

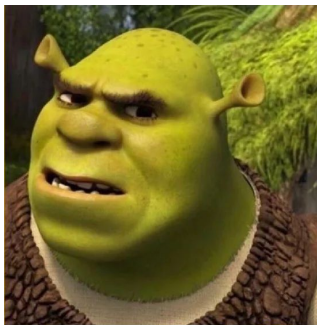
UCLA

**Computer
Science**



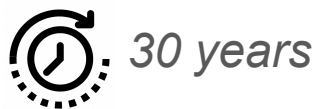
Symbolic Reasoning in the Age of Large Language Models

Guy Van den Broeck



Reasoning with Symbolic AI

- Logic and probabilistic
- **Deductive** reasoning algorithms
- Correct on *all* problems
- Limited scope
- Intractable

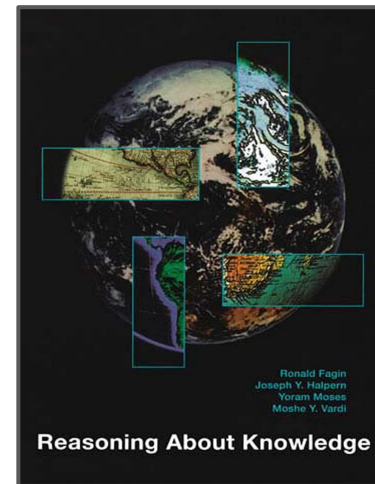


30 years



Reasoning with Transformers

- Build chains of thought
- **Inductive & Transductive** reasoning from data
- Correct on *many* problems
- Unlimited scope
- Tractable



Use the digits [0-9] to make three numbers: x,y,z so that $x+y=z$

Thought for 270 seconds ^

Okay, so I need to use each digit from 0 to 9 exactly once to create three numbers x, y, and z such that $x + y = z$. Hmm, let me think about how to approach this.

...

Verification:

- The sum $752 + 346 = 1098$ is correct.
- All digits from 0 to 9 are used exactly once.

Thus, the final answer is:

`752 + 346 = 1098`

<https://www.datacamp.com/blog/deepseek-r1-vs-v3>

Questions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of LLMs?
2. Can we put “probabilistic model checking” into the LLM?
3. Can we put “LLM probability” into the symbolic reasoning?
4. Do deductive reasoning algorithms still have a purpose in the age of LLMs? (reprise)

Questions for this talk:



1. **Do deductive reasoning algorithms still have a purpose in the age of LLMs?**
2. Can we put “probabilistic model checking” into the LLM?
3. Can we put “LLM probability” into the symbolic reasoning?
4. Do deductive reasoning algorithms still have a purpose in the age of LLMs? (reprise)

Why did this happen?

- **Not** because of **me**: *I don't know Rust and have written 0 LOC*
- **Not** because of **the LLM**: *it makes too many mistakes to write this code*
- **Not** because of **me and the LLM** working together:
I don't have enough time to understand diffs for 274,604 LOC

Why?

- Called 100,000 **symbolic** tools (rust compiler, profiler, sat solvers, ...)
- Strong **symbolic** guardrails: 1000 unit tests and ground-truth answers
- Strong **symbolic** objective: runtime on benchmarks

**It happened because of
formal methods/neurosymbolic AI!**

Math and CS Theory have become Neurosymbolic

```
equational.lean 1 M X
...
GROUP 1
  abbrev Equation1689 (M : Type _) (Magma M) := ∀ x y z : M, x =
    (y ∘ x) ∘ ((x ∘ z) ∘ z)
GROUP 2
  abbrev Equation2 (M : Type _) (Magma M) := ∀ x y : M, x = y
variable (M : Type _) (Magma M)
/-
  Human-readable proof that  $\forall x = (y \circ (x \circ z)) \circ z$  (equation 1689)
  implies the singleton law (equation 2). -/
19
20 /- We denote  $\$S_x(x) = (x \circ z) \circ z$ . -/
21
22 /- ... and  $\$S(x,y) = x \circ_z(y) = x \circ (y \circ z)$ . -/
23
24 abbrev f (x y : M) := x ∘ (S y x)
25
26 lemma f_eq (x y : M) : f x y = x ∘ ((x ∘ y) ∘ y) := rfl
27
28 /- The main equation is  $\$S(x,y) \circ_x(x) \circ \$S$ . -/
29
30 lemma main_eq (x y z : M) : x = (y ∘ x) ∘ z ∘ x := by
  rw [f_eq]
  -- unfold f
  rw [f_S_x]
  -- unfold S
  rw [S_eq]
```

expMath: Exponentiating Mathematics

Home | Research | Programs | ExpMath: Exponentiating Mathematics

FAQs | Contact us

Summary

Mathematics is the source of significant technological advances. However, progress in math is slow for two primary reasons.

1. Decomposing problems into useful lemmas is a laborious and manual process. To advance the field of mathematics, mathematicians use their knowledge and experience to explore candidate lemmas, which, when composed together, prove theorems. Ideally, these lemmas are generalizable beyond the specifics of the current problem so they can be easily understood and ported to new contexts.

How AI is changing the nature of mathematical research

By Michael Kearns, Aaron Roth
March 9, 2026 | 10 min read

What machine learning theorists learned using AI agents to generate proofs — and what comes next.

Modern AI coding tools have revolutionized software engineering, with developers now using AI assistants to write a [substantial fraction of their code](#) across a range of applications. As scientists studying the theory of machine learning, we're already seeing a similar transformation in basic scientific methodology, especially for research of a mathematical nature.

More precisely, AI tools are now able to develop and write rigorous mathematical proofs only from prompts providing high-level proof sketches. These proofs are written in longstanding "languages" for detailing mathematical arguments, in the same way that code is written in formal programming languages like Python. AI seems to have become proficient in both kinds of languages and their underlying logics.

We came to this realization during a three-week period last summer, when we used agentic AI tools to write a mathematical paper that normally would have taken months. The [50-page paper](#) describes and solves an optimization problem based on concepts from graph theory and machine learning. A typical prompt we would give the AI to set up the general framework for our paper looked like this: "Imagine a directed acyclic network of linear least-squares learning agents, each of which shares a common

"Working with proof-based AI tools is akin to collaborating with a smart, broadly educated but occasionally error-prone colleague."

The world has become Neurosymbolic

Victory!

- Software engineering has become Neurosymbolic
- Math and CS Theory have become Neurosymbolic
- Robotics has become Neurosymbolic
- Algorithms, Architecture has become Neurosymbolic (AlphaEvolve)
- AI4Science has become Neurosymbolic
- Computer Vision has become Neurosymbolic



Question:
Is the carriage to the right of a horse?

Codex
(Few-Shot Prompting)

In-Context Examples

```
# Image 1: On which side of the picture is the rug?  
img = open_image("Image1.jpg")  
rug_pos_x, rug_pos_y = get_pos(img, "rug")  
if rug_pos_x < (LEFT+RIGHT)/2:  
    answer = "left"  
else:  
    answer = "right"  
...
```

Code Generation

```
horse_exists = query(img, "Is there a horse?")  
answer = "no"  
  
if horse_exists == "yes":  
    carriage_pos_x, carriage_pos_y = get_pos(img, "carriage")  
    horse_pos_x, horse_pos_y = get_pos(img, "horse")  
    if carriage_pos_x > horse_pos_x:  
        answer = "yes"
```

Execute Code

`query(img, "Is there a horse?")`

Captions:

1. 'a police horse pulled by a fire policeman in a wagon',
2. 'man riding a horse drawn carriage pulling horse next to a officer', ...

`returns "yes"`

`get_pos(img, "carriage")` `get_pos(img, "horse")`



`returns 5, 11`



`returns 12, 11`

`carriage_pos_x < horse_pos_x`

Answer:
No

Why did neurosymbolic AI “win”? *No thanks to us.*

Success stories **despite** not understanding the symbolic side well.

- Ask a Computer Vision researcher about the VQA code they generate. Answer: I don't know, it's just Python.
- AlphaEvolve generated *hundreds of millions of programs*.
No attempt to be smart about it.
- A \$200-per-month Claude Code subscription uses \$5,000 in compute, subsidized by Anthropic [forbes]

Scale and brute force compensate for this lack of understanding.

