Models That Prove Their Own Correctness

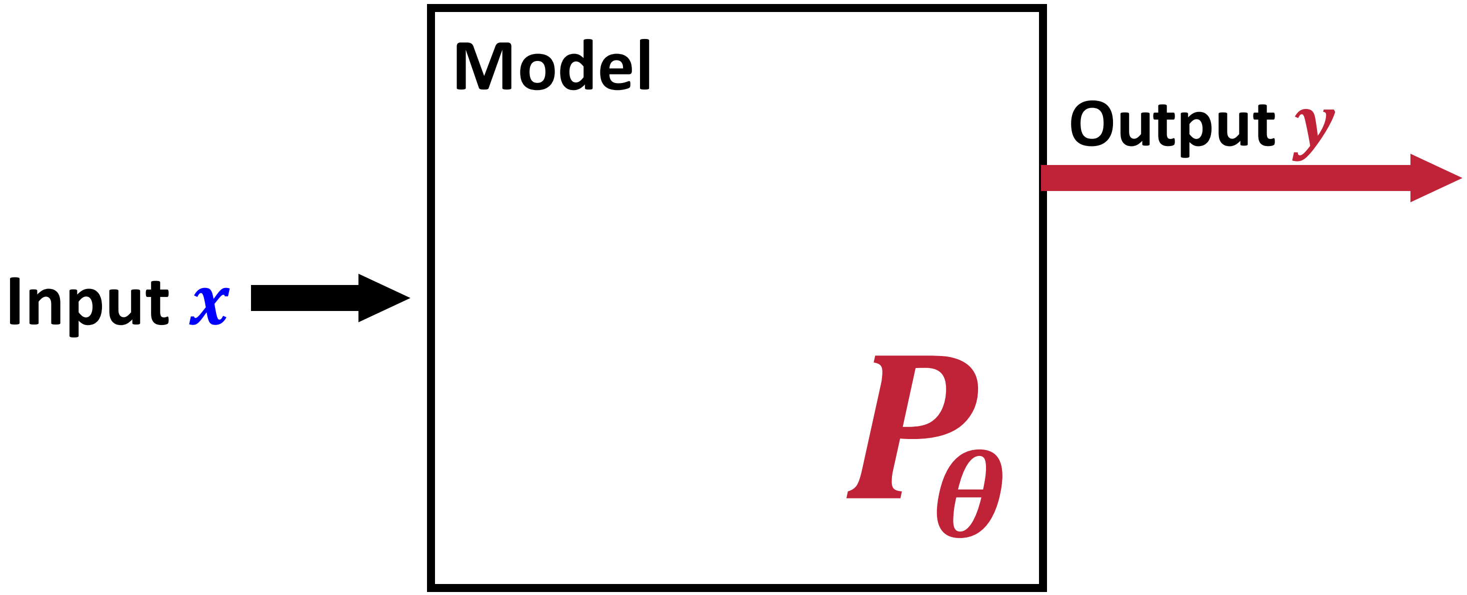
Orr Paradise

Simons Institute

October 15th, 2024

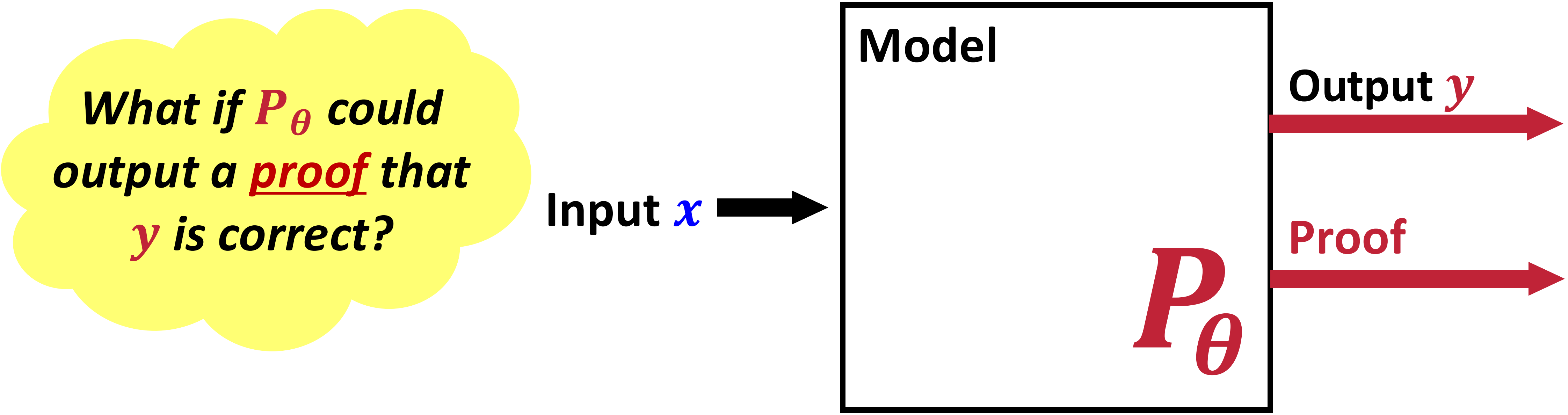
Motivation: Worst-case guarantees for ML

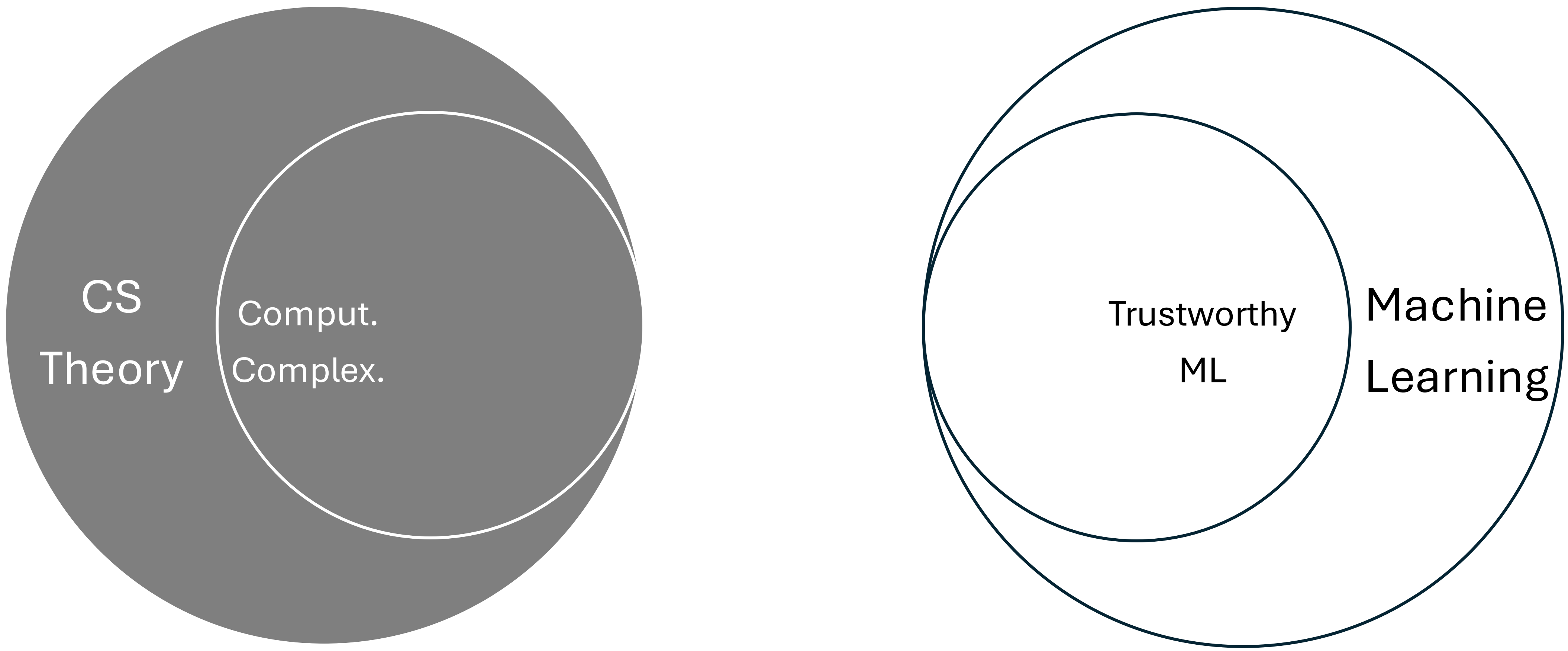
- Model P_θ has 99% accuracy on benchmarks for a task T
- You have an input x, for which the model generates $y \sim P_\theta(x)$
- **Should you trust that y is correct?**

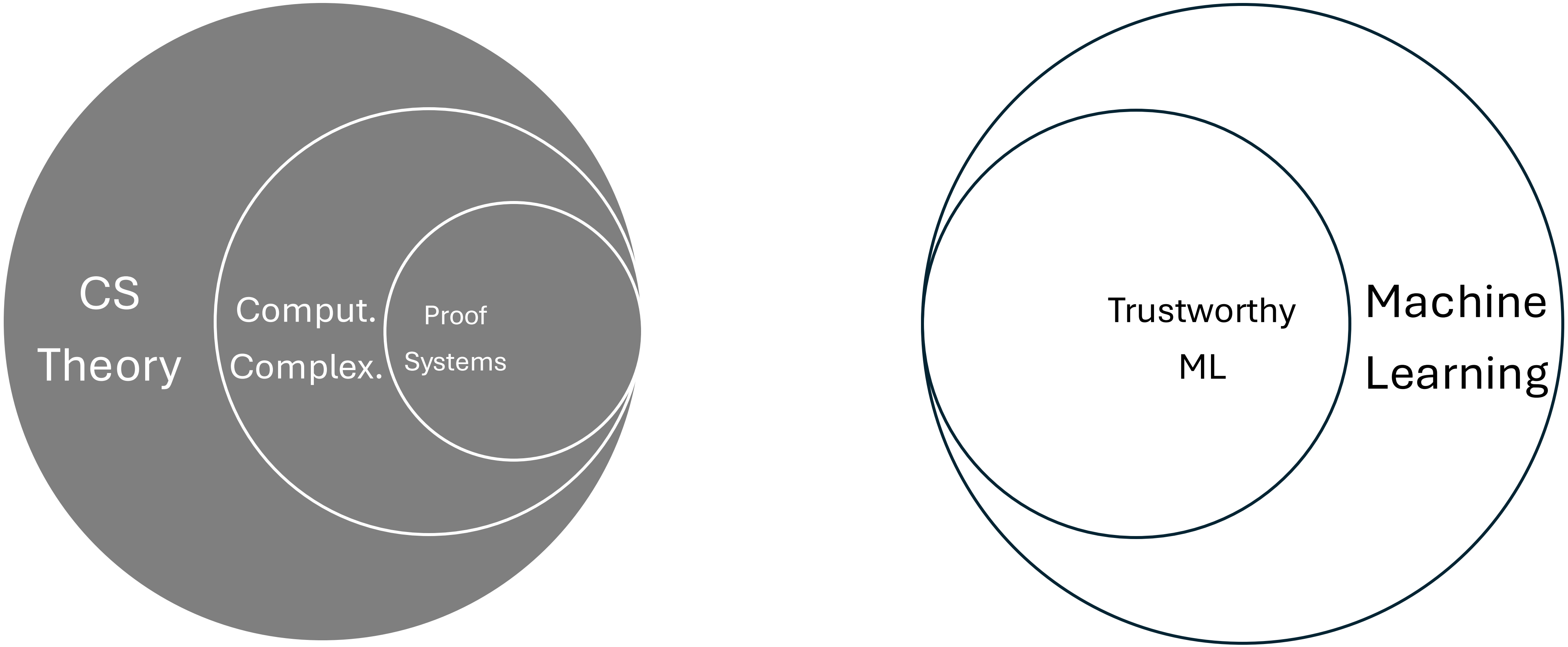


Motivation: Worst-case guarantees for ML

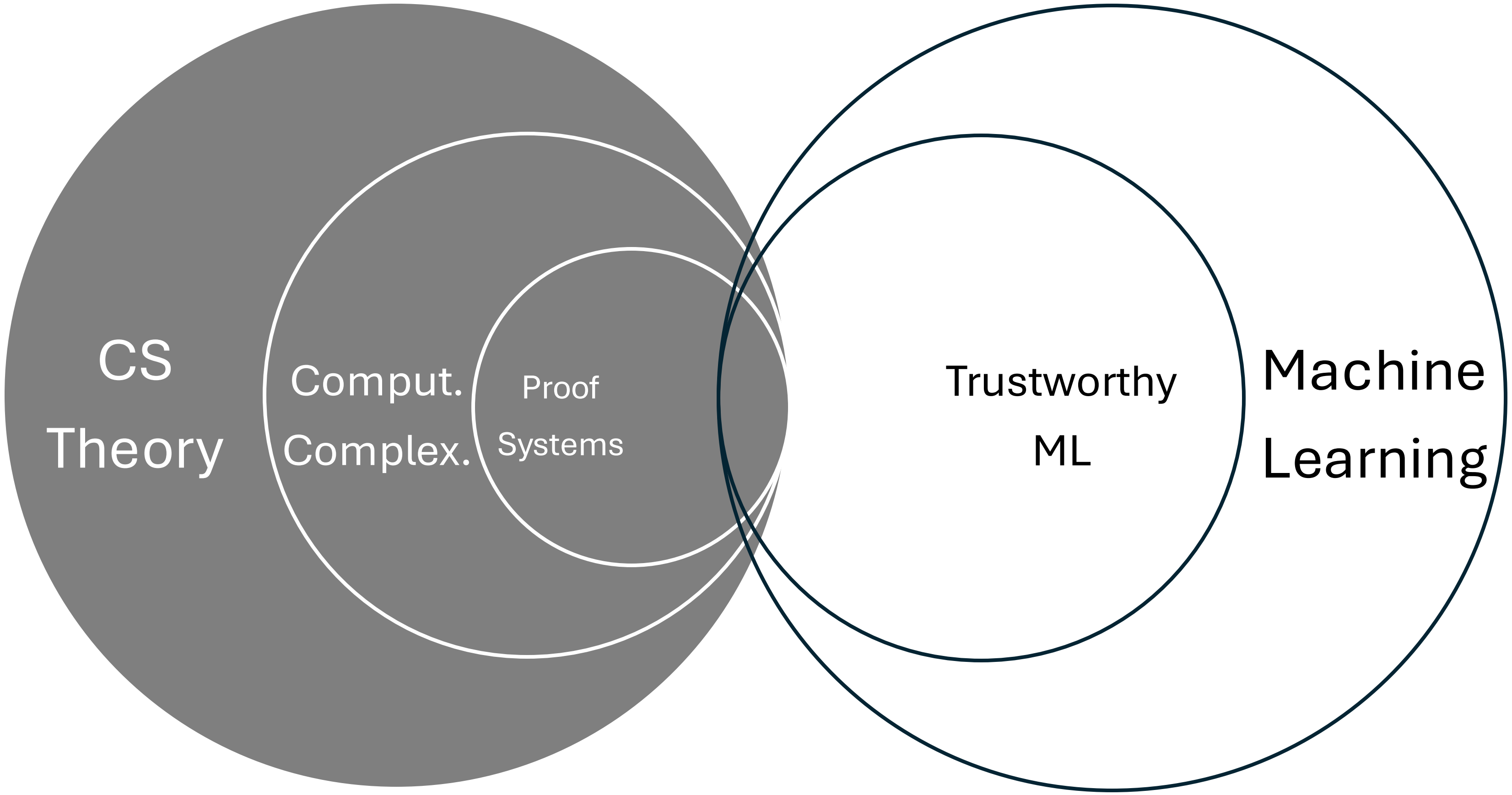
- Model P_θ has 99% accuracy on benchmarks for a task T
- You have an input x, for which the model generates $y \sim P_\theta(x)$
- **Should you trust that y is correct?**



