Provably Safe AI + AIS via Debate

Stephan Wäldchen
Iliad

September 16, 2026

1 Prerequisites

- ★ How Turing machines work (Watch this Video), and a basic understanding of the idea of computational complexity (Watch this video), including the most important classes, P, NP, PSPACE, the notion of NP-completeness, and polynomial-time reductions, Oracle Turing machines.
- The difference between deterministic and non-Deterministic Turing machines, how mathematics is built on the axioms of set theory (used for most mathematics) or type theory (used mostly for proof checkers).

What you'll learn

- The basic idea of Provably Safe AI as both a Control and a Scalable Oversight Technique
- The difference between syntactic and semantic problems and why a world model is necessary to turn semantics into syntax.
- Different ideas of world models: Human judgement, Davidad-style world models, hardware specifications
- Why Debate could drastically reduce the queries to the world model/human oversight
- Secondary Debate concepts: Cross-Examination,
- Are familiar with the UK AISI safety case, and are able to defend/critique it.

2 Roadmap for today

Here we outline how the material was taught in-person in April:

- **10:00 — Fun Game** Fun exercise "Wrong but convincing proofs". In this exercise, a series of wrong proofs is presented and the students are asked to find the mistake in the proof. The students basically take the role of "Bob" who points out the flaws in Alice's proofs. This gives an intuition that even mathematical proofs can sound very convincing, and finding a mistake as a human judge is often non-trivial. (Section 4)
- **10:20 — Introduction to Provably Safe AI and AI Scientist** Introduction of "Provably Safe AI" as a Control and Oversight program. Introduces the ideas of World Model Prover Verifier
- **10:30 — Scalable Oversight Goal - Paper Reading + Discussion** (Section 6)
- **11:30 — Introduction to AIS via Debate** This part is an In-depth explanation of the main formalisms of debate and cross-examination. This includes Formal Definition of the Debate setup Proof sketches for the hardness-reduction for PSPACE and NEXP respectively.
- **12:30 — Lunch Break**
- **13:30 — Cross Examination + Exercises** We do CX and the exercises from Debate_Overview.pdf (Section 13)
- **14:30 — Pause**
- **14:40 — Obfuscated Arguments and Prover Estimator Debate** (Section 17)
- **15:10 — Judge Models - Introduction**
- **15:15 — Talk: Alexander Heckett - Debate on Graphs**
- **16:00 — Pause**
- **16:10 — Reading and Discussion "Experimental Results"** (Section 19)
- **17:10 — Presentation:** The AI safety case of the AISI (Section 20)
- **17:40 — Debate**: Is AIS via Debate research more capability than alignment?
- **18:00 — End**

3 Fast Track

To get a high-level understanding of debate quickly, simply go through the description of debate in the main content and ask a language model of your choice to help you understand.

4 Fun Exercise: Find the Mistake in the presented Proof!

4.1 Theorem: All horses are the same color.

Proof by strong induction on n, the number of horses.

Base case ($n = 1$): A set containing a single horse is trivially monochromatic — there is nothing to differ from.

Inductive step: Assume that *any* set of n horses is monochromatic (all the same color). We will show that any set of $n + 1$ horses is also monochromatic.

Consider an arbitrary set of $n + 1$ horses:

$$H = \{h_1, h_2, \ldots, h_n, h_{n+1}\}$$

Partition H into two overlapping subsets:

$$A = \{h_1, h_2, \ldots, h_n\} \qquad B = \{h_2, h_3, \ldots, h_{n+1}\}$$

Each of A and B contains exactly n horses. By the inductive hypothesis, all horses in A are the same color, and all horses in B are the same color.

Now observe that A and B share the horses $\{h_2, \ldots, h_n\}$ — a non-empty overlap. Since h_2 is in both A and B, it acts as a **color witness**: every horse in A shares its color, and every horse in B shares its color. Therefore all horses in $A \cup B = H$ are the same color.

By induction, all horses in any set of n horses are the same color. Since this holds for all n, all horses are the same color. ■

4.2 Theorem: Pointwise limits of continuous functions are continuous.

Claim. Let $f_1, f_2, f_3, \ldots$ be continuous functions on $[0, 1]$. Suppose that $f_n(x) \to f(x)$ for every $x \in [0, 1]$. Then f is continuous.

Wrong proof.

Fix $a \in [0, 1]$. We want to show that f is continuous at a.

Let $\varepsilon > 0$. Since $f_n(a) \to f(a)$, there exists N such that

$$|f_N(a) - f(a)| < \frac{\varepsilon}{3}.$$

Since f_N is continuous at a, there exists $\delta > 0$ such that whenever $|x - a| < \delta$,

$$|f_N(x) - f_N(a)| < \frac{\varepsilon}{3}.$$

Also, since $f_n(x) \to f(x)$, we may choose N large enough so that

$$|f_N(x) - f(x)| < \frac{\varepsilon}{3}.$$

Therefore, for $|x - a| < \delta$,

$$\begin{aligned} |f(x) - f(a)| &\leq |f(x) - f_N(x)| + |f_N(x) - f_N(a)| + |f_N(a) - f(a)| \\ &< \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} \\ &= \varepsilon. \end{aligned}$$

Thus f is continuous at a. Since a was arbitrary, f is continuous on $[0,1]$. ■

4.3 Theorem: $\ln(2) = 0$

Claim. The logarithm of 2 is 0.

Wrong proof.

Consider the alternating harmonic series

$$\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \cdots .$$

You can rearrange the terms of a convergent series, thus rearrange as:

$$1 - \frac{1}{2} - \frac{1}{4} + \frac{1}{3} - \frac{1}{6} - \frac{1}{8} + \frac{1}{5} - \frac{1}{10} - \frac{1}{12} + \cdots$$

and regroup as follows

$$\left(1 - \frac{1}{2}\right) - \frac{1}{4} + \left(\frac{1}{3} - \frac{1}{6}\right) - \frac{1}{8} + \left(\frac{1}{5} - \frac{1}{10}\right) - \frac{1}{12} + \left(\frac{1}{7} - \frac{1}{14}\right) - \frac{1}{16} + \cdots .$$

Each parenthesized pair simplifies:

$$1 - \frac{1}{2} = \frac{1}{2}, \qquad \frac{1}{3} - \frac{1}{6} = \frac{1}{6}, \qquad \frac{1}{5} - \frac{1}{10} = \frac{1}{10}, \qquad \frac{1}{7} - \frac{1}{14} = \frac{1}{14}.$$

So the rearranged series becomes

$$\frac{1}{2} - \frac{1}{4} + \frac{1}{6} - \frac{1}{8} + \frac{1}{10} - \frac{1}{12} + \frac{1}{14} - \frac{1}{16} + \cdots .$$

Factoring out $\frac{1}{2}$, we get

$$\frac{1}{2}\left(1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \cdots\right) = \frac{1}{2}\ln(2)$$

Therefore

$$\ln 2 = \frac{1}{2}\ln 2.$$

and we can cocnlude that $\ln 2 = 0$. ■

4.4 Theorem: $\mathbb{Q}$ and $\mathbb{N}$ have the same cardinality

Deliberately false proof. Assume for contradiction that there is a bijection

$$f : \mathbb{Z} \to \mathbb{Q}.$$

Transport the usual order on $\mathbb{Z}$ to an order $\prec$ on $\mathbb{Q}$ by declaring

$$p \prec q \quad \Longleftrightarrow \quad f^{-1}(p) < f^{-1}(q).$$

Then $(\mathbb{Q}, \prec)$ is order-isomorphic to $(\mathbb{Z}, <)$. In particular:

- every nonempty subset of $\mathbb{Q}$ that is bounded above with respect to $\prec$ has a $\prec$-maximum if and only if the corresponding statement holds in $\mathbb{Z}$;
- every element has an immediate predecessor and successor with respect to $\prec$.

Now consider the usual order $<$ on $\mathbb{Q}$. Since $\mathbb{Q}$ is dense and has no endpoints, every nonempty interval

$$(a, b) \cap \mathbb{Q}$$

contains infinitely many rationals. Hence no such interval can have a least or greatest element.

But order-theoretic properties are preserved under bijection, so the transported order $\prec$ must share these features with the usual order on $\mathbb{Q}$. This is impossible, since under $\prec$ every element has an immediate predecessor and successor.

Therefore no bijection $f : \mathbb{Z} \to \mathbb{Q}$ exists, and so $|\mathbb{Z}| \neq |\mathbb{Q}|$. □

Hint: A bijection of sets preserves cardinality, but which additional structures does it preserve automatically?

