Self-Proving Models

Orr Paradise

MIT 6.S976 and 18.S996

March, 2026

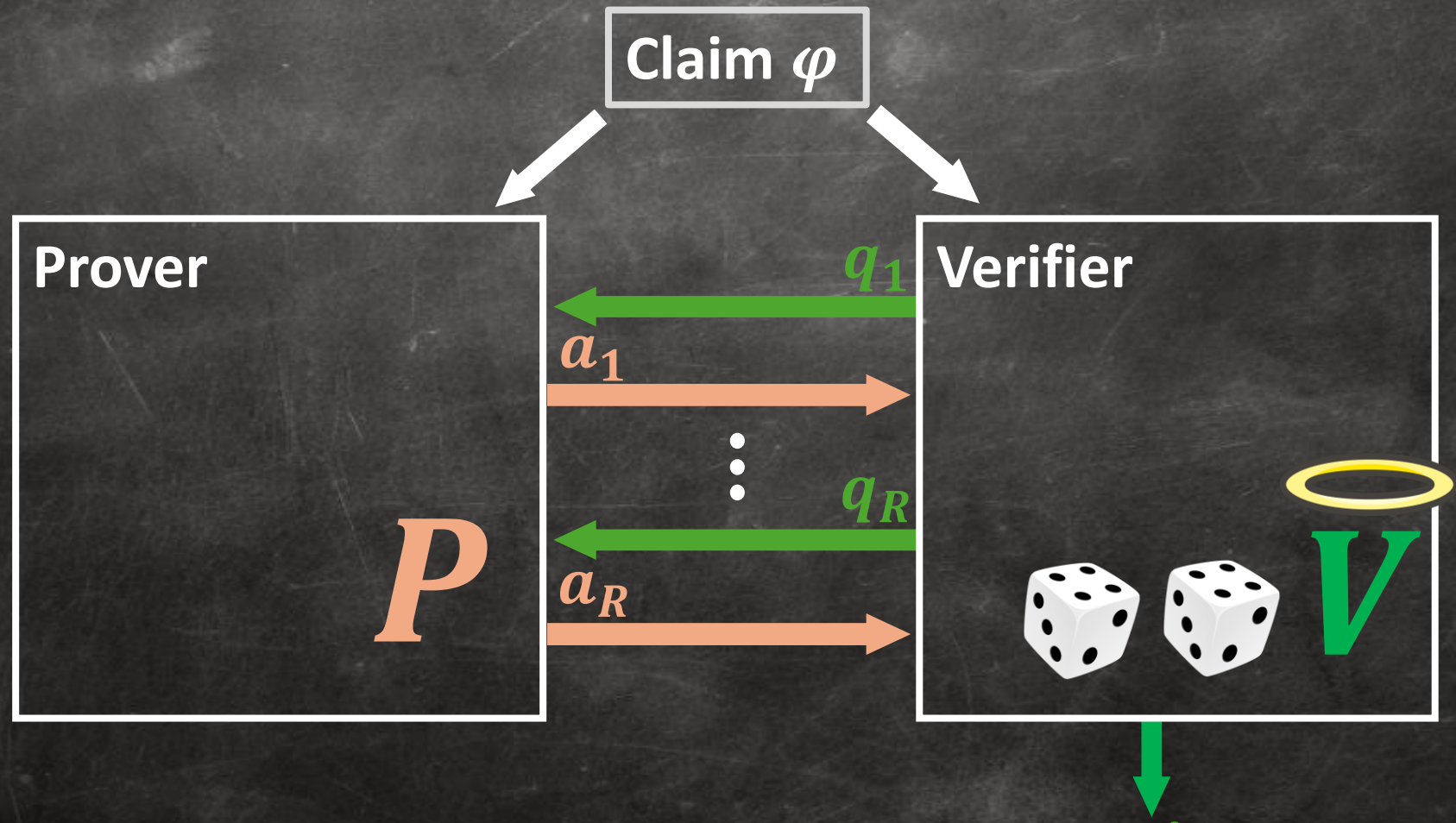
Lesson plan

- Reminder: Proofs and Models
- Defining Self-Proving models
- Learning Self-Proving models
  - Transcript Learning (TL)
  - Reinforcement Learning from Verifier Feedback (RLVF)
- Self-Proving models in practice

Lesson plan

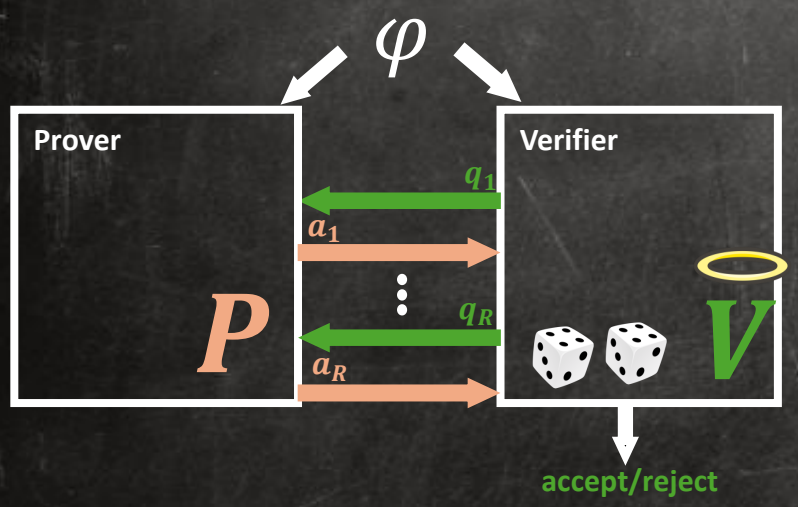
- **Reminder: Proofs and Models**
- Defining Self-Proving models
- Learning Self-Proving models
  - Transcript Learning (TL)
  - Reinforcement Learning from Verifier Feedback (RLVF)
- Self-Proving models in practice

Interactive Proof systems (IPs)



R: Number of rounds

Interactive Proof systems (IPs)



(1) Completeness
There exists an "honest prover" $\bar{P}$
such that for any correct claim φ,

$$\Pr[\bar{P} \text{ convinces } V \text{ to accept } \varphi] = 100\%$$

(2) Soundness (with δ error)
For any incorrect claim φ
and for any "lying prover" P,

$$\Pr[P \text{ convinces } V \text{ to accept } \varphi] \leq \delta$$

(3) Efficiency
Polytime Verifier, minimal #rounds, short messages…
→ *"More efficient than the Prover"*

The power of interaction

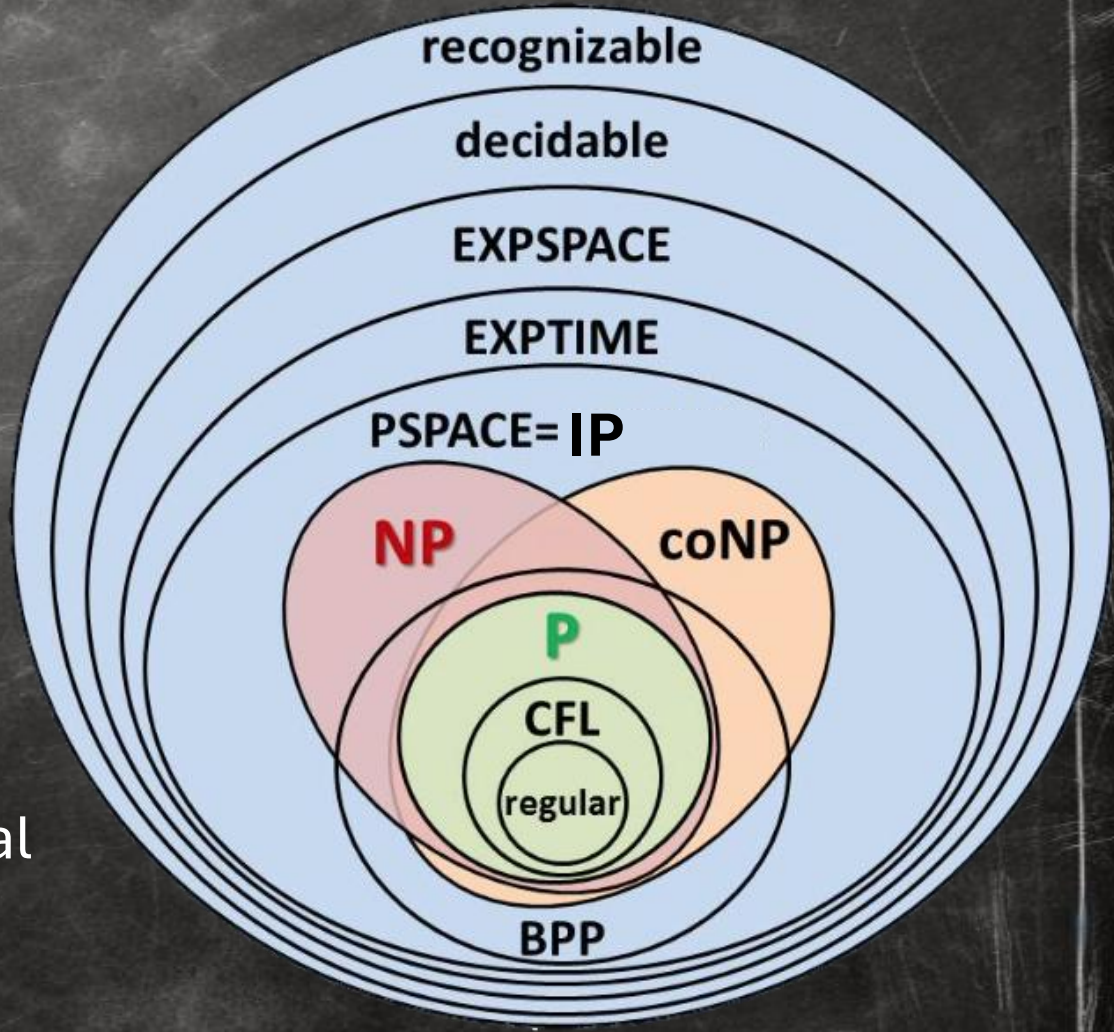
- We saw that Interactive Verification is much more powerful: IP = PSPACE [LFKN, Shamir 90]

  → For any **polynomial-space** decidable claim, there exists an efficient (Probabilistic Polynomial Time) Interactive Verifier.

- *"Any problem you could dream of solving, can be efficiently Verified with the help of an untrusted Prover."*

- IPs allow an efficient verifier to establish trust with a **more powerful, untrusted** computational entity…

- Who might this powerful entity be?



What is a model?

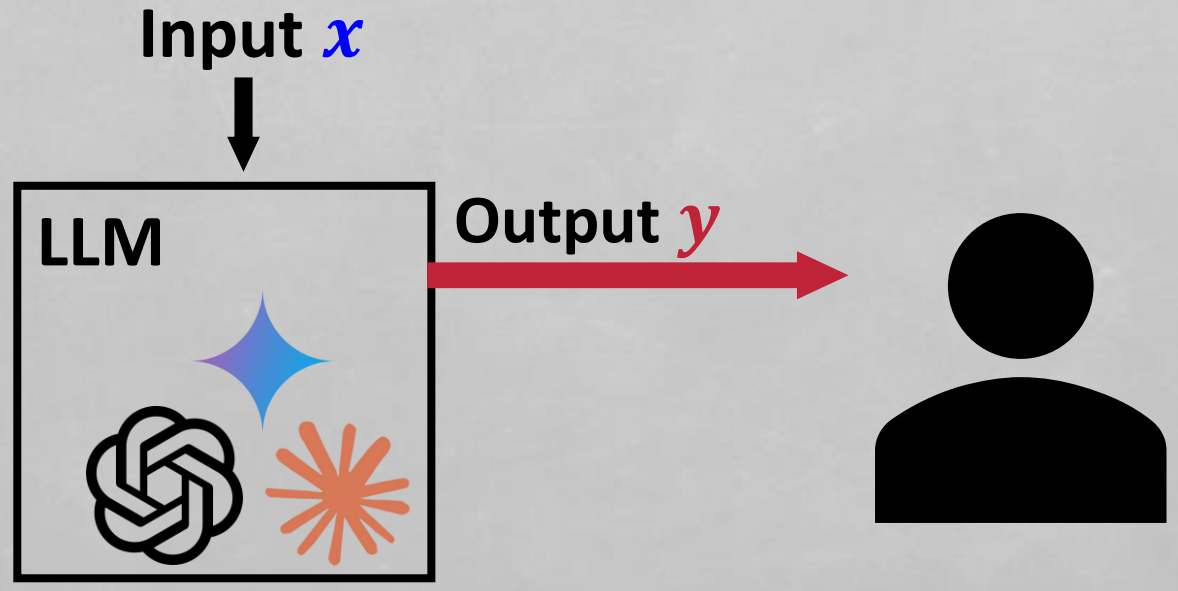
- Fix a ground-truth functionality $f^*: X \to Y$. For simplicity, let's take deterministic functions.
- f^* might be worst-case hard to compute.
  - We might not even *know* f^* is! Only know some samples $(x_1, f^*(x_1)), \ldots, (x_m, f^*(x_m))$
- In practice, we only need to compute f^* on "natural" inputs.
  - "Nature" is an unknown distribution $\boldsymbol{\mu}$ over $\boldsymbol{X}$ we can sample from.
- Fix a set of parameters $\boldsymbol{\Theta}$. A model family is a collection of (randomized) functions $\{\boldsymbol{f_\theta: X \to Y}\}_{\boldsymbol{\theta \in \Theta}}$
- Model training: Given input $S = (x_1, f^*(x_1)), \ldots, (x_m, f^*(x_m))$ for $x_1, \ldots, x_m \sim \mu$, output $\widehat{\boldsymbol{\theta}} \in \boldsymbol{\Theta}$
- A model $\widehat{\boldsymbol{\theta}}$ is $(1 - \varepsilon)$-correct (w.r.t $\boldsymbol{\mu}, \boldsymbol{f^*}$) if

$$\Pr_{\substack{x \sim \mu \\ \hat{y} \sim f_{\widehat{\theta}}(x)}} [\hat{y} = f^*(x)] \geq 1 - \varepsilon$$

In what sense do we “trust” a model?

- Average-case: Test if model f_θ has 99% accuracy w.r.t μ, f^* (last week’s lectures)
- Worst-case: You have an input x, for which the model generates $y \sim f_\theta(x)$

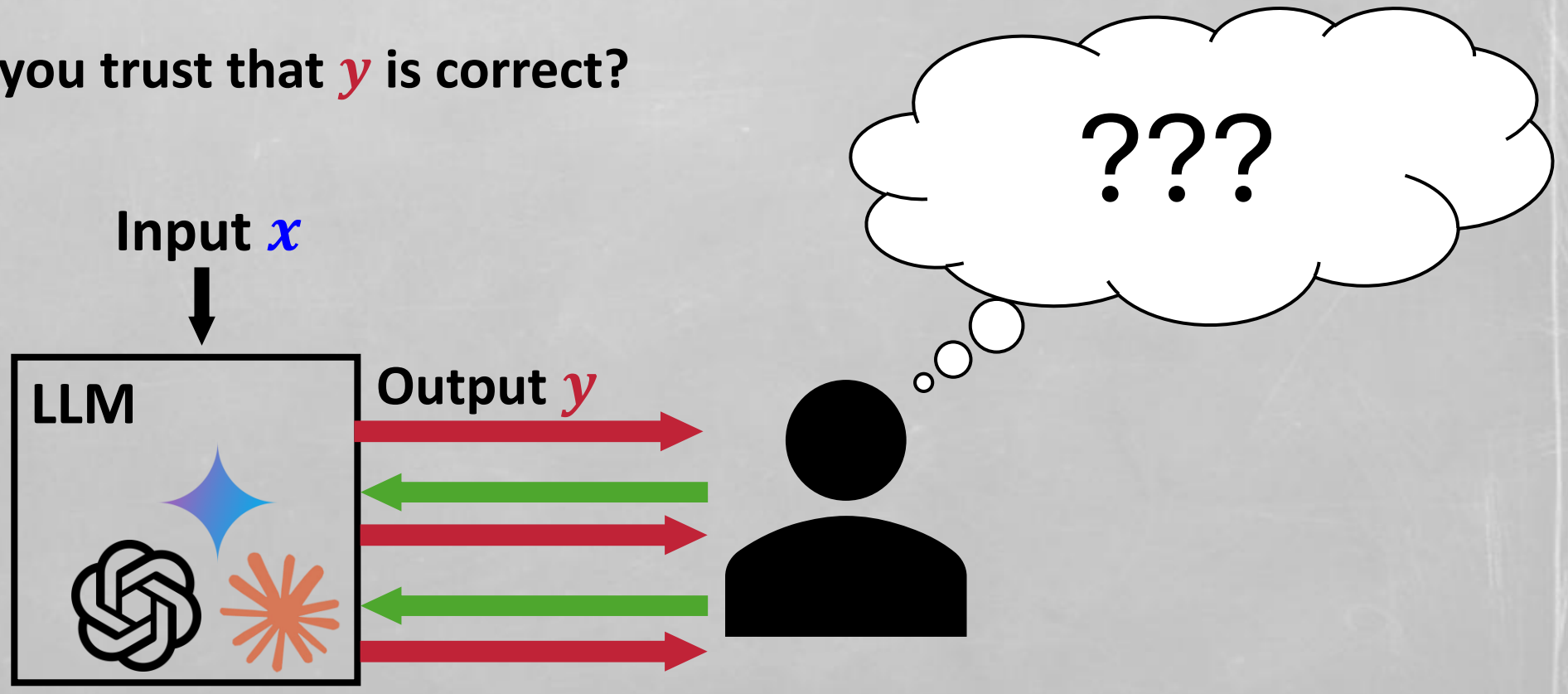
Should you trust that y is correct?



In what sense do we “trust” a model?