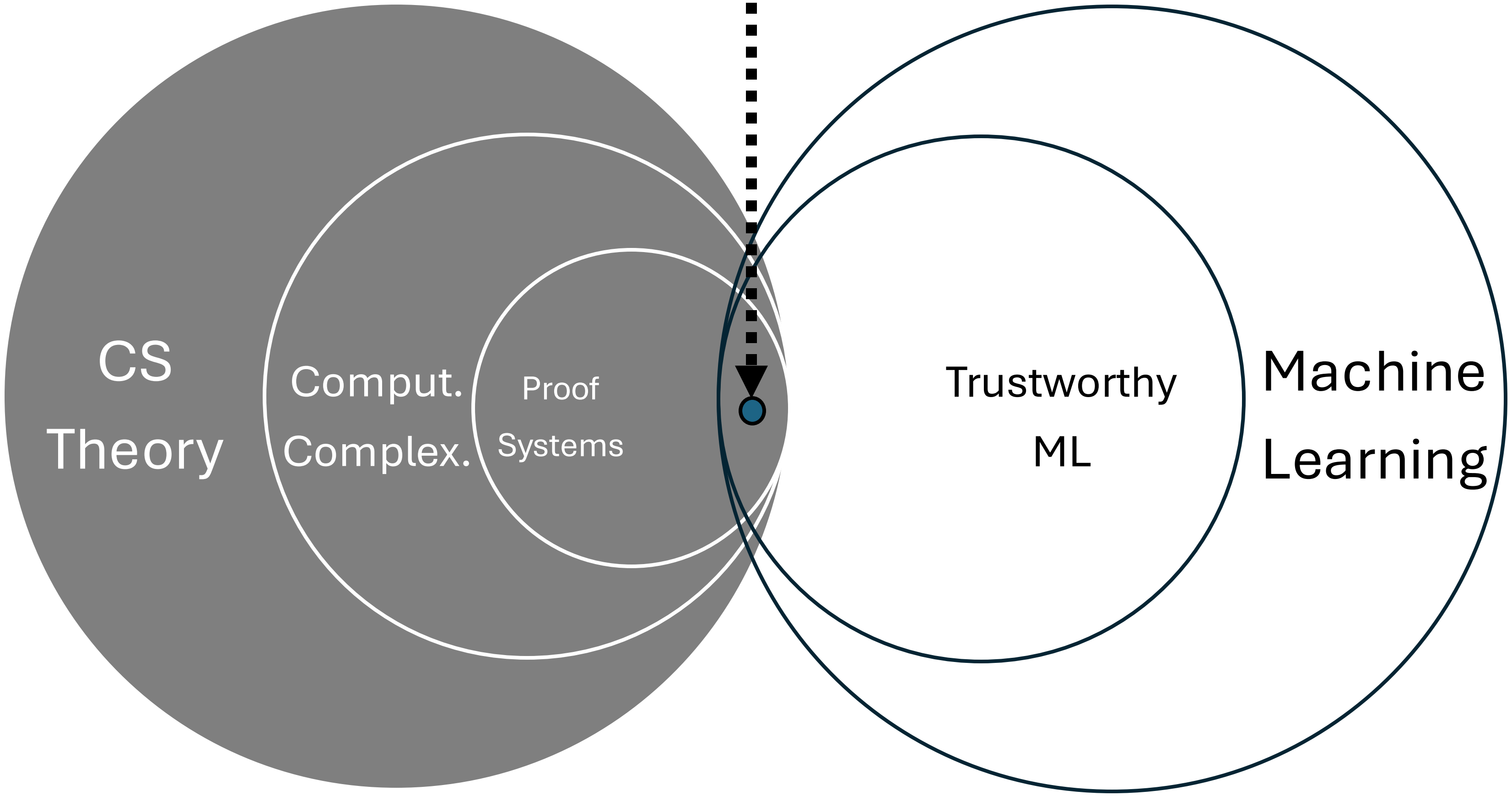

CS
Theory
Comput.
Complex.
Trustworthy
ML
Machine
Learning



CS
Theory
Comput.
Complex.
Proof
Systems
Trustworthy
ML
Machine
Learning


CS
Theory
Comput.
Complex.
Proof
Systems
Trustworthy
ML
Machine
Learning

Self-Proving models



Outline

- Self-Proving models
- Related literature
- Results
  - Theory
  - Experiments
- Future directions

Outline

- Self-Proving models
- Related literature
- Results
- Future directions

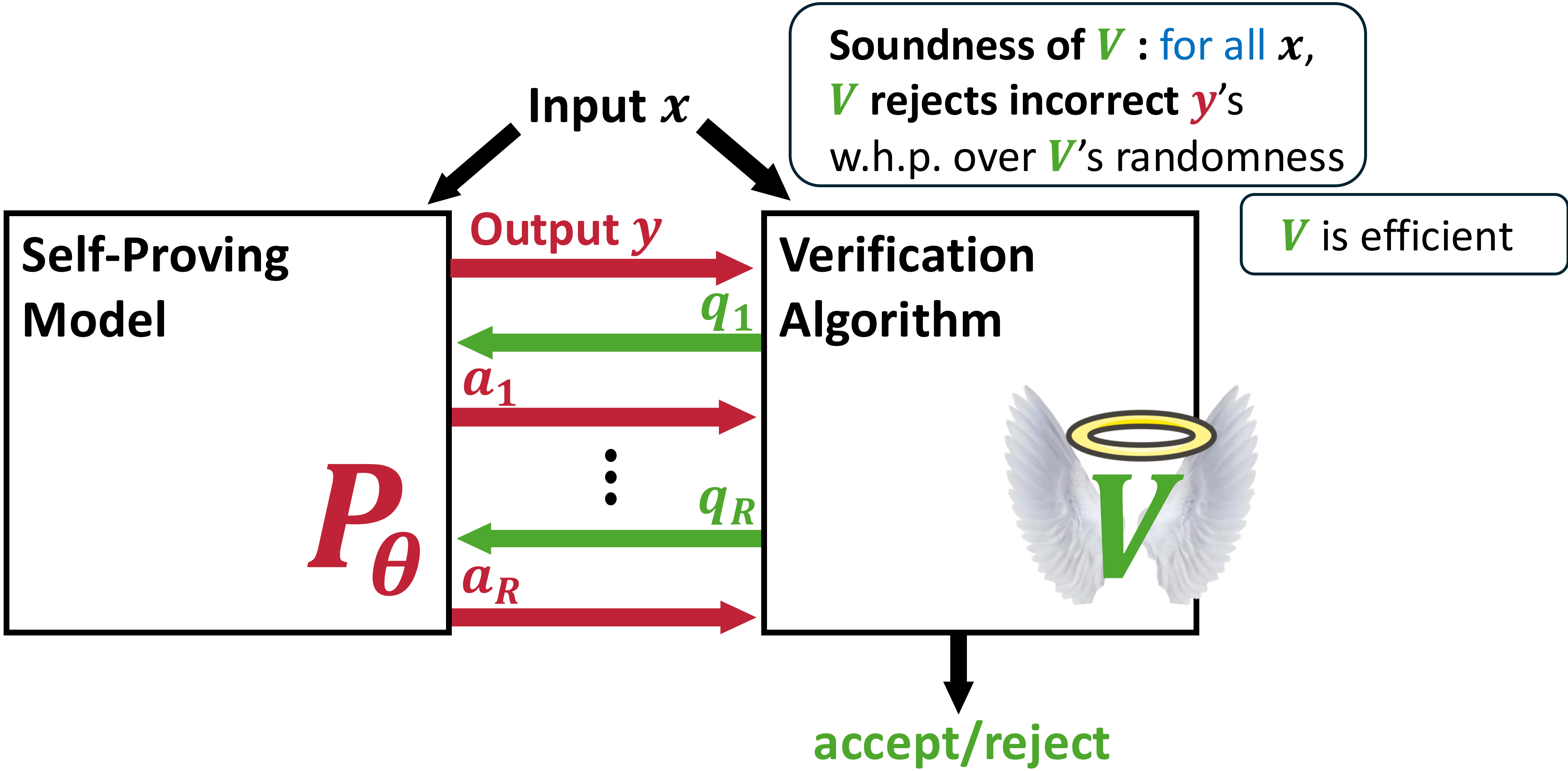
Joint work with:


Noga Amit

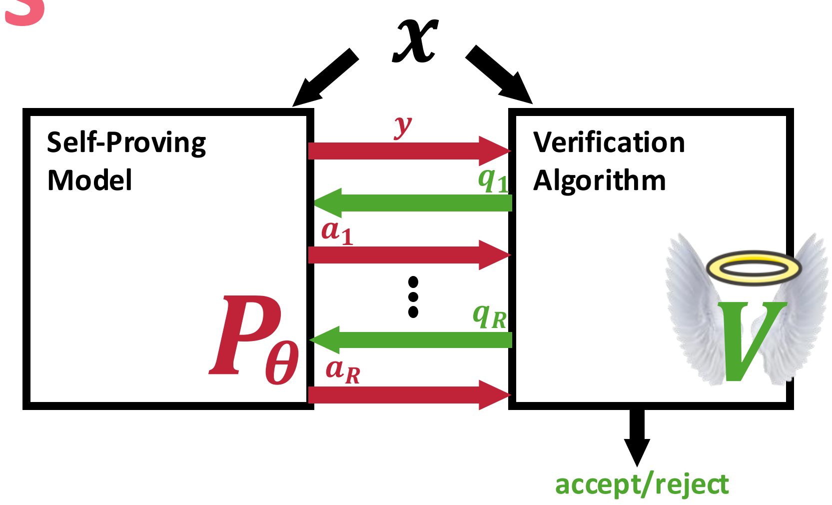

Shafi Goldwasser


Guy N. Rothblum

Self-Proving Models



Self-Proving Models



Setting

- μ = distribution over inputs x.
- V = an efficient and **sound** verification algorithm
- P_θ = sequence-to-sequence model with parameters $\theta \in \mathbb{R}^d$

Definition (Self-Provability): P_θ is β-self-proving w.r.t. V, μ if

$\Pr[P_\theta$ convinces V to **accept** $y] \geq \beta$ for $x \sim \mu$, $y \sim P_\theta(x)$

Self-Provability + Soundness ⇒ Standard accuracy (w.r.t μ):

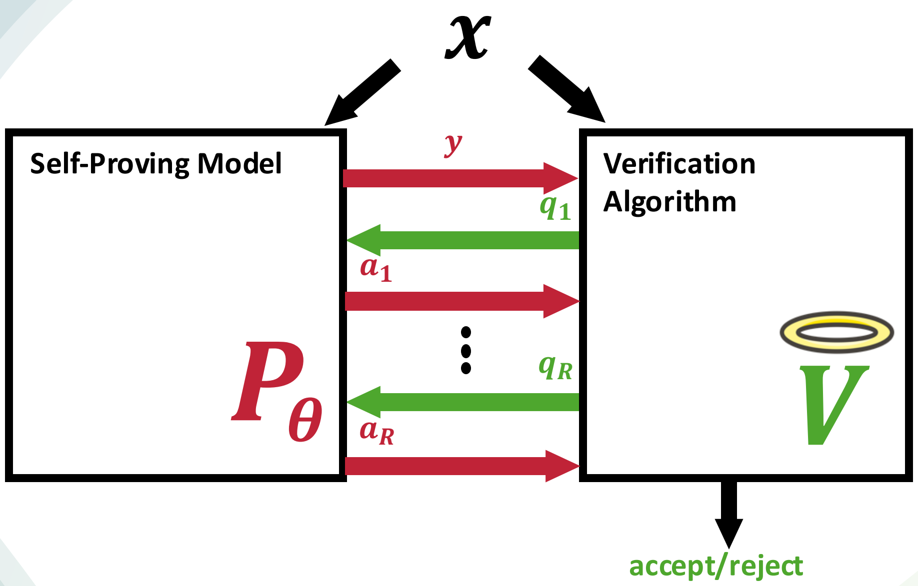
For $x \sim \mu$, $\Pr[P_\theta$ outputs correct $y] \geq \alpha$.

Outline

- Self-Proving models
- Related literature
  - Learning to Prove/Verify
  - Safety
  - Interpretability
  - Verifier-in-the-loop
- Results
- Future directions

