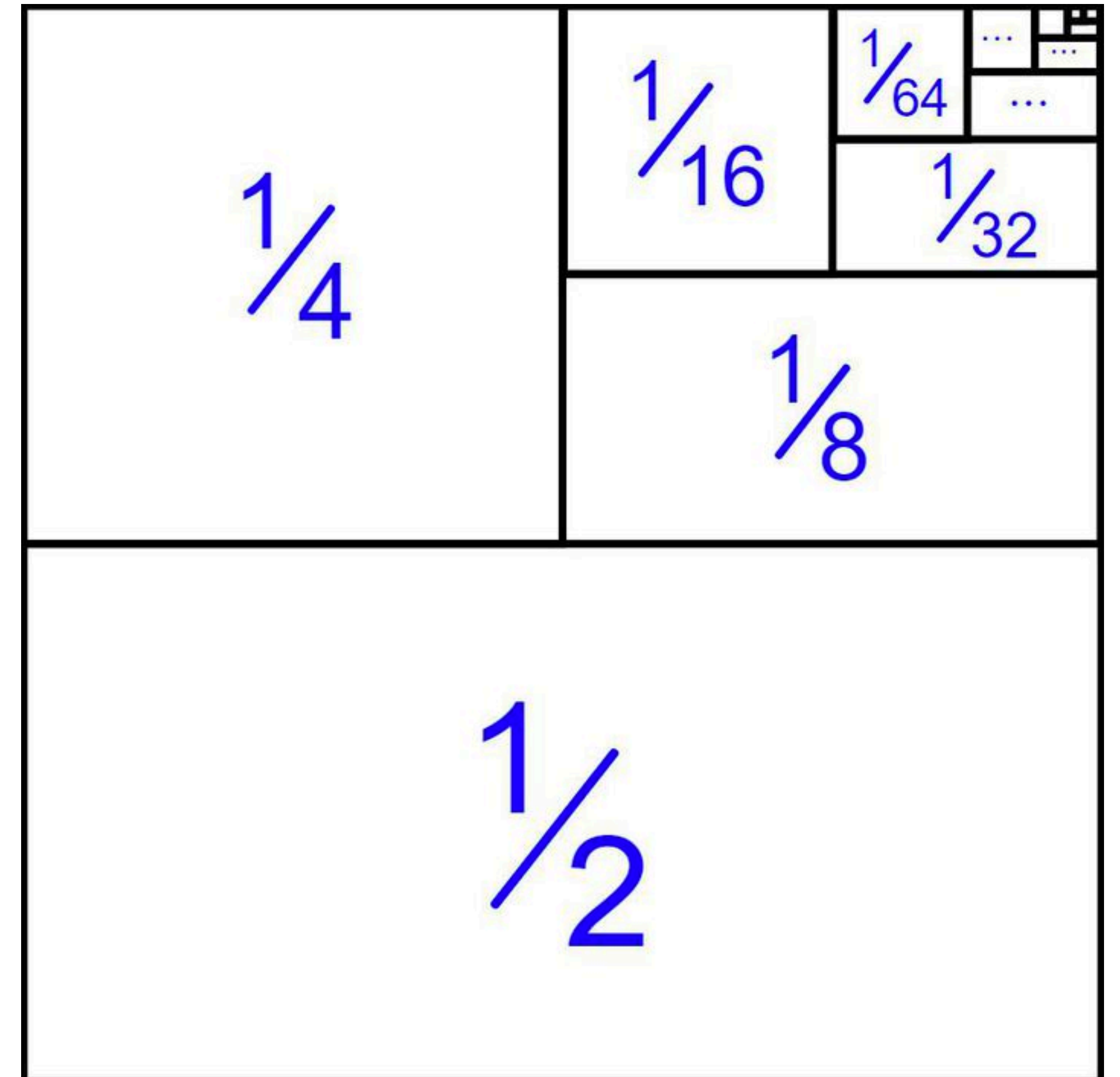
- The  $k$ -length Halting Problem
  - I give you a machine  $M$
  - Does  $M$  stop in  $k$  steps or less for inputs that are most size  $\log k$ ?
- The Halting Problem is undecidable
- But the  $k$ -length Halting Problem is finite
  - ...but requires evaluating an exponential number of possible inputs.



**Can we cache answers  
efficiently?**

# String Compression Again

- Imagine a string length  $l$
- I want to compress it by one bit.
- How many such strings can I compress?
- Using any possible compression scheme, I can compress at most half of the available strings.
- Why?



**Okay, so what do we do?**



# Crispy's *N* Strategies

# #1 Argue Towards Realism

- What stuff already gets done?
  - Traveling Salesman
  - Factorization
  - 3 Body Problem
- What subset of problems does it look like we can solve?
- Or can we use natural instances to solve by simulation?

# Gossip as Simulation





# #2 Measure the Tolerance

- Human needs are always based on tolerances and threshold
- Exact solutions were never even a “thing” until math
- Approximation is the hottest applied stuff in theory these days, because feeling sad about imperfection gets old quickly





# #3 Look for the Economy

- How hard is it to solve certain sub-problems?
  - Is one of them the thing you actually care about?
- Are there more general problems that solve most of what you care about, and allow you to fill in the details later?
- Can you bound properties of the problem to solve them within those bounds?
- Can you setup a system that amortizes the cost of solving over time?
- Can you use information from previous problems you've solved to make this easier?