Why did neurosymbolic AI “win”?

The philosophy was always right.

Neurosymbolic won because of the overwhelming gravitational pull of **tools** and **languages** that provide

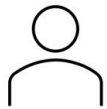
{
Compositionality
Abstraction
Soundness
}

Neurosymbolic data, training, and inference is vastly more powerful because of these properties!

Questions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of LLMs?
2. **Can we put “probabilistic model checking” into the LLM?**
3. Can we put “LLM probability” into the symbolic reasoning?
4. Do deductive reasoning algorithms still have a purpose in the age of LLMs? (reprise)



Generate a sentence using "frisbee", "caught", and "dog" in that order



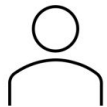
The frisbee flew through the air and the dog caught it with excitement.

GPT 4.1-mini (apr 2025)



The energetic dog leapt into the air, caught the frisbee in mid-flight, and proudly trotted back to its owner.

Llama-3.2-3B (sep 2024)



You got the order wrong, please respect the given order



Here is a new sentence with the words in that order: The dog's owner was about to catch the frisbee that had been flying through the air.



A frisbee is caught by a dog.

A pair of frisbee players are caught in a dog fight.

Ctrl-G (jun 2024)

Reasoning about all Future Tokens: *Constraints*

$p(\text{next-token} \mid \alpha, \text{prefix})$

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} = \text{"the weather is"})$

Lexical Constraint α : sentence contains keyword "winter"

Reasoning about all Future Tokens: *Constraints*

$p(\text{next-token} \mid \alpha, \text{prefix})$

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} = \text{"the weather is"})$

Lexical Constraint α : sentence contains keyword "winter"

$\propto p(\text{next-token} \mid \text{prefix}) \cdot p(\alpha \mid \text{next-token}, \text{prefix})$



Bayes' rule lets us reason backwards in time!

Reasoning about all Future Tokens: *Constraints*

$$p(\text{next-token} \mid \alpha, \text{prefix})$$

cold	0.025
warm	0.001

$$\propto p(\text{next-token} \mid \text{prefix})$$

cold	0.05
warm	0.10

Constrained Generation: $\Pr(x_{t+1} \mid \alpha, x_{1:t} = \text{"the weather is"})$

Lexical Constraint α : sentence contains keyword "winter"

$$p(\alpha \mid \text{next-token}, \text{prefix})$$

cold	0.50
warm	0.01



Reasoning about all Future Tokens

$$p_{lm}(\text{next-token} \mid \alpha, \text{prefix})$$

Using Bayes rule,

$$\propto p_{lm}(\text{next-token} \mid \text{prefix}) \cdot \cancel{p_{lm}(\alpha \mid \text{next-token}, \text{prefix})}$$



Intractable



Reasoning about all Future Tokens

$$p_{lm}(\text{next-token} \mid \alpha, \text{prefix})$$

Abusing Bayes rule,

$$\propto p_{lm}(\text{next-token} \mid \text{prefix}) \cdot p_{circuit}(\alpha \mid \text{next-token}, \text{prefix})$$



Use a tractable circuit model distilled from the transformer LLM...

A 'tractable digital twin'

Reasoning about all Future Tokens: Constraints

$$p_{lm}(\text{next-token} \mid \alpha, \text{prefix})$$

Abusing Bayes rule,

$$\propto p_{lm}(\text{next-token} \mid \text{prefix}) \cdot p_{circuit}(\alpha \mid \text{next-token}, \text{prefix})$$



Theorem. Given

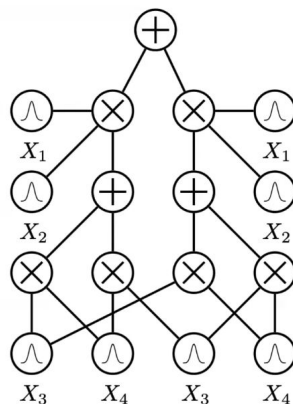
1. a deterministic finite automata constraint α with m edges and
 2. a probabilistic circuit $p(\cdot)$ with h hidden states (representing a Hidden Markov Model),
- computing $p(\alpha \mid x_{1:t})$ over a sequence of n future tokens takes $O(nmh^2)$ time.

Reasoning about all Future Tokens: Constraints

$$p_{lm}(\text{next-token} \mid \alpha, \text{prefix})$$

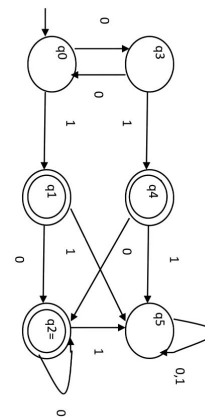
Abusing Bayes rule,

$$\propto p_{lm}(\text{next-token} \mid \text{prefix}) \cdot p_{circuit}(\alpha \mid \text{next-token}, \text{prefix})$$



weights/circuit p

X



models/dfa α

Reasoning about all Future Tokens: Constraints

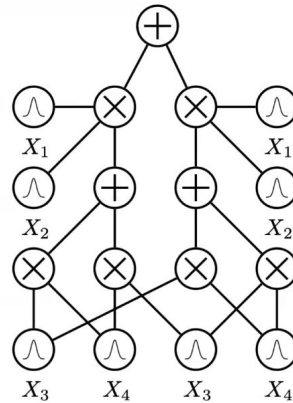
$$p_{lm}(\text{next-token} \mid \alpha, \text{prefix})$$

Abusing Bayes rule,

$$\propto p_{lm}(\text{next-token} \mid \text{prefix}) \cdot p_{circuit}(\alpha \mid \text{next-token}, \text{prefix})$$

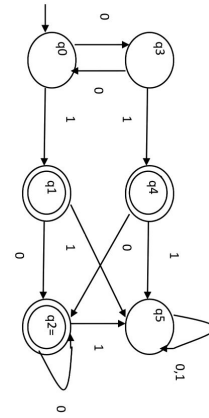


\sum future



weights/circuit p

\times



models/dfa α

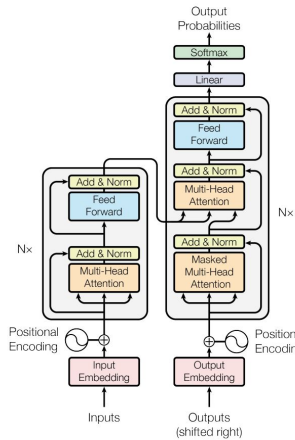
Reasoning about all Future Tokens: Constraints

$$p_{lm}(\text{next-token} \mid \alpha, \text{prefix})$$

Abusing Bayes rule,

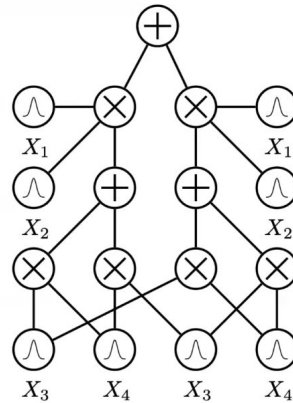


$$\propto p_{lm}(\text{next-token} \mid \text{prefix}) \cdot p_{circuit}(\alpha \mid \text{next-token}, \text{prefix})$$

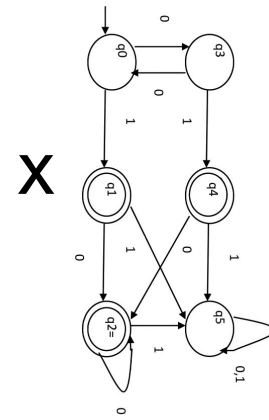


transformer p

$\cdot \sum_{\text{future}}$



weights/circuit p



models/dfa α

Interactive Text Editing

“First they've defeated a small squad [BLANK] are few humans left, and despite their magical power, their numbers are getting fewer.”

Interactive Text Editing

User: given the following context, generate infilling text for [BLANK] using key phrases "alien mothership", "far from over"; generated text must contain 25 - 30 words.

"First they've defeated a small squad [BLANK] are few humans left, and despite their magical power, their numbers are getting fewer."