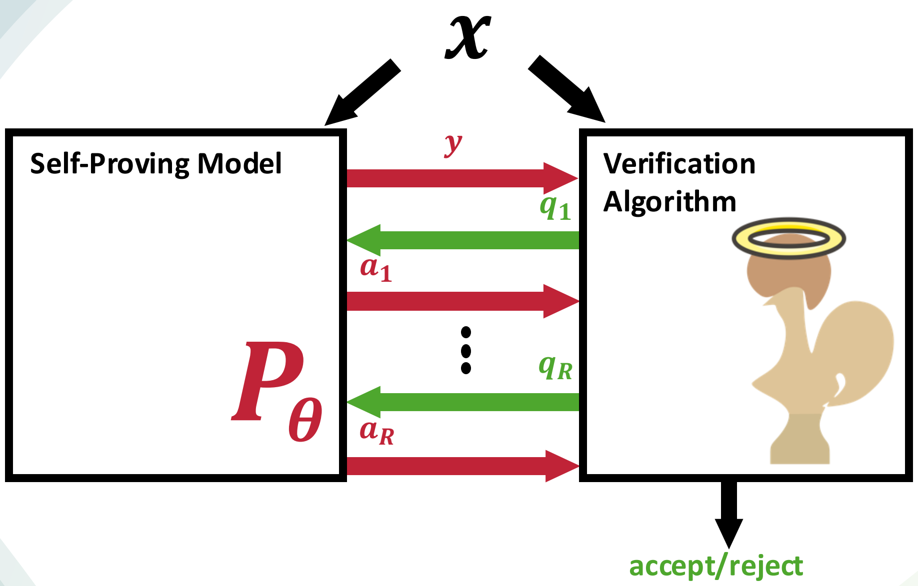
Related literature

- Learning to Prove…



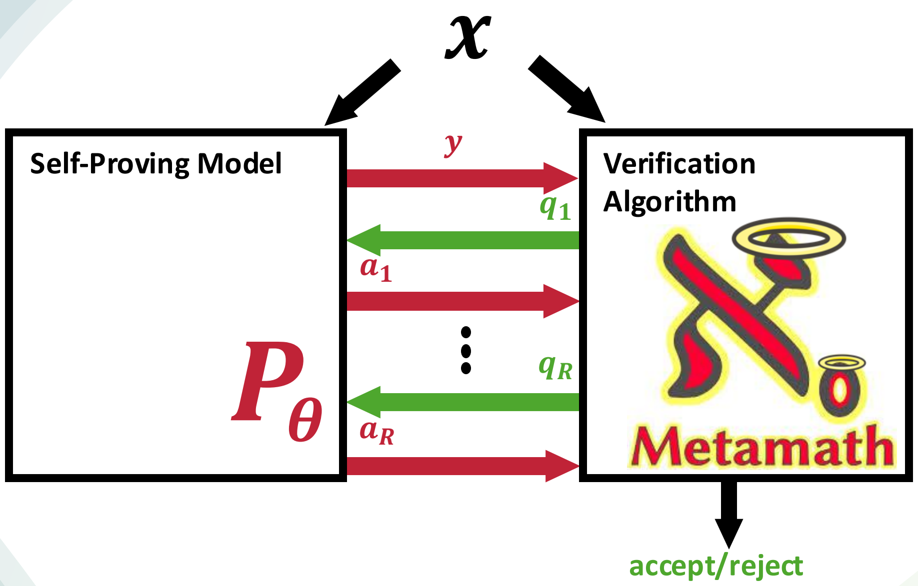
Related literature

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



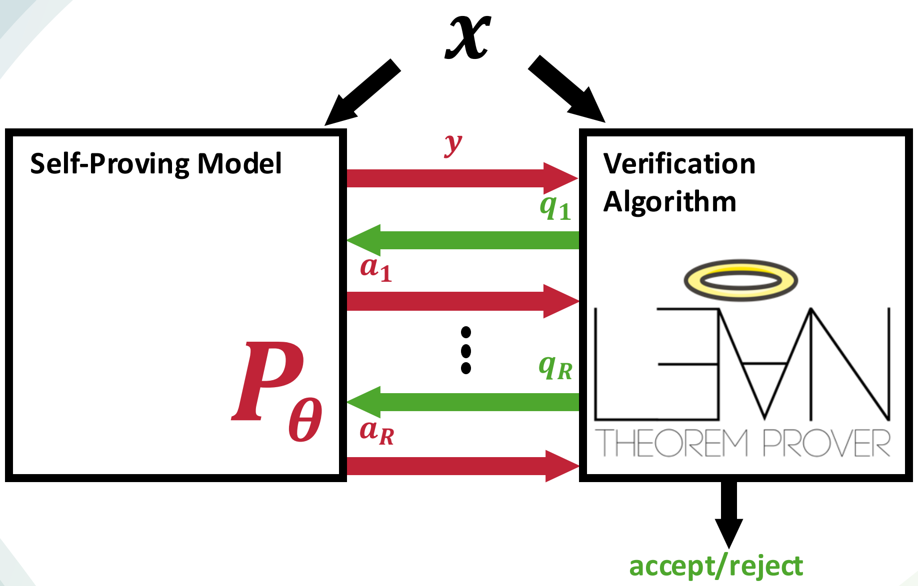
Related literature

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



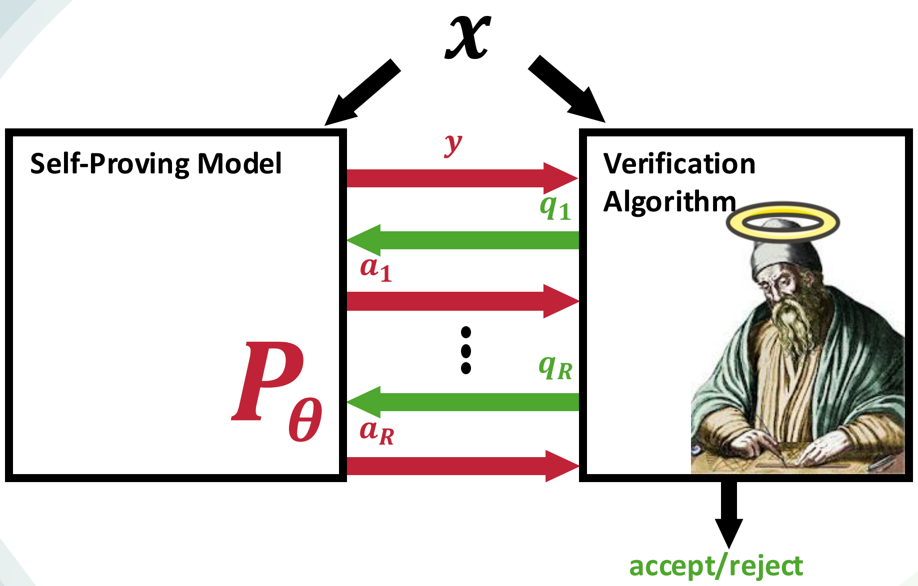
Related literature

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



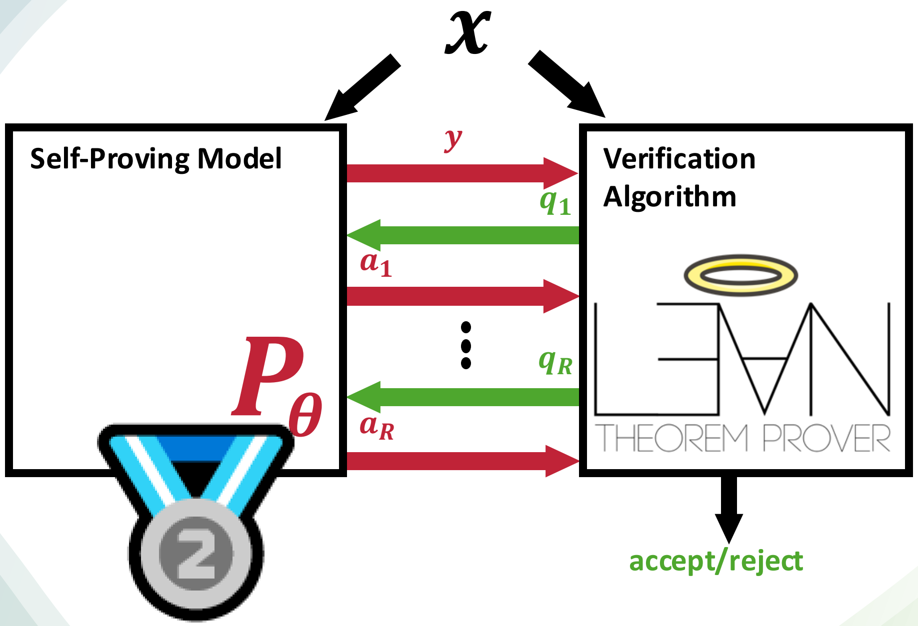
Related literature

- 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)**



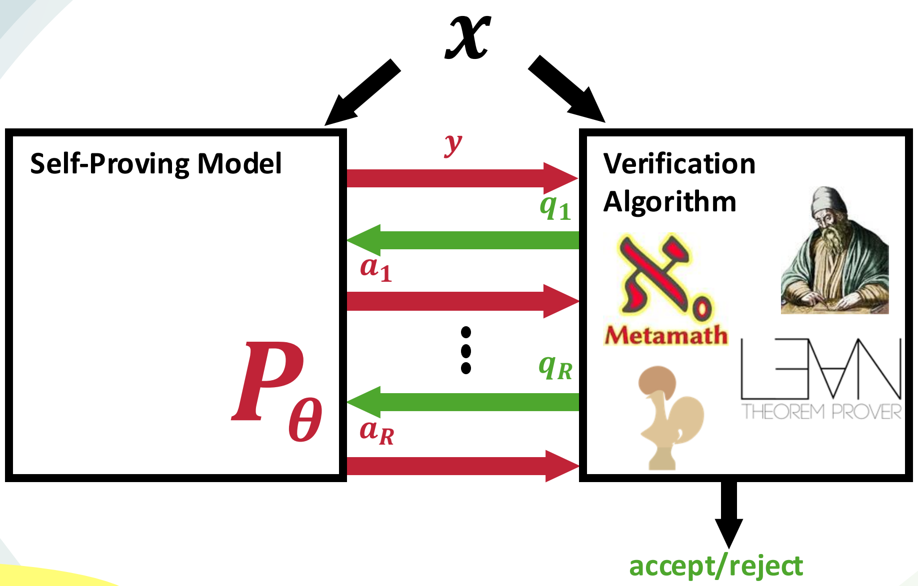
Related literature

- 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)**



Related literature

- 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)



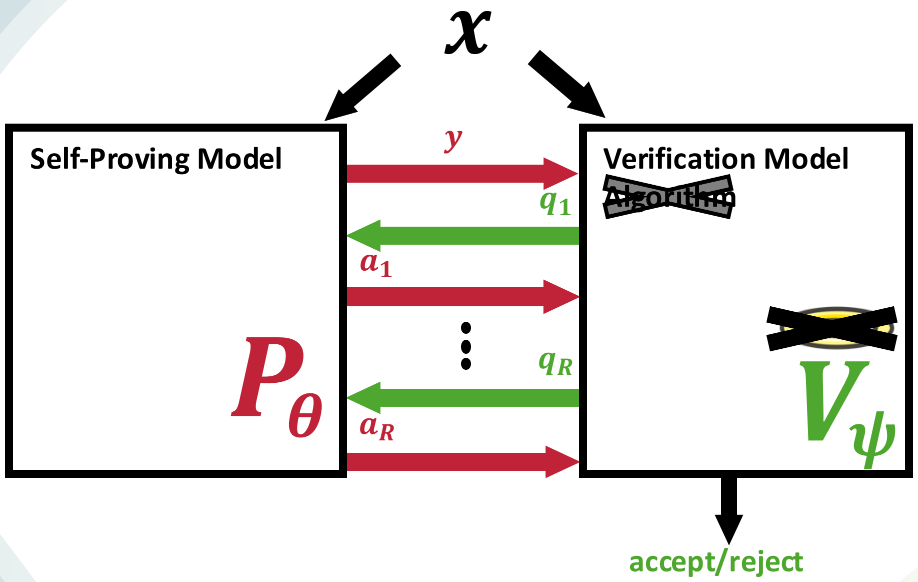
Self-Proving models: prove correctness via an Interactive Proof system

IP: Weak Verifier vs. powerful *yet untrusted* Prover (Goldwasser et al., 1985)

IP=PSPACE (Shamir, 1990)

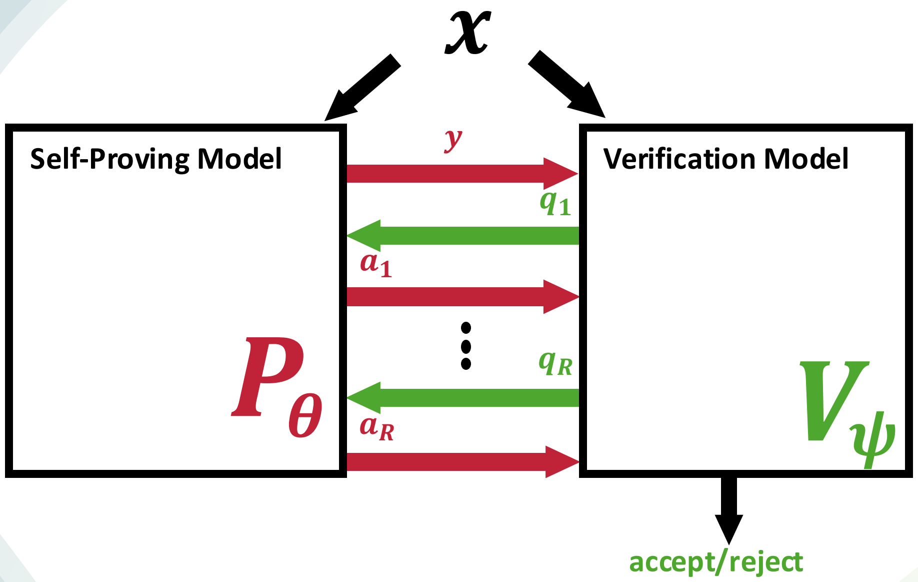
Related literature

- 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**



Related literature

- 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)**



Related literature

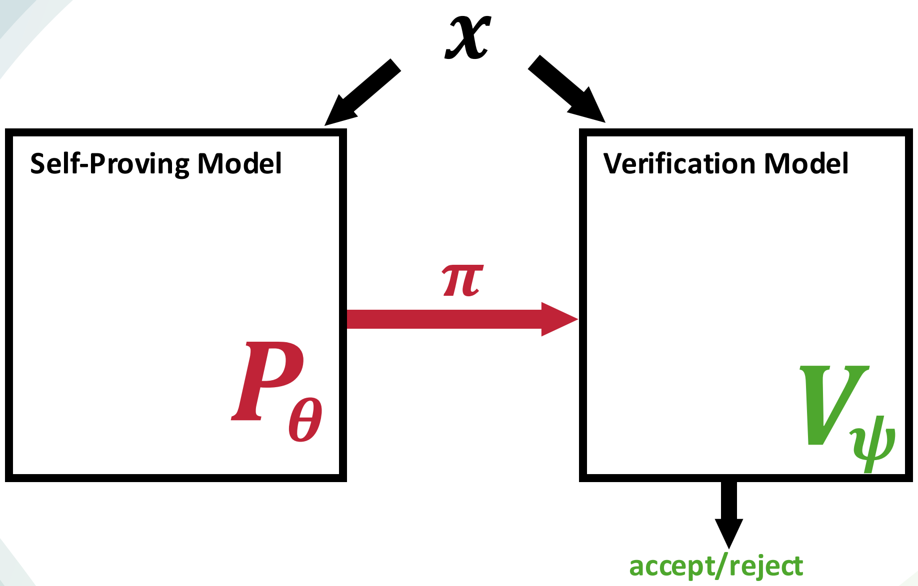
- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - **Prover Verifier Games (Anil et al., 2021)**

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

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

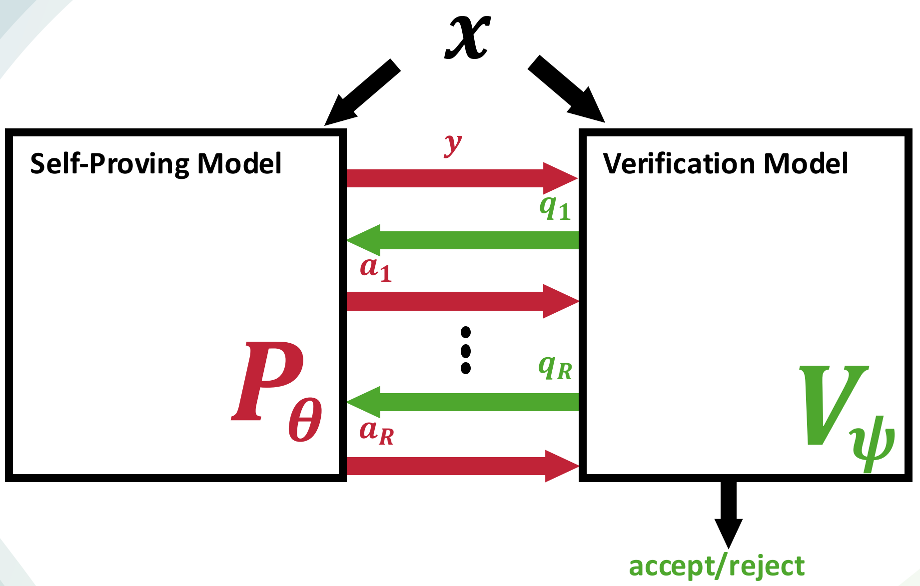
Does an equilibrium ⚖ → Completeness and Soundness? Does C&S → ⚖?