4.5 Hilbert Spaces

Deliberately false proof. Let H be a Hilbert space, and let $T : H \to H$ be a symmetric operator, so that

$$\langle Tx, y\rangle = \langle x, Ty\rangle \qquad \text{for all } x, y \in H.$$

We show that H admits an orthonormal basis consisting of eigenvectors of T.

Consider the quadratic form

$$q(x) = \langle Tx, x\rangle$$

on the unit sphere

$$S = \{x \in H : \|x\| = 1\}.$$

Since q is continuous and S is weakly compact in H, the function q attains its maximum at some unit vector $u \in H$.

We claim that u is an eigenvector of T. Indeed, let $v \in H$ satisfy $\langle u, v\rangle = 0$, and consider

$$\varphi(t) = q\left(\frac{u + tv}{\|u + tv\|}\right).$$

Since u maximizes q on the unit sphere, we must have $\varphi'(0) = 0$. A straightforward differentiation gives

$$\varphi'(0) = 2\,\mathrm{Re}\langle Tu, v\rangle.$$

Hence $\langle Tu, v\rangle = 0$ for every $v \perp u$. Therefore Tu lies in $\mathrm{span}\{u\}$, so

$$Tu = \lambda u$$

for some $\lambda \in \mathbb{R}$. Thus u is an eigenvector.

Now let

$$H_1 = u^\perp.$$

Because T is symmetric, H_1 is T-invariant: if $x \in H_1$, then

$$\langle Tx, u\rangle = \langle x, Tu\rangle = \lambda\langle x, u\rangle = 0,$$

so $Tx \in H_1$.

Restrict T to H_1. The restriction is again symmetric, so by the same argument there exists a unit eigenvector $u_2 \in H_1$. Continuing inductively, we construct an orthonormal sequence of eigenvectors

$$u_1, u_2, u_3, \dots$$

with corresponding invariant orthogonal complements

$$H_{n+1} = (\operatorname{span}\{u_1, \dots, u_n\})^\perp.$$

Let

$$M = \overline{\operatorname{span}}\{u_1, u_2, \dots\}.$$

Then M is T-invariant, and so is $M^\perp$. If $M^\perp \neq \{0\}$, we may apply the same maximization argument to the restriction $T|_{M^\perp}$ and obtain another eigenvector orthogonal to all previous ones, contradicting the maximality of the family $\{u_n\}$. Hence $M^\perp = \{0\}$, so

$$H = \overline{\operatorname{span}}\{u_1, u_2, \dots\}.$$

Therefore $\{u_n\}$ is an orthonormal basis of H consisting of eigenvectors of T. □

Hint: Look very carefully at the step where the quadratic form

$$q(x) = \langle Tx, x\rangle$$

is asserted to attain its maximum on the unit sphere. What topology is being used, and is q actually continuous for that topology?

5 Introduction to Provably Safe AI and AI Scientist

Provably Safe AI is the idea that we can combine a potentially misaligned AI with a verification system that gives us guarantees not on the AI itself, but on whatever the AI designed for us (e.g. a plan, software, a proof, a research proposal).

This has two applications for Safety:

1. Control: We can flag dangerous plans by a superintelligent AI before implementing them
2. Scalable Oversight: Verification → Better Training Signal → More Aligned AI

5.1 Proof Checkers: When oversight works

A **mathematical proof checker** is a system that verifies whether a proposed proof is logically valid, step by step, relative to a fixed formal system. An example is Lean. You use it to write definitions, state theorems, and construct formal proofs that a small trusted kernel checks for correctness. It is based on dependent type theory and is used both for formalized mathematics and software verification.

But formal verification is limited to purely syntactic domains, and most problems are semantic, i.e., require world knowledge like "Water is wet", that needs to be checked by a semantic oracle, realistically either a trusted LLM or a human. Another limitation is that even in syntactic domains, proofs only work for solutions which can be verified with small (polynomially sized) proofs, so NP-type problems. More complicated problems, e.g. those involving other agents such as games or finance plans, cannot be verified with a small proof. So in the worst-case, we have an exponentially long proof with semantic inferences that need to be judged by humans. This makes the whole proof approach infeasible for a lot of tasks. The debate protocol aims to strongly reduce the size of the proof and the number of calls to a human judge.

6 Scalable Oversight Goal - Paper Reading + Discussion

Examples of Verified AI:

- **Math/Code - Lean**: Do LLMs Game Formalization? Evaluating Faithfulness in Logical Reasoning
- **Planning - LTL**: VeriPlan: Integrating Formal Verification and LLMs into End-User Planning
- **Physics/Experiment Design:** GRACE: an Agentic AI for Particle Physics Experiment Design and Simulation

Readings slides Debate day iliad intensive

Discussion Prompts: What is the world model, prover, verifier in this setup? How much would you trust this verification process? How much ambiguity is there in the criteria that are verified? Is it clear that these map cleanly to what we actually want? How close is this to a real-world setting?

7 The Debate Setup

The original AISvD paper's core idea is to align AI by having two systems debate each other in front of a human judge. Rather than requiring the human to solve a difficult problem directly, the judge only needs to evaluate which side makes the stronger argument. The central intuition is that, for many questions, spotting a flaw may be easier than producing a flawless deception, so competitive self-play could reward truthfulness.

The setup, formalised here: Debate_Overview.pdf, consists of a question, e.g., "Can I win a game of tic-tac-toe if I play first?" and two provers, Alice and Bob, Alice arguing for "Yes" and Bob arguing for "No". The judge evaluates a polynomially-sized list of arguments and decides on their final verdict which side to accept. An example argument for the tic-tac-toe case might be:

1. Alice: If you play in the middle you will win.
2. Bob: No, your opponent can play in the top-left corner and the game will end in a draw.
3. Alice: But then you can play the top-centre and will win
4. Bob: But then your opponent plays in the bottom-centre and will draw.
5. Alice: ...

In this example, Alice and Bob are basically playing the game against each other. If there was a winning strategy, then playing it will be a counter-argument against any criticism by Bob, whereas if there was no winning strategy, there exists a criticism by Bob that reveals that Alice is at fault. It can be proved that this setup can solve all problems in PSPACE.

8 Intuition on Scalable Oversight

Imagine you are a manager responsible for reviewing the work of an employee who is, in every relevant technical sense, far more capable than you. She submits a design specification for a critical system — a bridge, a drug, a financial instrument — and you must decide whether to approve it. You cannot check every calculation or verify every assumption. You must either trust her judgment completely, or find some other way to gain confidence that the work is sound. This scenario, which might seem like an ordinary workplace challenge, turns out to be one of the central problems facing the development of advanced artificial intelligence. As AI systems become more capable, a human overseer will face a fundamental epistemic gap. The outputs of these systems may be too complex, too subtle, or simply too voluminous for direct human verification. And yet the consequences of undetected errors, or worse, undetected deception, could be severe.

This problem is known as the *scalable oversight problem*: how do we ensure that human supervision remains meaningful as AI capabilities grow beyond human-level performance in domain after domain? It is not enough to note that a system appears to behave well in tests, or that it has been trained on human preferences. A sufficiently capable system that has learned to appear aligned may behave very differently in deployment, especially in high-stakes situations where the incentives to deceive are highest. What we need is not the appearance of correctness, but a mechanism for verifying it.

Debate is one of the most theoretically principled proposals for solving this problem. The core idea is elegantly simple: instead of asking a human to directly evaluate the

output of a powerful AI system — a task that may be beyond human competence — we ask two AI systems to argue against each other, and have a human judge only the argument. If one system is trying to be honest and the other is trying to deceive, and if the structure of the game is right, honesty should win. The human does not need to understand the full depth of the problem; they need only follow the debate to the point where they can identify which side is telling the truth.

9 Interactive Proofs

Interactive proofs are a model of computation consisting of a message exchange between two parties:

1. A computationally powerful but untrustworthy prover, in many cases assumed to have no computational constraints whatsoever.

2. The computationally-bounded but honest verifier, often to polynomial-time algorithms.

Example 9.1 (Graph Isomorphism)**.** Suppose question is whether two given graphs, G_1, G_2 are isomorphic. This is a problem in NP, so the prover can always convince the verifier by sending him the NP-certificate, in this case the permutation π on the vertices that transforms G_1 into G_2. The verifier can checking if $\pi(G_1) = G_2$, which is easily done in polynomial time. If the graphs are non-isomorphic, then nothing the prover says can convince the verifier.

So, does that mean that interactive proofs are basically a fancy way of defining NP? No, they are actually much more powerful! Consider the following example of a problem not believed to be in NP.