Ctrl-G



"First they've defeated a small squad of aliens, then a larger fleet of their ships. Eventually they've even managed to take down the alien mothership. But their problems are far from over. There are few humans left, and despite their magical power, their numbers are getting fewer."

Interactive Text Editing with key phrase (K) or length (L) constraints



	K	L	K&L
<i>Quality</i>			
TULU2	2.64	2.78	2.74
GPT3.5	2.22	2.27	2.31
GPT4	3.33	3.53	3.10
Ctrl-G	3.56	3.73	3.59
<i>Success</i>			
TULU2	12%	20%	3%
GPT3.5	22%	54%	10%
GPT4	60%	20%	27%
Ctrl-G	100%	100%	100%
<i>Overall</i>			
TULU2	7%	10%	1%
GPT3.5	0%	5%	2%
GPT4	41%	17%	14%
Ctrl-G	76%	78%	82%

→ *How many stars by humans?*

→ *Follows instructions?*

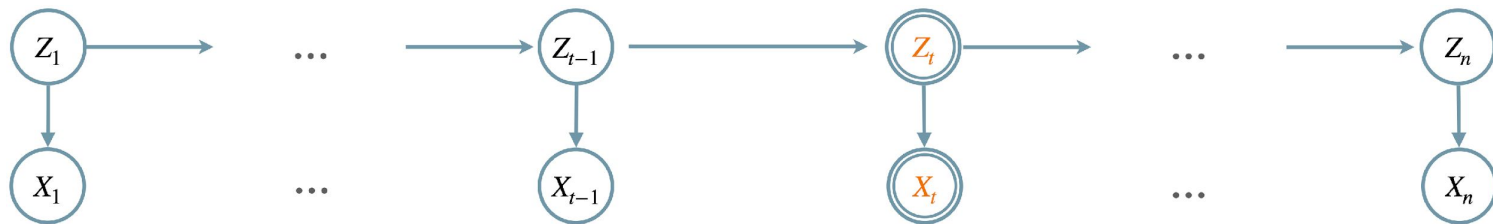
→ ★★☆☆☆ & Up + *Follows instructions?*

→ **Ctrl-G based on Llama2-7B wipes the floor with GPT4, which is a >100x bigger LLM**

Probabilistic Circuit Language Model

How did we train a probabilistic circuit to solve Ctrl-G?

Keep it simple... just a classic **Hidden Markov Model (HMM)** with 32,768 hidden states and 2 billion parameters... on the GPU



Grade School Math Benchmark

Question: Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?

Vanilla LLM Answer: The price of the 2nd glass is $(16 / 2) * 60\% = 8$ dollars. So one pair of glasses costs $16 + 8 = 24$ dollars. **So the answer is 24.**

Grade School Math Benchmark

Question: Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?

Vanilla LLM Answer: The price of the 2nd glass is $(16 / 2) * 60\% = 8$ dollars. So one pair of glasses costs $16 + 8 = 24$ dollars. **So the answer is 24.**

Ctrl-G Answer: The second glass costs $5 * .6 = \$3$. So each set of two glasses actually costs $5 + 3 = \$8$. He wants $16 / 2 = 8$ sets of two. That means he needs to pay $8 * 8 = \$64$. **So the answer is 64.**

Which constraint improves accuracy?

Grade School Math Benchmark

Question: *Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?*

Vanilla LLM Answer: The price of the 2nd glass is $(16 / 2) * 60\% = 8$ dollars. So one pair of glasses costs $16 + 8 = 24$ dollars. So the answer is 24.

Ctrl-G Answer: The second glass costs $5 * .6 = \$3$. So each set of two glasses actually costs $5 + 3 = \$8$. He wants $16 / 2 = 8$ sets of two. That means he needs to pay $8 * 8 = \$64$. So the answer is 64.

Use all the numbers in the problem statement!

Robotics has become **Neurosymbolic** (EAI Benchmark)

Task: use the rag to clean the refrigerator

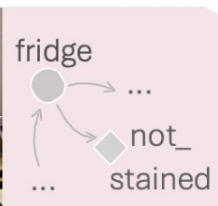
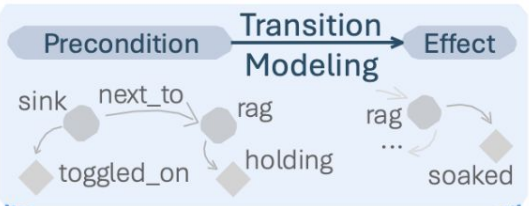
<https://arxiv.org/pdf/2410.07166>

stained(fridge) *next_to(rag, sink)* *toggled_on(sink)*

Subgoal Decomposition

soaked(rag) ...

not_stained(fridge)



GRASP(rag) PLACE_NEXTTO(sink) TOGGLE_ON(sink)

SOAK(rag)

Action Sequencing

Goal Interpretation

[TASK] Transform initial → target environment state.

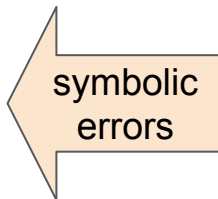
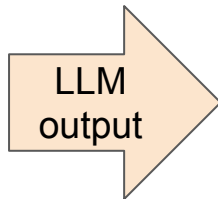
[OUTPUT] JSON list (execution order): {"action": ACTION_NAME, "object": OBJ}; use "OBJ1, OBJ2" only for *_NEXTTO_ONTOP.

[ACTIONS] From catalog: GRASP (LEFT/RIGHT), PLACE_(INSIDE/ONTOP/NEXTTO), OPEN/CLOSE, TOGGLE_ON/OFF, CLEAN, SOAK, DRY, SLICE. One object per hand; PLACE releases. OPEN/CLOSE/... require ≥1 empty hand & valid properties. Act only if target not inside a closed container.

[STATE] ['predicate', obj1, (obj2?)].

[INPUT] initial states..., target states..., interactable objects...

[TARGET (this instance)] rags NEXTTO sink_82; soap_0 INSIDE sink_82; trays (all) INSIDE fridge_97 & NOT stained; bowl_0 NEXTTO sink_82 & NOT dusty; fridge_97 NOT stained.



```
[{"action": "RIGHT_GRASP", "object": "rag_0"},
{"action": "RIGHT_PLACE_INSIDE", "object": "sink_82"},
{"action": "TOGGLE_ON", "object": "sink_82"},
{"action": "SOAK", "object": "rag_0"},
{"action": "RIGHT_GRASP", "object": "soap_0"},
{"action": "RIGHT_PLACE_INSIDE", "object": "sink_82"},
{"action": "LEFT_GRASP", "object": "rag_0"},
{"action": "TOGGLE_OFF", "object": "sink_82"},
```

```
{"action": "OPEN", "object": "fridge_97"},
{"action": "CLEAN", "object": "tray_0"},
{"action": "CLEAN", "object": "tray_1"},
{"action": "CLEAN", "object": "fridge_97"},
```

```
{"action": "RIGHT_GRASP", "object": "bowl_0"},
{"action": "RIGHT_PLACE_NEXTTO", "object": "sink_82"},
{"action": "CLEAN", "object": "bowl_0"},
```

```
{"action": "LEFT_PLACE_NEXTTO", "object": "sink_82"},
...
```

Beyond Token-Level Control

Decompose problem:

1. Token-level syntax
DFA
2. Meta-DFA capturing
high-level semantics
(preconditions)

Now Llama 3 8B
success rate goes
from 24% to 92%!

On par with closed LLMs.