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



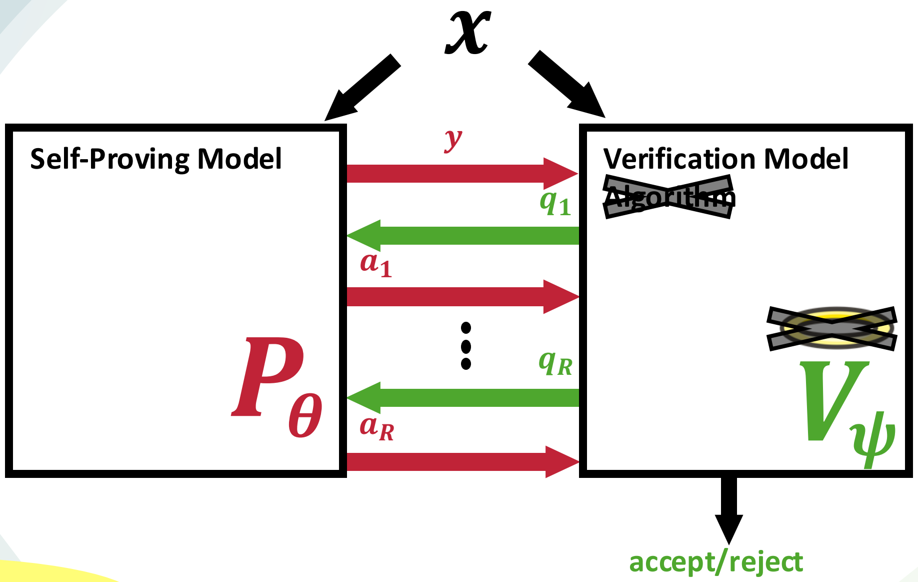
Related literature

- 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)**



Related literature

- 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)



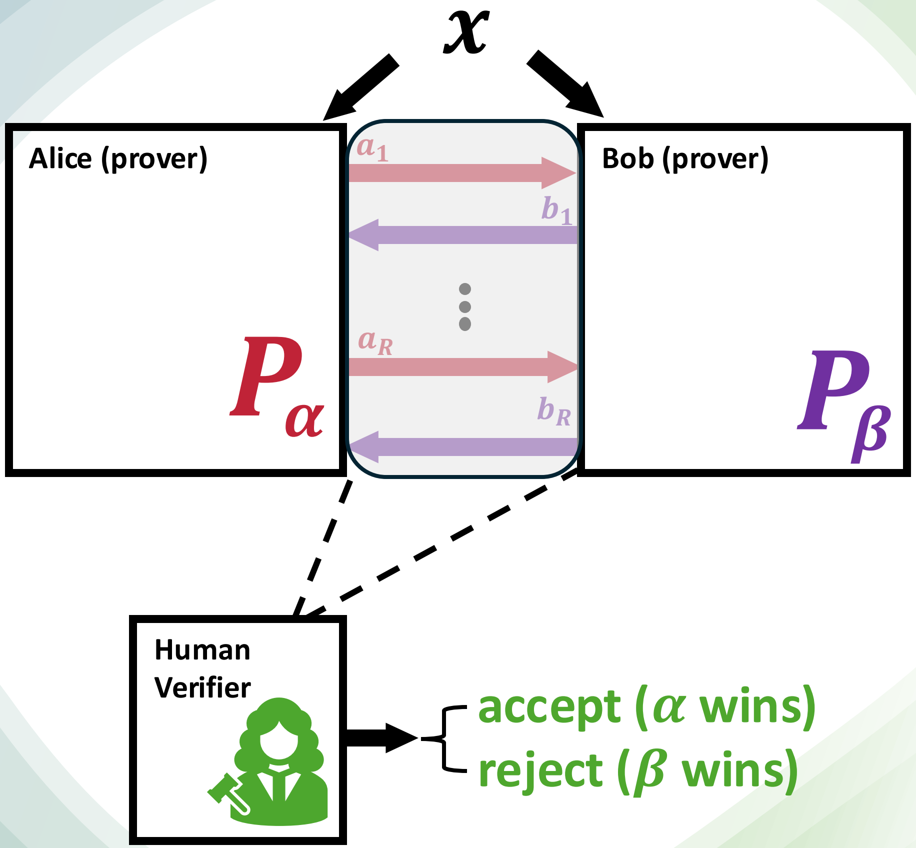
Self-Proving models: prove correctness
to Sound Verifiers---a formal guarantee.

Related literature

- 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)**

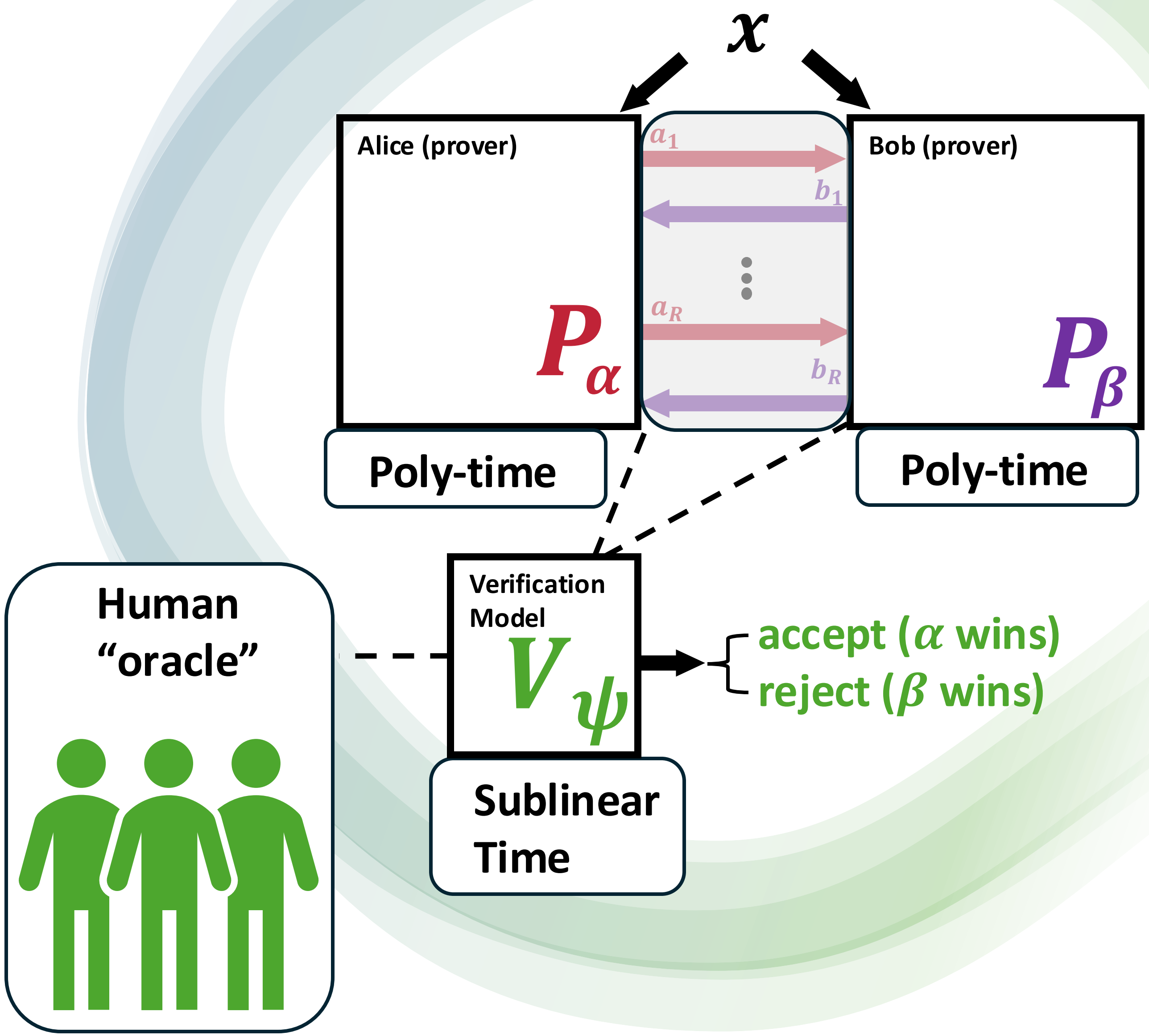
Related literature

- 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)**



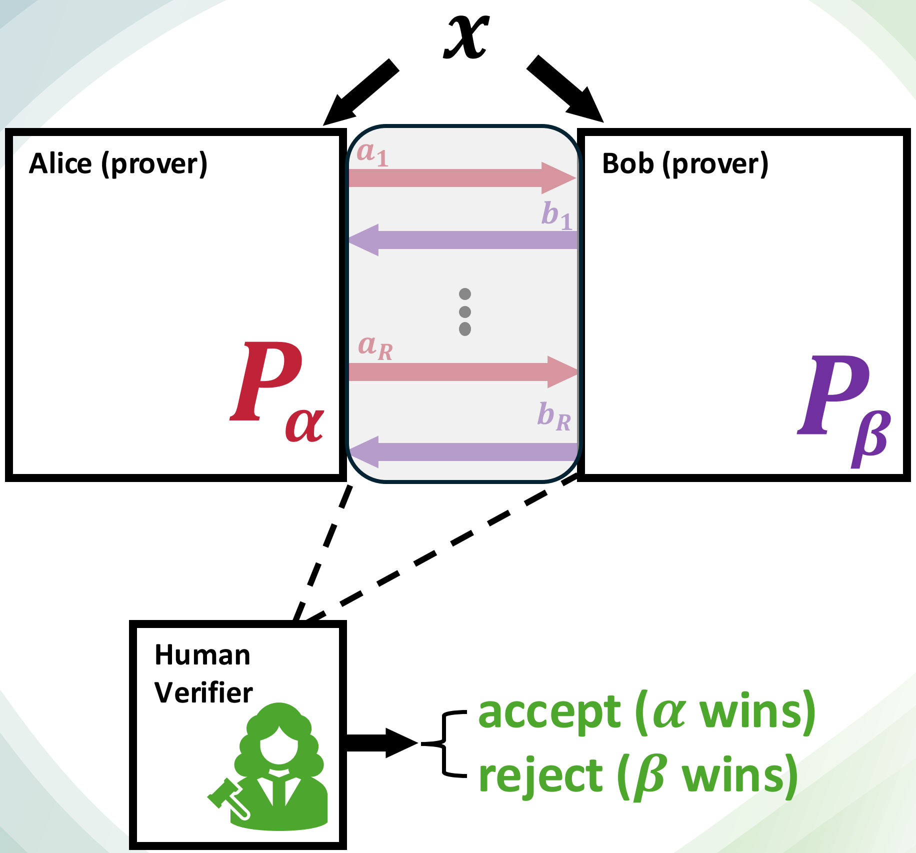
Related literature

- 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)**



Related literature

- 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)



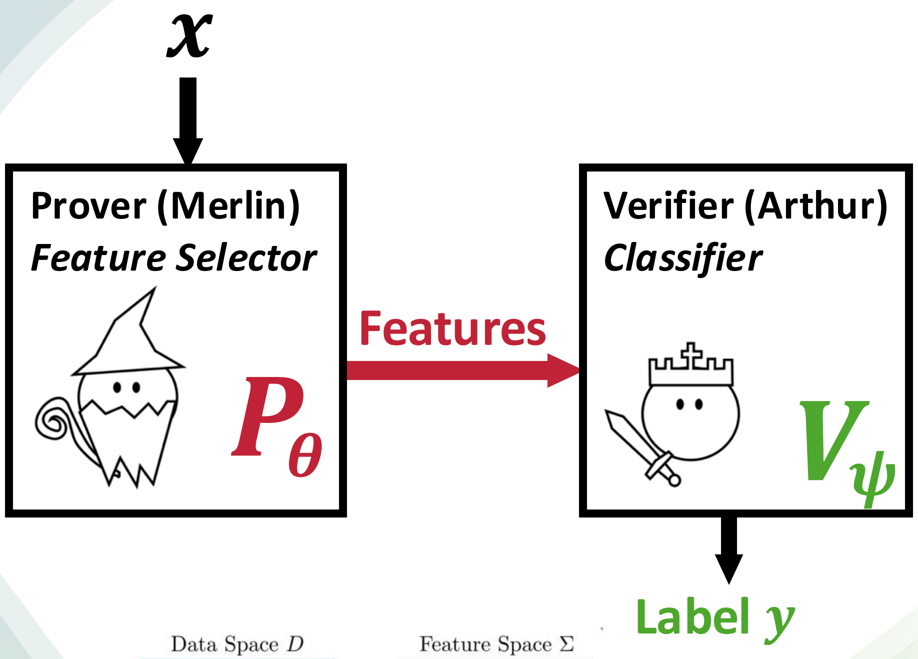
Self-Proving models: single prover, sound verifier

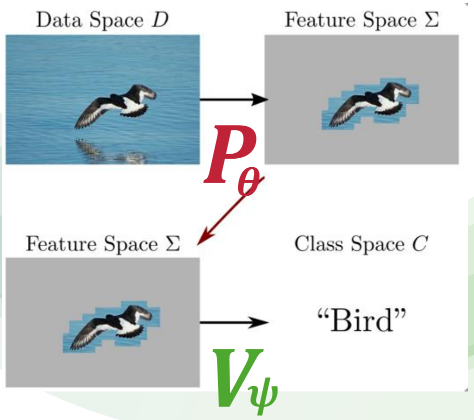
Related literature

- 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**

Related literature

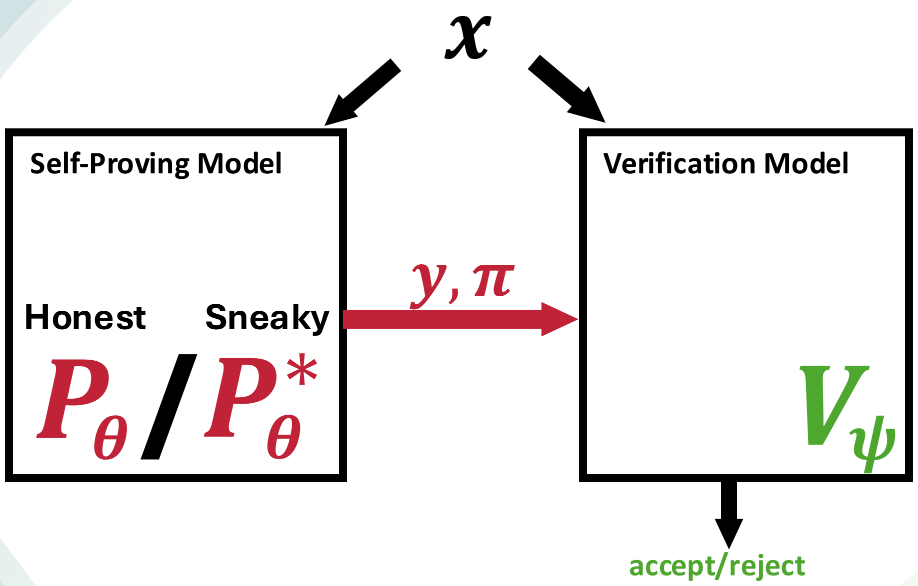
- Learning to Prove…
  - … in Coq (Gransden et al., 2015)
  - … in Metamath (Polu and Sutskever, 2020)
  - … in Lean (Yang et al., 2023)
  - … in synthetic geometry (Trinh et al., 2024)
  - … in Lean 🥈 (DeepMind, 2024)
- Learning to Verify
  - Prover Verifier Games (Anil et al., 2021)
  - Neural Interactive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