- Average-case: Test if model f_θ has 99% accuracy w.r.t μ, f^* (last week’s lectures)
- Worst-case: You have an input x, for which the model generates $y \sim f_\theta(x)$

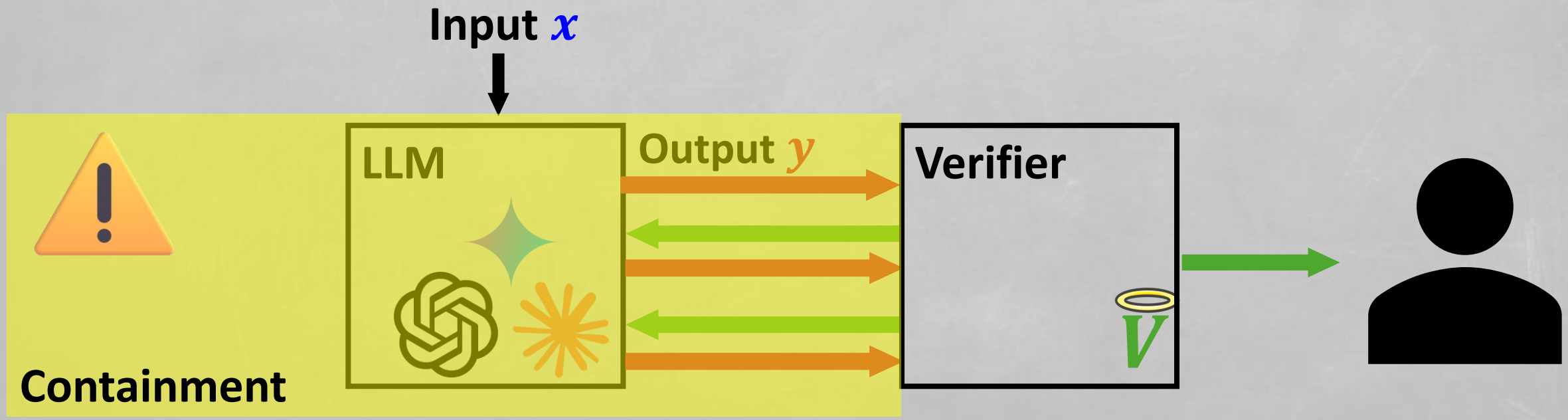
Should you trust that y is correct?



In what sense do we “trust” a model?

- Average-case: Test if model f_θ has 99% accuracy w.r.t μ, f^* (last week’s lectures)
- Worst-case: You have an input x, for which the model generates $y \sim f_\theta(x)$

Should you trust that y is correct?



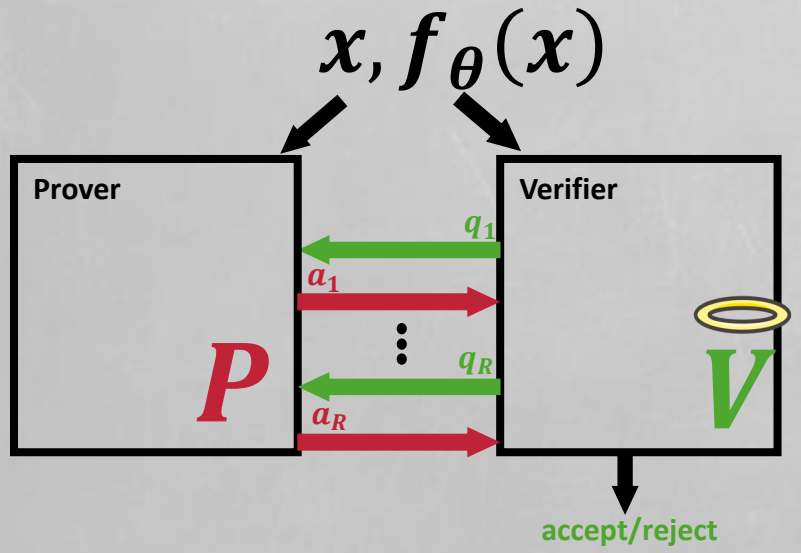
Towards Guaranteed Safe AI. May 2024. Dalrymple, Skalse, Bengio, Russell, Tegmark, Seshia, Omohundro, Szegedy, Goldhaber, Ammann, Abate, Halpern, Barrett, Zhao, Zhi-Xuan, Wing, Tenenbaum.

Lesson plan

- Reminder: Proofs and Models
- **Defining Self-Proving models**
- Learning algorithms for Self-Proving models
  - Transcript Learning (TL)
  - Reinforcement Learning from Verifier Feedback (RLVF)
- Self-Proving models in practice

IPs for model outputs

Fix a model $f_\theta : X \to Y$ (deterministic, for simplicity)
The "claim" is that $f_\theta(x) = f^*(x)$.



(1) Completeness

There exists an "honest prover" $\tilde{P}$
such that for any input x,

$$\Pr\left[\tilde{P} \text{ convinces } V \text{ to accept } \left(x, f^*(x)\right)\right] = 100\%$$

(2) Soundness

For any input x and incorrect output $f_\theta(x) \neq f^*(x)$,
for any "lying prover" P

$$\Pr\left[P \text{ convinces } V \text{ to accept } \left(x, f_\theta(x)\right)\right] \leq \delta$$

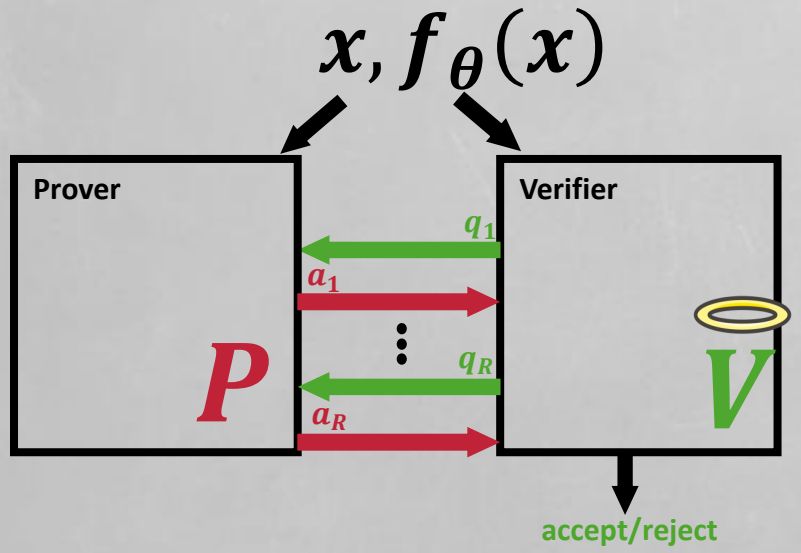
(3) Efficiency

Polytime verifier, minimal #rounds, short messages…

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. May 2024 (NeurIPS 2025)

IPs for model outputs

Fix a model $f_\theta : X \to Y$ (deterministic, for simplicity)
The "claim" is that $f_\theta(x) = f^*(x)$.



Even if we learned a model f_θ, where do we get this Prover?

(1) Completeness

There exists an "honest prover" $\tilde{P}$
such that for any input x,

$$\Pr\left[\tilde{P} \text{ convinces } V \text{ to accept } \left(x, f^*(x)\right)\right] = 100\%$$

(2) Soundness

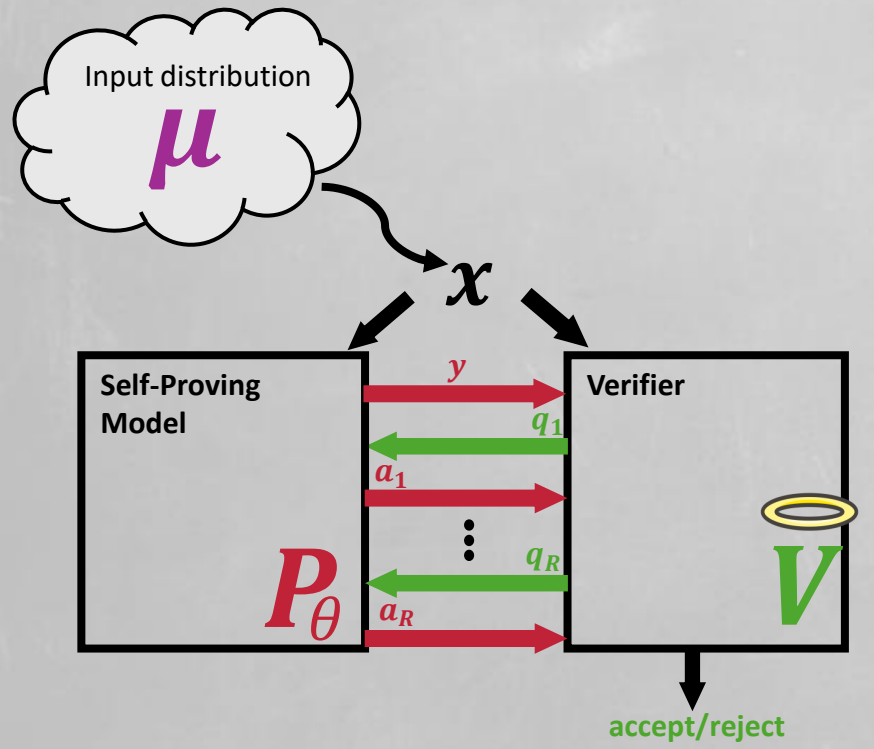
For any input x and incorrect output $f_\theta(x) \neq f^*(x)$,
for any "lying prover" P

$$\Pr\left[P \text{ convinces } V \text{ to accept } \left(x, f_\theta(x)\right)\right] \leq \delta$$

Why not learn them *jointly*?
Let the model *prove its own output!*

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. May 2024 (NeurIPS 2025)

Self-Proving models



Definition (Self-Proving Model with error ε)

A model P_θ is **Self-Proving** w.r.t V, μ

If $\Pr[P_\theta$ convinces V to accept $y] \geq 1 - \varepsilon$

Over $x \sim \mu$, $y \sim P_\theta(x)$, $q_1, a_1, \ldots, q_R, a_R$

Exercise (Soundness+Self-Prov. → Correctness):

Fix a Verifier V for f^* with soundness error δ.

If P is Self-Proving with err ε w.r.t V, μ, then

$$\Pr_{x \sim \mu}[P_\theta(x) = f^*(x)] \geq 1 - \delta - \varepsilon$$

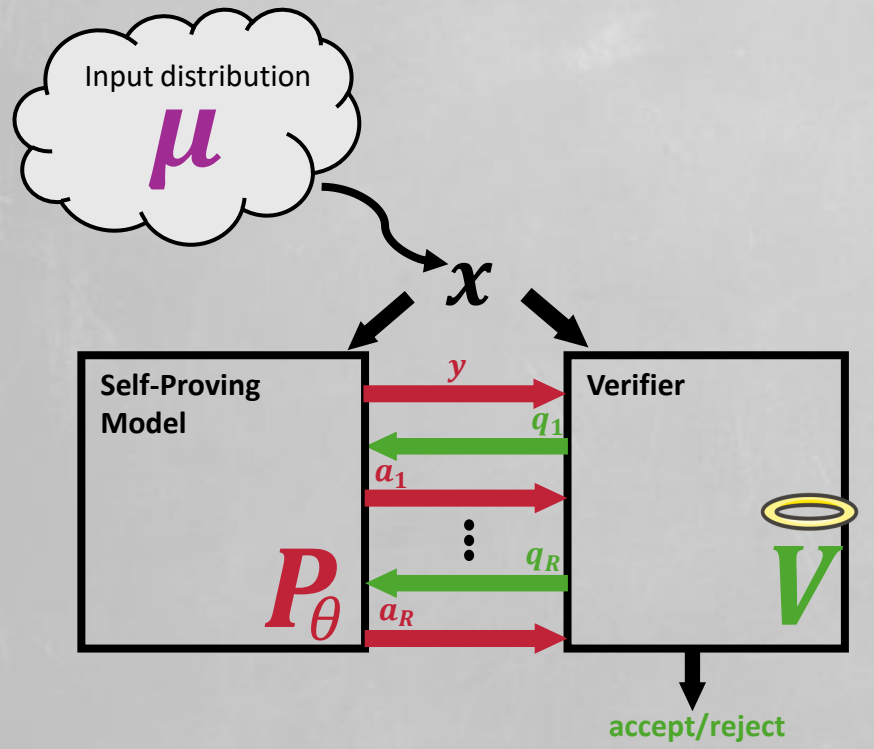
Goal: Learn θ such that P_θ is Self-Proving

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. May 2024 (NeurIPS 2025)

Lesson plan

- What is a proof? What is a model?
- Defining Self-Proving models
- **Learning Self-Proving models**
  - **Transcript Learning (TL)**
  - **Reinforcement Learning from Verifier Feedback (RLVF)**
- Self-Proving models in practice
- Proving convergence guarantees for TL and RLVF

Reminder: Self-Proving models



Definition (Self-Proving Model with error ε)

A model P_θ is **Self-Proving** w.r.t V, μ

If $\Pr[P_\theta$ convinces V to accept $y] \geq 1 - \varepsilon$

Over $x \sim \mu$, $y \sim P_\theta(x)$, $q_1, a_1, \ldots, q_R, a_R$

Exercise (Soundness+Self-Prov. → Correctness):

Fix a Verifier V for f^* with soundness error δ.

If P is Self-Proving with err ε w.r.t V, μ, then

$$\Pr_{x \sim \mu}[P_\theta(x) = f^*(x)] \geq 1 - \delta - \varepsilon$$

Goal: Learn θ such that P_θ is Self-Proving

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. May 2024 (NeurIPS 2025)

Learning Self-Proving models

What do we need in order to learn a Self-Proving model?

- Sequence-to-sequence model family: $\{P_\theta : \Sigma^* \to \Sigma^*\}_{\theta \in \Theta}$ where $\Theta = \mathbb{R}^d$.
- Autoregressive generation: For any parameter setting θ and input string $z \in \Sigma^*$,
  - Next token ("forwards pass"): RV $p_\theta(z)$ over Σ, determined by the *logits:* $\log p_\theta(z) \in \mathbb{R}^{|\Sigma|}$
  - Keep generating until a special token $\mathrm{EOS} \in \Sigma$ is sampled.
- Differentiability ("backwards pass"): Can compute $\nabla_\theta \log \Pr_{\tau \sim p_\theta(z)}[\tau = \sigma]$ for each $\sigma \in \Sigma$.
- Access to the input distribution $x \sim \mu$
- Access to the Verifier V.
- **Transcript Learning:** assume also access to an "honest prover"
  - (Slightly weaker access to an "honest transcript generator" suffices)

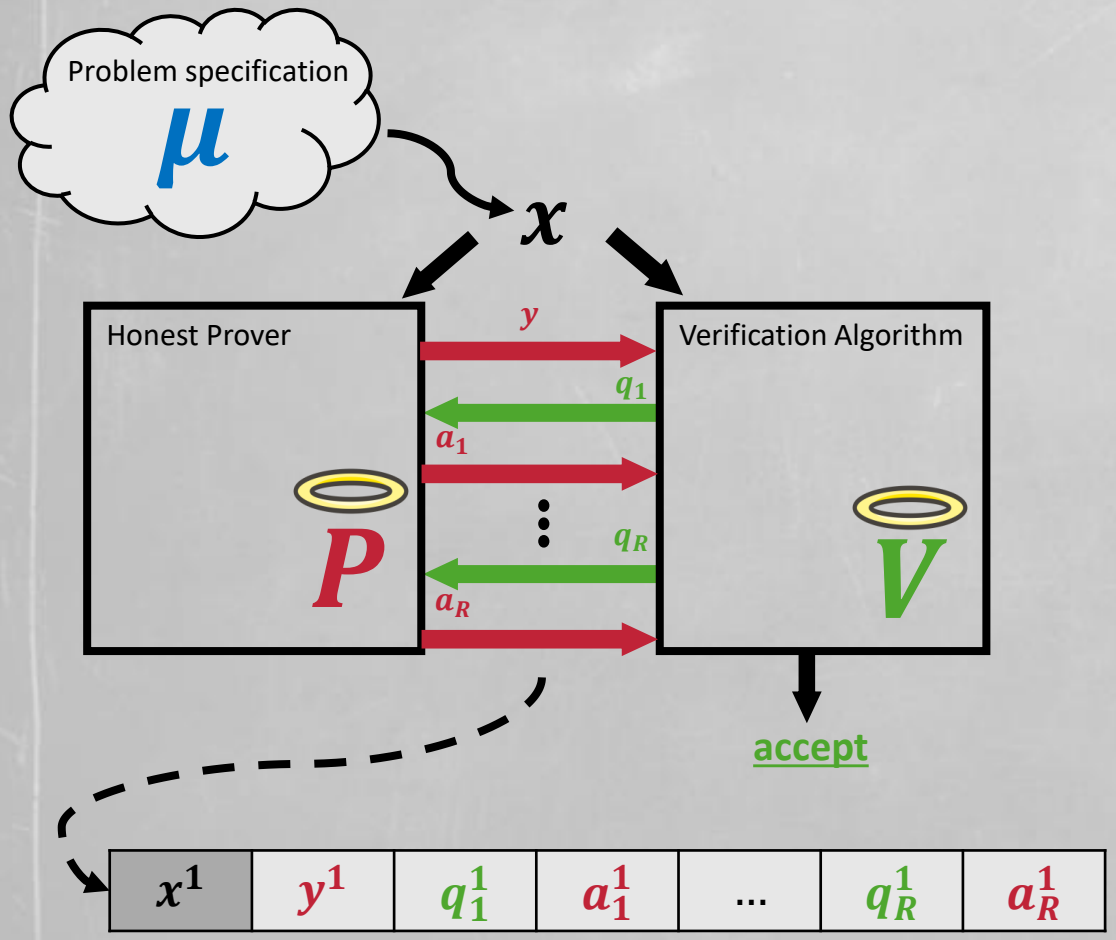
Transcript Learning

Step 1: Transcribe ”honest” interactions

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

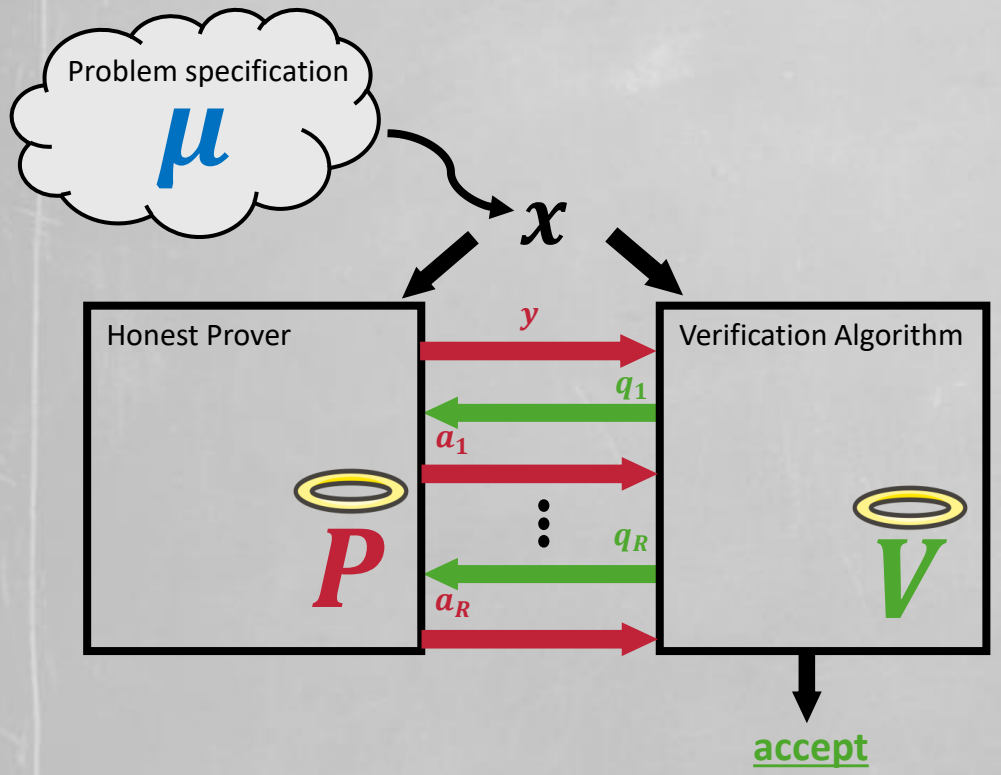
Step 1: Transcribe ”honest” interactions



Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

Step 1: Transcribe ”honest” interactions

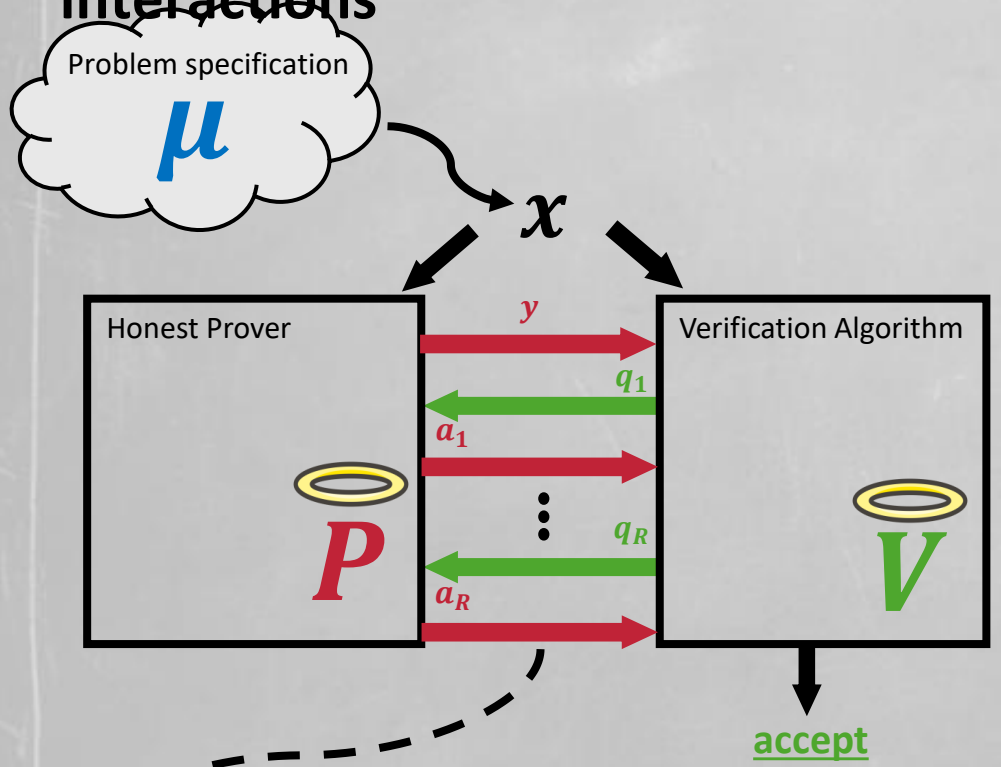


x^1	y^1	q_1^1	a_1^1	...	q_R^1	a_R^1

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

Step 1: Transcribe ”honest” interactions



x^1	y^1	q_1^1	a_1^1	...	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	...	q_R^N	a_R^N

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

Step 1: Transcribe ”honest” interactions

Step 2: Transcript Cloning

x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	…	q_R^N	a_R^N

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

Step 1: Transcribe ”honest” interactions

Step 2: Transcript Cloning

For each transcript:

x	y	q_1	a_1	…	q_R	a_R

Self-Proving Autoregressive Model

P_θ

x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	…	q_R^N	a_R^N

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

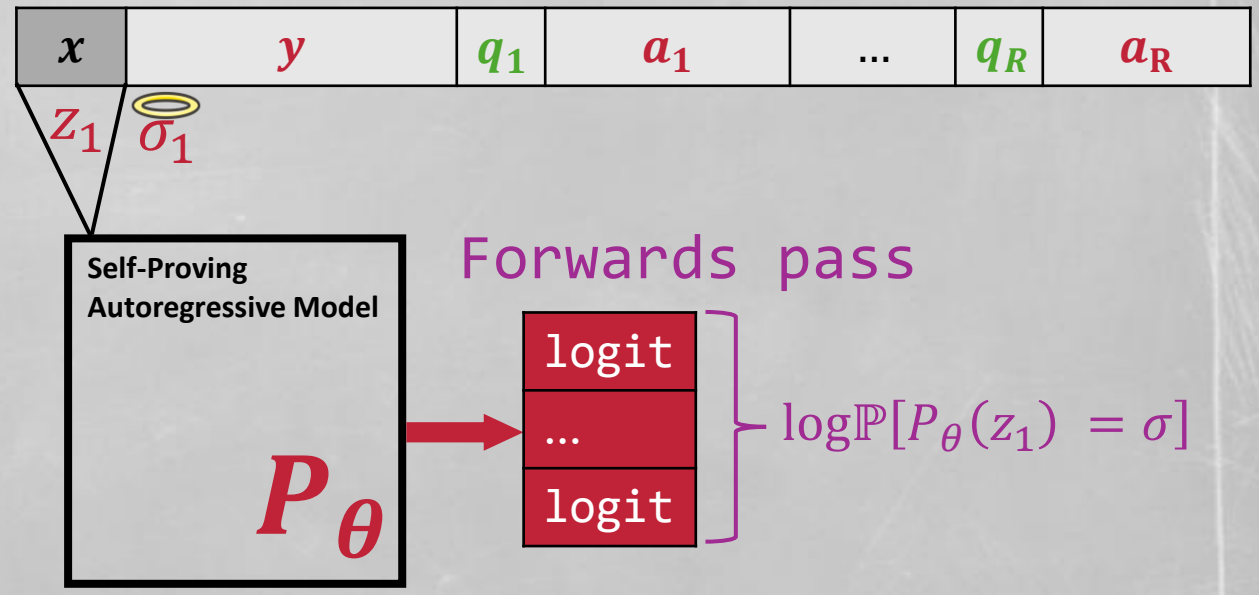
Transcript Learning

Step 1: Transcribe ”honest” interactions

x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	…	q_R^N	a_R^N

Step 2: Transcript Cloning

For each transcript:



Backwards pass $\overrightarrow{d_1} := \nabla_\theta \log\mathbb{P}[P_\theta(z_1) = \sigma_1]$

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

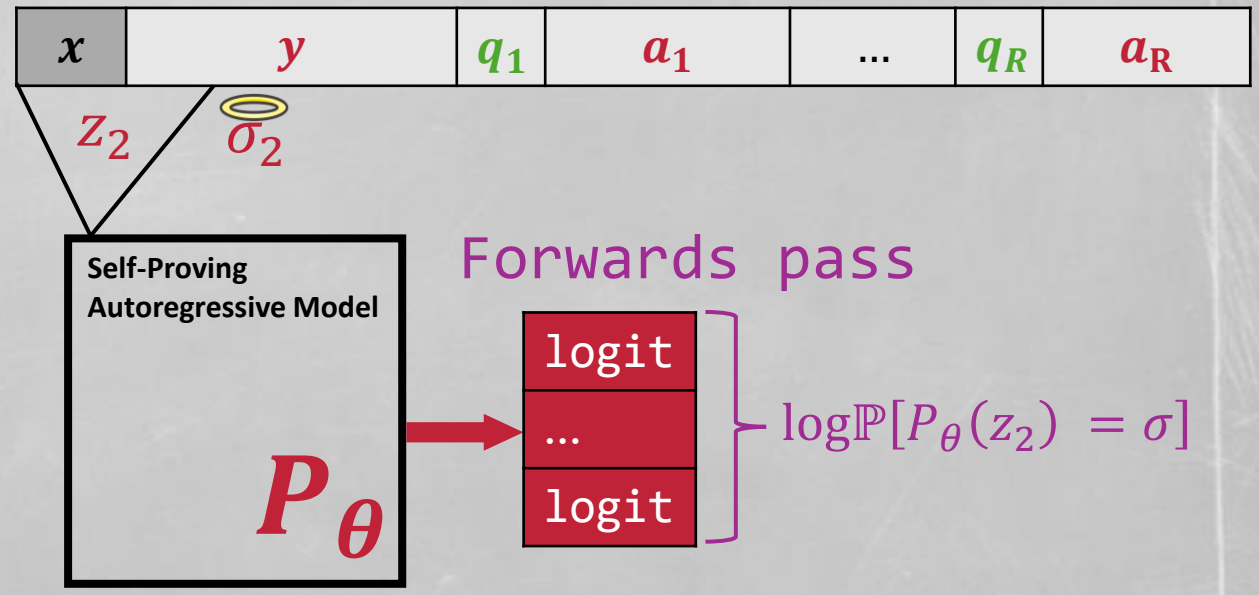
Transcript Learning

Step 1: Transcribe ”honest” interactions

x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	…	q_R^N	a_R^N

Step 2: Transcript Cloning

For each transcript:



Backwards pass $\overrightarrow{d_2} \coloneqq \nabla_\theta \log\mathbb{P}[P_\theta(z_2) = \sigma_2]$

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

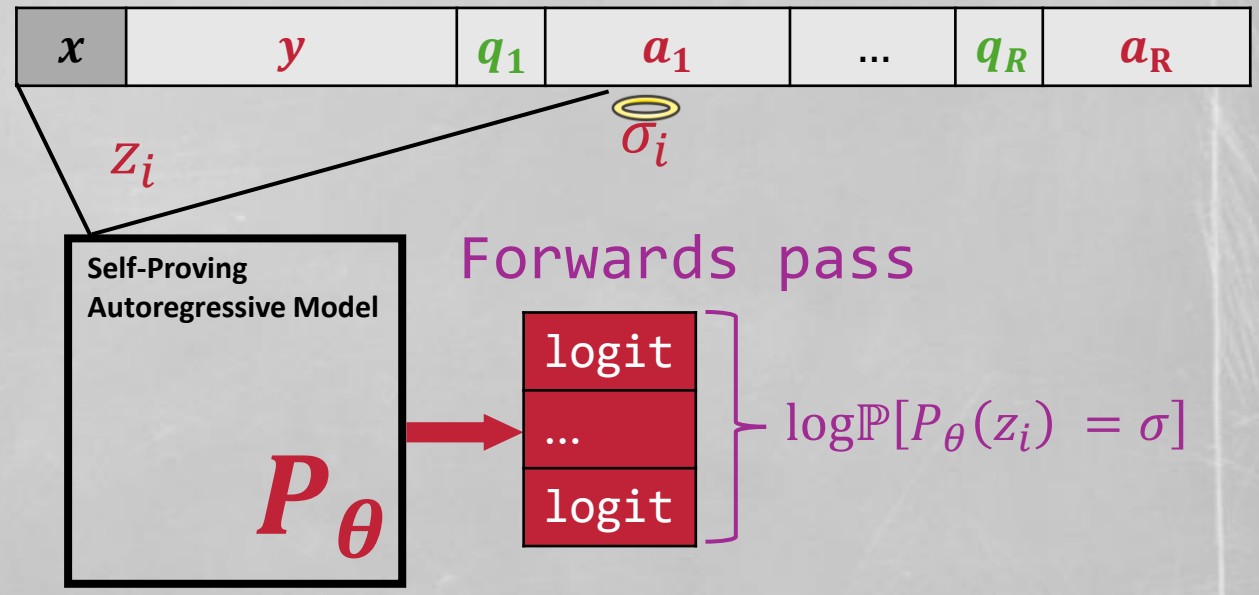
Transcript Learning

Step 1: Transcribe ”honest” interactions

x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	…	q_R^N	a_R^N

Step 2: Transcript Cloning

For each transcript:



Backwards pass $\overrightarrow{d_i} := \nabla_\theta \log\mathbb{P}[P_\theta(z_i) = \sigma_i]$

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

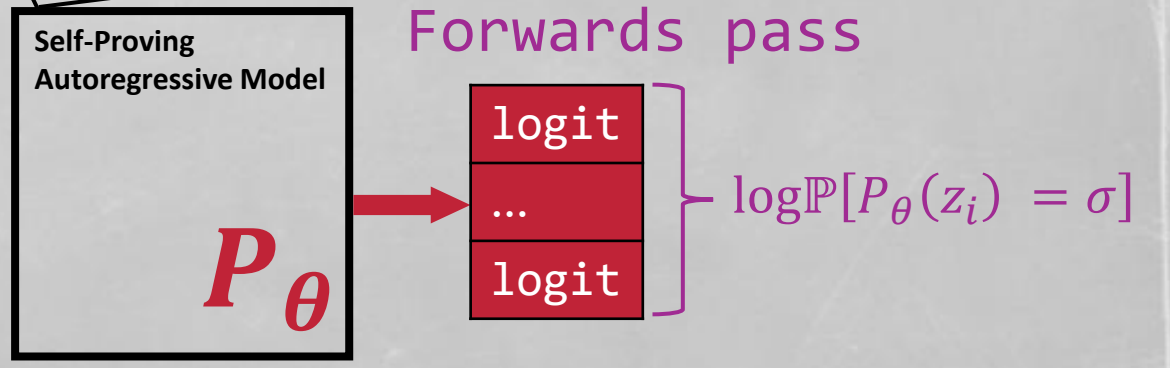
Step 1: Transcribe "honest" interactions

x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1
x^N	y^N	q_1^N	a_1^N	…	q_R^N	a_R^N

Step 2: Transcript Cloning

For each transcript:

x	y	q_1	a_1	…	q_R	a_R



Backwards pass $\overrightarrow{d_i} := \nabla_\theta \log\mathbb{P}[P_\theta(z_i) = \sigma_i]$

Update params $\theta \leftarrow \theta + \lambda \cdot \mathbb{P}\left[\overrightarrow{P_\theta}(x) = z\right] \cdot \sum_i \overrightarrow{d_i}$

Forwards Backwards

Models That Prove Their Own Correctness. Noga Amit, Guy N. Rothblum, Orr Paradise, Shafi Goldwasser. In Submission.

Transcript Learning

Assumptions:

- Access to a dataset of honest transcripts.
- The total number of tokens sent by the prover in any interaction is $< \boldsymbol{C}$.
- The surrogate objective $A(\theta) \coloneqq \mathbb{P}_x[\pi_\theta(x) = \tilde{\pi}(x) \mid V'\text{s rand}]$ is concave and differentiable in θ.
- The logits of $\boldsymbol{P_\theta}$ are $\boldsymbol{B_1}$-Lipschitz in $\boldsymbol{\theta}$.
- For $\boldsymbol{\varepsilon} > 0$ let $\boldsymbol{B_2}$ such that:
  - There exists $\boldsymbol{\theta^*}$ with $\|\boldsymbol{\theta^*}\| < \boldsymbol{B_2}$ such that $A(\boldsymbol{\theta^*}) \geq 1 - \boldsymbol{\varepsilon}/2$.

"Necessary" for reasoning about NN convergence

<u>The TL Theorem</u>: Under the assumptions,

Transcript Learning outputs a $(\boldsymbol{1-\epsilon})$-**Self-Proving model** when trained on

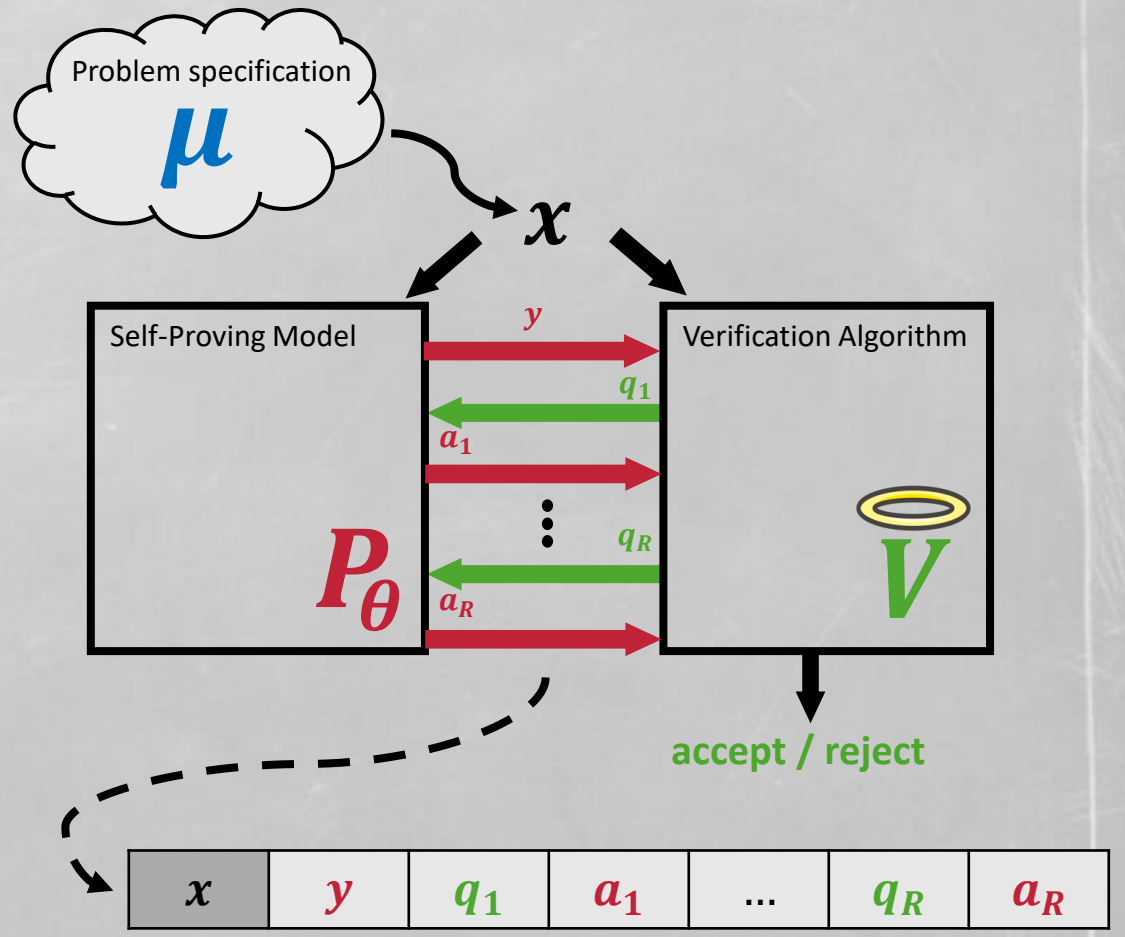
$$\boldsymbol{N \geq 4\left(C \cdot B_1 \cdot B_2 \cdot \frac{1}{\epsilon}\right)^2}$$ samples