Model	Action Sequencing		Subgoal Decomposition	
	Task SR	Execution SR	Task SR	Execution SR
Claude-3 Haiku	54.8	60.7	78.4	82.8
Claude-3 Sonnet	58.0	63.3	83.1	86.4
Claude-3 Opus	64.6	69.5	86.7	89.9
Claude-3.5 Sonnet	76.1	81.3	89.1	92.0
Cohere Command R	44.9	44.3	71.3	78.1
Cohere Command R+	54.1	65.2	77.8	83.7
Gemini 1.0 Pro	45.6	56.7	70.4	84.6
Gemini 1.5 Flash	69.5	75.4	89.1	94.1
Gemini 1.5 Pro	76.7	83.6	87.0	91.1
GPT-3.5-turbo	24.9	40.7	69.2	81.4
GPT-4-turbo	60.0	65.2	85.5	94.1
GPT-4o	71.5	81.3	87.6	91.1
o1-mini	71.5	76.4	79.3	84.6
o1-preview	65.2	72.5	89.4	93.2
Llama 3 8B Instruct	21.3	23.6	48.8	58.0
Llama 3 70B Instruct	59.0	66.6	78.4	87.3
Mistral Large	78.4	84.6	84.3	92.0
Mixtral 8x22B MoE	63.3	67.9	80.5	90.2
ctrlg				
Llama 3 8B Instruct	21.3	23.6	48.8	58.0
+Syntax Ctrlg	48.7	51.5	60.1	79.0
+Semantic Ctrlg	85.3	91.8	79.8	82.5

Advantages of Ctrl-G:

1. Constraint α is guaranteed to be satisfied:

if next-token makes α unsatisfiable, $p_{lm}(\text{next-token} \mid \alpha, \text{prefix}) = 0$.

$$p_{lm}(\text{next-token} \mid \text{prefix}) \cdot p_{circuit}(\alpha \mid \text{next-token}, \text{prefix}) = 0$$

2. Generalizes well to unseen reasoning tasks, because all tasks are unseen :-)
(training on a distribution over tasks is slow and brittle!)
3. Bayesian = goal-oriented (\leftrightarrow structured generation tools)

The Weather in Australia Is Beautiful, the Kangaroos Are TACAS

You can control an intractable generative model using a generative model that is *tractable for symbolic reasoning*.

Questions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of LLMs?
2. Can we put “probabilistic model checking” into the LLM?
3. **Can we put “LLM probability” into the symbolic reasoning?**
4. Do deductive reasoning algorithms still have a purpose in the age of LLMs? (reprise)

Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM.

```
def compute_answer():  
    total_students = 200  
    smallest_group = (total_students - 10) // 3  
    return smallest_group
```

Not exactly correct! Let's ask again.

Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM a second time.

```
def compute_answer():  
    total_students = 200  
    smallest_group = (total_students + 10) // 3  
    return smallest_group
```

Not exactly correct! Let's ask again.

Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM a third time.

```
def compute_answer():  
    total_students = 200  
    smallest_group = (total_students + 10) // 3  
    return smallest_group
```

Same program
Ask again.

Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM a fourth time.

```
def compute_answer():  
    total_students = 200  
    smallest_group = (total_students // 3) - 10  
    return smallest_group
```

Not exactly correct! Let's ask again.

Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM a fifth time.

```
def compute_answer():  
    total_students = 200  
    difference = 10  
    group1 = group2 = (total_students + difference) // 3  
    group3 = group1 - difference  
    return group3
```

Phew!
finally

Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

Why do we need so many samples from the language model?

Deterministic Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

Why do we need so many samples from the language model?

LLM is a distribution over token sequences.

We sample a deterministic program and throw away the uncertainty!!!

Probabilistic Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM but let's not throw away the uncertainty

```
def compute_answer():  
    total_students = x12 x13 x14  
    smallest_group = (total_students + x25 x26) // x27  
    return smallest_group
```

- Interpret digits as random variables.

Probabilistic Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

🤔 Let's ask an LLM but let's not throw away the uncertainty

```
def compute_answer():  
    total_students = x12 x13 x14  
    smallest_group = (total_students z23 x25 x26) z28 x27  
    return smallest_group
```

- Interpret digits as random variables.
- Interpret operators as random variables.

 **A Probabilistic Program!!!**

Probabilistic Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

```
def compute_answer():  
    total_students = X12 X13 X14  
    smallest_group = (total_students Z23 X25 X26) Z28 X27  
    return smallest_group
```

Sample from probabilistic program

- Much cheaper than sampling from LLM
- Even do full probabilistic program analysis

```
def compute_answer():
```

...

```
def compute_answer():
```

...

```
def compute_answer():
```

...

```
def compute_answer():
```

...

Probabilistic Programs-of-thought for grade school math

Question: A class of 200 students is split into 3 groups such that 2 of them are equal in number and the last one (which is the smallest) is 10 less than each of the other groups. How many students are in this (smallest) group?

```
def compute_answer():  
    total_students = X12 X13 X14  
    smallest_group = (total_students X23 X25 X26) Z28 X27  
    return smallest_group
```

```
def compute_answer():  
    ...
```

```
def compute_answer():  
    total_students = 200  
    smallest_group = (total_students + 20) // 3  
    return smallest_group  
    ...
```

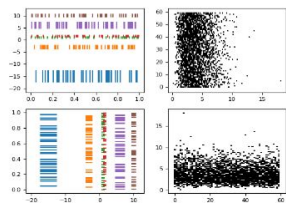


```
def compute_answer():  
    ...
```

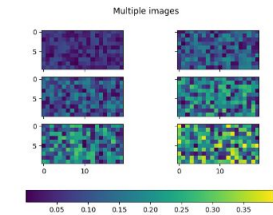
Sample from probabilistic program

- Much cheaper than sampling from LLM
- Even do full probabilistic program analysis

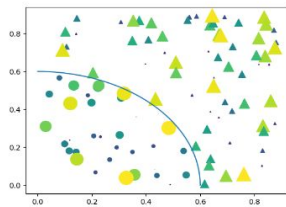
Plot2Code



ground-truth



ground-truth



ground-truth

```

--- LLM sample
+++ PPoT sample
- data1 = np.random.randn(5, 50)
+ data1 = np.random.randn(6, 50)
- data3 = np.random.gamma(1, 1, (60, 50))
+ data3 = np.random.gamma(2, 1, (60, 50))
+ fig, axs = plt.subplots(2, 2,
    figsize=(80, 80))
- fig, axs = plt.subplots(2, 2,
    figsize=(10, 10))
    
```

diff

```

--- LLM sample
+++ PPoT sample
# Create a figure with a grid of
subplots
- fig, axs = plt.subplots(2, 2,
    figsize=(80, 1))
+ fig, axs = plt.subplots(3, 2,
    figsize=(10, 8))
for ax, d in zip(axs.flat, data):
    im = ax.imshow(d,
        vmin=np.min(data),
        vmax=np.max(data))
    
```

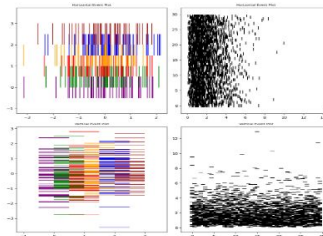
diff

```

--- LLM sample
+++ PPoT sample
- x = np.random.rand(100) * 0.9
+ x = np.random.rand(200) * 0.9
- plt.scatter(x, y, s=marker_area,
    c=marker_color, marker='^',
    alpha=0.5, label='Region 1')
+ plt.scatter(x, y, s=marker_area,
    c=marker_color, marker='^',
    alpha=0.6, label='Region 1')
    
```

diff

ERROR

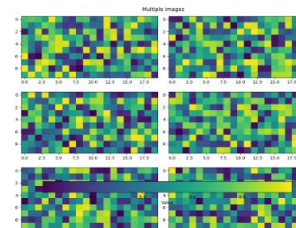


LLM sample

PPoT sample

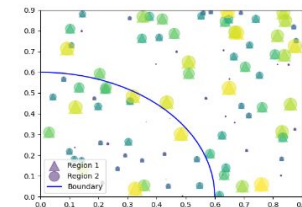


LLM sample



PPoT sample

ERROR



LLM sample

PPoT sample

Probabilistic Programs-of-thought

Best-of-n performance:

1.5% - 7% boost
for the same amount of GPU compute

math reasoning

program inversion

plot generation

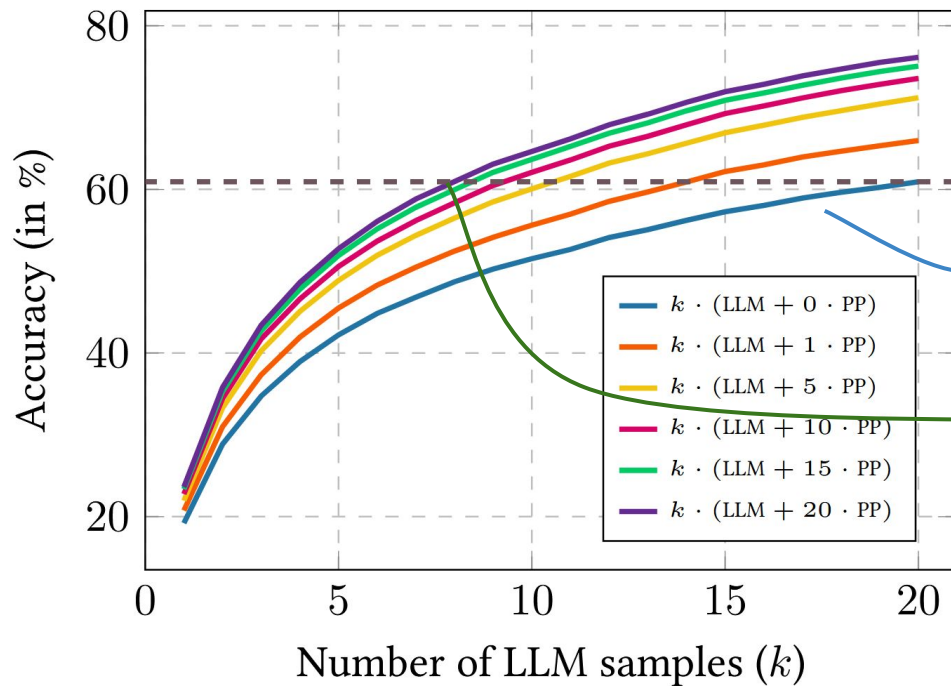
PPoT samples

LLM samples

$k \cdot (\text{LLM} + m \cdot \text{PP})$

	GSM8k			CRUXEval			Plot2Code	
	0.5B	3B	7B	0.5B	3B	7B	3B	7B
$1 \cdot (\text{LLM} + 0 \cdot \text{PP})$	24.94	72.63	82.71	30.87	40.5	57.12	39.44	36.70
$1 \cdot (\text{LLM} + 5 \cdot \text{PP})$	28.51	76.19	84.76	37.37	45.87	60.62	41.89	42.48
$5 \cdot (\text{LLM} + 0 \cdot \text{PP})$	42.53	86.20	91.20	46.13	59.13	73.37	63.86	69.92
$5 \cdot (\text{LLM} + 5 \cdot \text{PP})$	49.13	89.08	93.02	53.13	63.87	76.87	64.84	71.95
Metric	Accuracy						Text Match	

Probabilistic Programs-of-thought



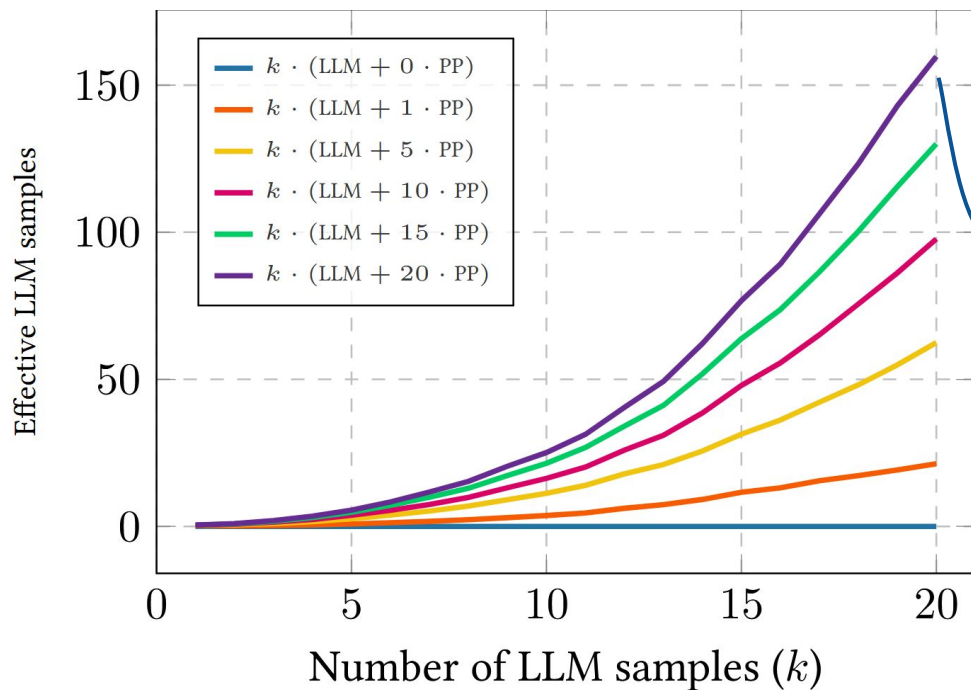
How much can we gain from PPOt?

This curve is your usual LLM best-of- n setting

Same accuracy for less than half the number of LLM calls!

Qwen 2.5 Coder on GSM8k

Probabilistic Programs-of-thought: scaling laws



How much can we gain from PPOt?

You need >150 LLM samples to achieve the same accuracy as PPOt!

Qwen 2.5 Coder on GSM8k

Questions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of LLMs?
2. Can we put “probabilistic model checking” into the LLM?
3. Can we put “LLM probability” into the symbolic reasoning?
4. **Do deductive reasoning algorithms still have a purpose in the age of LLMs? (reprise)**

Reasoning about all Future Tokens: *Alignment*

$p(\text{next-token} \mid \alpha, \text{prefix})$

Prefix: It's a pain ...

Constraint α : non-toxic

Reasoning about all Future Tokens: *Alignment*

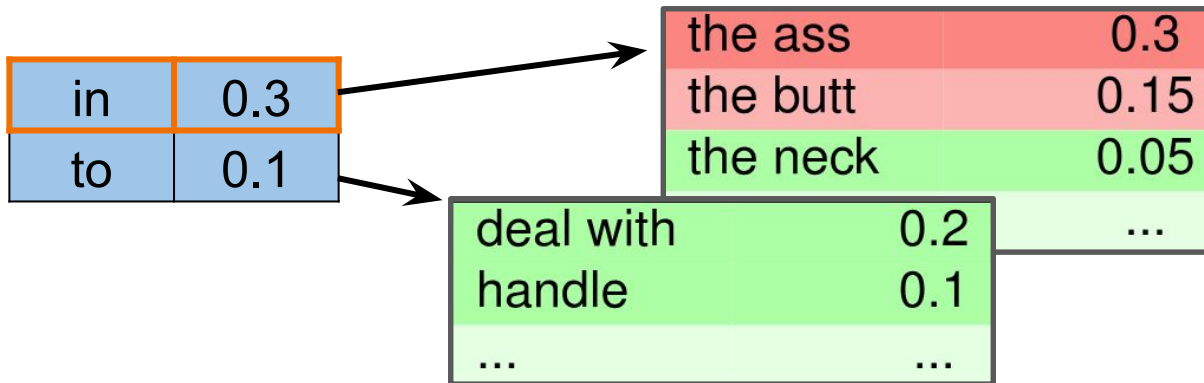
$p(\text{next-token} \mid \alpha, \text{prefix})$

Prefix: It's a pain ...

Constraint α : non-toxic

$\propto p(\text{next-token} \mid \text{prefix})$

$p(\alpha \mid \text{next-token}, \text{prefix})$



Reasoning about all Future Tokens: *Alignment*

$$p(\text{next-token} \mid \alpha, \text{prefix})$$

in	0.03
to	0.08

Prefix: It's a pain ...