Related literature

- 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)**



Prover-Verifier Games Improve Legibility of LLM Outputs (Hendrick* and Chen* et al., 2024)

Idea: Introduce a sma

FT Prover LLM to

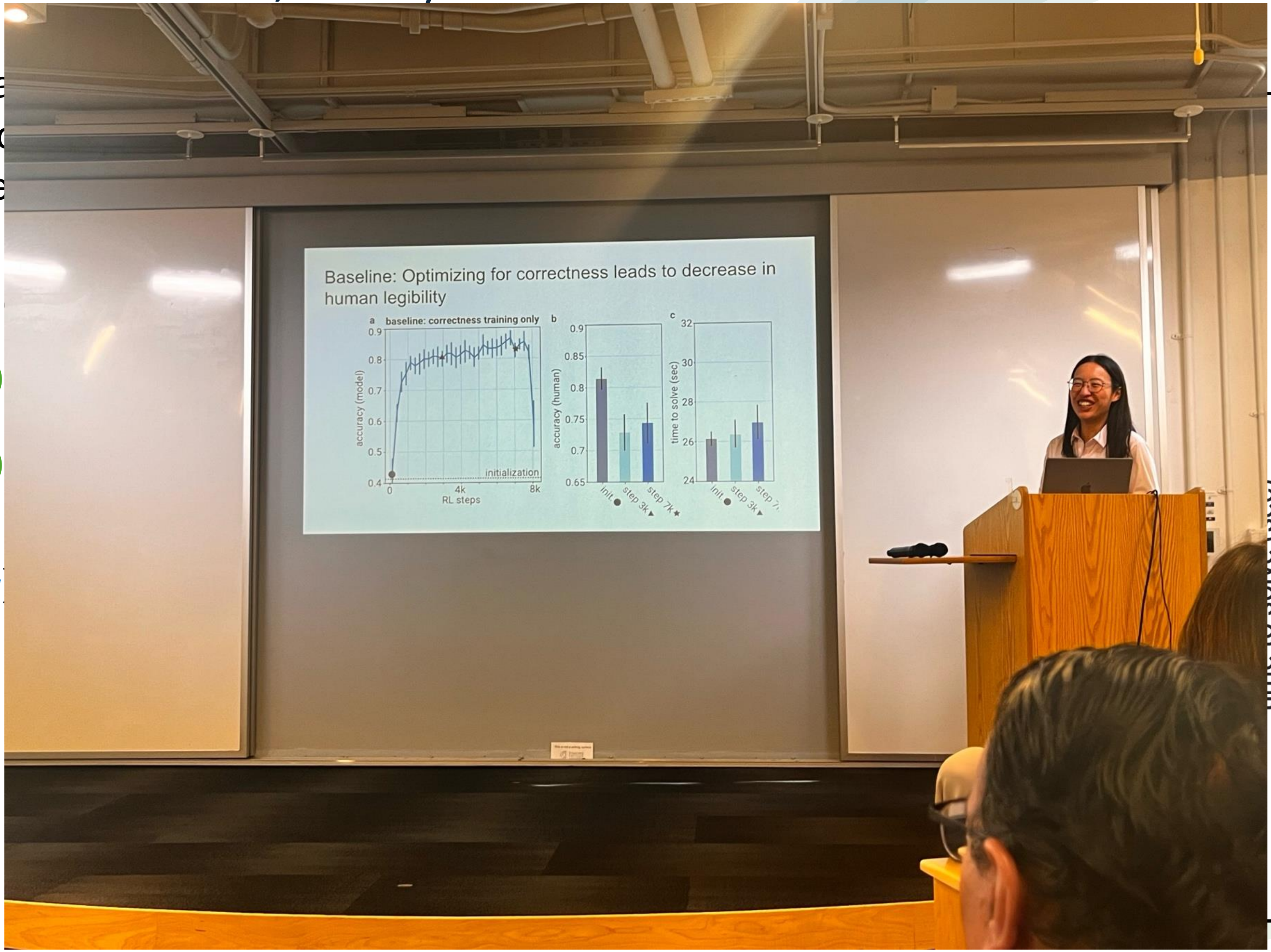
- Does this increase LLM's answers?

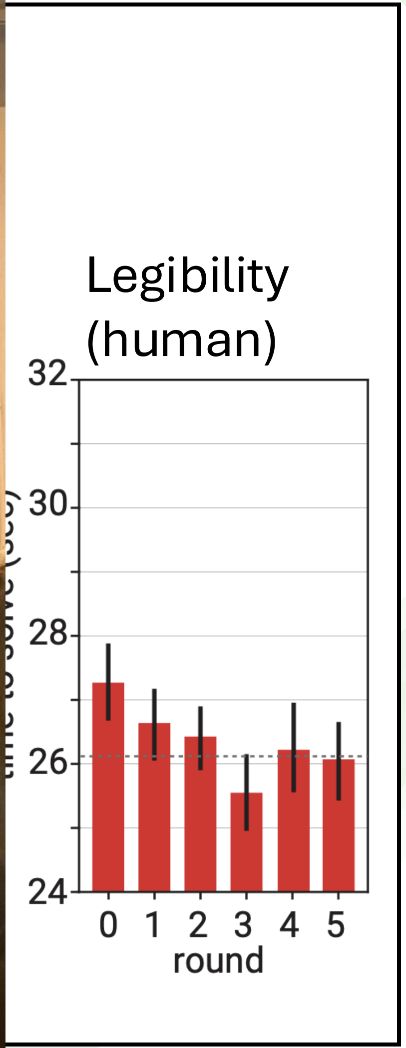
Joint training (simplifi

$$\psi \underset{\text{argmin}}{\longleftarrow} L(\psi \mid \theta)$$

$$\ell(\psi \mid P)$$

$$\theta \underset{\text{argmin}}{\longleftarrow} \mathbb{E}_x \left[\langle V_\psi, P_\theta \rangle (x \right.$$





Related literature

- 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)

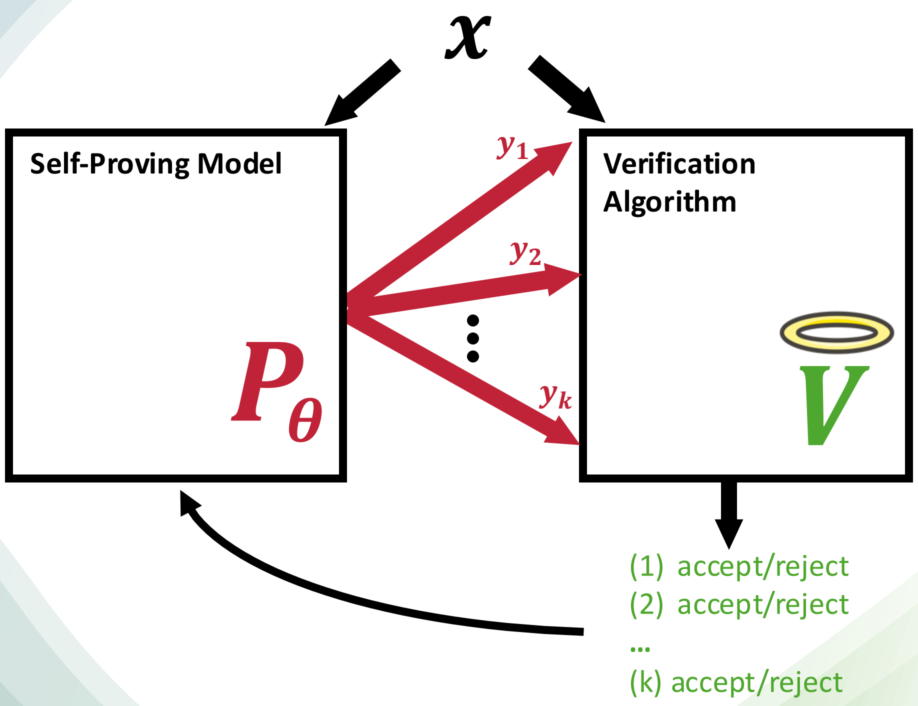
Self-Proving models:

Guarantee provably correct outputs from a model.

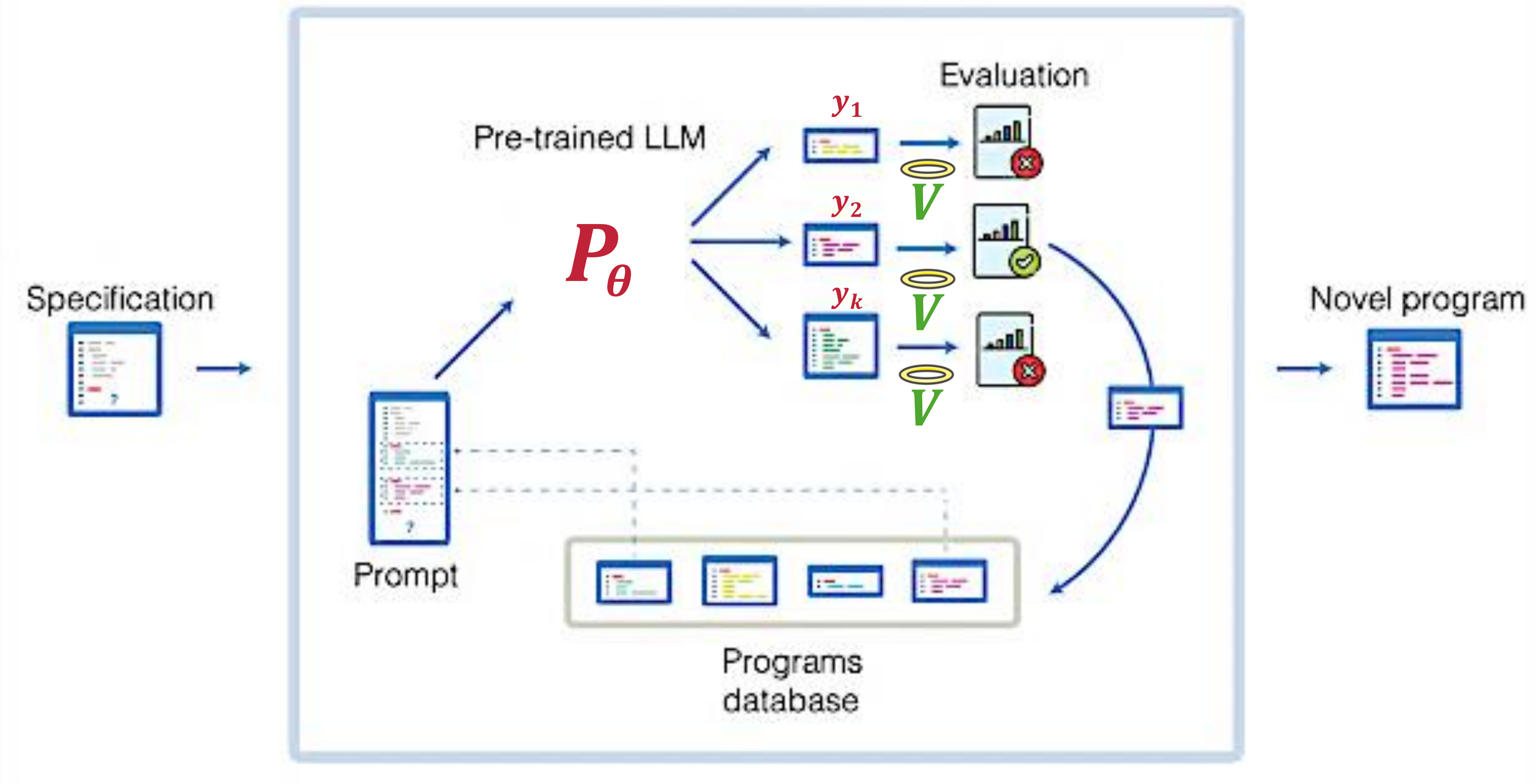
Related literature

- 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)
- Verifier in-the-loop
  - **FunSearch (Romera-Paredes et al., 2024)**

Goal: find optimal y for x



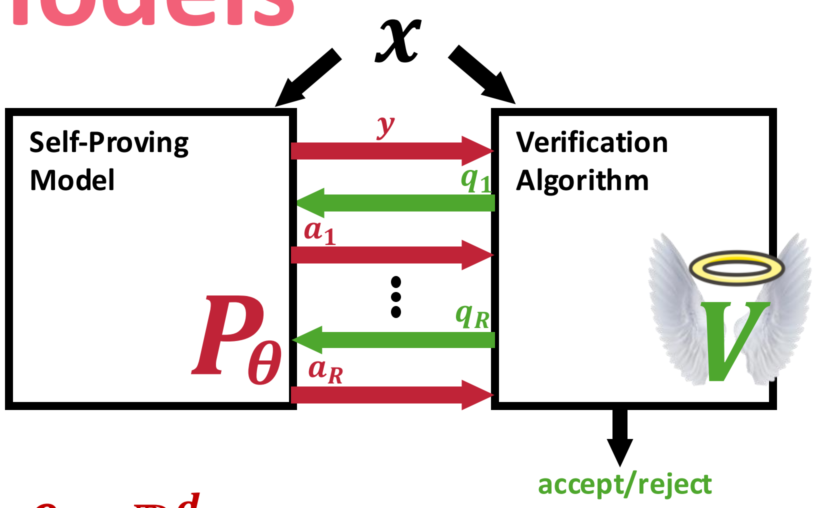
Mathematical discoveries from program search with LLMs (Romera-Paredes et al., 2024)



Outline

- Self-Proving models
- Related literature
- Results
  - Theory
  - Experiments
- Future directions

Reminder: Self-Proving Models



Setting

- μ = distribution over inputs x.
- V = an efficient and **sound** verification algorithm
- P_θ = sequence-to-sequence autoregressive model with parameters $\theta \in \mathbb{R}^d$

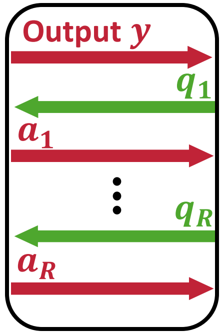
Definition: (Worst-case) **Self-Provability:** P_θ is β-self-proving w.r.t. V, μ if

$$\Pr[P_\theta \text{ convinces } V \text{ to accept } y] \geq \beta \quad \text{for } x \sim \mu,\ y \sim P_\theta(x)$$

Goal: **Learn θ that maximizes $\Pr[P_\theta$ convinces V to accept $y]$**

Learning Self-Proving Models

Fix input distribution $\boldsymbol{\mu}$, verifier $\boldsymbol{V}$

Idea: Transcripts $\boldsymbol{\pi} =$ 

are sequences, learn to generate $\boldsymbol{x} \to \boldsymbol{\pi}$

<u>Theorem</u>: Under convexity and Lipschitzness assumptions, given access to accepted transcripts, Transcript Learning outputs a $(\mathbf{1} - \boldsymbol{\epsilon})$-**Self-Proving model** after $\boldsymbol{N \geq 4\left(C \cdot B_1 \cdot B_2 \cdot \frac{1}{\epsilon}\right)^2}$ iterations