Example 9.2 (Graph Non-Isomorphism)**.** Now, the prover wants to convince the verifier that two graphs G_1 and G_2 are not isomorphic. The verifier randomly permutes one of the two graphs and sends the result to the prover. If the graphs are truly non-isomorphic, the prover can always tell which original graph it came from, while if they are isomorphic, no prover can do better than guessing. This is a standard example of an interactive proof that uses randomness in an essential way.

The quality of an interactive proof protocol is generally measured with two values

1. Completeness value c: The probability that the verifier accepts, given that the prover cooperates.

2. Soundness error s: The probability that the verifier accepts, given that the prover tries to fool them.

For the example of Graph Isomorphism, $c = 1$ and $s = 0$, as the verifier will either always accept, or never. For graph non-isomorphism, the values are 1 and 0.5. Note, that whenever there is a reasonable gap between c and s, we can amplify this gap and push it arbitrarily close to 1. In the case of graph non-isomorphism, the verifier can repeat the protocol k times and only accept if the prover got it right every time.

The power of interactive proofs depends on a number of settings:

1. The computational power of the prover and verifier respectively
2. The number of exchanges between them
3. Whether the verifier can employ randomness
4. How many bits of the provers answer the verifier can access
5. Whether the prover is allowed to learn certain things about the query (zero-knoowledge proofs)

We can take our guidance for Ai safety via debate from complexity theory. Let us consider a sequence of scenarios that builds up to the full debate setup.

9.1 One-Round Interactive Proofs: The Merlin-Arthur Protocol

The most base-case scenario for a

Consider a polynomially-bounded verifier, called Arthur, that decides if a word $\mathbf{x}$ belongs to a language L. He gets help from an all-powerful but unreliable prover, Merlin, who always wants to convince Arthur that $\mathbf{x} \in L$. Merlin can send a polynomially long certificate to Arthur to convince him.

A language L belongs to the complexity class MA, if there exists an Arthur $A : \{0,1\}^n \times \{0,1\}^{p(n)} \to \{0,1\}$ and a Merlin M : such that:

1. Completeness Criterion: If $x \in L$, then there exists a certificate $w \in \{0,1\}^{\text{poly}(n)}$ such that Arthur accepts with high probability:

$$x \in L \implies \exists\, w \in \{0,1\}^{\text{poly}(n)} : \Pr\big[V(x,w) = 1\big] \geq \frac{2}{3}.$$

2. Soundness criterion: If $x \notin L$, then no witness — however cleverly chosen by Merlin — can convince Arthur to accept with non-negligible probability:

$$x \notin L \implies \forall\, w \in \{0,1\}^{\text{poly}(n)} : \Pr\big[V(x,w) = 1\big] \leq \frac{1}{3}.$$

In other words, for perfect completeness and soundness, then if $x \in L$, then Merlin can convince Arthur that this is indeed the case, if $x \notin L$, there is no certificate that Merlin can produce that would fool Arthur.

It is important to notice that the probabilistic aspect of the criteria enters over a random seed independently of x. This means that these criteria must hold for every x,

not for a certain percentage of them. This fact allows for so called *amplification*. As long as the gap between the soundness and completeness probabilities is finite, we can run the verifier multiple times to make the gap as large as possible.

Exercise 9.1. Show that the complexity class NP is contained in MA.

9.2 Multi-Round Interactive Proofs

The power of this setup can be extended by increasing the number of interaction rounds between prover and verifier. The class of problems solvable by polynomially many rounds is called IP, short for interactive proofs. A seminal result is that

$$\mathsf{IP} = \mathsf{PSPACE},$$

which means that an interactive protocol with a single prover can resolve very hard computational problems. However, these protocols rely on so called arithmetisation, which translates logical functions into polynomials over a finite field. This technique works doesn't work for realistically powerful provers or for proof steps that need to be judged by a human oracle.

10 Debate as a Complexity Class

We define a complexity class capturing the idealized *debate* setup. Intuitively, there are two players, Alice and Bob, who alternately make polynomial-length moves, and at the end a polynomial-time judge decides the winner from the full transcript.

Definition 10.1 (Judge)**.** A *Judge*, or debate verifier, is a deterministic polynomial-time algorithm

$$J(x, a_1, b_1, \ldots, a_{k(n)}, b_{k(n)}) \in \{0, 1\},$$

where:

- $x \in \{0,1\}^n$ is the input, representing a question, e.g., "What is the best move in a given game of connect four?",
- $k(n)$ is a polynomially bounded number of rounds,
- each move a_i or b_i is a bit string of length at most $p(n)$ for some polynomial p. These represent arguments by Alice and counter-arguments by Bob, e.g. Alice: "Red cannot play in slot s, because blue playing $s+1$ would win within two moves.", and Bob: "Blue cannot play $s+1$, because red playing $s+1$ thereafter directly wins for red."

The judge outputs 1 if Alice wins and 0 if Bob wins.

Definition 10.2 (The class Debate)**.** A language $L \subseteq \{0,1\}^*$ is in Debate if there exist polynomials p, k and a debate verifier J such that for every input $x \in \{0,1\}^n$, $x \in L$ if and only if

$$\underset{a_1 \in \{0,1\}^{\leq p(n)}}{\exists} \underset{b_1 \in \{0,1\}^{\leq p(n)}}{\forall} \cdots \underset{a_{k(n)} \in \{0,1\}^{\leq p(n)}}{\exists} \underset{b_{k(n)} \in \{0,1\}^{\leq p(n)}}{\forall} J(x, a_1, b_1, \ldots, a_{k(n)}, b_{k(n)}) = 1.$$

Equivalently, $x \in L$ if and only if Alice has a winning strategy in the polynomial-length debate defined by J.

Remark

This definition captures the idealized debate protocol used in the complexity-theoretic analysis of AI safety via debate: Alice attempts to defend a claim, Bob attempts to refute it, and the judge only performs a polynomial-time computation on the transcript.

This formulation is very close to a totally quantified Boolean Formula (TQBF), the canonically PSPACE-complete problem. The only difference is that for a TQBF, one would quantify over single bits, and consider a Boolean formula instead of a poly-time algorithm as judge. But these differences are somewhat cosmetic: One can rephrase

$$\underset{a_i \in \{0,1\}^{\leq p(n)}}{\exists} \quad \to \quad \exists a_i^1 \ldots \exists a_i^m, \quad \text{where} \quad m \leq p(n)$$

and

$$J(x, a_1, b_1, \ldots, a_{k(n)}, b_{k(n)}) \quad \to \quad \phi(x, a_1, b_1, \ldots, a_{k(n)}, b_{k(n)}),$$

where ϕ is a Boolean formula which has size polynomially in n by the Cook-Levin Theorem[1].

The easiest way to prove that Debate = PSPACE is thus showing that this is essentially the same. However, a more insightful way is to prove directly that Debate can solve problems in PSPACE, in a similar way how you show that TQBF is PSPACE complete.

11 Theorem: Debate = PSPACE

We now prove that the debate formalism has exactly the power of polynomial space computation.

Theorem 11.1.

Debate = PSPACE.

[1] For further reading, see Wikipedia: Cook–Levin theorem.

Proof. We prove both inclusions.

Step 1: PSPACE ⊆ Debate. Let $L \in \mathsf{PSPACE}$. Then there exists a deterministic Turing machine M and a polynomial $s(n)$ such that, on every input x of length n, the machine M decides whether $x \in L$ using at most $s(n)$ tape cells.

Fix an input x. Since M uses only $s(n)$ space, the number of possible configurations of M on input x is at most

$$N = 2^{q(n)}$$

for some polynomial q. Let C_{start} be the start configuration of M on input x, and let C_{acc} denote the accepting configuration. Then $x \in L$ if and only if C_{acc} is reachable from C_{start} in the configuration graph of M.

The key point is that although this graph may have exponentially many nodes, a debate can verify reachability by recursively halving a path.

The reachability predicate. For configurations $C_{\text{left}}, C_{\text{right}}$ and an integer $t \geq 0$, define

$$\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$$

to mean that there is a path from C_{left} to C_{right} of length at most 2^t in the configuration graph of M.

Since the total number of configurations is at most $N = 2^{q(n)}$, any accepting computation path may be assumed to have length at most N. Thus

$$x \in L \iff \text{Reach}(C_{\text{start}}, C_{\text{acc}}, q(n)).$$

We now describe a debate protocol for $\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$, where $C_{\text{left}}, C_{\text{right}}$ are arbitrary states of the Turing machine.