RL from Verifier Feedback (RLVF)

RL from Verifier Feedback (RLVF)

Repeat the following:

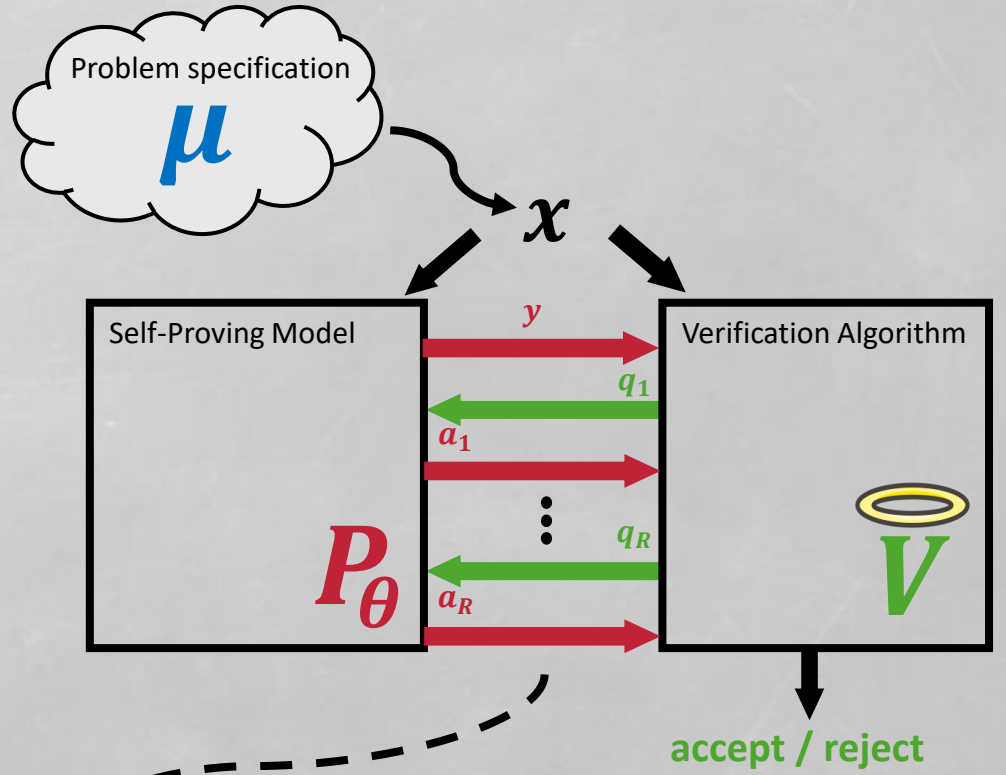
1. Generate transcript batch with P_θ. Keep only transcripts accepted by the Verifier.



RL from Verifier Feedback (RLVF)

Repeat the following:

1. Generate transcript batch with P_θ. Keep only transcripts accepted by the Verifier.

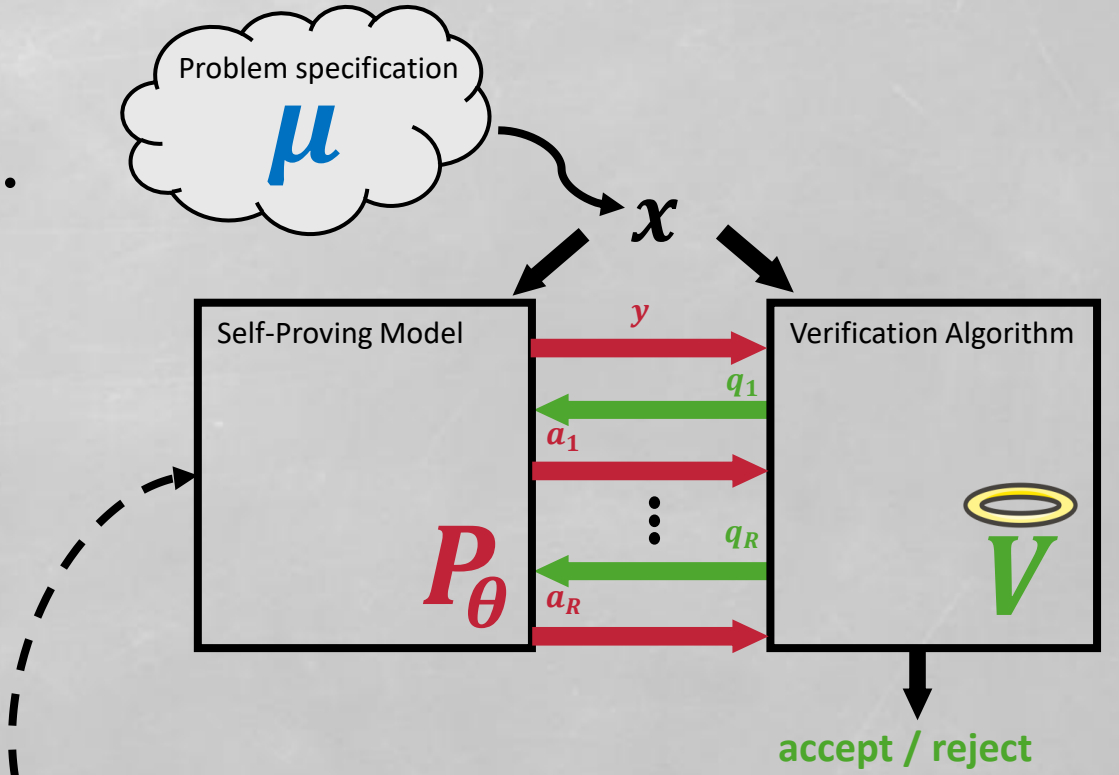


x	y	q_1	a_1	...	q_R	a_R
x	y	q_1	a_1	...	q_R	a_R
x	y	q_1	a_1	...	q_R	a_R

RL from Verifier Feedback (RLVF)

Repeat the following:

1. Generate transcript batch with P_θ. Keep only transcripts accepted by the Verifier.
2. Update θ towards accepted trans.



x	y	q	a	...	q	a
x	y	q	a	...	q	a
x	y	q	a	...	q	a

Lesson plan

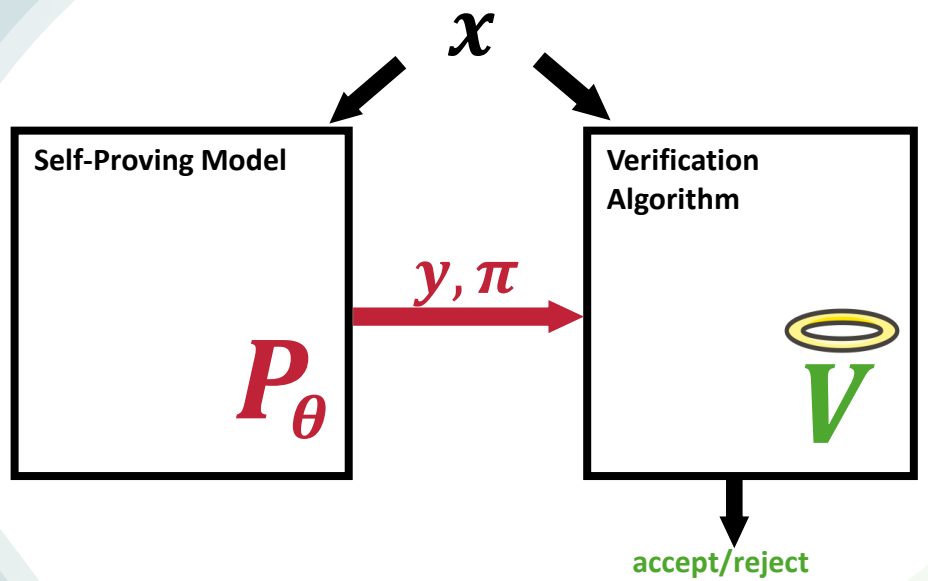
- Reminder: Proofs and Models
- Defining Self-Proving models
- Learning algorithms for Self-Proving models:
  - Transcript Learning (TL)
  - Reinforcement Learning from Verifier Feedback (RLVF)
- **Self-Proving models in practice**
- Proving convergence guarantees for TL and RLVF

Self-Proving models in practice

- “Learning to Prove” is a natural thing to do.
- It has been explored in many settings, mostly in practice.
- Let’s look at some of them:
  - In the ML literature
  - Experiments directly on TL and RLVF
  - RLVF adoption (sort of)

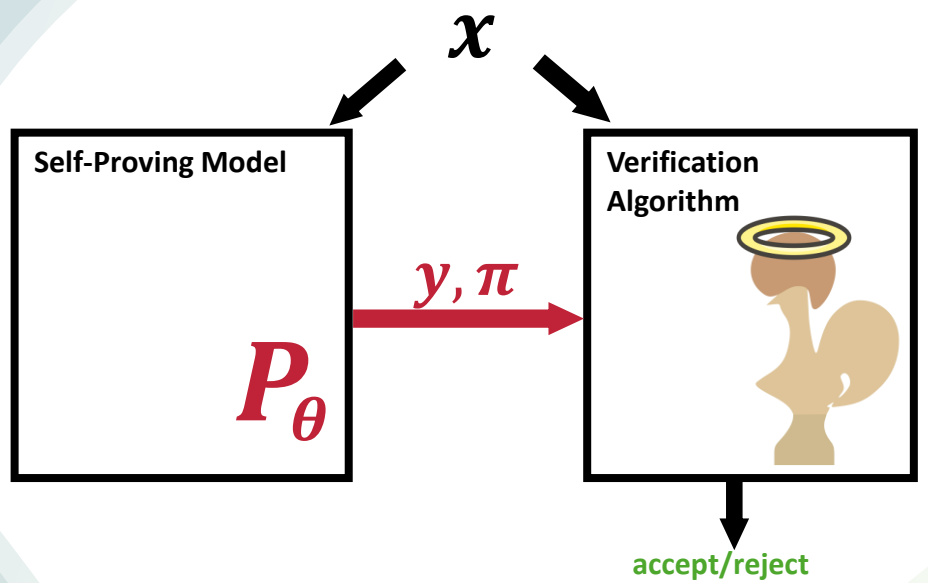
IP∩ML [May 2024]

- Learning to Prove…



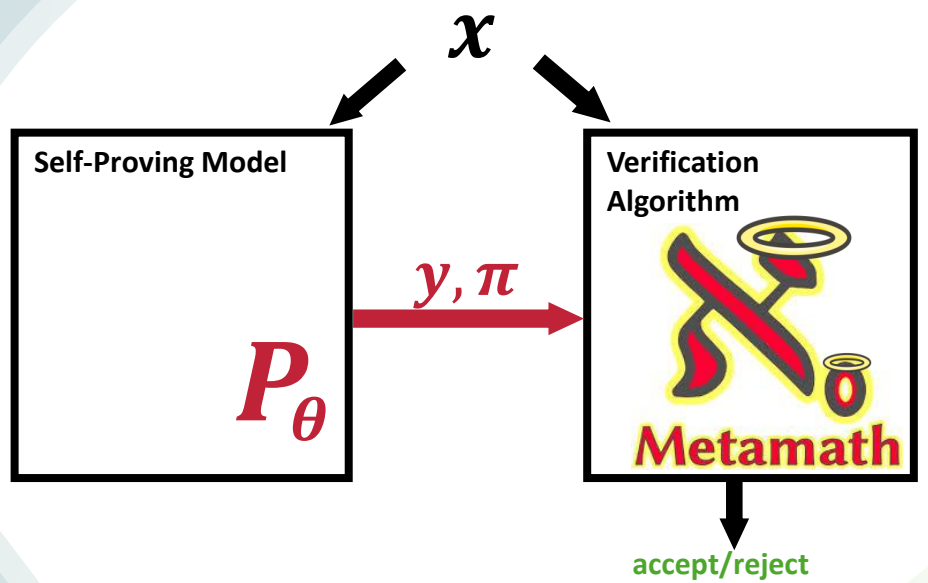
IP∩ML [May 2024]

- Learning to Prove…
  - **… in Coq (Gransden et al., 2015)**



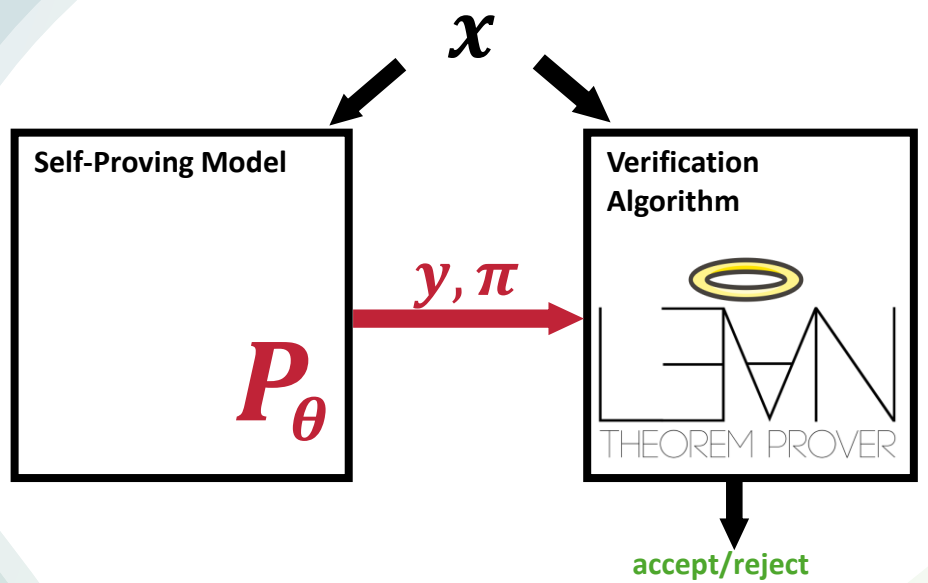
IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - **… in Metamath (Polu and Sutskever, 2020)**



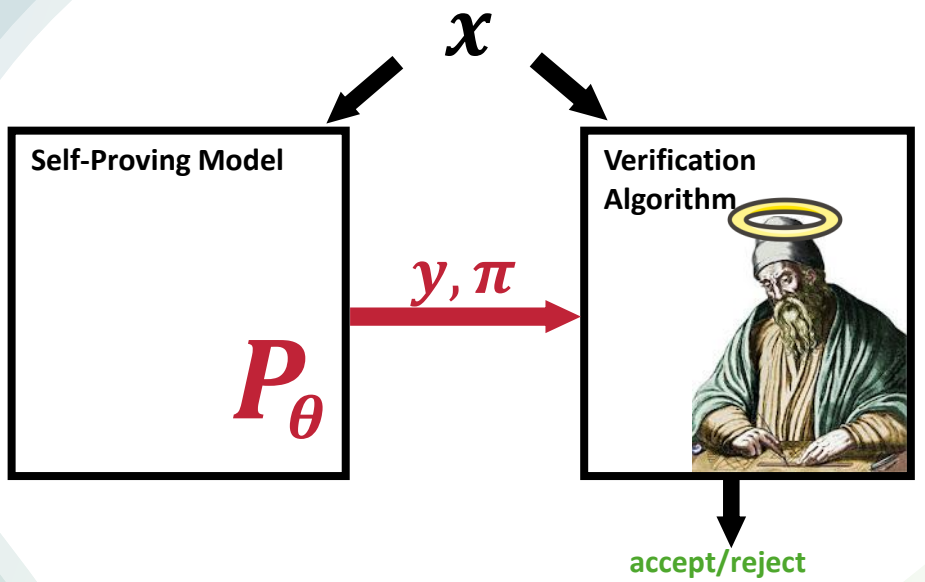
IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - **… in Lean (Yang et al., 2023)**



IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - **… in synthetic geometry (Trinh et al., 2024)**



Article | Open access | Published: 17 January 2024

Solving olympiad geometry without human demonstrations

Trieu H. Trinh ✉, Yuhuai Wu, Quoc V. Le, He He & Thang Luong ✉

Nature **625**, 476–482 (2024) | Cite this article

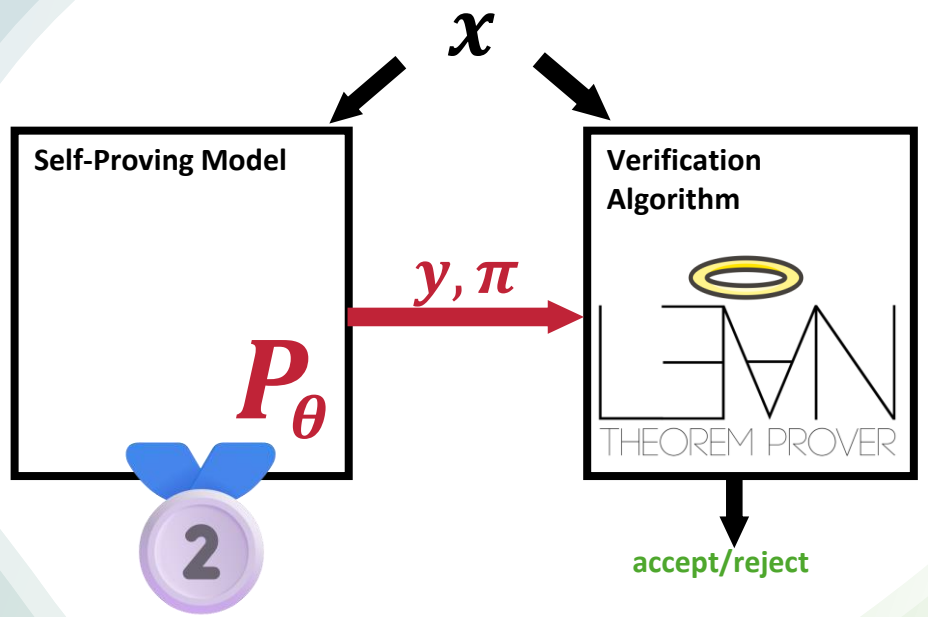
IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - **… in Lean 🥈 (DeepMind, 2024)**

AI achieves silver-medal standard solving International Mathematical Olympiad problems