Total message length of $\boldsymbol{P_\theta}$'s

Bound on $\|\nabla \log \boldsymbol{P_\theta}\|$

Bound on $\|\boldsymbol{\theta}\|$

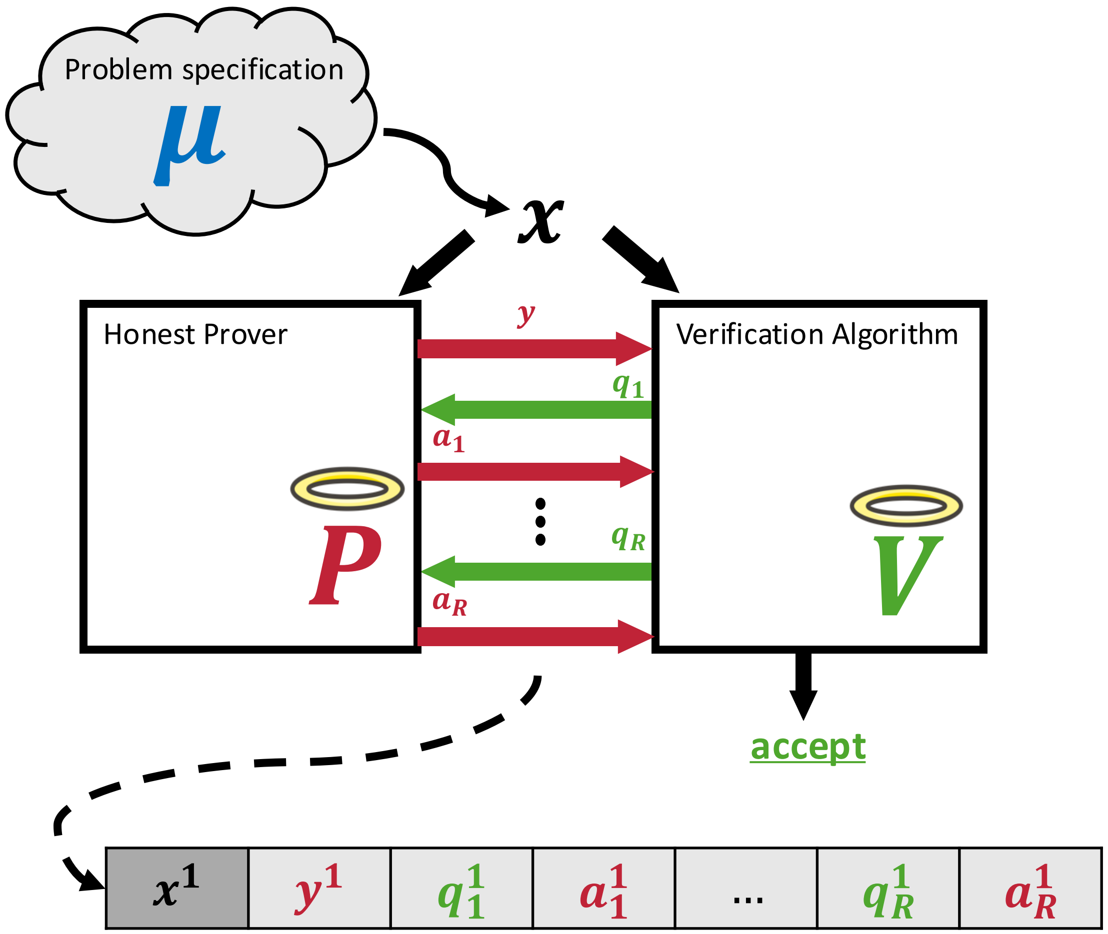
- Proof by reduction to SGD convergence bounds

Transcript Learning

Step 1: Collect accepted transcripts

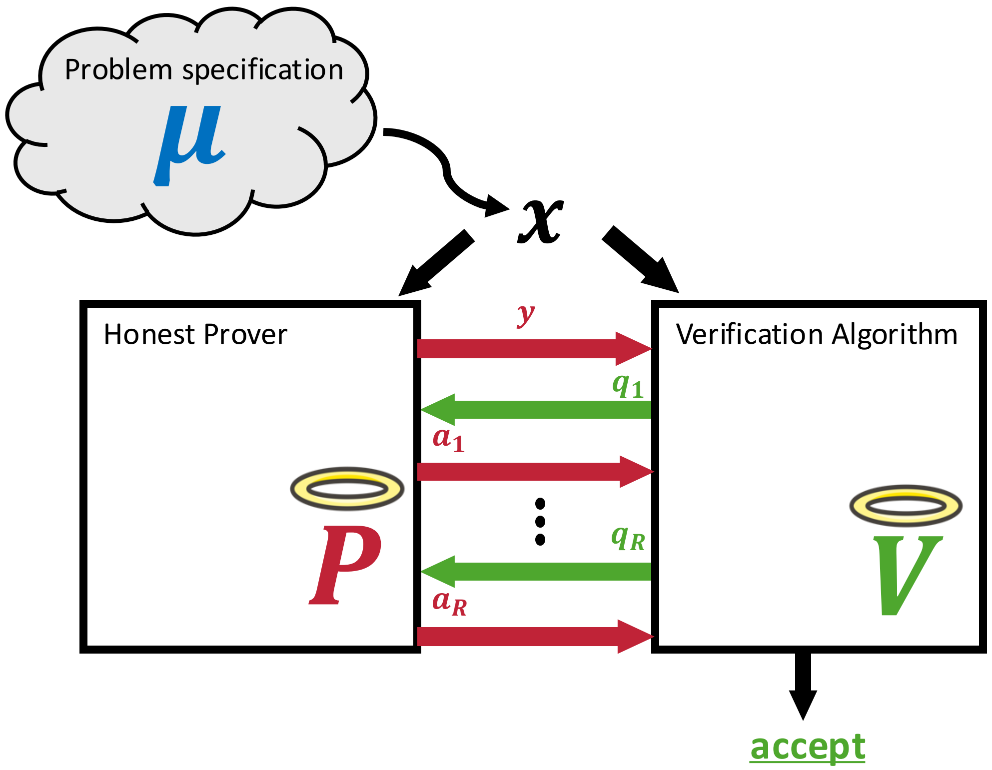
Transcript Learning

Step 1: Collect accepted transcripts



Transcript Learning

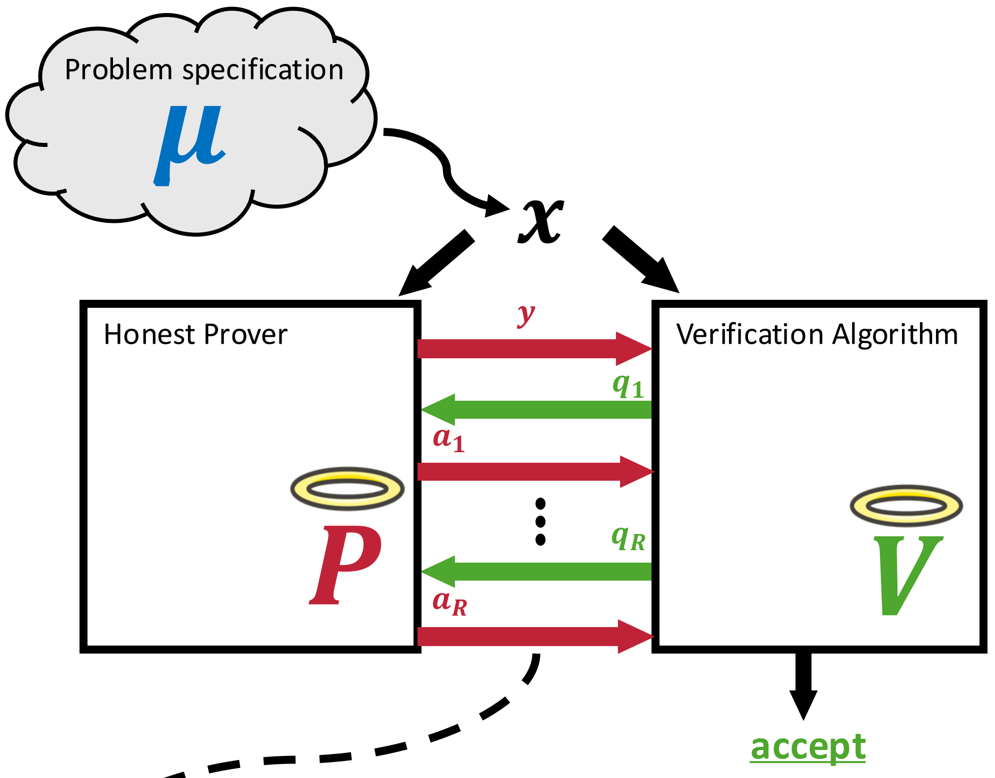
Step 1: Collect accepted transcripts



x^1	y^1	q_1^1	a_1^1	…	q_R^1	a_R^1

Transcript Learning

Step 1: Collect accepted transcripts



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

Transcript Learning

Step 1: Collect accepted transcripts

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

Transcript Learning

Step 1: Collect accepted transcripts

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

Self-Proving
Autoregressive Model

P_θ

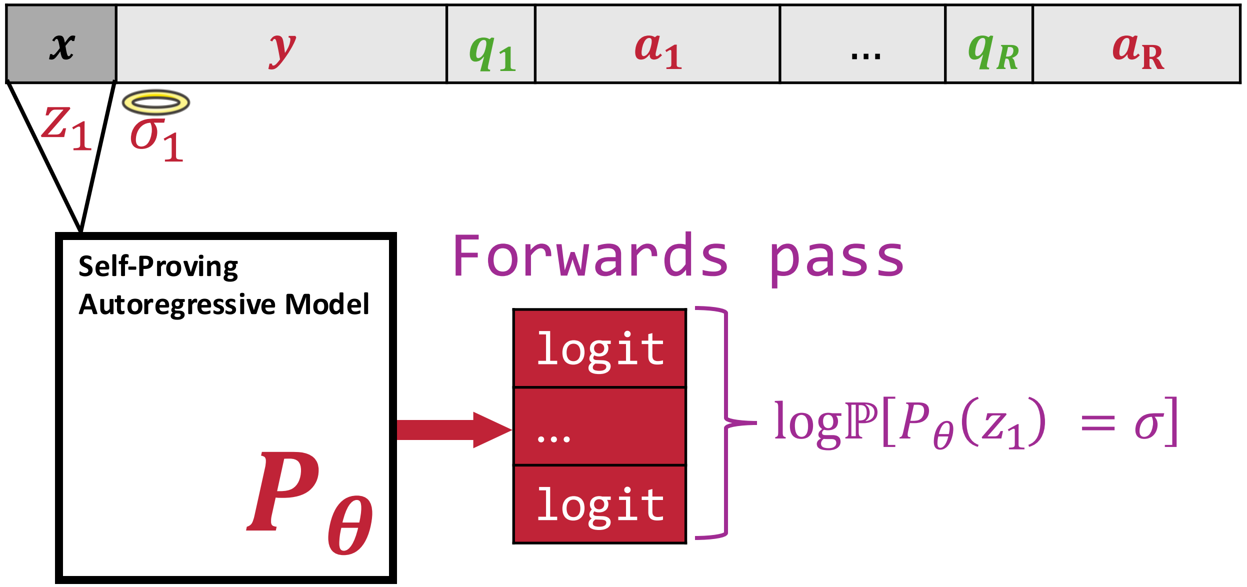
Transcript Learning

Step 1: Collect accepted transcripts

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

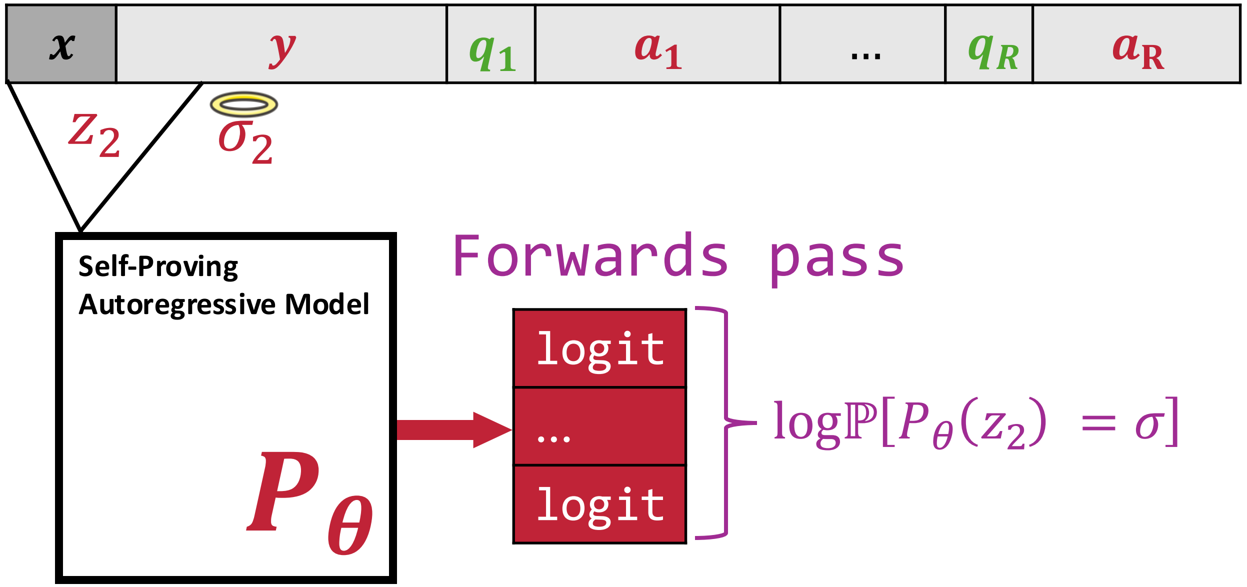
Transcript Learning

Step 1: Collect accepted transcripts

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

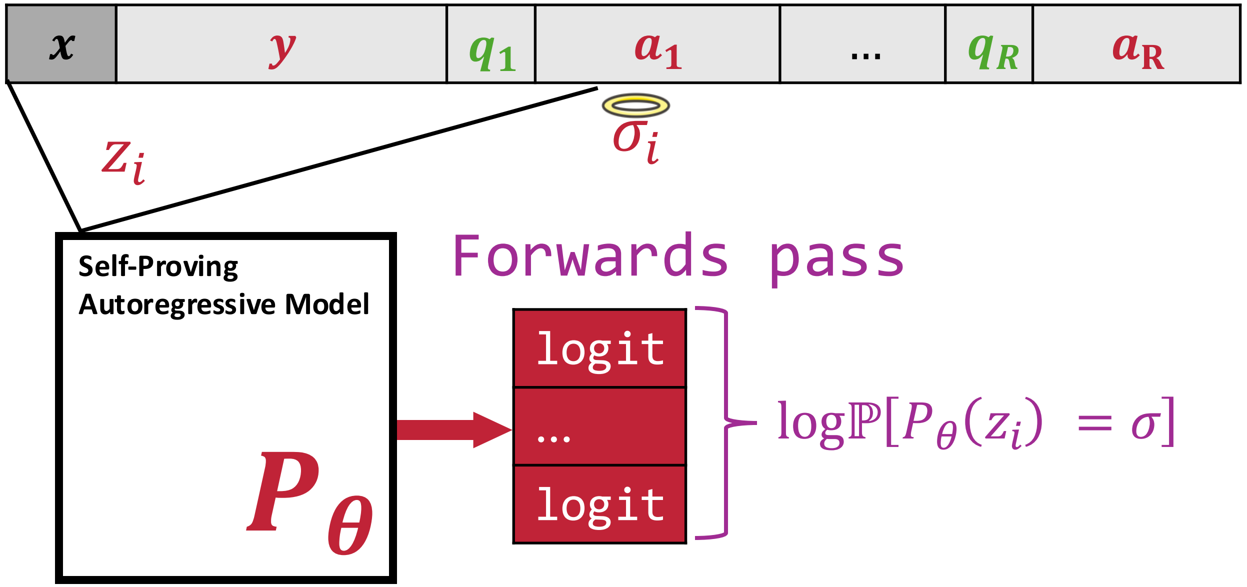
Transcript Learning

Step 1: Collect accepted transcripts

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 $\vec{d_i} \coloneqq \nabla_\theta \log \mathbb{P}[P_\theta(z_i) = \sigma_i]$