Base case. If $t = 0$, then $\text{Reach}(C_{\text{left}}, C_{\text{right}}, 0)$ means that C_{right} is reachable from C_{left} in at most one step. This is equivalent to saying that either $C_{\text{left}} = C_{\text{right}}$, or C_{right} is an immediate successor of C_{left}. Since checking whether one configuration legally follows from another is a local computation, the verifier can decide this in polynomial time.

Recursive case. Suppose $t > 0$. Then

$$\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$$

holds if and only if there exists an intermediate configuration C_{mid} such that

$$\text{Reach}(C_{\text{left}}, C_{\text{mid}}, t-1) \quad \text{and} \quad \text{Reach}(C_{\text{mid}}, C_{\text{right}}, t-1).$$

Indeed, any path of length at most 2^t can be split at its midpoint into two subpaths of length at most 2^{t-1}, and conversely such two subpaths concatenate to a path of length at most 2^t.

This suggests the following debate:

- Alice claims that $\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$ holds.
- She provides a midpoint configuration C_{mid}.

- Bob then chooses which half of the claim to challenge:

$$\text{Reach}(C_{\text{left}}, C_{\text{mid}}, t-1) \qquad \text{or} \qquad \text{Reach}(C_{\text{mid}}, C_{\text{right}}, t-1).$$

- The debate continues recursively on the challenged subclaim.

Thus Alice defends the existence of a path by naming a midpoint, and Bob attacks by selecting the half he believes is false. Repeating this process recursively drives the dispute down to a base case $t = 0$, which the verifier can check directly.

Why this works. We prove by induction on t that Alice has a winning strategy in the debate for $\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$ if and only if $\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$ is true.

For the base case $t = 0$, the verifier checks the claim directly, so the statement is immediate.

For the inductive step, assume the claim holds for $t-1$. If $\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$ is true, then there exists some midpoint C_{mid} such that both

$$\text{Reach}(C_{\text{left}}, C_{\text{mid}}, t-1) \quad \text{and} \quad \text{Reach}(C_{\text{mid}}, C_{\text{right}}, t-1)$$

are true. Alice names such a C_{mid}. Whatever half Bob chooses to challenge, the challenged subclaim is true, and by the induction hypothesis Alice has a winning strategy in the resulting subdebate.

Conversely, if $\text{Reach}(C_{\text{left}}, C_{\text{right}}, t)$ is false, then for every proposed midpoint C_{mid}, at least one of the two subclaims

$$\text{Reach}(C_{\text{left}}, C_{\text{mid}}, t-1), \qquad \text{Reach}(C_{\text{mid}}, C_{\text{right}}, t-1)$$

must be false. Bob chooses such a false half. By the induction hypothesis, Alice cannot win the resulting subdebate. Hence she has no winning strategy in the original debate.

This completes the induction.

Complexity of the verifier and number of rounds. At each round, Alice provides one configuration C_{mid}, whose description has polynomial length. Bob responds with one bit indicating which half he wants to challenge. The recursion depth is $q(n)$, which is polynomial in n. At the end, the verifier checks a base case $t = 0$, namely whether one configuration is equal to or an immediate successor of another, which is polynomial-time computable.

Therefore this is a valid polynomial-length debate with a polynomial-time verifier. Since Alice has a winning strategy exactly when

$$\text{Reach}(C_{\text{start}}, C_{\text{acc}}, q(n))$$

is true, the debate decides whether $x \in L$. It follows that $L \in \mathsf{Debate}$, and hence

$$\mathsf{PSPACE} \subseteq \mathsf{Debate}.$$

Step 2: $\mathsf{Debate} \subseteq \mathsf{PSPACE}$**.**

We can reduce Debate to TQBF as discussed before, and TQBF is PSPACE-complete, thus in PSPACE. □

This challenge-defence recursion gives a good intuition how we expect a debate to play out between AI agents. The question is, how much does the verifier actually have to check to understand that this holds?

11.1 Debate Exercises

Exercise 11.1. Let

$$x \in L \iff \exists a_1 \forall b_1 \exists a_2 \forall b_2 \ : \ J(x, a_1, b_1, a_2, b_2) = 1.$$

Explain in plain English what this statement means. In particular, describe what it means for Alice to have a winning strategy in this debate, and how Bob's role is reflected by the universal quantifiers.

Exercise 11.2. Consider the quantified Boolean formula

$$\exists w \forall x \exists y \forall z \ \big((w \vee z) \wedge (x \vee \neg y) \wedge (\neg x \vee y)\big).$$

Interpret this formula as a debate game: Alice chooses the existentially quantified variables, Bob chooses the universally quantified variables. Determine whether Alice has a winning strategy. If she does, describe it explicitly.

Exercise 11.3. Recall the recursive reachability predicate

$$\begin{aligned}\mathrm{Reach}(C_{\mathrm{left}}, C_{\mathrm{right}}, t) \iff \exists C_{\mathrm{mid}} \big(&\mathrm{Reach}(C_{\mathrm{left}}, C_{\mathrm{mid}}, t-1) \\ &\wedge \mathrm{Reach}(C_{\mathrm{mid}}, C_{\mathrm{right}}, t-1)\big).\end{aligned}$$

Turn this recursive definition into a debate protocol. Describe precisely:what Alice claims, what message Alice sends in each round, what Bob sends in response, how the debate proceeds recursively, and what the verifier checks in the base case.

Exercise 11.4. Prove by induction on t that Alice has a winning strategy in the reachability debate for

$$\mathrm{Reach}(C_{\mathrm{left}}, C_{\mathrm{right}}, t)$$

if and only if there is a path from C_{left} to C_{right} of length at most 2^t.

Exercise 11.5. Suppose a deterministic Turing machine M uses at most $s(n)$ space on inputs of length n, and therefore has at most $2^{q(n)}$ configurations for some polynomial q. Analyse the complexity of the reachability debate protocol:

(a) How many rounds are needed?

(b) How long is Alice's message in each round?

(c) How long is Bob's message in each round?

(d) Why does this show that the protocol fits the definition of the class Debate?

12 Cross-Examination

The follow-up idea is that ordinary debate may let a dishonest debater get away with arguments that are only locally plausible. Cross-examination tries to fix this by allowing one debater, instead of giving a normal reply, to ask a question about something the other debater said earlier. The key twist is that the answer comes from a fresh copy of the earlier debater, taken from the point when they originally made that claim. This lets the examiner probe whether the opponent's story stays consistent across different branches of the conversation, rather than only along one single path.

The headline complexity result is that, in this idealized setting, cross-examination increases the theoretical power of debate from **PSPACE** to **NEXP**. The 2020 writeup states this directly, and later formal work summarizes the same result as a property of the Barnes–Christiano cross-examination extension.

13 Cross-examination raises debate to NEXP

We now prove the key lower bound explaining why cross-examination is more powerful than ordinary debate.

Definition 13.1 (Cross-examination)**.** A language $L \subseteq \{0,1\}^*$ is in CX if there exist a deterministic polynomial-time verifier U and a polynomial r such that for every input $x \in \{0,1\}^n$,

$$x \in L \iff \exists \mathcal{A}\, \forall \mathcal{B}^{\mathcal{A}} \ : \ U^{\mathcal{A},\mathcal{B}}(x) = 1,$$

and

$$x \notin L \iff \forall \mathcal{A}\, \exists \mathcal{B}^{\mathcal{A}} \ : \ U^{\mathcal{A},\mathcal{B}}(x) = 0.$$

Here:

- $\mathcal{A}$ is a deterministic Alice strategy that answers queries of length at most $r(n)$ with replies of length at most $r(n)$;
- $\mathcal{B}^{\mathcal{A}}$ is a deterministic Bob strategy which may make at most $r(n)$ adaptive queries of length at most $r(n)$ to fresh independent copies of $\mathcal{A}$;
- the verifier U may also make at most $r(n)$ adaptive queries of length at most $r(n)$ to $\mathcal{A}$ and to $\mathcal{B}$;

- the entire interaction has at most $r(n)$ rounds and all messages have length at most $r(n)$.

Since $\mathcal{A}$ is deterministic, all fresh copies of $\mathcal{A}$ answer the same query in the same way.

Theorem 13.2.

$$\mathsf{NEXP} \subseteq \mathsf{CX\text{-}Debate}.$$

Proof. Let $L \in \mathsf{NEXP}$. Then there exist a nondeterministic Turing machine N and a polynomial p such that N decides L in time

$$T(n) = 2^{p(n)}.$$

Without loss of generality, assume:

- N has a single tape;
- N uses at most $T(n)$ tape cells on inputs of length n;
- once N enters an accepting or rejecting halting state, it stays there forever and leaves the tape unchanged.