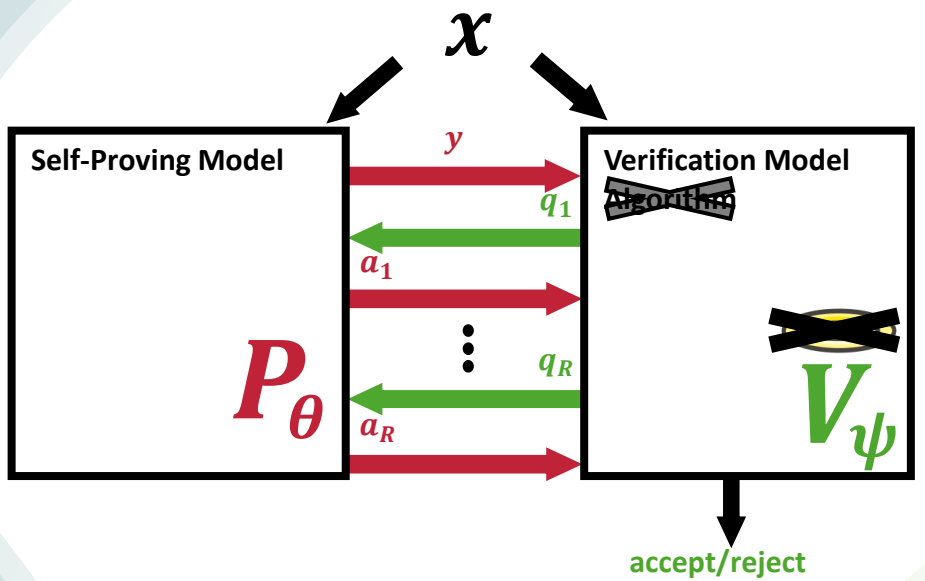
25 JULY 2024

AlphaProof and AlphaGeometry teams



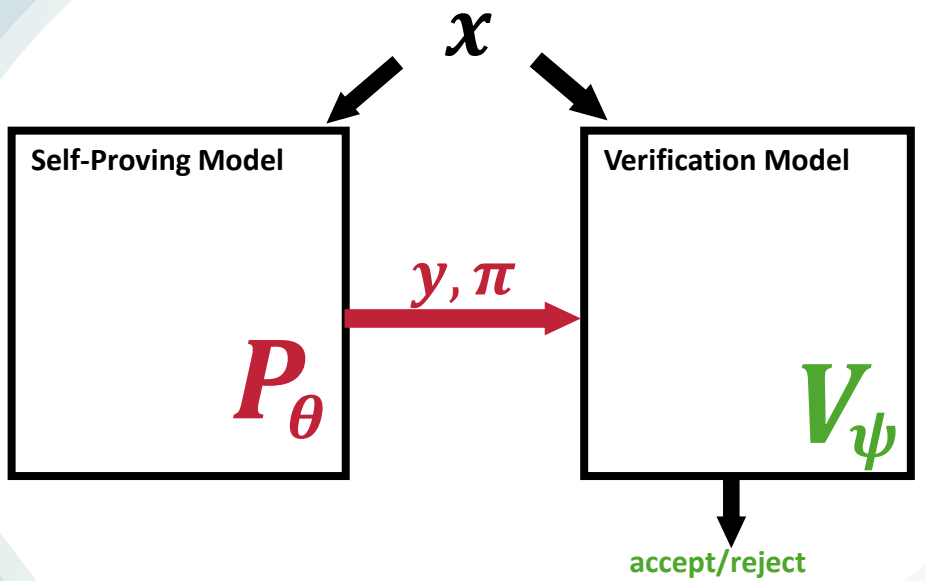
IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- **Learning to Verify**



IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - **Prover Verifier Games (Anil et al., 2021)**



Consider the class NP: (1) Non-interactive proofs, (2) Decision problems.

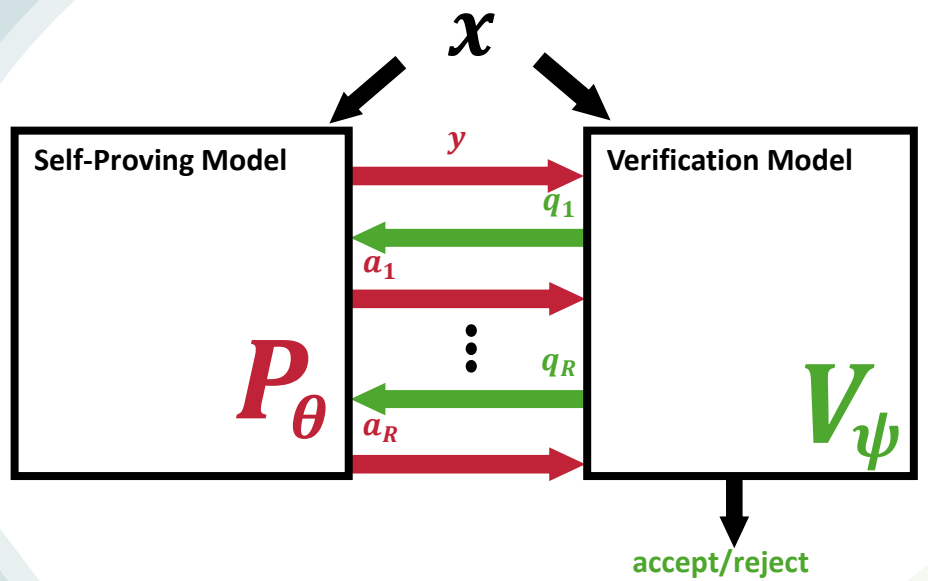
Consider a strategy-finding game with two players: P_θ and V_ψ.

Does an equilibrium ⚖️ → Completeness and Soundness? Does C&S → ⚖️ ?

Order	⚖️ →**C&S**	**C&S**→ ⚖️
P_θ first	No	No
V_ψ first	**Yes!**	**Yes!**
$\{P_\theta, V_\psi\}$ simul.	No	**Yes!**

IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - **Neural Interactive Proofs (Hammond & Adam-Day, 2024)**

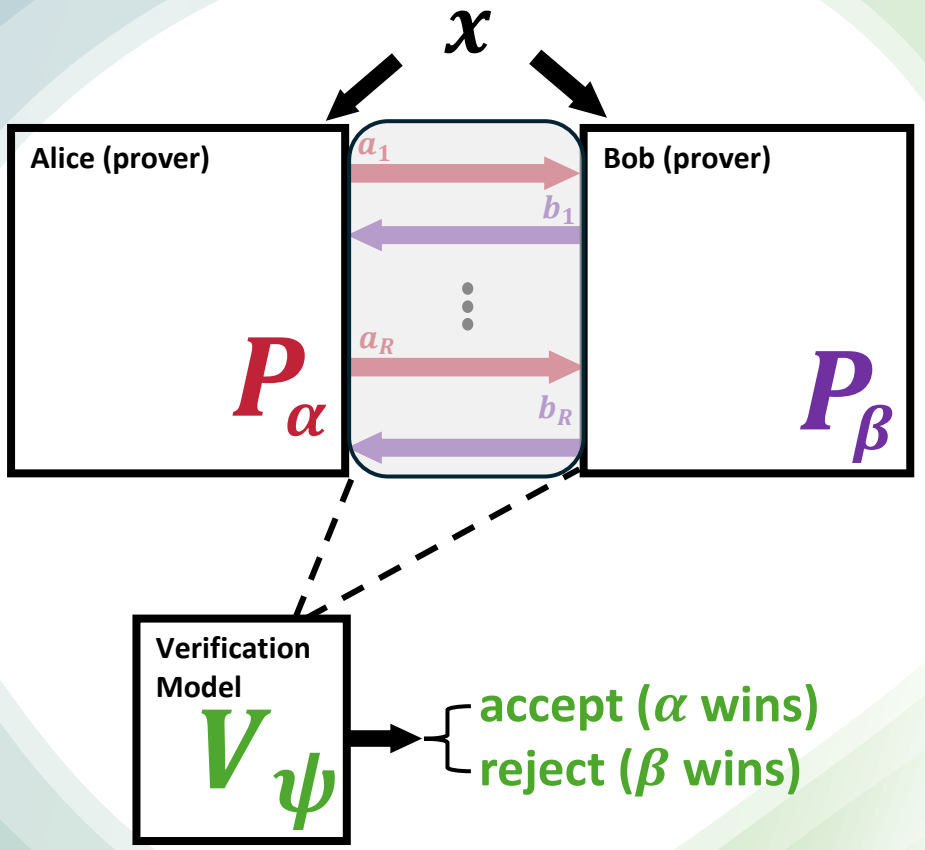


IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive Proofs (Hammond & Adam-Day, 2024)
- Safety and alignment
  - **Debate Systems for AI Safety (Irving et al., 2017)**

AI safety via debate

Geoffrey Irving* Paul Christiano Dario Amodei



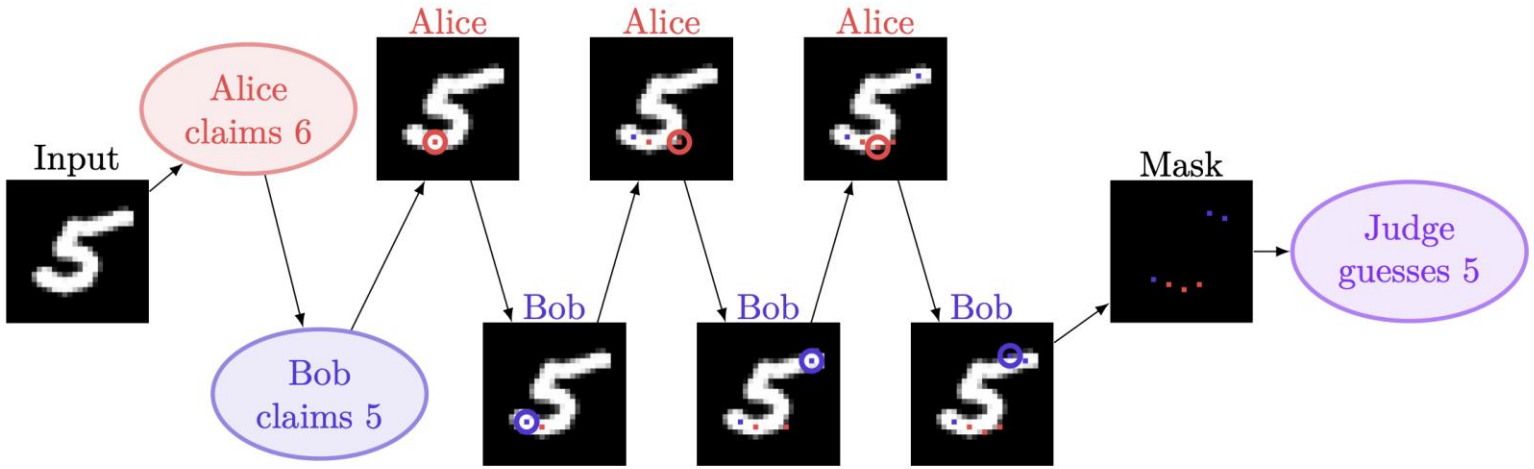
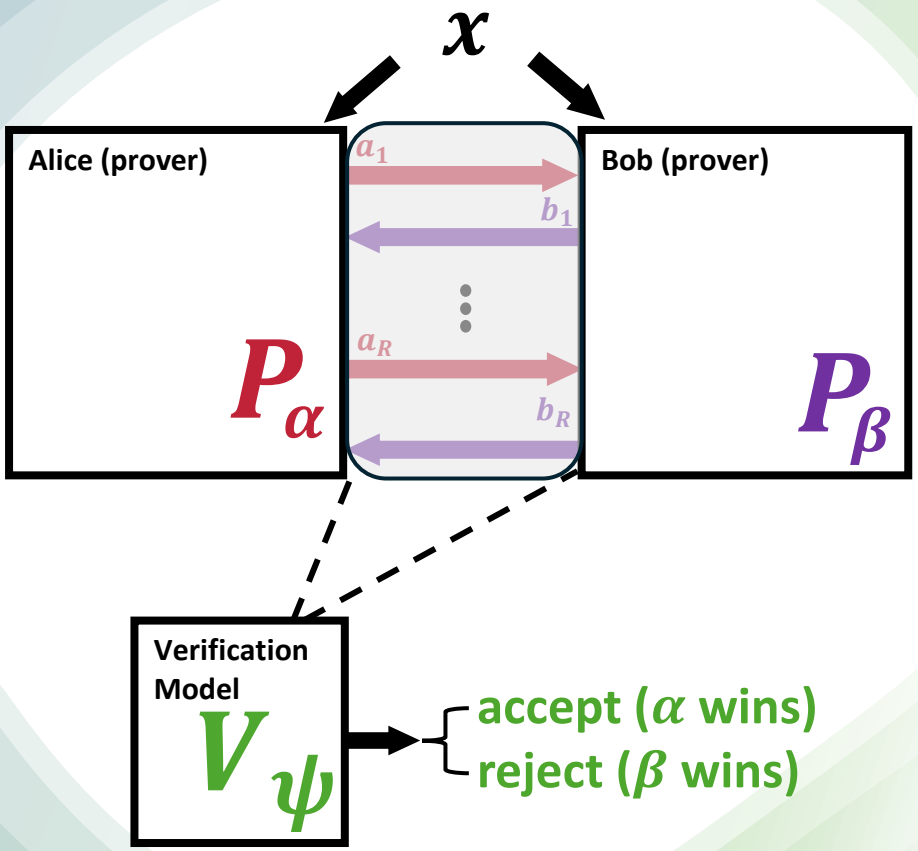


Figure 2: The MNIST debate game. A random MNIST image is shown to the two debating agents but not the judge. The debaters state their claimed label up front, then reveal one nonzero pixel per turn to the judge up to a total of 4 or 6. The judge sees the sparse mask of 4 or 6 pixels and chooses the winner based on which of the two labels has higher logit. The judge is trained in advance to recognize MNIST from random masks of nonzero pixels.

Pixels	First player	Judge accuracy (random pixels)	Honest win rate	
			No precommit	Precommit
4	honest	48.2%	51.0%	83.8%
	liar		68.4%	86.7%
	mean		59.7%	**85.2%**
6	honest	59.4%	67.4%	87.4%
	liar		81.5%	90.4%
	mean		74.4%	**88.9%**

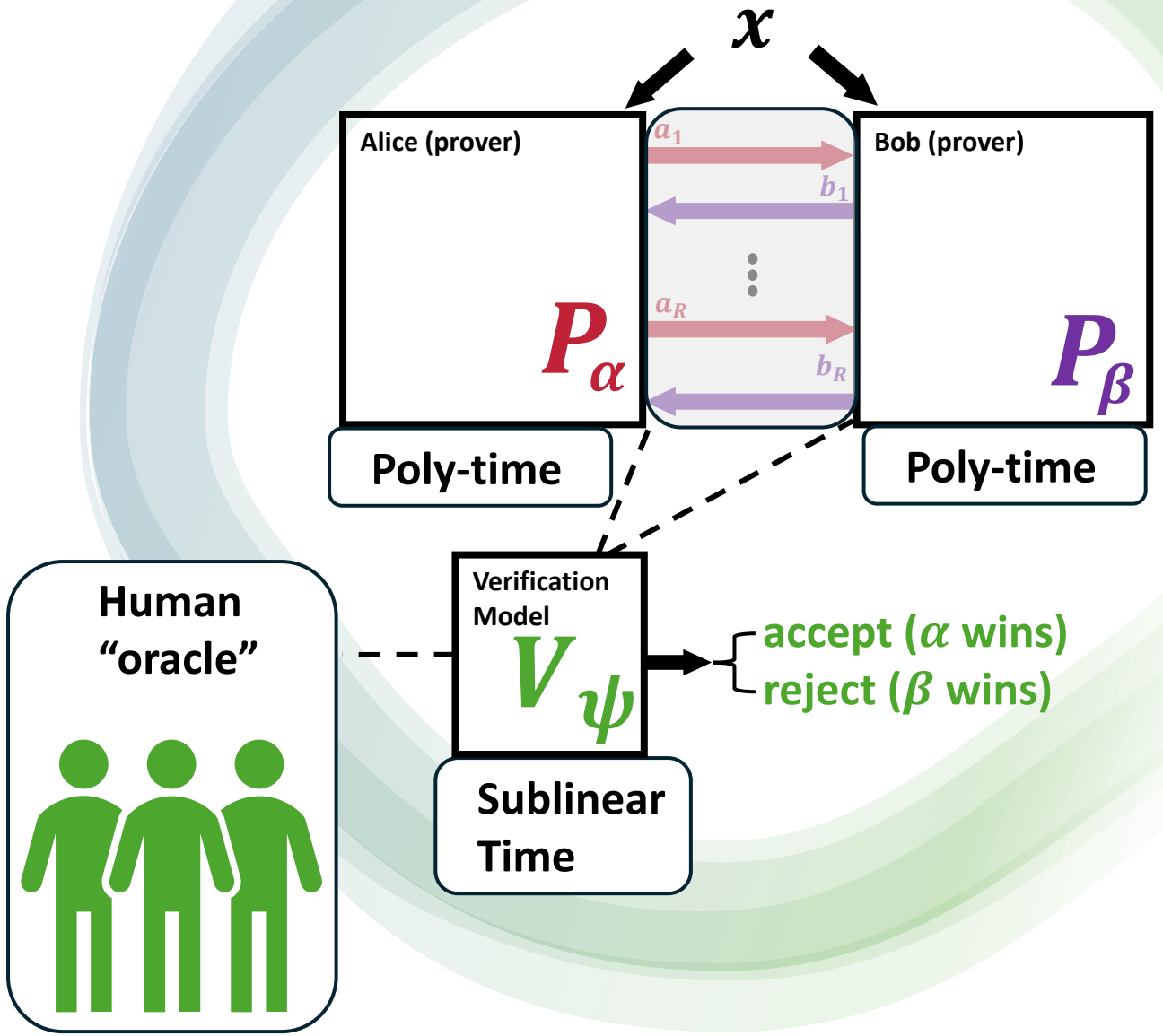
Table 2: Results for debate on MNIST. We prespecify one player as honest and one as liar: when the honest player wins honesty is the best strategy. *No precommit* means the liar wins for any incorrect guess by the judge, even if the incorrect guess differs for different parts of the game tree. Lying is harder in the *Precommit* case, where the liar states their claim in advance of making moves and cannot change their story. With or without precommit, the honest player wins more often than a random judge, showing that honesty has an advantage.