Constraint α : non-toxic

$$\propto p(\text{next-token} \mid \text{prefix})$$

in	0.3
to	0.1

$$\cdot p(\alpha \mid \text{next-token}, \text{prefix})$$

in	0.1
to	0.8



RL Hammer: Ad-Hoc Probabilistic Reasoning

Toxicity

Base LLM	0.385
PPO ⁽⁷⁾	0.218
Quark ⁽⁸⁾	0.196
DPO ⁽⁹⁾	0.180

lower = less toxic



Perplexity

Base LLM	25.57
PPO ⁽⁷⁾	14.27
Quark ⁽⁸⁾	12.47
DPO ⁽⁹⁾	21.59

lower = higher LLM probability



Base LLM	52.06	Lower entropy means
DPO	39.52	lower diversity

DPO “thinks” that 99.9996% of all internet text is toxic.

Attribute Probability



0 (toxic)

1 (nontoxic)

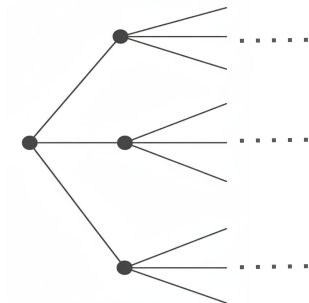
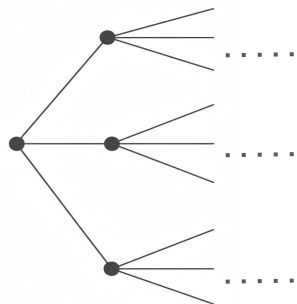
It's a pain

in

$p_{LM} = 0.3$

to

$p_{LM} = 0.1$



future text

the ass

the butt

the neck

...

...

Intractable to know
expected future toxicity



future text

deal with

handle

...

...

Attribute Probability



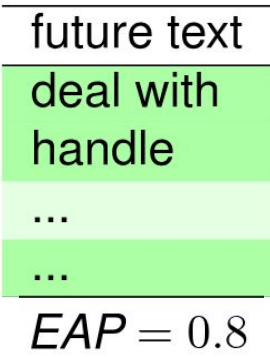
It's a pain

in

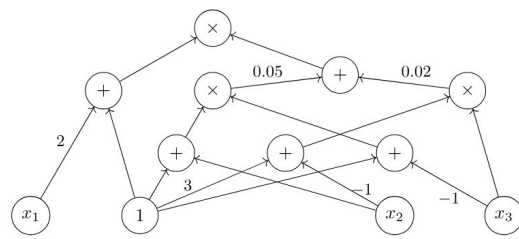
$$p_{LM} = 0.3$$

to

$$p_{LM} = 0.1$$



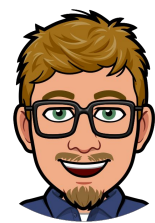
LLM continuations with *tractable probabilistic circuit*



+
Goal attribute with
log-linear classifier



=
**Efficient Expected
Attribute Probability!**



Attribute Probability



0 (toxic)

1 (nontoxic)

It's a pain

in

$$p_{LM} = 0.3$$

to

$$p_{LM} = 0.1$$

future text

the ass

the butt

the neck

...

...

$$EAP = 0.1$$

$$= p_{TRACE} \propto 0.03$$

future text

deal with

handle

...

...

$$EAP = 0.8$$

$$= p_{TRACE} \propto 0.08$$



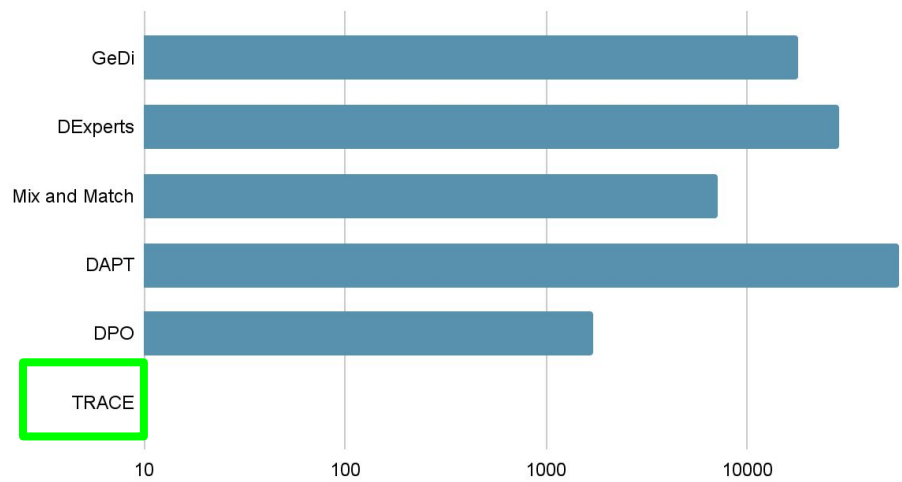
State-of-the-art LLM Detoxification

Model	Toxicity (↓)		Diversity (↑)		Fluency (↓)	Approach Type
	avg. max.	prob.	dist-2	dist-3		
GPT-2 Large Results						
GPT2	0.385	0.254	0.87	0.86	25.57	Baseline
DAPT ⁽¹⁾	0.428	0.360	0.84	0.84	31.21	Finetuning
GeDi ⁽²⁾	0.363	0.217	0.84	0.83	60.03	Decoding (Trained Guide)
FUDGE ⁽³⁾	0.302	0.371	0.78	0.82	12.97 *	Decoding (Trained Guide)
DExperts ⁽⁴⁾	0.314	0.128	0.84	0.84	32.41	Decoding (Trained Guide)
PPLM ⁽⁵⁾	0.520	0.518	0.86	0.86	32.58	Decoding (Logit Control)
MuCoLa ⁽⁶⁾	0.308	0.088	0.82	0.83	29.92	Decoding (Sampling)
PPO ⁽⁷⁾	0.218	0.044	0.80	0.84	14.27 *	RL
Quark ⁽⁸⁾	0.196	0.035	0.80	0.84	12.47 *	RL
DPO ⁽⁹⁾	0.180	0.026	0.76	0.78	21.59 *	RL
TRACE	0.163	0.016	0.85	0.85	29.83	Decoding (HMM Reasoning)
Gemma-2B Results						
Gemma-2B	0.359	0.23	0.86	0.85	15.75	Baseline
DPO ⁽⁹⁾	0.222	0.06	0.74	0.77	14.39 *	RL
TRACE	0.189	0.02	0.86	0.85	17.68	Decoding (HMM Reasoning)

TRACE is Blazingly Fast

Given a language model, and its tractable circuit twin,
train log-linear attribute classifier

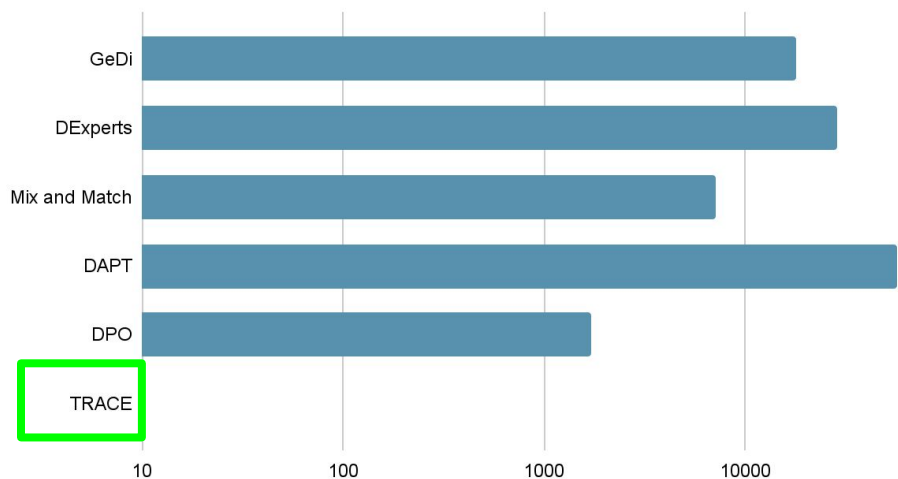
Training Time per Attribute (seconds)