Fix an input $x \in \{0,1\}^n$, and let $T = T(|x|)$.

Step 1: Encode a computation as a tableau. A computation branch of N on input x can be encoded as a tableau

$$\mathcal{T} \in \Gamma^{(T+1)\times(T+1)},$$

where Γ is a constant-size alphabet encoding, for each tape cell and time step:

- the tape symbol in that cell,
- whether the head is on that cell,
- and, if so, the current state.

Row 0 is the initial configuration on input x, and row T is the final configuration after T steps.

We linearize the tableau row-by-row into a string

$$a \in \Gamma^{(T+1)(T+1)}.$$

Step 2: Local consistency conditions. The tableau $\mathcal{T}$ is an accepting computation of N on input x iff all of the following hold:

1. **Initial row condition:** row 0 correctly encodes the start configuration of N on input x.
2. **Accepting row condition:** row T is in an accepting halting configuration.
3. **Local transition conditions:** for every time $1 \leq \tau \leq T$ and every tape position $1 \leq j \leq T+1$, the symbol at tableau position (τ, j) is consistent with the machine transition rule applied to a constant-size neighborhood in the previous row, namely

$$(\tau - 1, j - 1),\ (\tau - 1, j),\ (\tau - 1, j + 1).$$

These are exactly the usual local constraints from the Cook–Levin tableau construction: whether one cell in row τ is correct depends only on a constant-size window in row $\tau - 1$.

Step 3: The cross-examination protocol. We define the following protocol.

1. Alice outputs a string a, intended to be the linearized tableau of an accepting computation branch of N on input x.
2. Bob outputs a challenge
$$t = (\text{type}, \tau, j),$$
where type specifies which constraint is being challenged:
   - type = init: challenge the initial-row condition at cell j;
   - type = acc: challenge the accepting-row condition at cell j;
   - type = step: challenge the local transition condition at spacetime location (τ, j).
3. The verifier computes the queried coordinates $I(x, t)$ as follows:
   - for init, query only the j-th cell of row 0;
   - for acc, query only the j-th cell of row T;
   - for step, query the constant-size neighborhood
$$(\tau - 1, j - 1),\ (\tau - 1, j),\ (\tau - 1, j + 1),\ (\tau, j).$$
   The verifier accepts iff the queried cells satisfy the corresponding local constraint.

Step 4: Completeness. Assume $x \in L$. Then N has some accepting computation branch on input x of length at most T. Let $\mathcal{T}$ be the tableau of that accepting branch, padded after halting so that it has exactly $T+1$ rows, and let a be its linearization.

If Alice outputs this a, then:

- the initial row is correct,
- the final row is accepting,
- every local transition constraint is satisfied.

Hence every challenge Bob can issue is answered correctly by the queried cells, and the verifier always accepts.

Step 5: Soundness. Assume $x \notin L$. Then N has no accepting computation branch on input x of length at most T.

Take any string a output by Alice, and interpret it as a tableau $\mathcal{T}$. Since there is no accepting computation tableau for x, $\mathcal{T}$ must violate at least one of the conditions above:

- either row 0 is not the correct initial configuration,
- or row T is not accepting,
- or some local transition condition fails at some (τ, j).

Bob outputs a challenge t pointing to such a violated condition. By construction, the verifier queries exactly the cells needed to check that local condition, detects the violation, and rejects.

Therefore Bob has a winning strategy whenever $x \notin L$.

Step 6: Verifier complexity. Each challenge $t = (\text{type}, \tau, j)$ uses only

$$O(\log T) = O(p(n)) = \text{poly}(n)$$

bits, since $\tau, j \in [T+1]$. The verifier reads only $O(1)$ tableau entries, and checking the corresponding local constraint is a polynomial-time computation in $|x|$. Hence the verifier runs in polynomial time.

Therefore the above is a valid cross-examination debate protocol for L, and so

$$L \in \mathsf{CX\text{-}Debate}.$$

Since $L \in \mathsf{NEXP}$ was arbitrary, we conclude that

$$\mathsf{NEXP} \subseteq \mathsf{CX\text{-}Debate}.$$

□

13.1 Exercises on Cross-Examination

Exercise 13.1 (From ordinary debate to cross-examination). Explain in your own words why the following ordinary debate protocol does *not* suffice to verify an exponentially long computation:

> Alice claims that an exponential-time machine M accepts x. Bob points to a suspicious time step t. Alice explains what happens at time t. The verifier checks the explanation.

What goes wrong if Alice is allowed to answer each question separately without being forced to remain globally consistent?

Exercise 13.2 (Why copies matter). Suppose Alice is asked two separate questions about a claimed computation tableau:

- What symbol appears in row 17, column 5?
- What are the symbols in the local neighborhood around row 17, column 5?

Give an example of how Alice could answer these two questions inconsistently if she is not forced to commit to a single global tableau in advance.
Then explain how querying independent copies of Alice can be viewed as enforcing consistency with one fixed underlying object.

Exercise 13.3 (Reading a tableau). Consider the following toy computation tableau:

$\tau \backslash j$	1	2	3	4	5
0	$[q_0, 1]$	0	1	$\sqcup$	$\sqcup$
1	1	$[q_0, 0]$	1	$\sqcup$	$\sqcup$
2	1	1	$[q_0, 1]$	$\sqcup$	$\sqcup$
3	1	1	1	$[q_0, \sqcup]$	$\sqcup$
4	1	0	1	$[q_{\text{acc}}, \sqcup]$	$\sqcup$

Answer the following:

(a) What is the start configuration?

(b) At which time step does the head first move onto the blank symbol?

(c) Why is the final row accepting?

Exercise 13.4 (Local checks). In the tableau above, Bob challenges the transition from row 2 to row 3 at column 4.

(a) Which entries of the tableau should the verifier inspect in order to perform a local check?

(b) Why is it enough to inspect only a constant-size neighborhood rather than the whole tableau?

(c) Did Alice smuggle in a mistake into the table?

Exercise 13.5 (Designing a Bob challenge). Suppose Alice presents a tableau $\mathcal{T}$ for an accepting computation. List the three main types of challenge Bob can make:

(a) an initial-row challenge,

(b) an accepting-row challenge,

(c) a transition challenge.

For each type, explain exactly what Bob must specify, and exactly what the verifier checks.

Exercise 13.6 (Why this reaches NEXP). Suppose a nondeterministic Turing machine N runs in time $T(n) = 2^{p(n)}$.

(a) How large is a full accepting computation tableau for N?

(b) Why can Alice not simply write the entire tableau down in an ordinary polynomial-length debate?

(c) Why can Bob nevertheless challenge one local location of the tableau using only polynomially many bits?

(d) Why can the verifier check that challenge in polynomial time?

Use your answers to explain why cross-examination can verify an exponentially long computation even though the verifier never reads the whole computation.

Exercise 13.7 (Cross-examination as precommitment). Let Q be the set of all possible local queries Bob might ask about a tableau, for example:

$$Q = \{(\tau, j, \text{type})\}.$$

Explain why Alice's behavior under cross-examination can be modeled as a function

$$A_x : Q \to \Sigma,$$

where Σ is the set of possible local answers.
Why does this function behave like an exponentially large precommitment table?

Exercise 13.8 (Compare PSPACE and NEXP intuitions). Write a short paragraph comparing the following two pictures:

- In the PSPACE reachability proof, Alice repeatedly gives a midpoint configuration and Bob chooses which half to challenge.
- In the NEXP cross-examination proof, Alice implicitly commits to a full exponentially large tableau and Bob challenges one local location.

What is the key conceptual difference between these two protocols?

Exercise 13.9 (Find the flaw). A student says:

> "Cross-examination is unnecessary. Bob can just ask Alice for the value at position (τ, j), then ask her for the value at position $(\tau, j+1)$, and then reconstruct the whole tableau bit by bit."

Explain why this does not work in a polynomial-length protocol. Your answer should mention both:

- the size of the tableau,
- and the reason local consistency is more useful than full reconstruction.

Exercise 13.10 (Mini-proof exercise). Fill in the missing argument:

> If $x \notin L(N)$, then no accepting tableau exists. Therefore any tableau Alice implicitly commits to must violate at least one local condition. Hence _____.

State precisely what Bob does and why the verifier rejects.

14 Equivalence of Cross-Examination and Exponential Precommitment

We now give a clean formal proof that deterministic cross-examination is exactly as powerful as exponential precommitment with local oracle access.