Debate Systems for AI Safety (Irving et al., 2017)

IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive Proofs (Hammond & Adam-Day, 2024)
- Safety and alignment
  - Debate Systems for AI Safety (Irving et al., 2017)
  - **Doubly-efficient debates for scalable AI safety (Brown-Cohen et al., 2024)**

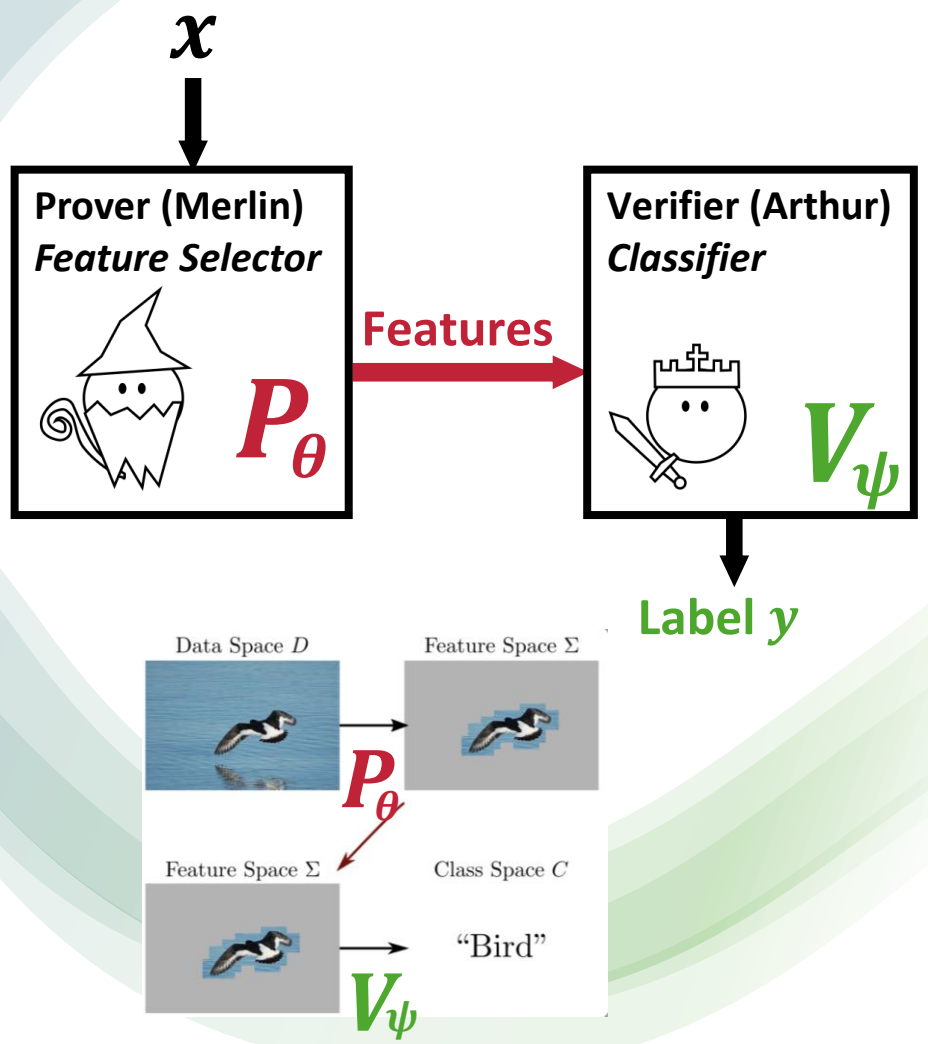


IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive Proofs (Hammond & Adam-Day, 2024)
- Safety and alignment
  - Debate Systems for AI Safety (Irving et al., 2017)
  - Doubly-efficient debates for scalable AI safety (Brown-Cohen et al., 2024)
- **Interpretability**

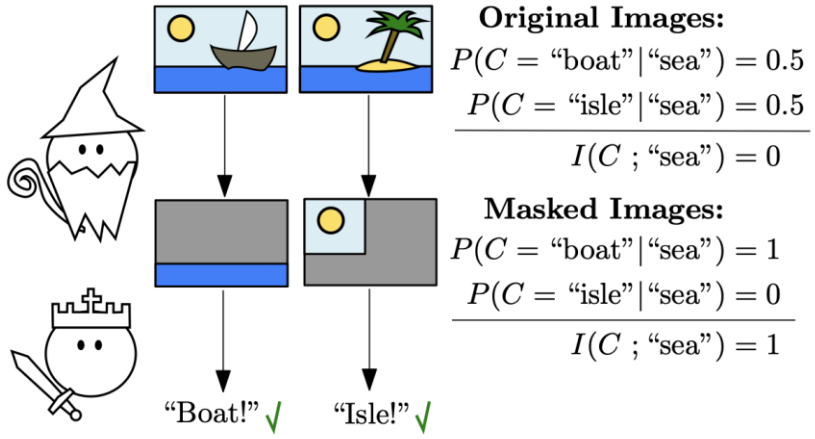
IP∩ML [May 2024]

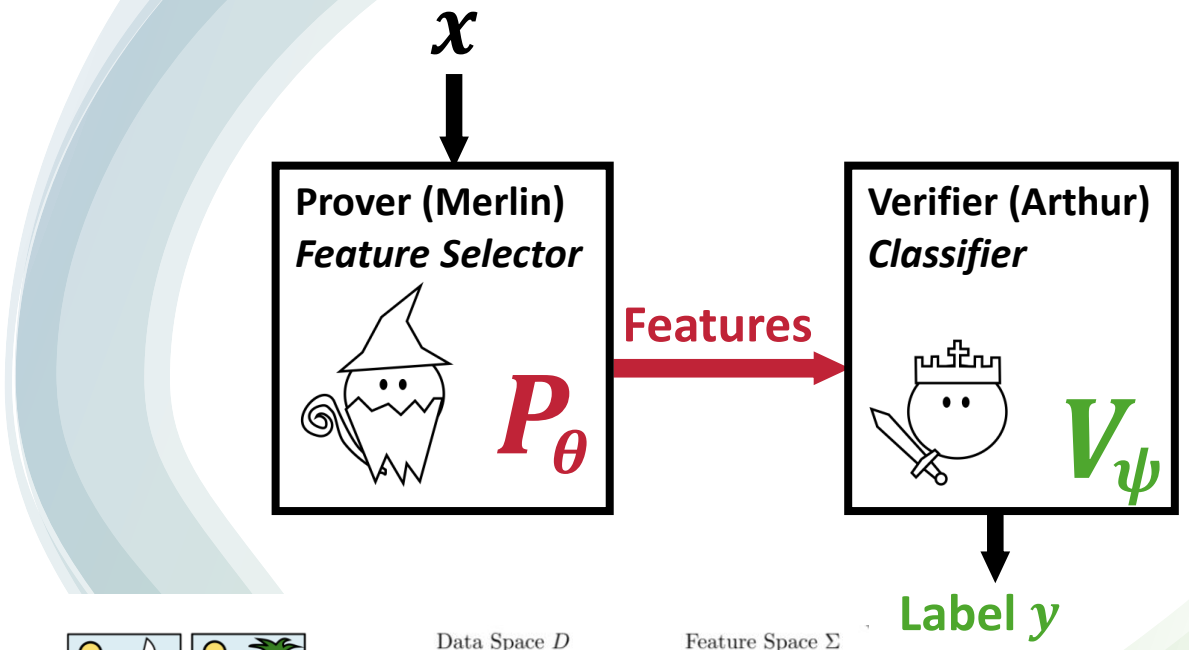
- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive Proofs (Hammond & Adam-Day, 2024)
- Safety and alignment
  - Debate Systems for AI Safety (Irving et al., 2017)
  - Doubly-efficient debates for scalable AI safety (Brown-Cohen et al., 2024)
- Interpretability
  - **MA Classifiers (Wäldchen et al., 2022)**



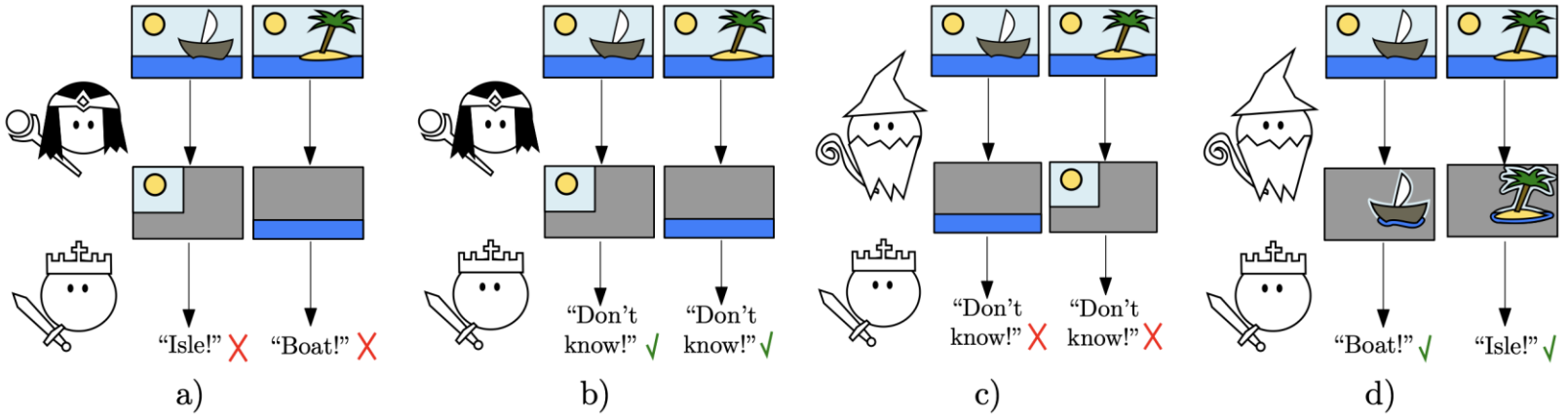
IP∩ML [May 2024]

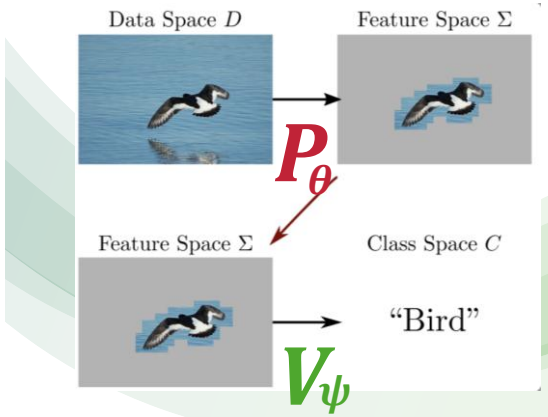
Problem: “Spurious features”





Solution: Soundness!

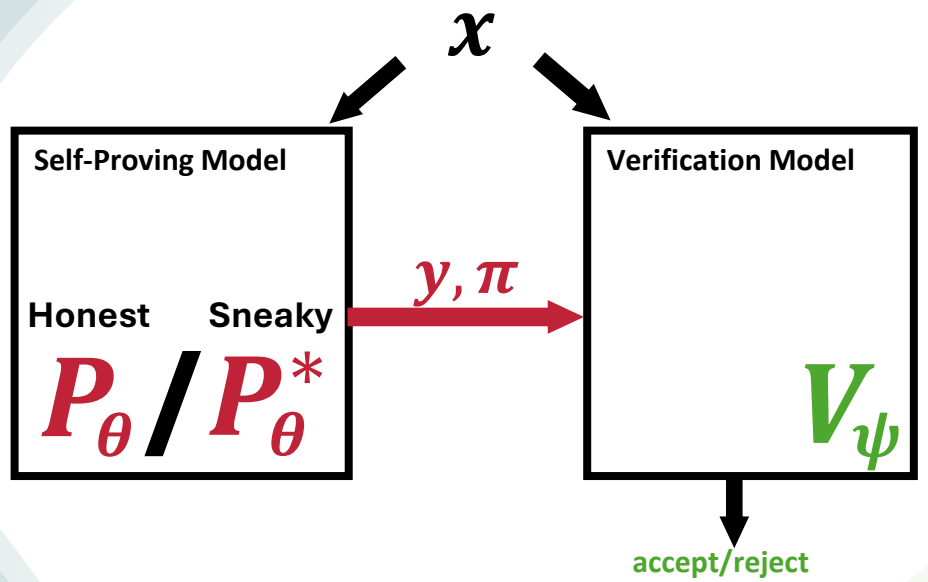




MA Classifiers (Wäldchen et al., 2022)

IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive Proofs (Hammond & Adam-Day, 2024)
- Safety and alignment
  - Debate Systems for AI Safety (Irving et al., 2017)
  - Doubly-efficient debates for scalable AI safety (Brown-Cohen et al., 2024)
- Interpretability
  - MA Classifiers (Wäldchen et al., 2022)
  - **Prov. Ver. Games for legibility (Hendrick* and Chen* et al., 2024)**



IP∩ML [May 2024]

Idea: Introduce a smaller LLM as a Verifier.
FT Prover LLM to act as honest / sneaky* prover

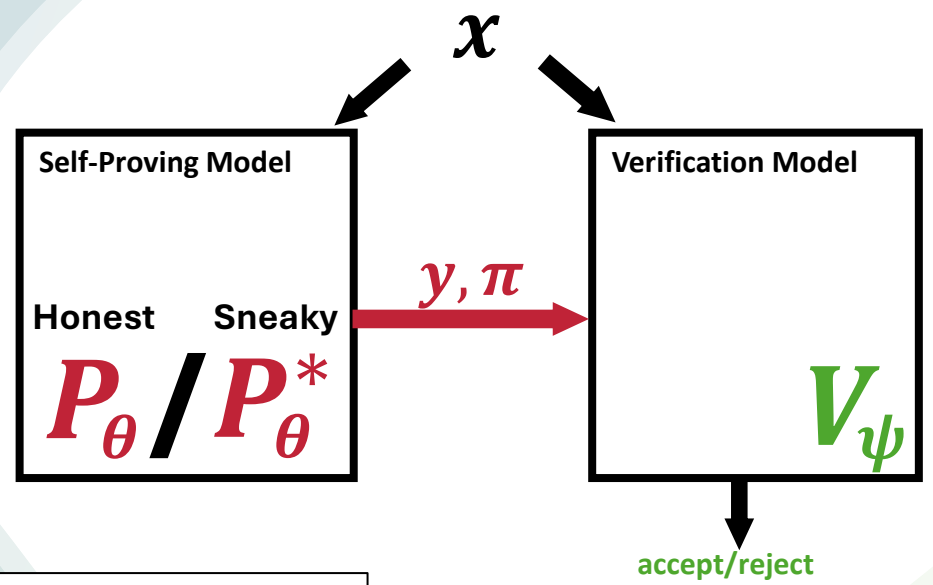
- Does this increase the (human) legibility of the Prover LLM's answers?

Joint training (simplified):

$$\psi \xleftarrow[\text{argmin}]{} L(\psi \mid \theta) \coloneqq \ell(\psi \mid P_\theta) + \ell(\psi \mid P_\theta^*)$$

$$\ell(\psi \mid P) \coloneqq -\mathbb{E}_x\left[H\left(\langle V_\psi, P\rangle(x), \mathbb{I}_{P(x)=y(x)}\right)\right]$$

Cross-entropy loss between V_ψ's decision logits (over 0,1) and the correctness of the prover.



IP∩ML [May 2024]

Idea: Introduce a smaller LLM as a Verifier.
FT Prover LLM to act as honest / sneaky* prover

- Does this increase the (human) legibility of the Prover LLM's answers?

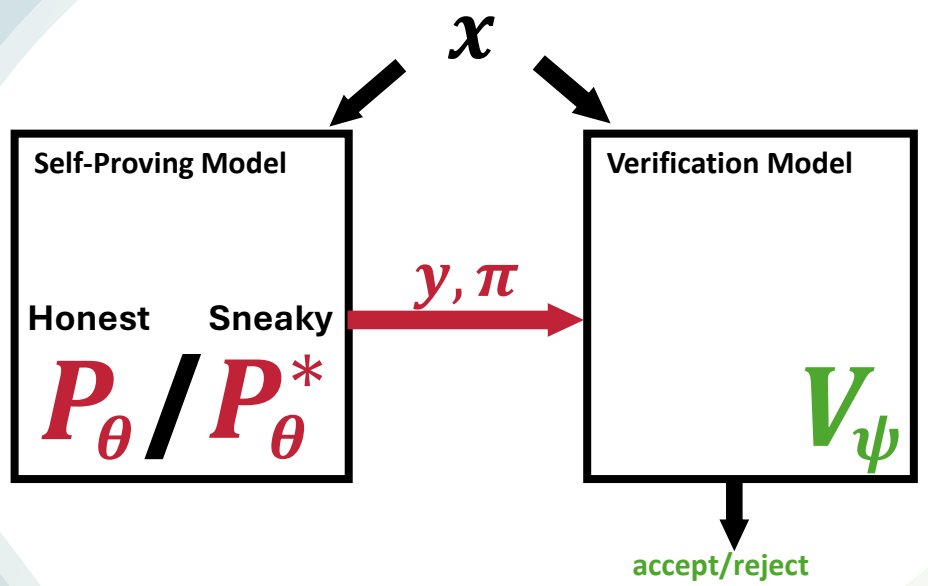
Joint training (simplified):

$$\psi \xleftarrow[\text{argmin}]{} L(\psi \mid \theta) := \ell(\psi \mid P_\theta) + \ell(\psi \mid P_\theta^*)$$

$$\ell(\psi \mid P) := -\mathbb{E}_x\left[H\left(\langle V_\psi, P\rangle(x), \mathbb{I}_{P(x)=y(x)}\right)\right]$$

$$\theta \xleftarrow[\text{argmin}]{} \mathbb{E}_x\left[\langle V_\psi, P_\theta\rangle(x) \cdot \mathbf{1}_{P_\theta(x)=y(x)} + \langle V_\psi, P_\theta^*\rangle(x) \cdot \mathbf{1}_{P_\theta^*(x)\neq y(x)}\right]$$

(simplified)



IP∩ML [May 2024]

Idea: Introduce a smaller LLM as a Verifier.
FT Prover LLM to act as honest / sneak