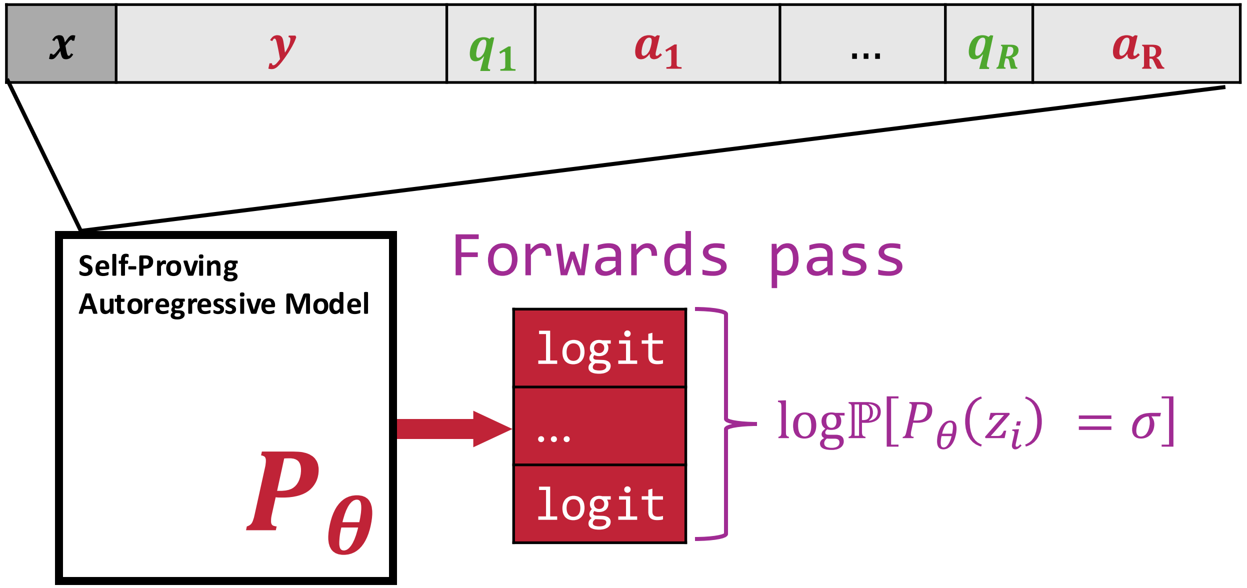
Transcript Learning

Step 1: Collect accepted transcripts

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

Update params $\theta \leftarrow \theta + \lambda \cdot \underbrace{\mathbb{P}[\overrightarrow{P_\theta}(x) = z]}_{\text{Forwards}} \cdot \underbrace{\sum_i \vec{d_i}}_{\text{Backwards}}$

Transcript Learning Sample Complexity

Assumptions:

Theorem: Under the assumptions, given access to accepted transcripts, Transcript Learning outputs a $\boldsymbol{(1-\epsilon)}$-**Self-Proving model** after

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

Transcript Learning Sample Complexity

Assumptions:

- The surrogate objective $A(\theta) \coloneqq \mathbb{P}_x[\pi_\theta(x) = \pi(x)]$ is concave and differentiable in θ.
- The total number of tokens sent by the prover in any interaction is $< \boldsymbol{C}$.
- 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 $A(\boldsymbol{\theta}^*) \geq 1 - \boldsymbol{\varepsilon}/2$.
- Access to a dataset of honest transcripts.

Theorem: Under the assumptions, given access to accepting transcripts, Transcript Learning outputs a $(\mathbf{1} - \boldsymbol{\epsilon})$-**Self-Proving model** after

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

Transcript Learning Sample Complexity

Assumptions:

- The surrogate objective $A(\theta) \coloneqq \mathbb{P}_x[\pi_\theta(x) = \pi(x)]$ is concave and differentiable in θ.
- The total number of tokens sent by the prover in any interaction is $< \boldsymbol{C}$.
- 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 $A(\boldsymbol{\theta}^*) \geq 1 - \boldsymbol{\varepsilon}/2$.
- Access to a dataset of honest transcripts.

<u>**Theorem**</u>: Under the assumptions, given access to accepting transcripts, Transcript Learning outputs a $(\mathbf{1} - \boldsymbol{\epsilon})$-**Self-Proving model** after

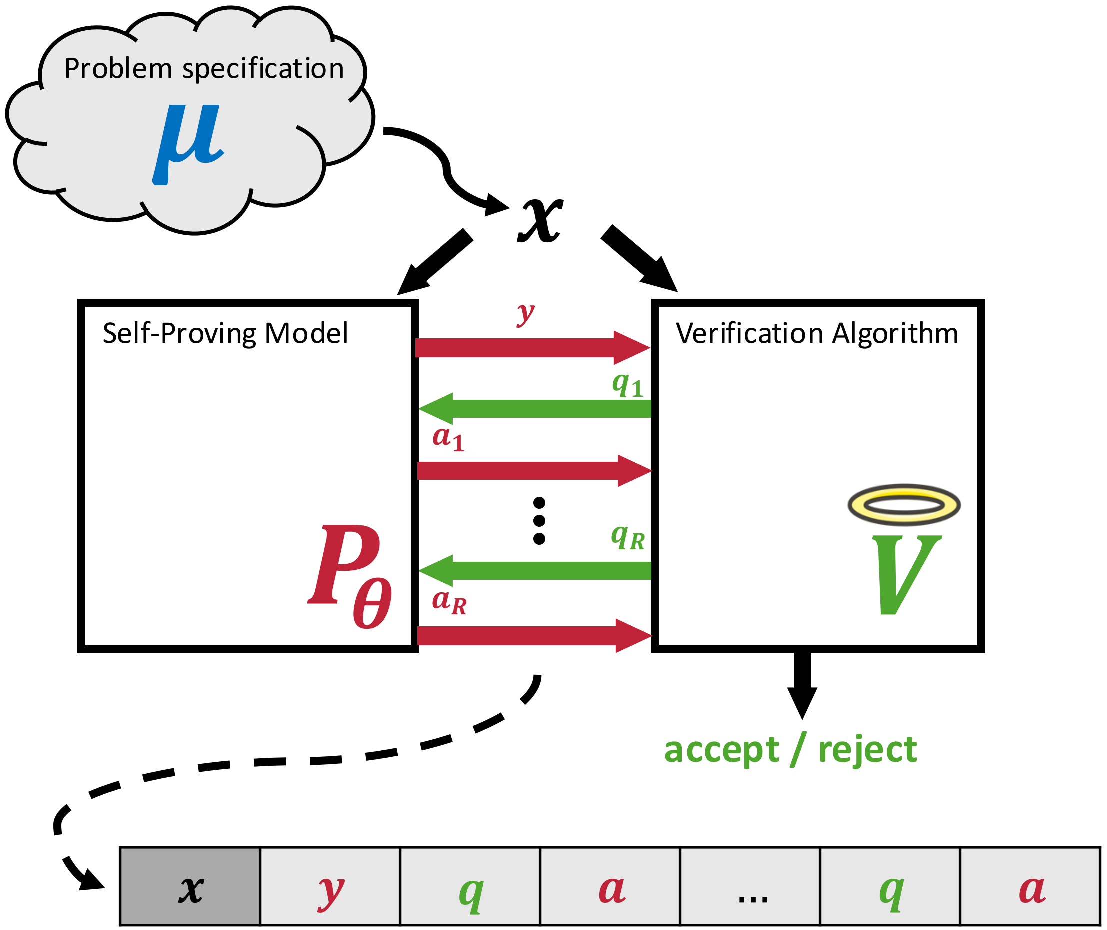
$$\boldsymbol{N} \geq 4\left(\boldsymbol{C} \cdot \boldsymbol{B_1} \cdot \boldsymbol{B_2} \cdot \frac{1}{\epsilon}\right)^2 \text{ iterations}$$

RLVF

RLVF

Repeat the following:

1. Generate transcript batch with P_θ.
   Keep only accepted transcripts.

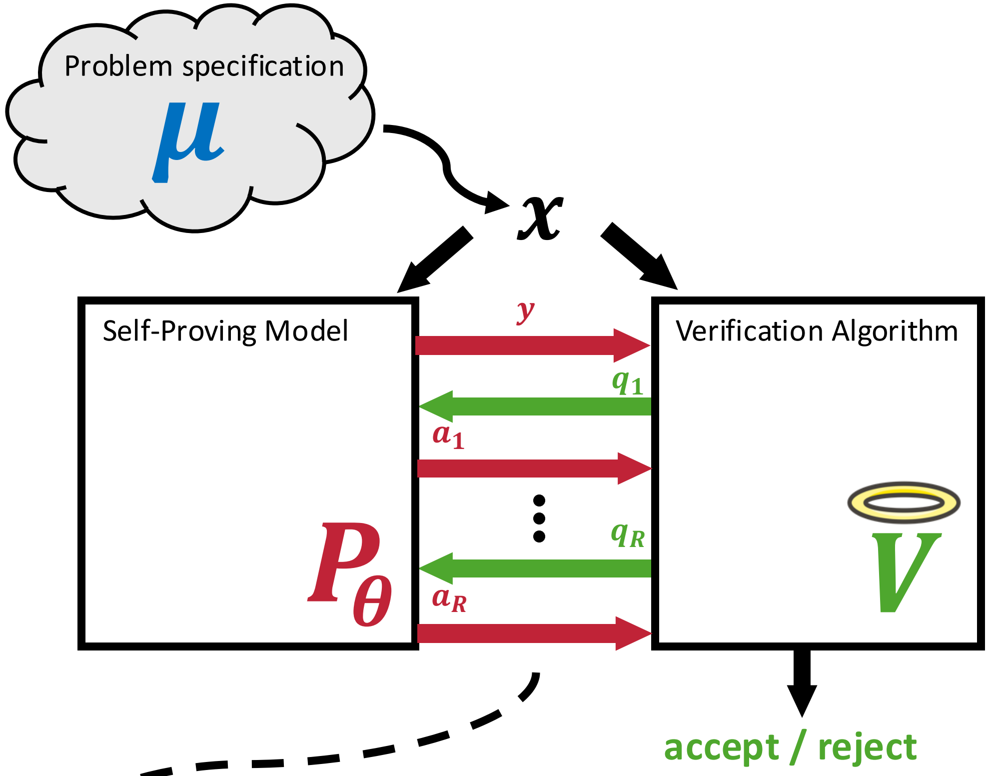


Reinforcement Learning from Verifier Feedback

RLVF

Repeat the following:

1. Generate transcript batch with P_θ. Keep only accepted transcripts.



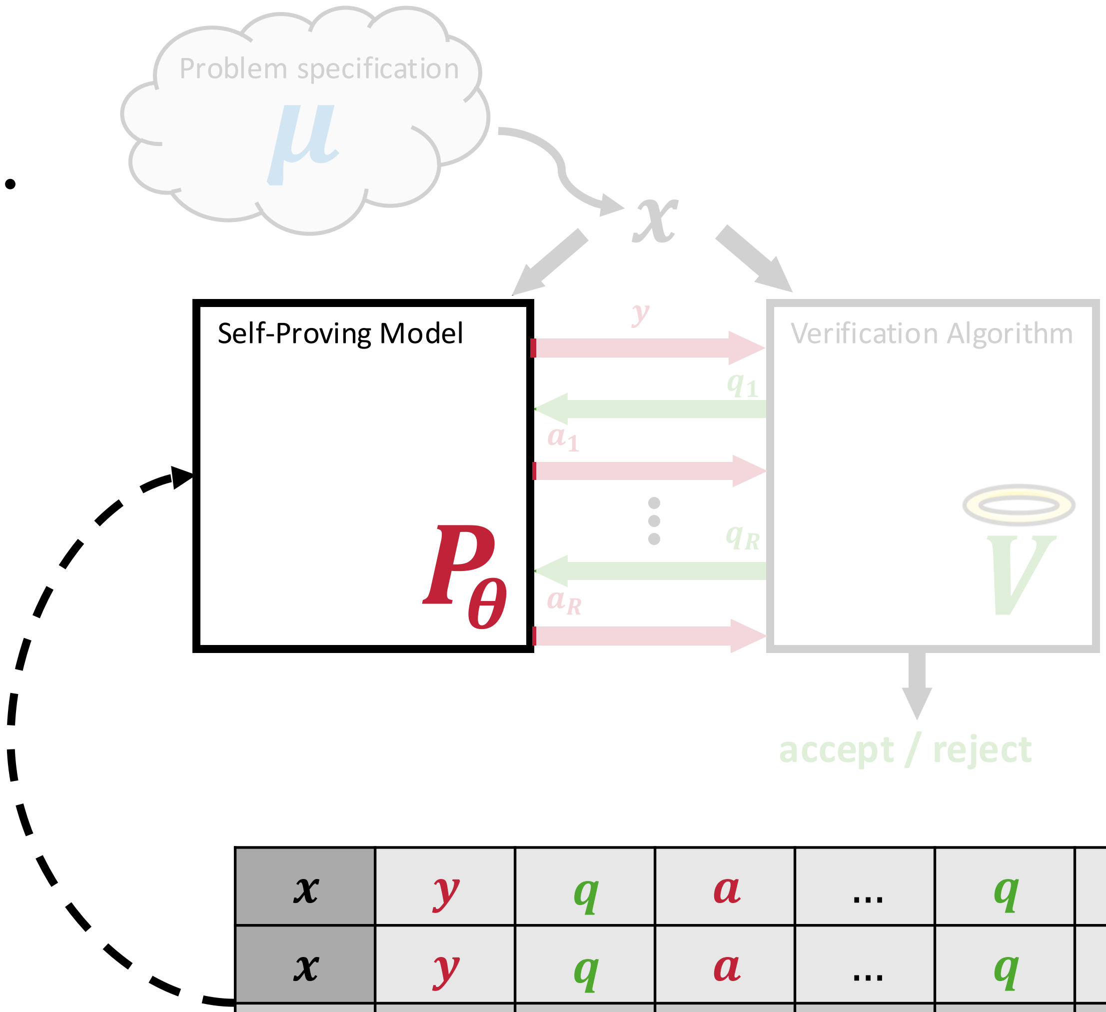
x	y	q	a	…	q	a
x	y	q	a	…	q	a
x	y	q	a	…	q	a