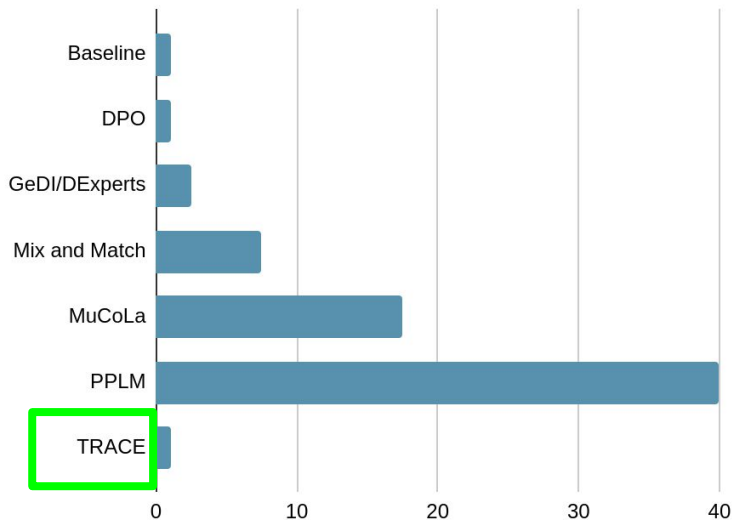
TRACE is Blazingly Fast

Given a language model, and its tractable circuit twin,
train log-linear attribute classifier,
then use Bayesian logits at decoding time (**weighted model counting**)

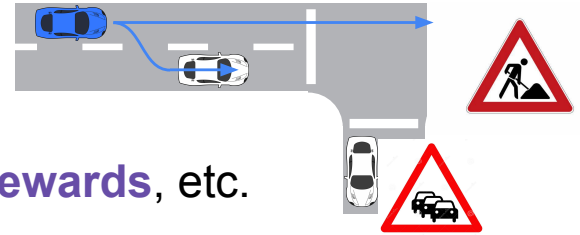
Training Time per Attribute (seconds)



Inference Time



Reasoning about all Future Tokens: *Offline RL*



Training: model the joint distribution over **states**, **actions**, **rewards**, etc.

Inference: sample next **states** and **actions**, as well as **constraints**.



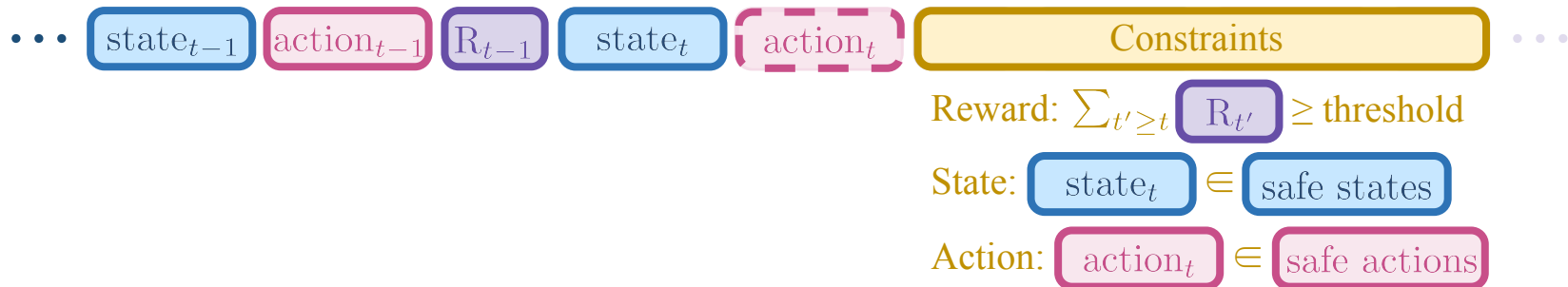
Reward: $\sum_{t' \geq t} R_{t'} \geq \text{threshold}$

State: state_t \in safe states

Action: action_t \in safe actions

$$p(\text{action} \mid \alpha, \text{prefix}) \propto p(\text{action} \mid \text{prefix}) \cdot p(\alpha \mid \text{action}, \text{prefix})$$

Reasoning about all Future Tokens: *Offline RL*



Inference: sample actions condition on past **states** and **actions**, as well as **constraints**.

$$\begin{aligned}
 & p(\text{action}_t \mid \text{state}_{\leq t}, \text{action}_{< t}, \text{Constraints}) \\
 \propto & \underbrace{p(\text{action}_t \mid \text{state}_{\leq t}, \text{action}_{< t})}_{\text{Autoregressive Transformers (GPTs)}} \cdot \underbrace{p(\text{Constraints} \mid \text{state}_{\leq t}, \text{action}_{< t})}_{\text{Probabilistic Circuits (PCs)}}
 \end{aligned}$$

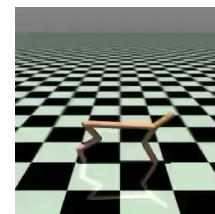
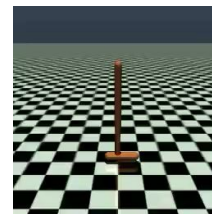
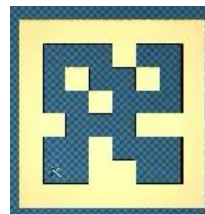
Bayes' rule



Condition on Various Constraints in Offline RL

- Condition on **high reward**: SoTA performance on standard offline RL benchmarks.

Dataset	Environment	TT		TT(+Q)		DT		DD	IQL	CQL	%BC	TD3(+BC)
		base	Trifle	base	Trifle	base	Trifle					
Med-Expert	HalfCheetah	95.0±0.2	95.1 ±0.3	82.3±6.1	89.9 ±4.6	86.8±1.3	91.9 ±1.9	90.6	86.7	91.6	92.9	90.7
Med-Expert	Hopper	110.0±2.7	113.0 ±0.4	74.7±6.3	78.5 ±6.4	107.6±1.8	/	111.8	91.5	105.4	110.9	98.0
Med-Expert	Walker2d	101.9±6.8	109.3 ±0.1	109.3±2.3	109.6 ±0.2	108.1±0.2	108.6 ±0.3	108.8	109.6	108.8	109.0	110.1
Medium	HalfCheetah	46.9±0.4	49.5 ±0.2	48.7±0.3	48.9 ±0.3	42.6±0.1	44.2 ±0.7	49.1	47.4	44.0	42.5	48.3
Medium	Hopper	61.1±3.6	67.1 ±4.3	55.2±3.8	57.8 ±1.9	67.6±1.0	/	79.3	66.3	58.5	56.9	59.3
Medium	Walker2d	79.0±2.8	83.1 ±0.8	82.2±2.5	84.7 ±1.9	74±1.4	81.3 ±2.3	82.5	78.3	72.5	75.0	83.7
Med-Replay	HalfCheetah	41.9±2.5	45.0 ±0.3	48.2±0.4	48.9 ±0.3	36.6±0.8	39.2 ±0.4	39.3	44.2	45.5	40.6	44.6
Med-Replay	Hopper	91.5±3.6	97.8 ±0.3	83.4±5.6	87.6 ±6.1	82.7±7.0	/	100.0	94.7	95.0	75.9	60.9
Med-Replay	Walker2d	82.6±6.9	88.3 ±3.8	84.6±4.5	90.6 ±4.2	66.6±3.0	73.5 ±0.1	75.0	73.9	77.2	62.5	81.8
Average Score		78.9	83.1	74.3	77.4	74.7	/	81.8	77.0	77.6	74.0	75.3