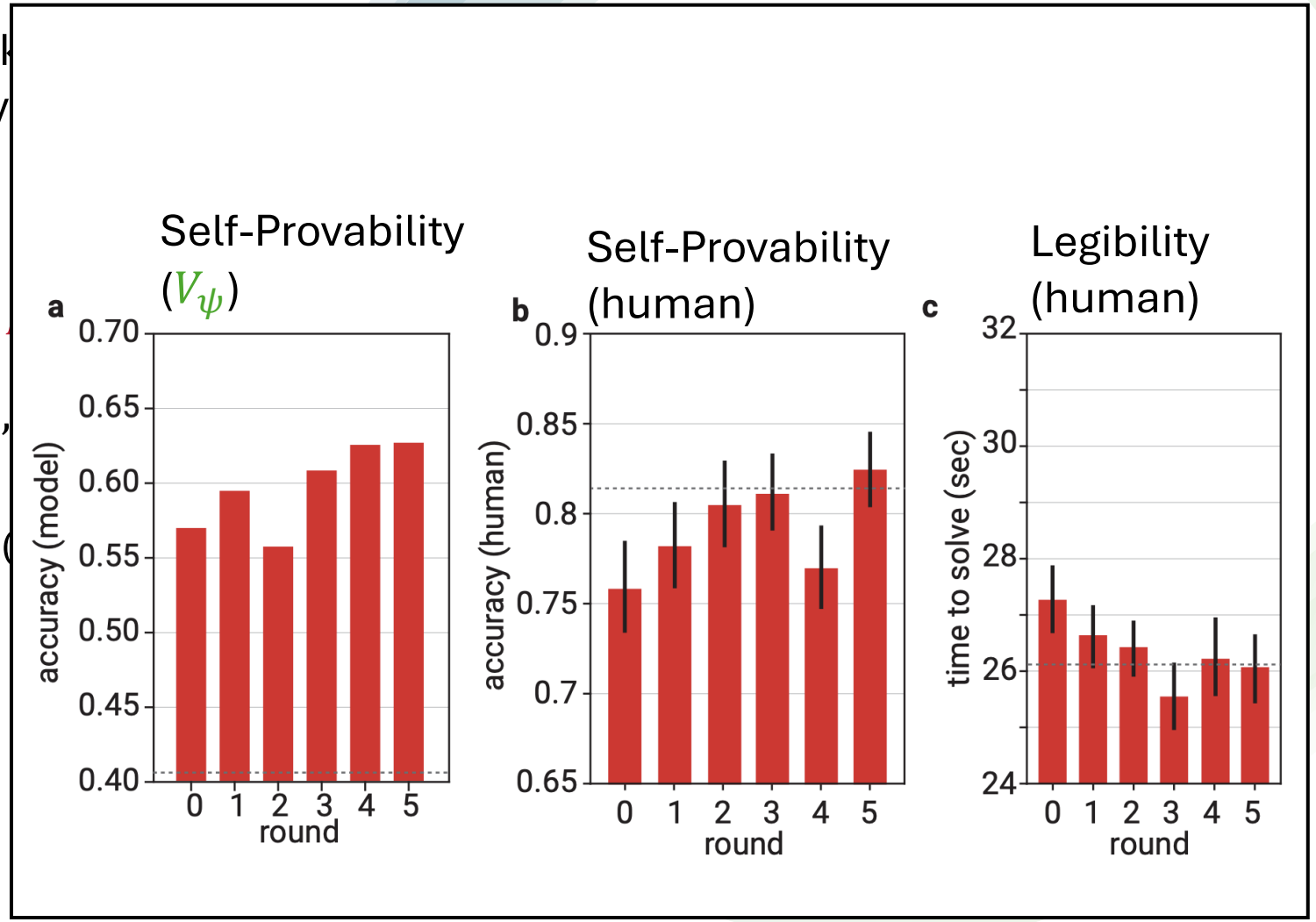
- Does this increase the (human) legibility LLM's answers?

Joint training (simplified):

$$\psi \xleftarrow[\text{argmin}]{} L(\psi \mid \theta) \coloneqq \ell(\psi \mid P_\theta) + \ell(\psi \mid$$

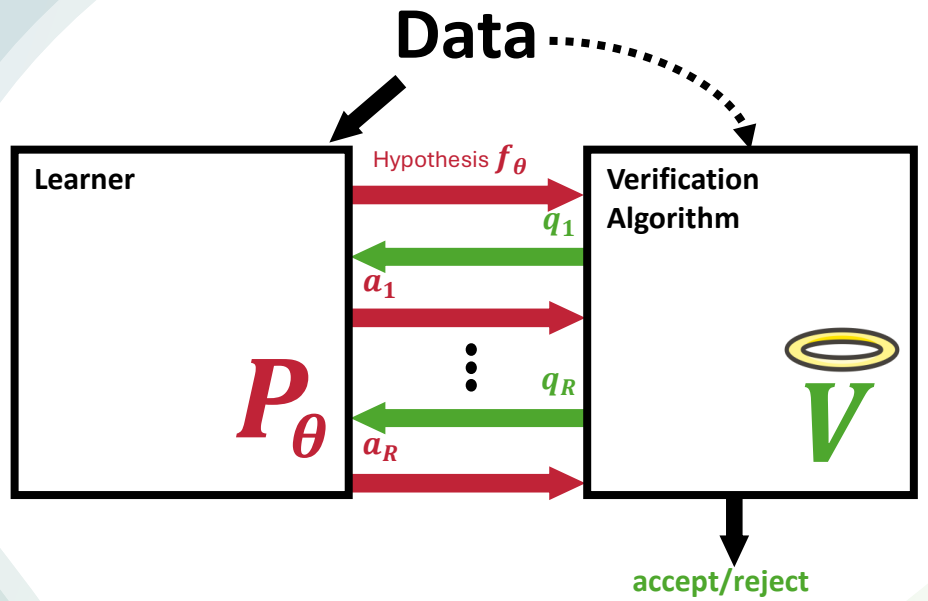
$$\ell(\psi \mid P) \coloneqq -\mathbb{E}_x\left[H\left(\langle V_\psi, P\rangle(x),\right.\right.$$

$$\theta \xleftarrow[\text{argmin}]{} \mathbb{E}_x\left[\langle V_\psi, P_\theta\rangle(x) \cdot \mathbf{1}_{P_\theta(x)=y(x)} + \langle V_\psi, P_\theta^*\rangle(\right.$$



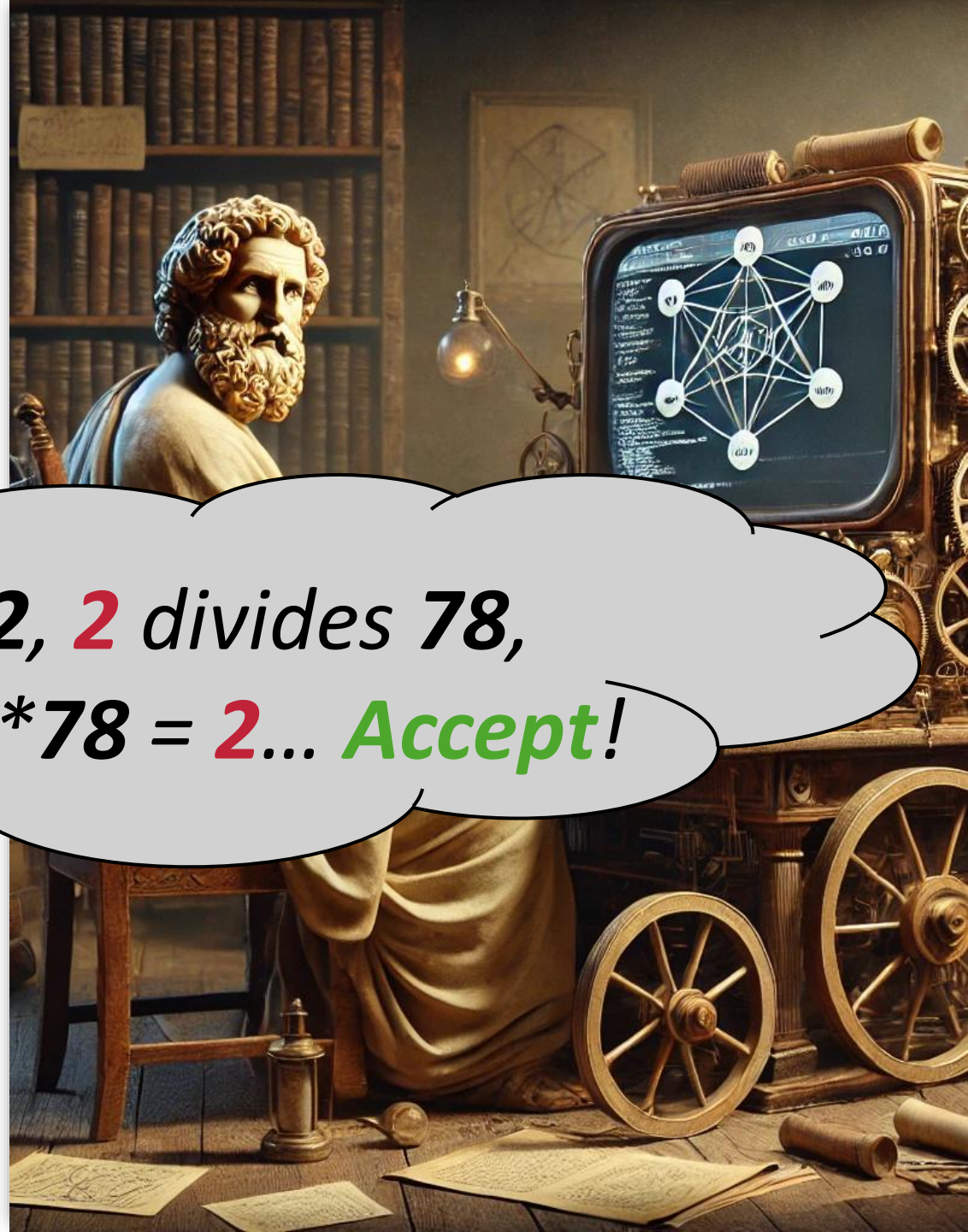
IP∩ML [May 2024]

- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive Proofs (Hammond & Adam-Day, 2024)
- Safety and alignment
  - Debate Systems for AI Safety (Irving et al., 2017)
  - Doubly-efficient debates for scalable AI safety (Brown-Cohen et al., 2024)
- Interpretability
  - MA Classifiers (Wäldchen et al., 2024)
  - Prov. Ver. Games for legibility (Hendrick* and Chen* et al., 2024)
- **Verifying global accuracy**
  - **(Goldwasser etl al., 2021; Mutreja and Shafer, 2023)**



Transcript Learning: Ancient Experiments [May '24]

- Charton (2024) showed that small GPT can learn to compute the GCD.
  *Can it prove that its answer is **correct**?*
- A proof system for GCD:
  - Bézout's identity: Let $(x_1, x_2) \in \mathbb{N}$
    if $z_1x_1 + z_2x_2$ divides x_1 and x_2,
    then $z_1x_1 + z_2x_2 = \mathbf{GCD}(x_1, x_2)$
  - V_{GCD} accepts iff $z_1x_1 + z_2x_2$ divides x_1 and x_2, a…



Transcript Learning experiments

Learning method	Correctness	Self-Provability
GPT (baseline)	99.8%	-
GPT + TL	98.8%	60.3%

Transcript Learning experiments

Learning method	Correctness	Self-Provability
GPT (baseline)	99.8%	-
GPT + TL	98.8%	60.3%
GPT + Annotated TL	98.6%	96.7%

- In practice, annotations speed up learning
  - Intermediate steps in Euclid's algorithm

Annotations

Extended Euclidean algorithm

Input: Nonzero integers $x_0, x_1 \in \mathbb{N}$.

Output: Integers (y, z_0, z_1), such that $y = GCD(x_0, x_1)$ and (z_0, z_1) are Bézout coefficients for (x_0, x_1).

1: Initialize $r_0 = x_0$, $r_1 = x_1$, $s_0 = 1$, $s_1 = 0$, and $q = 0$.
2: **while** $r_1 \neq 0$ **do**
3: Update $q := \lfloor r_0 / r_1 \rfloor$.
4: Update $(r_0, r_1) := (r_1, r_0 - q \times r_1)$.
5: Update $(s_0, s_1) := (s_1, s_0 - q \times s_1)$.
6: Output GCD $y = r_0$ and Bézout coefficients $z_0 := s_0$ and $z_1 := (r_0 - s_0 \cdot x_0)/x_1$.

Annotation

Input		GCD	Annotation			Bézout coefs	
x_0	x_1	y	$\vec{s_0}$	$\vec{r_0}$	$\vec{q}$	z_0	z_1
46	39		1	46	1		
			0	39	5		
			1	7	1		
			−5	4	1		
			6	3	3		
		1				−11	13

Annotations

Extended Euclidean algorithm

Input: Nonzero integers $x_0, x_1 \in \mathbb{N}$.

Output: Integers (y, z_0, z_1), such that $y = GCD(x_0, x_1)$ and (z_0, z_1) are Bézout coefficients for (x_0, x_1).

1: Initialize $r_0 = x_0$, $r_1 = x_1$, $s_0 = 1$, $s_1 = 0$, and $q = 0$.
2: **while** $r_1 \neq 0$ **do**
3: Update $q := \lfloor r_0 / r_1 \rfloor$.
4: Update $(r_0, r_1) := (r_1, r_0 - q \times r_1)$.
5: Update $(s_0, s_1) := (s_1, s_0 - q \times s_1)$.
6: Output GCD $y = r_0$ and Bézout coefficients $z_0 := s_0$ and $z_1 := (r_0 - s_0 \cdot x_0)/x_1$.

Annotation

Input GCD Annotation Bézout coefs

x_0	x_1	y	$\vec{s_0}$	$\vec{r_0}$	$\vec{q}$	z_0	z_1
46	39		1	46	1		
			0	39	5		
			1	7	1		
			−5	4	1		
			6	3	3		
		1				−11	13

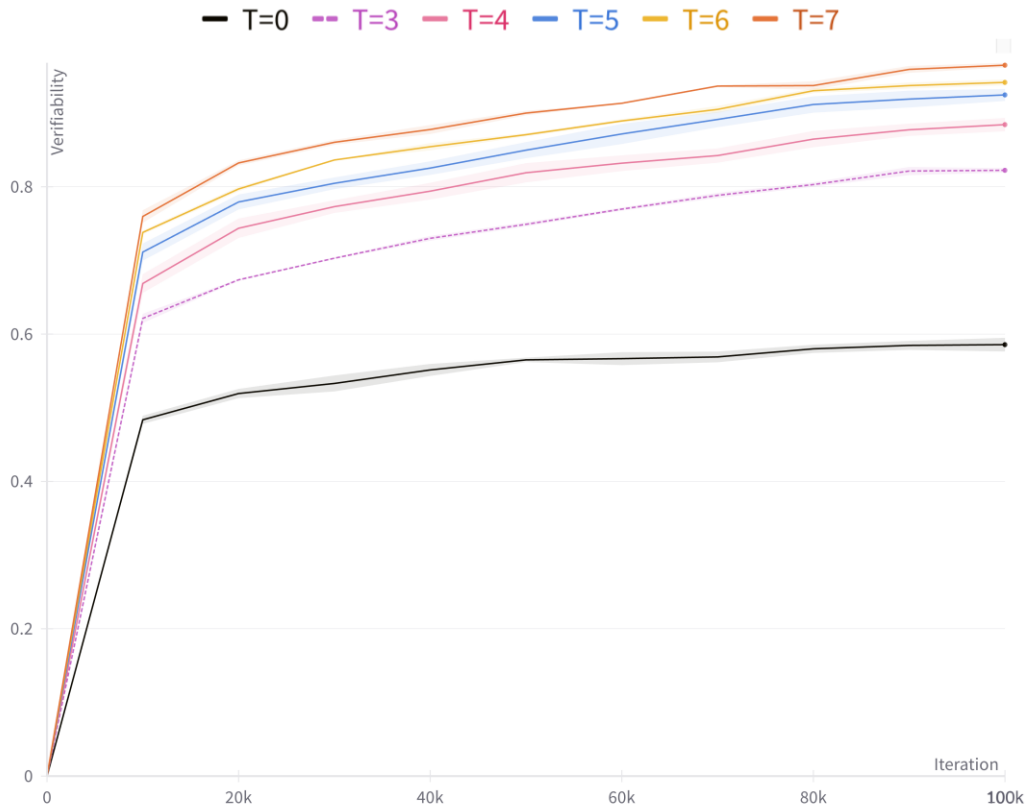
Decimal encoding

+,4,6,x0,+,3,9,x1,+,1,y,
+,1,z0',+,4,6,z1',+,1,q',
+,0,z0'',+,3,9,z1'',+,5,q''
+,1,z0''',+,7,z1''',+,1,q''',
-,1,1,z0,+,1,3,z1

sign tokens
digits
delimiters

Annotated Transcript Learning

Longer annotations help



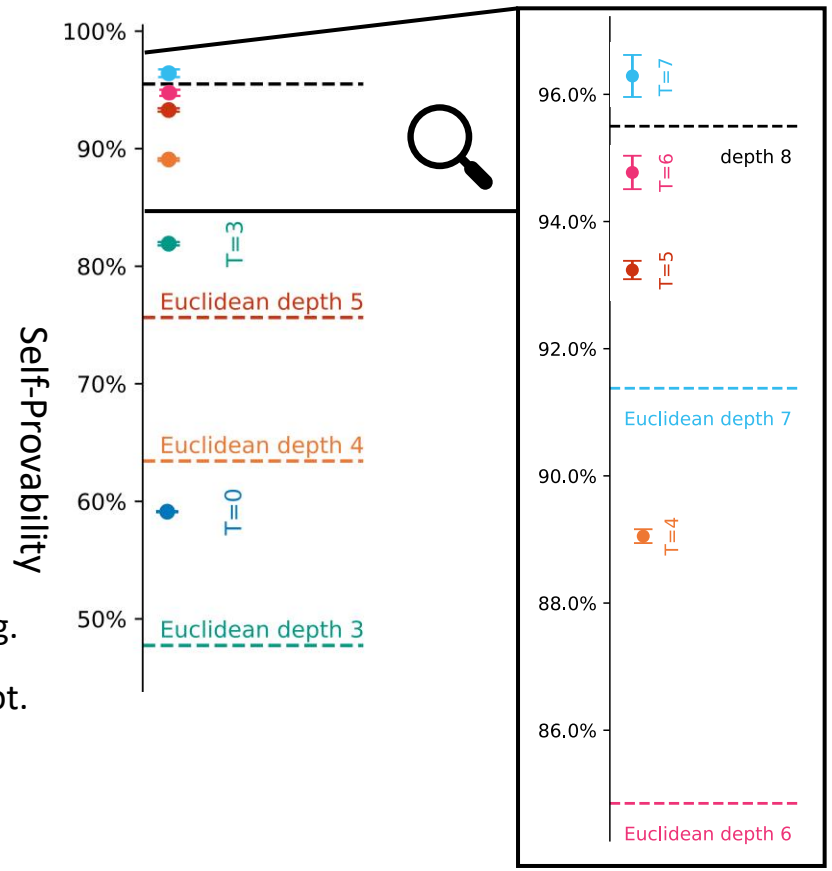
Models generalize beyond annotations

Input | Annotation

x_0	x_1	y	$\vec{s_0}$	$\vec{r_0}$	$\vec{q}$
46	39		1	46	1
			0	39	5
			1	7	1
			−5	4	1
			6	3	3
		1			

Depth(x_0, x_1) = #steps in Euclidean alg.