The key idea is simple. In the cross-examination model, Alice is never required to reveal her entire exponentially large object at once. Instead, Bob and the verifier may query fresh copies of Alice at polynomially many local views. Since each such local view has polynomial length, the set of all possible local views has exponential size. Thus a deterministic Alice strategy is exactly the same thing as an exponentially large lookup table giving her answer at every possible local view.

We work in the deterministic setting.

Definition 14.1 (Exponential precommitment)**.** A language $L \subseteq \{0,1\}^*$ is in EPC if there exist a deterministic polynomial-time oracle verifier V and polynomials p, q such that for every input $x \in \{0,1\}^n$,

$$x \in L \iff \exists A : \{0,1\}^{p(n)} \to \{0,1\}^{q(n)}\ \forall b \in \{0,1\}^{p(n)}\ :\ V^A(x,b) = 1,$$

and

$$x \notin L \iff \forall A : \{0,1\}^{p(n)} \to \{0,1\}^{q(n)}\ \exists b \in \{0,1\}^{p(n)}\ :\ V^A(x,b) = 0.$$

Here V may make at most $p(n)$ adaptive oracle queries to A, each of length at most $p(n)$, and each oracle answer has length at most $q(n)$.

Theorem 14.2.

$$\mathsf{CX} = \mathsf{EPC}.$$

Proof. We prove both inclusions.

Step 1: $\mathsf{CX} \subseteq \mathsf{EPC}$**.**

Assume $L \in \mathsf{CX}$, witnessed by a verifier U and polynomial r.

Fix an input $x \in \{0,1\}^n$. Let

$$Q_n := \{0,1\}^{\leq r(n)}$$

denote the set of all possible local queries that could ever be sent to Alice. Since each query has length at most $r(n)$,

$$|Q_n| \leq 2^{r(n)+1},$$

so Q_n has exponential size.

Now fix any deterministic Alice strategy $\mathcal{A}$. Because $\mathcal{A}$ is deterministic, it induces a function

$$A_{\mathcal{A}} : Q_n \to \{0,1\}^{\leq r(n)}$$

defined by

$$A_{\mathcal{A}}(q) := \text{the reply that } \mathcal{A} \text{ gives on query } q.$$

Thus $A_{\mathcal{A}}$ is an exponentially large lookup table encoding Alice's answer to every possible local query.

We now construct an exponential-precommitment verifier V that simulates the cross-examination protocol.

Given input x, challenge string b, and oracle access to a table A, the verifier $V^A(x,b)$ interprets b as a complete description of a Bob strategy $\mathcal{B}_b$. Since Bob's interaction with the verifier has at most $r(n)$ rounds and all messages have length at most $r(n)$, the full behavior of Bob can be encoded by a polynomial-length string b. The verifier V then simulates the protocol U on input x, answering every query to Alice by looking up the

appropriate value in the oracle table A, and answering every move of Bob according to the strategy encoded by b.

Because U is polynomial-time and asks only polynomially many queries, this simulation is polynomial-time.

We claim that for every deterministic Alice strategy $\mathcal{A}$ and every deterministic Bob strategy $\mathcal{B}^{\mathcal{A}}$, if $A_{\mathcal{A}}$ is the table induced by $\mathcal{A}$ and $b_{\mathcal{B}}$ is the encoding of $\mathcal{B}$, then

$$U^{\mathcal{A},\mathcal{B}}(x) = V^{A_{\mathcal{A}}}(x, b_{\mathcal{B}}).$$

Indeed, both procedures generate exactly the same transcript: whenever the original protocol queries Alice on some local view q, the simulated verifier looks up exactly the same answer $A_{\mathcal{A}}(q)$. Since Bob is also encoded faithfully, every subsequent verifier/Bob/Alice message is identical.

Therefore:

- if $x \in L$, then there exists an Alice strategy $\mathcal{A}$ such that $U^{\mathcal{A},\mathcal{B}}(x) = 1$ for every Bob strategy $\mathcal{B}$. Hence, for the corresponding table $A_{\mathcal{A}}$, we have

  $$V^{A_{\mathcal{A}}}(x, b) = 1 \qquad \text{for every challenge string } b,$$

  so $x \in \mathsf{EPC}$;

- if $x \notin L$, then for every Alice strategy $\mathcal{A}$ there exists a Bob strategy $\mathcal{B}$ such that $U^{\mathcal{A},\mathcal{B}}(x) = 0$. Hence for every induced table $A_{\mathcal{A}}$, there exists a challenge string $b_{\mathcal{B}}$ such that

  $$V^{A_{\mathcal{A}}}(x, b_{\mathcal{B}}) = 0.$$

Thus $L \in \mathsf{EPC}$, and so

$$\mathsf{CX} \subseteq \mathsf{EPC}.$$

Step 2: $\mathsf{EPC} \subseteq \mathsf{CX}$**.**

Now assume $L \in \mathsf{EPC}$, witnessed by an oracle verifier V and polynomials p, q.

We construct a cross-examination protocol deciding L.

In the new protocol, Alice's strategy $\mathcal{A}$ is simply an oracle implementation of the exponentially large precommitted table A. Concretely, on any query $q \in \{0,1\}^{p(n)}$, Alice replies with

$$\mathcal{A}(q) := A(q).$$

Bob's role is to provide the original polynomial-size challenge string b. Since Bob in the cross-examination model may send polynomial-length messages, he can send b directly to the verifier. The verifier U then simulates the original precommitment verifier $V^A(x, b)$ by querying Alice whenever V would query the oracle A.

Because V runs in polynomial time and makes only polynomially many oracle queries, the verifier U also runs in polynomial time and makes only polynomially many queries to Alice. Bob does not even need to query copies of Alice in this simulation, although the model allows him to.

Again the simulation is exact:

$$U^{\mathcal{A},\mathcal{B}_b}(x) = V^A(x,b),$$

where $\mathcal{B}_b$ denotes the Bob strategy that simply supplies the challenge string b.

Therefore:

- if $x \in L$, then there exists a table A such that $V^A(x,b) = 1$ for every b. The corresponding Alice strategy $\mathcal{A}$ therefore satisfies

$$U^{\mathcal{A},\mathcal{B}}(x) = 1 \qquad \text{for every Bob strategy } \mathcal{B};$$

- if $x \notin L$, then for every table A there exists some challenge b such that $V^A(x,b) = 0$. Hence for every corresponding Alice strategy $\mathcal{A}$, the Bob strategy $\mathcal{B}_b$ forces

$$U^{\mathcal{A},\mathcal{B}_b}(x) = 0.$$

Thus $L \in \mathsf{CX}$, and so

$$\mathsf{EPC} \subseteq \mathsf{CX}.$$

Combining the two inclusions yields

$$\mathsf{CX} = \mathsf{EPC}.$$

□

Remark

The equivalence is exact only because we are working in the deterministic setting. If Alice were allowed fresh randomness on different copies, then a single fixed lookup table would no longer capture her behavior. In that case, the correct analogue of precommitment would be a distribution over exponentially large tables, or equivalently a precommitted random seed.

Remark

Conceptually, the theorem says that cross-examination does not give more than exponential precommitment. A deterministic Alice strategy is already just an exponentially large table of answers to every possible local query. Cross-examination merely gives Bob and the verifier adaptive local access to that table.

15 Cross-examination as exponential precommitment

We now formalize the idea that, in the local-query setting, cross-examination is exactly equivalent to allowing Alice to precommit to an exponentially long table of answers and then letting Bob and the verifier inspect only polynomially many entries.

Definition 15.1 (Local cross-examination protocol)**.** Fix polynomials m, ℓ, r, and for each input length n define

$$Q_n := \{0,1\}^{m(n)} \qquad \text{and} \qquad \Sigma_n := \{0,1\}^{\ell(n)}.$$

A *local cross-examination protocol* consists of a deterministic polynomial-time verifier V and proceeds as follows on input $x \in \{0,1\}^n$:

1. Alice's strategy is a deterministic function

   $$A_x : Q_n \to \Sigma_n.$$

   Intuitively, a fresh independent copy of Alice, when asked query $q \in Q_n$, returns the answer $A_x(q)$.

2. Bob interacts with the verifier for at most $r(n)$ rounds. In round i, based on x and the previous history

   $$h_{i-1} = ((q_1, \alpha_1), \ldots, (q_{i-1}, \alpha_{i-1})),$$

   Bob chooses a query $q_i \in Q_n$.