Reinforcement Learning from Verifier Feedback

RLVF

Repeat the following:

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



x	y	q	a	…	q	a
x	y	q	a	…	q	a
x	y	q	a	…	q	a

Reinforcement Learning from Verifier Feedback

Outline

- Self-Proving models
- Related literature
- Results
  - Theory
  - **Experiments**
- Future directions

Experiments:
Greatest Common Divisor

- 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,

2 divides **92**, **2** divides **78**,

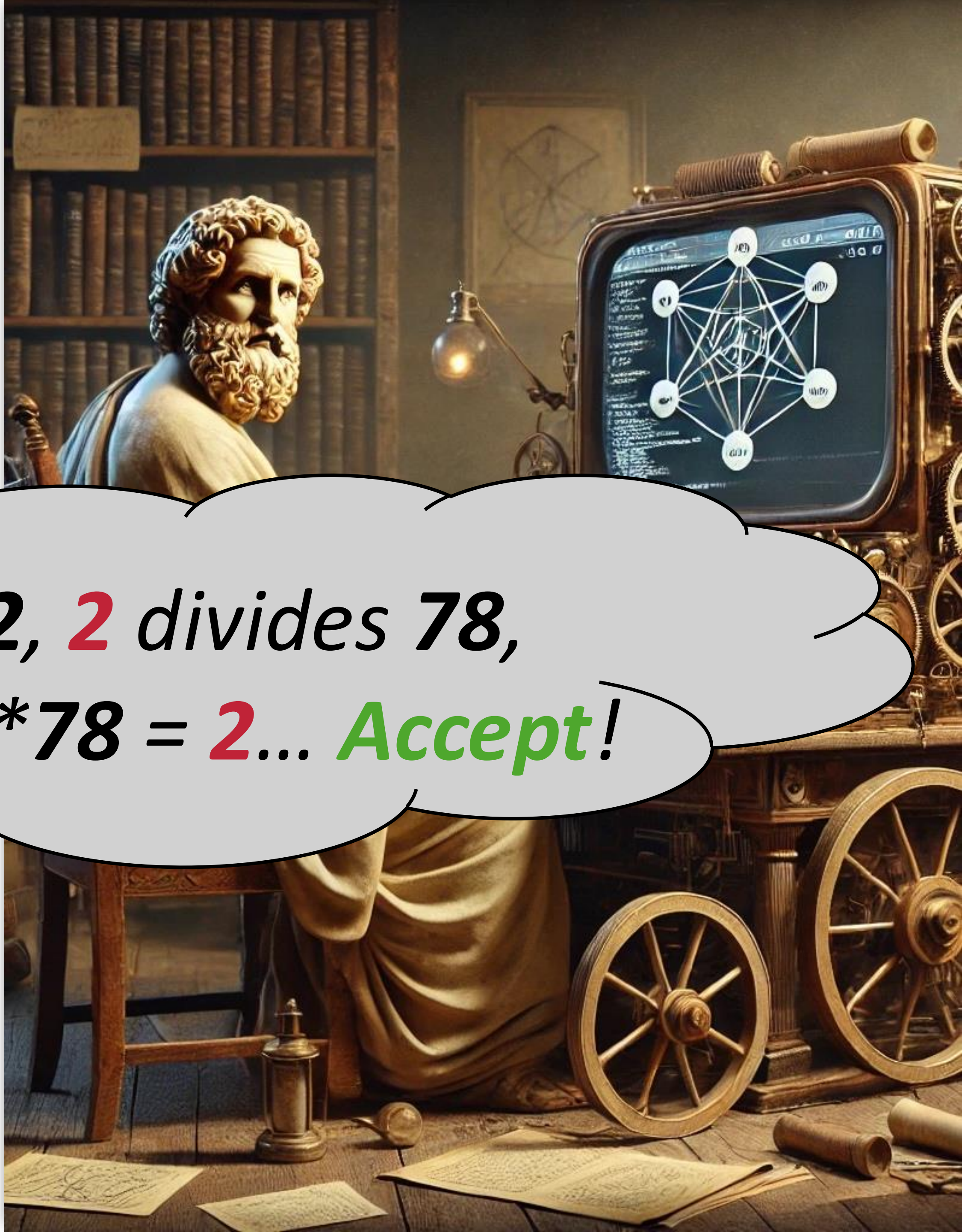
-11*92+13*78 = 2... ***Accept!***

*What is the GCD(**92, 78**)?*

P_θ → V

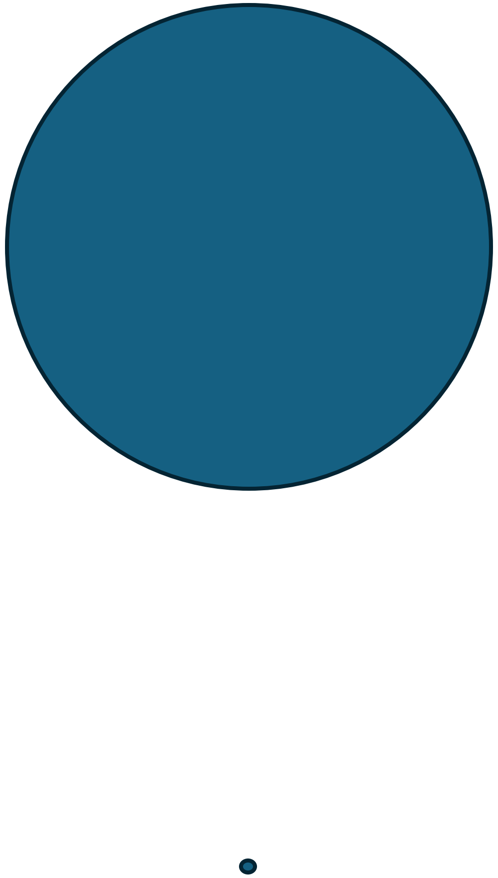
Answer: ***2***

Proof: z_1 ***= -11,*** z_2 ***= 13***

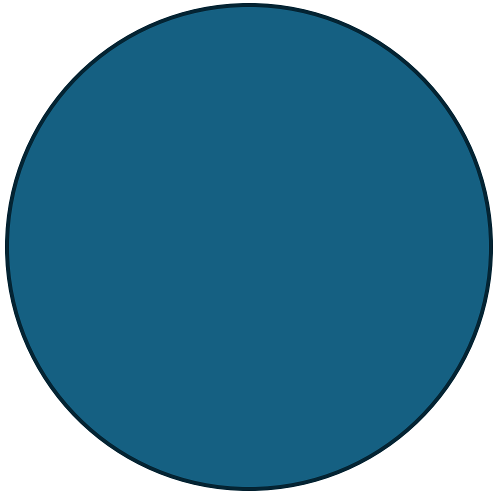


Experiments: “LLMs” in the theory group

Experiments: “LLMs” in the theory group



Experiments: “LLMs” in the theory group



GPT-N

trained on

towards

GPT (our exps)

trained on

```
+,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
```

towards computing the GCD,
and proving it to a sound verifier!

Experimental results: Transcript Learning

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

Experimental results: Transcript Learning

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

Γύρισα!
(I’m back!)

- 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).

Initialize $r_0 = x_0$, $r_1 = x_1$, $s_0 = 1$, $s_1 = 0$, and $q = 0$.
while $r_1 \neq 0$ **do**
    Update $q := \lfloor r_0 / r_1 \rfloor$.
    Update $(r_0, r_1) := (r_1, r_0 - q \times r_1)$.
    Update $(s_0, s_1) := (s_1, s_0 - q \times s_1)$.
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).

Initialize $r_0 = x_0$, $r_1 = x_1$, $s_0 = 1$, $s_1 = 0$, and $q = 0$.

while $r_1 \neq 0$ **do**

$$\left.\begin{array}{l} \text{Update } q := \lfloor r_0 / r_1 \rfloor. \\ \text{Update } (r_0, r_1) := (r_1, r_0 - q \times r_1). \\ \text{Update } (s_0, s_1) := (s_1, s_0 - q \times s_1). \end{array}\right\} \textbf{Annotation}$$

Output GCD $y = r_0$ and Bézout coefficients $z_0 := s_0$ and $z_1 := (r_0 - s_0 \cdot x_0)/x_1$.

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

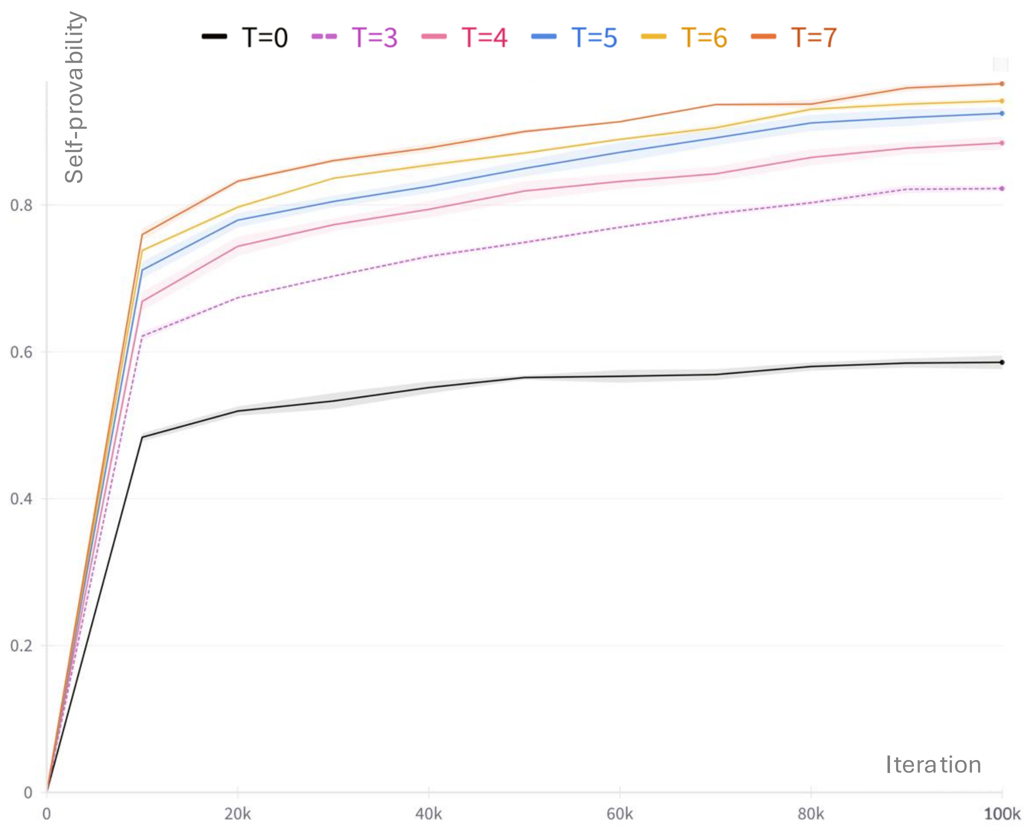
sign tokens

digits

delimiters

Annotated Transcript Learning

Longer annotations help

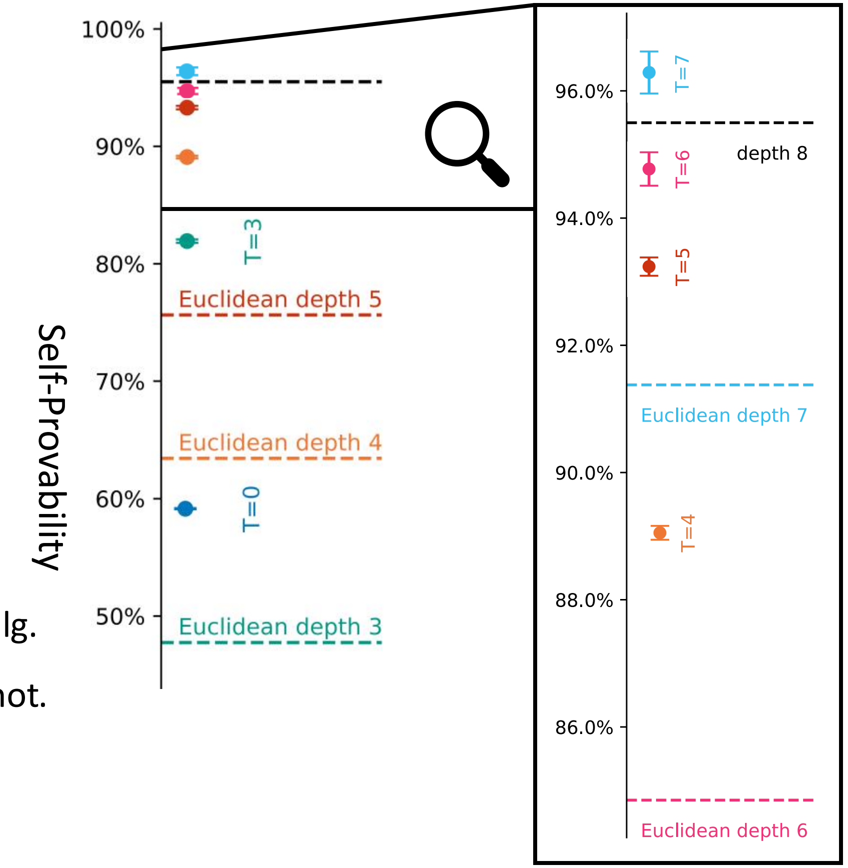


Models generalize beyond annotations

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.

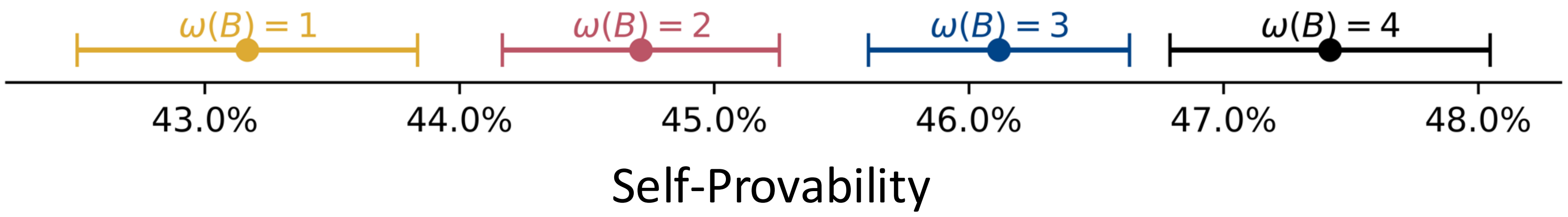
= length of “ideal” annot.



Base of Representation

"Early during training, transformers learn to predict products of divisors of the base $\boldsymbol{B}$ *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).



Outline

- Self-Proving models
- Related literature
- Results
- Future directions

Future directions

- Reinforcement Learning from Verifier Feedback (RLVF)
  - Sample complexity bounds via RL theory
- From Theory to Practice:
  - Proof system complexity: #rounds, randomized verifier, …
  - Problem complexity
    - Can you learn to prove in a low-accuracy baseline?
    - Does Self-Provability increase accuracy?
  - "Practical" settings: Proof of harmlessness.
  - Larger models.
- 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?
- Connections to existing AI Safety frameworks

Thank you!

Code here!

Paper here!