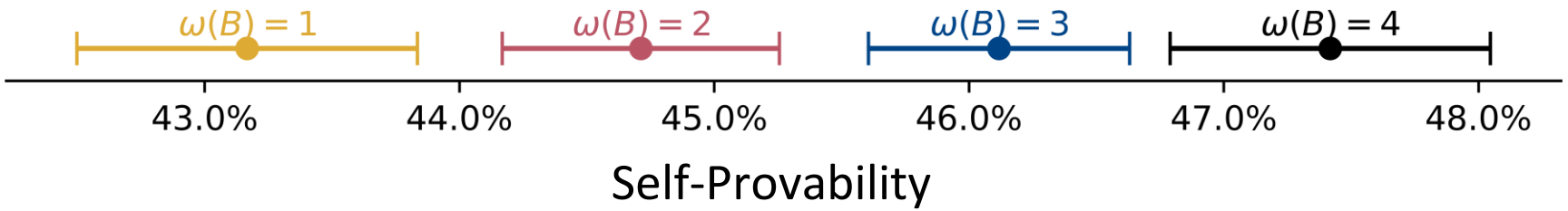
= length of “ideal” annot.



Base of Representation matters

"Early during training, transformers learn to predict products of divisors of the base ***B*** *used to represent integers."* (Charton, 2024)

- Let $\omega(B)$ denote the number of primes in the factorization of base B.
- Then $\omega(B)$ determines Self-Provability (similarly to Charton's observation).



RLVF: from Theory to Practice

RLVF introduced as a method for training Self-Proving models (Amit et al.)

May 2024

Lemma B.6 (RLVF gradient estimation). *Fix an input distribution μ over Σ^* a*
V with round complexity R and answer length L_a. For any transcript $(x, y, q_1, \ldots$
$\mathrm{Acc}_V(x, y, q_1, \ldots, a_R)$ denote the indicator random variable which equals 1 if and o
cepts the transcript. For any model P_θ, denote by $\mathrm{ver}(\theta)$ the verifiability of P_θ with resp
μ (Definition 3.4). Then, for any θ,

$$\nabla_\theta \mathrm{ver}(\theta) = \mathop{\mathbb{E}}_{\substack{x\sim\mu \\ y\sim P_\theta(x) \\ (q_r,a_r)_{r=1}^R}} \left[\mathrm{Acc}_V(x, y, q_1, \ldots, a_R) \cdot \sum_{\substack{r\in[R]\cup\{0\} \\ s\in[L_a]}} \vec{d}_s(\theta) \right]$$

where $(q_r, a_r)_{r=1}^R$ are as sampled in lines 5-6 of Algorithm 2, and $\vec{d}_s(\theta)$ is as defined in l

Algorithm 2: Reinforcement Learning from Verifier Feedback (RLVF)

Hyperparameters: Learning rate $\lambda \in (0, 1)$ and number of samples $N \in \mathbb{N}$.

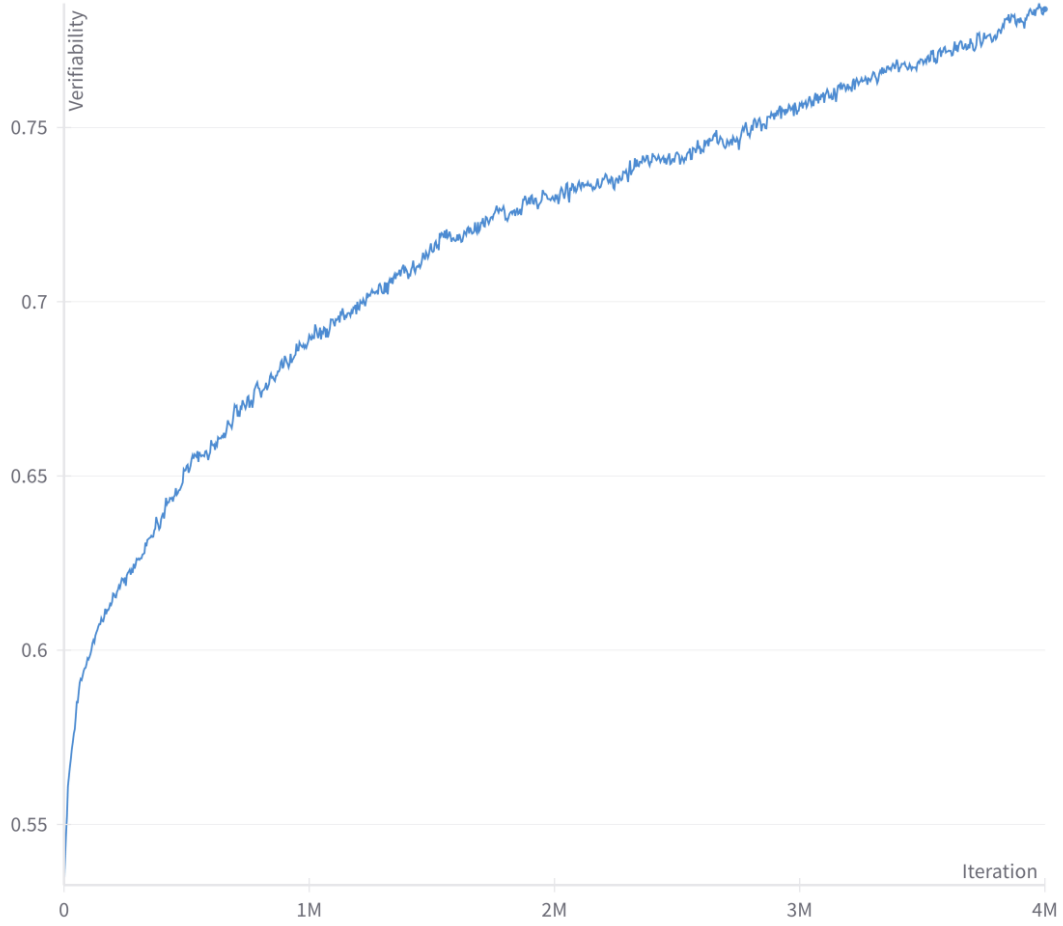


Figure 6: **RLVF Verifiability as a function of the number of samples N.** Starting from a base model with Verifiability 48% (obtained via Transcript Learning), in each iteration a batch of 2048 inputs are sampled; the model generates a proof for each; the Verifier is used to check which proofs are accepted; then, the model parameters are updated accordingly (see Algorithm 2). Verifiability was evaluated on a held-out dataset of 1k inputs.

RLVF: from Theory to Practice

RLVF introduced as a method for training Self-Proving models (Amit et al.)

May 2024

Nov 2024

RLVR (Lambert et al.) implements RLVF for Tülu3

+ KL regularization, RM initialization, data shuffling, No-EOS penalty, advantage normalization

+ ZeRO 3 (Rajbhandari et al. 2020), asynchronous RL (Noukhovitch et al. 2024), ablations…

→ Beats DPO on GSM8K, MATH, IFEval at 8B params

RLVF: from Theory to Practice

RLVF introduced as a method for training Self-Proving models (Amit et al.)

May 2024

Nov 2024

RLVR (Lambert et al.) implements RLVF for Tülu3

+ KL regularization, RM initialization, data shuffling, No-EOS penalty, advantage normalization

+ ZeRO 3 (Rajbhandari et al. 2020), asynchronous RL (Noukhovitch et al. 2024), ablations…

→ Beats DPO on GSM8K, MATH, IFEval at 8B params

Med-RLVR (Zheng et al.)

Feb 2025

RLVF: from Theory to Practice

RLVF introduced as a method for training Self-Proving models (Amit et al.)

May 2024

Nov 2024

RLVR (Lambert et al.) implements RLVF for Tülu3

+ KL regularization, RM initialization, data shuffling, No-EOS penalty, advantage normalization

+ ZeRO 3 (Rajbhandari et al. 2020), asynchronous RL (Noukhovitch et al. 2024), ablations…

→ Beats DPO on GSM8K, MATH, IFEval at 8B params

Feb 2025

🩺 **Med-RLVR** (Zheng et al.)

May 2025

👀 **RLVR-World** (Wu et al.)

🤖 **ManipLVM-R1** (Song et al.)

💸 **REcon** (Zhou et al.)

RLVF: from Theory to Practice

RLVF introduced as a method for training Self-Proving models (Amit et al.)

May 2024

Nov 2024

RLVR (Lambert et al.) implements RLVF for Tülu3

+ KL regularization, RM initialization, data shuffling, No-EOS penalty, advantage normalization

+ ZeRO 3 (Rajbhandari et al. 2020), asynchronous RL (Noukhovitch et al. 2024), ablations…

→ Beats DPO on GSM8K, MATH, IFEval at 8B params

Med-RLVR (Zheng et al.)

Feb 2025

May 2025

RLVR-World (Wu et al.)

ManipLVM-R1 (Song et al.)

REcon (Zhou et al.)


The Hills are Alive.....?


V
V
V
V
V
The Hills are Alive....?


So far, Verifiers have rarely been **provably sound**
RLVF was shown effective for post-training capabilities. What about **worst-case guarantees**?
The Hills are Alive....?

Lesson plan

- Reminder: Proofs and Models
- Defining Self-Proving models
- Learning algorithms for Self-Proving models:
  - Transcript Learning (TL)
  - Reinforcement Learning from Verifier Feedback (RLVF)
- Self-Proving models in practice
- **Proving convergence guarantees for TL and RLVF**

TL convergence

Algorithm 1: Transcript Learning (TL)

Hyperparameters: Learning rate $\lambda \in (0,1)$ and number of samples $N \in \mathbb{N}$.

Input: An autoregressive model family $\{P_\theta\}_{\theta \in \mathbb{R}^d}$, verifier specification (code) V, and sample access to an input distribution μ and an accepting transcript generator $\mathcal{T}^*_V(\cdot)$.

Output: A vector of parameters $\bar{\theta} \in \mathbb{R}^d$.

Initialize $\theta_0 := \vec{0}$.

for $i = 0, \ldots, N-1$ **do**

  Sample $x \sim \mu$ and $\pi^* = (y^*, q_1^*, a_1^*, \ldots, q_R^*, a_R^*) \sim \mathcal{T}^*_V(x)$. Denote $a_0 := y^*$.

  foreach *Round of interaction* $r = 0, \ldots, R$ **do**

    Let $S(r)$ denote the indices of the r^{th} answer a_r in π^*, and let $\pi_{<s}$ denote the prefix of the partial transcript $(y, q_1^*, a_1^*, \ldots, q_r^*)$.

    for $s \in S(r)$ **do**

      Compute `# Forwards and backwards pass`

$$\alpha_s(\theta_i) := \Pr_{\sigma \sim p_{\theta_i}(x\pi_{<s})}[\sigma = \pi_s^*]$$

$$\vec{d}_s(\theta_i) := \nabla_\theta \log \alpha_s(\theta_i) = \nabla_\theta \log \Pr_{\sigma \sim p_{\theta_i}(x\pi_{<s})}[\sigma = \pi_s^*].$$

  Update

$$\theta_{i+1} := \theta_i + \lambda \cdot \prod_{\substack{r \in [R] \cup \{0\} \\ s \in S(r)}} \alpha_s(\theta_i) \cdot \sum_{\substack{r \in [R] \cup \{0\} \\ s \in S(r)}} \vec{d}_s(\theta_i).$$

Output $\bar{\theta} := \frac{1}{N} \sum_{i \in [N]} \theta_i$.

Lemma (TL gradient estimation). *Fix an input distribution μ over Σ^* and a verifier V with round complexity R and answer length L_a. Fix an honest transcript generator $\mathcal{T}^*_V$. Let θ denote the parameters of the model P_θ, let $A(\theta)$ be as defined above and let the terms $S(r)$, $\alpha_s(\theta)$, and $\vec{d}_s(\theta)$ be as defined in Algorithm 1. Then,*

$$\nabla A(\theta) = \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ \pi^* \sim \mathcal{T}^*_V}} \left[\prod_{\substack{r \in [R] \cup \{0\} \\ s \in S(r)}} \alpha_s(\theta) \cdot \sum_{\substack{r \in [R] \cup \{0\} \\ s \in S(r)}} \vec{d}_s(\theta) \right].$$

Theorem *Fix a verifier V, an input distribution μ, an autoregressive model family $\{P_\theta\}_{\theta \in \mathbb{R}^d}$, and a norm $\|\cdot\|$ on $\mathbb{R}^d$. Fix an honest transcript generator $\mathcal{T}^*_V$, and assume that the* agreement function $A(\theta) := \Pr[\pi = \pi^*]$ *is concave in θ, where the verifier queries are the same in π^* and π, and the probability is over $x \sim \mu$, $\pi_\theta \sim \mathcal{T}^\theta_V(x)$, and $\pi^* \sim \mathcal{T}^*_V(x)$. For any $\varepsilon > 0$, let B_{Norm}, B_{Lip} and C be upper-bounds such that the following conditions hold:*

- *There exists $\theta^* \in \mathbb{R}^d$ with $\|\theta^*\| < B_{\text{Norm}}$ such that $A(\theta^*) \geq 1 - \varepsilon/2$.*
- *For all θ, the logits of P_θ are B_{Lip}-Lipschitz in θ. That is, $\sup_{\substack{\theta \in \mathbb{R}^d \\ z \in \Sigma^*}} \|\nabla_\theta \log p_\theta(z)\| \leq B_{\text{Lip}}$.*
- *In the proof system defined by V, the total number of tokens (over all rounds) is at most C.*

Denote by $\bar{\theta}$ the output of TL running for $N \geq \left(4 \cdot C^2 \cdot B_{\text{Norm}}^2 \cdot B_{\text{Lip}}^2\right)/\varepsilon^2$ iterations and learning rate $\lambda = B_{\text{Norm}}/CB_{\text{Lip}}\sqrt{N}$. Then the expected Verifiability (over the randomness of the samples collected by TL) of $\bar{\theta}$ is at least $1 - \varepsilon$. That is, $\mathbb{E}_{\bar{\theta}}[\text{ver}_{V,\mu}(\bar{\theta})] \geq 1 - \varepsilon$.

Towards RLVF convergence

Algorithm 2: Reinforcement Learning from Verifier Feedback (RLVF)

Hyperparameters: Learning rate $\lambda \in (0,1)$ and number of samples $N \in \mathbb{N}$.
Input: An autoregressive model family $\{P_\theta\}_{\theta \in \mathbb{R}^d}$, initial parameters $\theta_0 \in \mathbb{R}^d$, verifier specification (code) V, and sample access to an input distribution μ.
Output: A vector of parameters $\bar{\theta} \in \mathbb{R}^d$.

1 **for** $i = 0, \ldots, N-1$ **do**
2 Sample $x \sim \mu$.
3 Initialize $a_0 \coloneqq y \sim P_{\theta_i}(x)$.
4 **foreach** *Round of interaction* $r = 1, \ldots R$ **do**
5 Sample the r^{th} query `# Emulate the verifier`

$$q_r \sim V_q(x, a_0, q_1, a_1, \ldots, q_{r-1}, a_{r-1}).$$

  Sample the r^{th} answer `# Forwards pass`

$$a_r \sim P_{\theta_i}(x, a_0, q_1, a_1, \ldots, q_r).$$

  Let $\tau_r \coloneqq (a_0, q_1, \ldots, a_{r-1}, q_r)$.
6 **for** $s \in [L_a]$ **do**
7 Let $a_{r,s}$ denote the s^{th} token in a_r. Compute `# Backwards pass`

$$\vec{d}_s(\theta_i) \coloneqq \nabla_\theta \log \Pr_{\sigma \sim p_{\theta_i}(x\tau_r)}[\sigma = a_{r,s}].$$

8 **if** $V(x, y, q_1, a_1, \ldots, q_R, a_R)$ accepts **then**
9 Update

$$\theta_{i+1} \coloneqq \theta_i + \lambda \cdot \sum_{\substack{r \in [R] \cup \{0\} \\ s \in [L_a]}} \vec{d}_s(\theta_i).$$

10 Output $\bar{\theta} \coloneqq \frac{1}{N} \sum_{i \in [N]} \theta_i$.

Lemma (RLVF gradient estimation). *Fix an input distribution μ over Σ^* and a verifier V with round complexity R and answer length L_a. For any transcript $(x, y, q_1, \ldots, a_R)$ we let $\mathrm{Acc}_V(x, y, q_1, \ldots, a_R)$ denote the indicator random variable which equals 1 if and only if V accepts the transcript. For any model P_θ, denote by $\mathrm{ver}(\theta)$ the verifiability of P_θ with respect to V and μ Then, for any θ,*

$$\nabla_\theta \mathrm{ver}(\theta) = \mathop{\mathbb{E}}_{\substack{x \sim \mu \\ y \sim P_\theta(x) \\ (q_r, a_r)_{r=1}^R}} \left[\mathrm{Acc}_V(x, y, q_1, \ldots, a_R) \cdot \sum_{\substack{r \in [R] \cup \{0\} \\ s \in [L_a]}} \vec{d}_s(\theta) \right]$$

where $(q_r, a_r)_{r=1}^R$ are as in lines 5-6 of Algorithm 2, and $\vec{d}_s(\theta)$ is as defined in line 8 therein.

But proving RL convergence guarantees is challenging...

What next?

- (Multiprover) proof system for AI Scientist [joint w/ Yoshua Bengio, Shafi Goldwasser, Oliver Richardson, Avishay Tal]
- Catching up RLVF theory 😅
  - Sample complexity bounds [joint w/ Nicolas Flammarion]
  - RLVF with average-case soundness leads to…
    - A useful and meaningful notion! (Optimistic)
    - Verifier hacking (Pessimistic)
    - A position paper… (Pragmatic)
- Practical Self-Proving Models *for sound Verifiers*
  - Lean-based DSLs [joint w/ Thomas Bourgeat]
- Universal (“Foundation”) Self-Proving models
  - So far: $\forall \boldsymbol{V}\ \exists \boldsymbol{P_\theta}$. I.e., need to learn a different prover for each verifier.
  - Can we have $\exists \boldsymbol{P_\theta}\ \forall \boldsymbol{V}$ (in a restricted class)?
    I.e., can we learn a prover P_θ such that for all $V \in \mathcal{V}$,
    $P_\theta(x, V)$ outputs y, *and proves correctness to* V*?*
- “Fundamental Theorem of Self-Provable learning”
  - Is there a (combinatorial) dimension of the problem/proof system that captures the sample complexity of learning Self-Proving models?

Seeking strong students with
- Background in Theoretical CS
- Interest in Self-Proving models

Visit EPFL and work with us in 2026/7.
Funding available*

Apply: orrp.net/visitEPFL

Contact: orr.paradise@epfl.ch