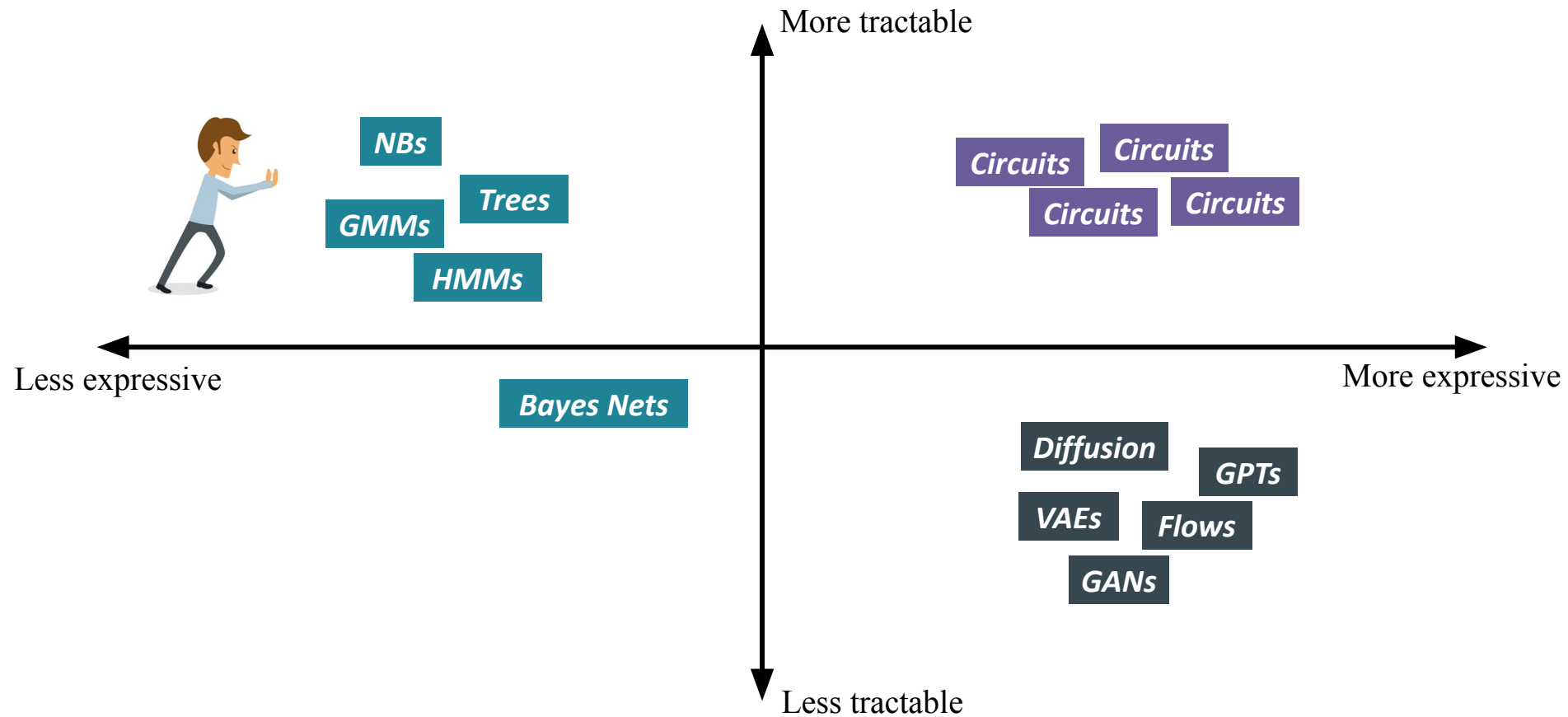
- Also works **in stochastic environments**



Methods	Taxi	FrozenLake		
		$\epsilon = 0.3$	$\epsilon = 0.5$	$\epsilon = 0.7$
m-Trifle	-57	0.61	0.59	0.37
s-Trifle	-99	0.62	0.60	0.34
TT [20]	-182	0.63	0.25	0.12
DT [6]	-388	0.51	0.32	0.10
DoC [47]	-146	0.58	0.61	0.23

- Condition on **safe actions**

Dataset	Environment	Trifle	TT
Med-Expert	Halfcheetah	81.9 ±4.8	77.8±5.4
Med-Expert	Hopper	109.6 ±2.4	100.0±4.2
Med-Expert	Walker2d	105.1 ±2.3	103.6±4.9



Generative Models

polynomials model **joint distributions**

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

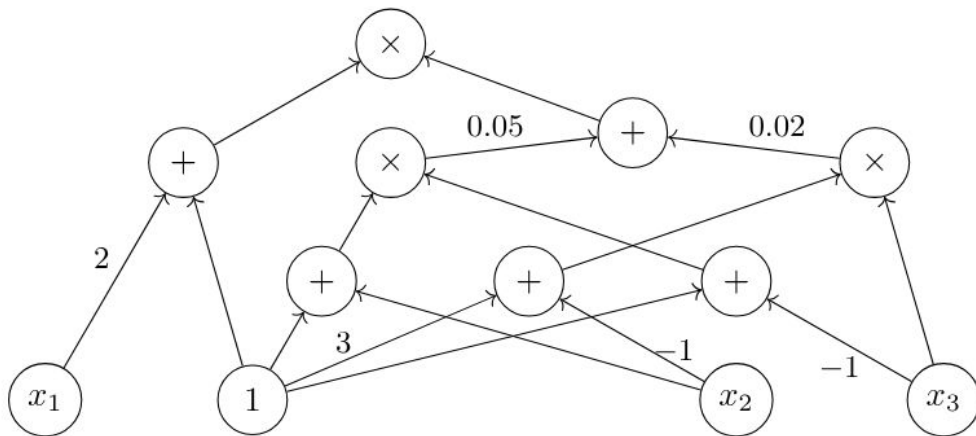
X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12

Deep Generative Models

circuit polynomials model **joint distributions** compactly
(and can have billions of trainable parameters)

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12



Probabilistic Reasoning Task

Marginal inference:

X_1	X_2	Pr
0	0	.1
0	1	.2
1	0	.3
1	1	.4

$$\begin{aligned}\Pr[X_1 = 1] &= \Pr[X_1 = 1, X_2 = 0] + \Pr[X_1 = 1, X_2 = 1] \\ &= 0.3 + 0.4 \\ &= 0.7\end{aligned}$$

Application: Ctrl-G



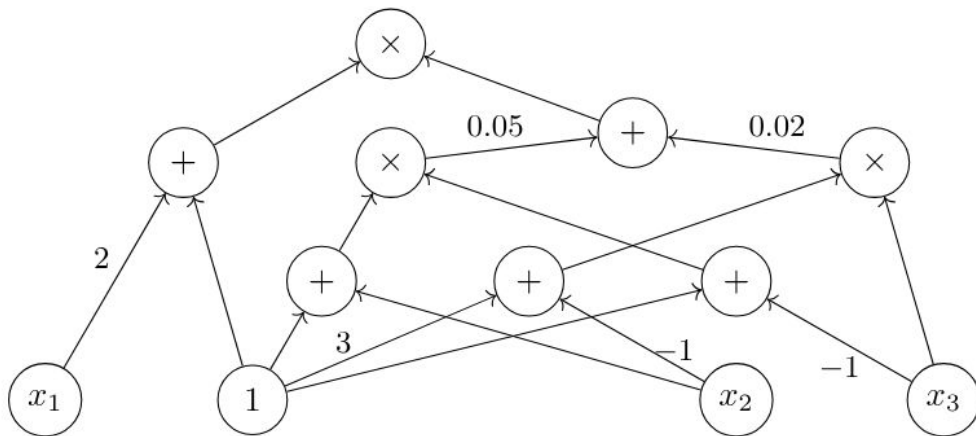
$p_{circuit}(\alpha \mid \text{next-token, prefix})$ is summing over all future text

Tractable Deep Generative Models

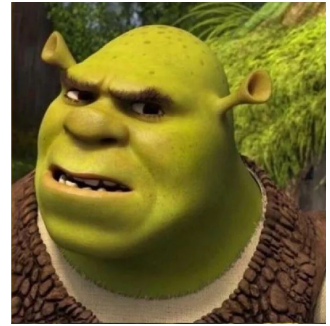
Multilinear circuit polynomials model **joint distributions** compactly *and* allow **efficient** probabilistic reasoning (marginalization)

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12



You Tricked Us



You promised us reasoning algorithms...

... and all we got was another lousy feedforward neural network!

***Theorem.** If there exists a polynomial time (real RAM) algorithm that computes (virtual evidence) marginal probabilities for a class of distributions, then there exist **poly-size circuits** for their **multilinear polynomials**.*



Questions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of LLMs?
2. Can we put “probabilistic model checking” into the LLM?
3. Can we put “LLM probability” into the symbolic reasoning?
4. Do deductive reasoning algorithms still have a purpose in the age of LLMs? (reprise)



Thanks

This was the work of many wonderful students/postdocs/collaborators!



References: <http://starai.cs.ucla.edu>