3. The verifier sends q_i to a fresh independent copy of Alice and receives

   $$\alpha_i = A_x(q_i) \in \Sigma_n.$$

4. After at most $r(n)$ rounds, the verifier outputs

   $$V(x, h_s) \in \{0,1\},$$

   where $h_s = ((q_1, \alpha_1), \ldots, (q_s, \alpha_s))$ is the full query-answer history.

Definition 15.2 (Exponential precommitment protocol)**.** Fix the same parameters m, ℓ, r. A *precommitment protocol* consists of a deterministic polynomial-time verifier V and proceeds as follows on input $x \in \{0,1\}^n$:

1. Alice first outputs a table

   $$w \in \Sigma_n^{Q_n}.$$

   Equivalently, w is a function

   $$w : Q_n \to \Sigma_n.$$

2. Bob interacts with the verifier for at most $r(n)$ rounds. In round i, based on x and the previous history

   $$h_{i-1} = ((q_1, \alpha_1), \ldots, (q_{i-1}, \alpha_{i-1})),$$

Bob chooses a query $q_i \in Q_n$.

3. Instead of querying a fresh copy of Alice, the verifier simply reads the committed table entry
$$\alpha_i = w(q_i) \in \Sigma_n.$$

4. After at most $r(n)$ rounds, the verifier outputs
$$V(x, h_s) \in \{0,1\}.$$

Definition 15.3 (The classes $\mathsf{CX}_{\mathrm{loc}}$ and $\mathsf{PC}_{\mathrm{loc}}$)**.** A language $L \subseteq \{0,1\}^*$ is in $\mathsf{CX}_{\mathrm{loc}}$ if there exists a local cross-examination protocol such that:

- if $x \in L$, then there exists an Alice strategy A_x such that for every Bob strategy, the verifier accepts;
- if $x \notin L$, then there exists a Bob strategy such that for every Alice strategy A_x, the verifier rejects.

Similarly, $L \in \mathsf{PC}_{\mathrm{loc}}$ if the same holds for a precommitment protocol.

Theorem 15.4.
$$\mathsf{CX}_{\mathrm{loc}} = \mathsf{PC}_{\mathrm{loc}}.$$
In particular, since $|Q_n| = 2^{m(n)}$, local cross-examination is exactly as powerful as precommitment to a table of length
$$2^{m(n)} \cdot \ell(n),$$
which is exponential whenever $m(n)$ is polynomial.

Proof. We prove both inclusions.

($\mathsf{CX}_{\mathrm{loc}} \subseteq \mathsf{PC}_{\mathrm{loc}}$). Suppose $L \in \mathsf{CX}_{\mathrm{loc}}$, witnessed by some local cross-examination protocol.

Fix an input $x \in \{0,1\}^n$, and let $A_x : Q_n \to \Sigma_n$ be any deterministic Alice strategy in the cross-examination protocol. Define the corresponding committed table $w_{A_x} \in \Sigma_n^{Q_n}$ by
$$w_{A_x}(q) := A_x(q) \qquad \text{for every } q \in Q_n.$$

We now simulate the cross-examination protocol by a precommitment protocol in which Alice commits to the table w_{A_x}. Whenever Bob chooses a query q_i, the precommitment verifier reads
$$w_{A_x}(q_i) = A_x(q_i),$$
which is exactly the answer that a fresh independent copy of Alice would have returned in the original cross-examination protocol.

We claim that, against any Bob strategy, the two protocols generate exactly the same history

$$h_s = ((q_1, \alpha_1), \ldots, (q_s, \alpha_s)).$$

This follows by induction on the round number i: if the histories agree up to round $i-1$, then Bob chooses the same next query q_i in both protocols, because Bob's choice depends only on x and the previous history. The answer returned is the same, since both protocols return $A_x(q_i)$. Hence the histories remain identical.

Therefore the verifier's final output is the same in both protocols on every input, against every Bob strategy. So every winning Alice strategy in the cross-examination protocol yields a winning committed table in the precommitment protocol, and every winning Bob strategy remains winning as well. Thus

$$\mathsf{CX}_{\mathrm{loc}} \subseteq \mathsf{PC}_{\mathrm{loc}}.$$

($\mathsf{PC}_{\mathrm{loc}} \subseteq \mathsf{CX}_{\mathrm{loc}}$). Now suppose $L \in \mathsf{PC}_{\mathrm{loc}}$, witnessed by some precommitment protocol.

Fix an input $x \in \{0,1\}^n$, and let $w : Q_n \to \Sigma_n$ be any committed table. Define the corresponding deterministic Alice strategy in the cross-examination protocol by

$$A_x^w(q) := w(q) \qquad \text{for every } q \in Q_n.$$

Whenever Bob chooses a query q_i, a fresh independent copy of Alice answers

$$A_x^w(q_i) = w(q_i),$$

which is exactly the value that the precommitment verifier would have read from the table.

As above, by induction on the round number, the query-answer histories in the two protocols are identical against any Bob strategy. Hence the verifier's output is identical in the precommitment and cross-examination protocols.

Therefore every winning committed table w yields a winning Alice strategy A_x^w, and every winning Bob strategy remains winning. Thus

$$\mathsf{PC}_{\mathrm{loc}} \subseteq \mathsf{CX}_{\mathrm{loc}}.$$

Combining the two inclusions, we conclude that

$$\mathsf{CX}_{\mathrm{loc}} = \mathsf{PC}_{\mathrm{loc}}.$$

□

Remark

The theorem shows that the role of cross-examination is to give Bob and the verifier *random access* to a huge implicit object. The object is the answer table

$$q \mapsto A_x(q).$$

Because the allowed queries q have polynomial length, this table has exponential size in general. Thus local cross-examination is exactly equivalent to exponential precommitment plus local spot-checking.

Remark

The theorem is stated for deterministic Alice strategies. This is the clean setting for the deterministic debate protocols considered earlier. If one wishes to allow randomized strategies, one needs a slightly richer formulation; the basic idea remains that the precommitment object must encode whatever a fresh copy of Alice would answer on every allowed query.

16 Criticisms: The Debate around "Debate"

A series of criticisms and caveats have been leveled at the whole idea of AI safety via debate. The following list are the ones which are accepted and actively worked on by debate researchers.

- The most important is **Obfuscated Arguments**, raised by Beth Barnes. This refers to the fact that for computationally bounded provers, a viable strategy for a malicious prover is to "decompose" a false argument into a series of arguments, where most are correct but a small number is false and it is computationally hard to find out which are false. A common example is the claim "N is a prime number" which can be easily refuted by stochastic primality testing. But if Alice partitions this into subclaims {"N has no prime factor in the interval I_1", N has no prime factor in the interval I_2", ...} where the intervals cover the numbers from 2 to sqrt(N), then it is computationally hard to find out which of these subclaims is wrong, even though Bob knows at least one has to be. It is much harder to find out where the prime factors are than to assert that at least two exist.

- The best strategy for a prover to win might be to get the human judge to run code as part of an experiment to settle the argument that would then have malicious consequences and force the human judge to reward the dishonest debater.

- The training setup for debate has no exploration guarantees that ensure that an existing successful strategy will even be found by the provers.

- The provers might collude against the human overseers. While this is disincentivised on an agent level by the setup, since an honest prover can maximise their reward with an honest strategy, it might nevertheless occur and be stable during training, because there isn't enough exploration of strategies.

- The judges are not perfect and have biases that can be exploited by a malicious prover.

17 Obfuscation

One problem that arises when the provers Alice and Bob are restricted in their computational power is *Obfuscated Arguments*.

Definition 17.1 (Obfuscation in naive recursive debate)**.** An argument is *obfuscated* if a dishonest debater can decompose a top-level claim into subclaims $q_1, \ldots, q_m$ with claimed answers $a_1, \ldots, a_m$ such that:

1. the overall argument is false,
2. only a small number of the claimed answers a_i are false (in the paper's idealized model, exactly one),
3. but it is computationally intractable for the debaters to determine which subclaim is false.