# #4 Memorize

- Can you just keep all the problem instances in memory instead of calculating the pattern?
  - This is super common for humans, e.g., memorizing the circle of 5ths.
- Can you memorize specific sub-problems that always come-up
  - Endgames in Chess, Go, etc.
- Can you memorize part of the specific problem as you go and reuse them?
  - Dynamic programming

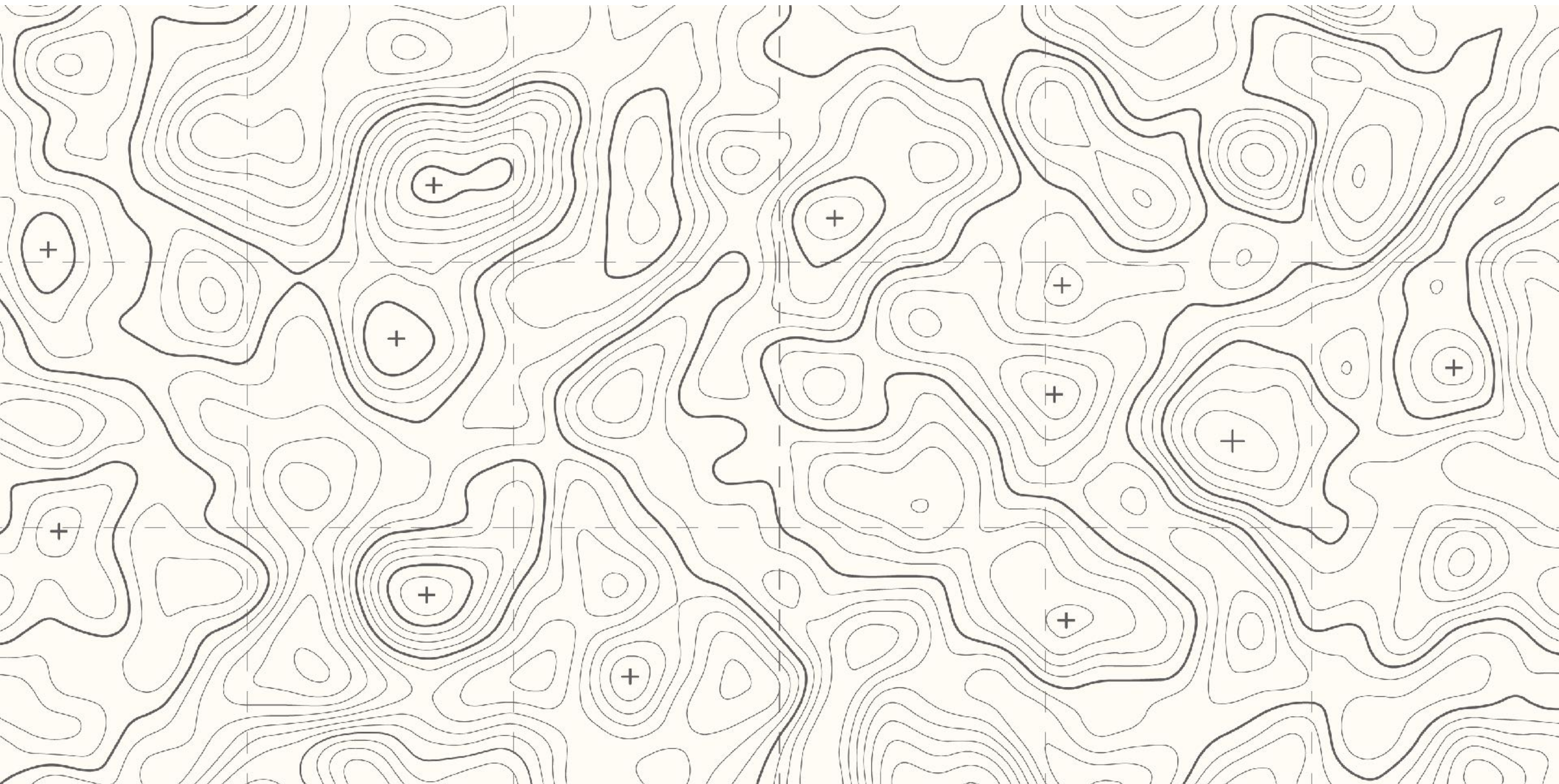
# #5 Consider the Description Length

- Problems that take a long time to specify, take a long time to solve.
  - ...unless most of the information was unimportant to the actual problem
- This is because if different variables interact with each other the problem gets combinatorially (usually exponentially) more complex
  - e.g., there are more legal Go boards than the number of atoms in the universe, *squared*
- Humans solve complex problems by relying on a few variables to decide on different sub-problems that need to be solved right now; it's not efficient it Just Works™











# #6 Use the Problem as a Locking Mechanism

- This is literally how all of Cryptography works:
  - Find a problem that's hard to solve and easy to verify
  - Use a solution to that problem as a signature, by backwards engineering the problem from the solution
- But it's also how inside jokes work.
  - Many require enough context to be able to mutate the joke rather than just parrot it back
  - e.g., memes, people who only repost memes are not in-group
  - you heard it here first