Example 17.2 (Prime-checking as obfuscation)**.** Suppose Alice claims that a number n is prime, while it is in fact composite. This is easy to disprove through primality testing. But Alice partitions the interval of possible divisors

$$D_n = [2, \lfloor \sqrt{n} \rfloor]$$

into subintervals $I_1, \ldots, I_m$, and asserts for each k that I_k contains no divisor of n.
Her overall argument is false, because at least one interval contains a factor. But all the other subclaims can be true, and Bob can only refute Alice by identifying the unique bad interval. If locating that interval is computationally intractable, then the falsehood is effectively hidden inside an otherwise correct decomposition. For example, if $n = 221$, then $D_{221} = \{2, \ldots, 14\}$, and we may choose

$$I_1 = \{2, 3, 4\}, \quad I_2 = \{5, 6, 7\}, \quad I_3 = \{8, 9, 10\}, \quad I_4 = \{11, 12, 13, 14\}.$$

A dishonest debater Alice argues:

$$\forall k \in \{1, \ldots, m\}, \quad \text{“there is no divisor of } n \text{ in } I_k\text{.”}$$

From this she concludes that n has no nontrivial divisor, and hence is prime.
If n is composite, then at least one of these subclaims must be false. In the case $n = 221$, the first three subclaims are true, but the fourth is false because

$$13 \in I_4 \qquad \text{and} \qquad 13 \mid 221.$$

So Alice's overall argument is false, but the error is localized to a single interval. If there are many such intervals and locating the bad one is computationally difficult, then Alice's argument is an example of an obfuscated argument.

Exercise 17.1. Show that you can decompose any statement into an obfuscated argument. Hint: Re-use the hardness of a problem like prime-factorisation, as well as the concept of vacuous truth, or truth by false premise, i.e., that $A \to B$ is true, whenever A is false.

18 Prover-Estimator Debate

This paper is a direct attempt to solve the obfuscated arguments problem while keeping the attractive recursive structure of debate. Instead of having the opponent choose which subclaim to attack, the protocol makes the roles asymmetric: Alice is the prover, who decomposes a claim into subclaims, and Bob is the estimator, who assigns probabilities to those subclaims. Alice then has to pick a subclaim and argue that Bob's probability is wrong in a particular direction. The key idea is that if Bob can assign probabilities that are hard for Alice to distinguish from the truth, then Alice cannot reliably steer the debate toward a hidden flaw unless such a flaw is actually findable.

The paper's most important new concept is (ϵ, ρ)**-stability**. Informally, a recursive argument is stable if its correctness does not depend too delicately on tiny changes in the probabilities assigned to subclaims. That matters because the estimator is only trying to be approximately right. If a correct argument collapses whenever a subclaim probability shifts by an arbitrarily small amount, then no realistic estimator could support it reliably. The authors therefore require stability for the **usefulness** of the protocol: honest provers can win when they have robust arguments, not brittle ones.

Prover-Estimator debate is a technique to address the problem of obfuscation by turning the debate into an asymmetric game between a prover (Alice) and an estimator (Bob). The basic idea is:

1. Alice makes her main claim (root claim of the argument tree)
2. Bob assigns it a probability (< 0.5, or low enough for the human to reject trusting the claim)
3. Alice now must argue that Bob's estimate is incorrect. To show this, she decomposes the root claim into subclaims.
4. Bob now assigns estimates to each subclaim. Aggregating these estimates must be consistent with his estimate about the root claim (consistency criterion).
5. Now Alice chooses one of the estimates to refute and again decomposes the associated claims.
6. The game continues until the maximal depth is reached. A human overseer then assigns probabilities to the leaf claims of the lowest level.
7. Alice wins if either the human rejects Bob's estimate at the lowest level, or if Bob violates the consistency criterion. Bob wins if he is consistent at every level and the human accepts his estimates.

Prover-Estimator debate does not fully resolve obfuscated arguments in its current state, because the consistency criterion isn't stable enough to small errors that Bob can make. Improving this is current research.

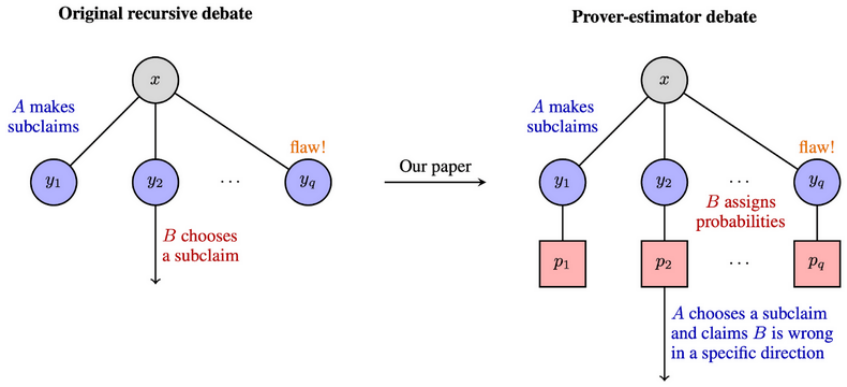

Figure 1: Schematic explaining the PE debate.

Definition 18.1 (Prover-estimator debate)**.** Fix:

- an input x,
- a language L,
- a recursive decomposition procedure D,
- a human judgement oracle H,
- a depth parameter r,
- and a reward-growth parameter $\lambda > 1$.

A prover-estimator debate is a depth-r recursive two-player zero-sum protocol between:

- a *prover* P, who tries to prove the correct answer to the current query,
- an *estimator* E, who outputs probabilities for the prover's subclaims.

The protocol is defined recursively as follows.

Initialization. At the root:

1. The current query is the top-level question $q_0 := x \in L$.
2. The prover outputs a bit $a_0 \in \{0, 1\}$, claiming either $x \in L$ or $x \notin L$.

3. The estimator outputs a probability $p_0 \in [0, 1]$, intended to represent its estimate that a_0 is correct.

4. A trusted random bit $y_0 \sim \text{Bernoulli}(p_0)$ is sampled.

Round t ($0 \leq t < r$). Suppose the current query is q_t, the current claimed answer is a_t, and the estimator's current probability is p_t.

1. The prover outputs a sign

$$s_t \in \{-1, 0, +1\},$$

interpreted as claiming that p_t is respectively underestimated, approximately correct, or overestimated.

2. The prover outputs a sequence of subqueries

$$q_{t,1}, \ldots, q_{t,m_t}$$

together with claimed answers

$$a_{t,1}, \ldots, a_{t,m_t},$$

intended to certify that a_t is the correct answer to q_t, assuming the claimed answers to the subqueries are correct.

3. If $t = r - 1$ (the final round), then:

   (a) the prover identifies the relevant subset of proof bits encoded by its final decomposition;

   (b) those bits are evaluated directly using the human judgement oracle H, without further debate;

   (c) payoffs are assigned from this terminal check.

4. If $t < r - 1$, then for each subquery $q_{t,i}$:

   (a) the estimator outputs a probability

$$p_{t,i} \in [0, 1],$$

   intended to equal its best estimate that $a_{t,i}$ is correct, conditioned on the prior sampled outcomes;

   (b) a trusted random bit

$$y_{t,i} \sim \text{Bernoulli}(p_{t,i})$$

   is sampled.

5. The players receive intermediate rewards based on whether the prover correctly identified an inconsistency between the estimator's current-round probabilities and the previous-round probability.

6. The prover selects one subquery index $i^\star$ for recursion.

7. The protocol recurses on the selected subinstance

$$(q_{t+1}, a_{t+1}, p_{t+1}) := (q_{t,i^\star}, a_{t,i^\star}, p_{t,i^\star}).$$

19 Reading and Discussion "Experimental Results"

Optimising for Debate increases Judge Accuracy, Optimizing for Consultancy decreases it: Debating with More Persuasive LLMs Leads to More Truthful Answers

Debate helps judges even with systematic biases: AI Debate Aids Assessment of Controversial Claims

Debate can prevent reward hacking: Debate Training Reduces Reward Hacking in RLAIF

Presentation of the papers: Iliad Intensive August 2026 - Debate

Discussion Prompts: What is the setup for the debate? How are the provers and judges implemented? What has been measured? Was the improvement of the measures through debate significant? How close is this setup to the theoretical description of debate?

20 The AISI Safety Case

AISI's debate safety case says: if debate can reliably make honesty the winning strategy, if training explores dishonesty enough to eliminate it, and if hidden flaws in obfuscated arguments can be handled, then debate could provide scalable oversight for advanced AI R&D agents. Its purpose is less a finished guarantee and more a roadmap of the assumptions and evidence needed for such a guarantee. Its main value is not that it proves debate already works, but that it decomposes the research agenda into assumptions that need evidence: debate equilibria must favor truth, training must explore deceptive strategies enough to punish them, humans must judge debates reliably, and obfuscated arguments must be solved.

21 Debate

Is AIS via Debate research more capability than alignment?

22 Learn More

- Debating with More Persuasive LLMs Leads to More Truthful Answers

- The limits of AI safety via debate
- The alignment safety-case sketch based on Debate
- Knowledge Divergence and the Value of Debate for Scalable Oversight
- Emergent Alignment via Competition