



Computer
Science



Agentic AI is Neurosymbolic AI

Guy Van den Broeck

BNAIC/BeNeLearn 2026, Maastricht - Oct 22 2026

My very first paper: BNAIC 2009, Eindhoven

Monte-Carlo Tree Search in Poker using Expected Reward Distributions*

Guy Van den Broeck^a Kurt Driessens^a Jan Ramon^a

^a *Dept. of Computer Science, K.U.Leuven*

1 Introduction

Poker playing computer bots can be divided into two categories. There are the *game-theoretic bots*, that play according to a strategy that gives rise to a Nash equilibrium. These bots are impossible to beat, but are also not able to exploit non-optimalities in their opponents. The other type of bot is the *exploiting bot* that employs game tree search and opponent modeling techniques to discover and exploit weaknesses of opponents. In this paper, we focus on the second type of bot.

Research in computer Poker has been mainly dealing with the limit variant of Texas Hold'em. In limit Poker, bet sizes are fixed as well as the total number of times a player can raise the size of the pot. In fact, all exploiting bots developed so far, deal with the heads-up limit version of the game. We investigate the use of Monte-Carlo Tree Search (MCTS) in multiplayer no-limit Texas Hold'em Poker. To extend to use of MCTS to incomplete information games, we propose new backpropagation and selection strategies that take into account the consequences of having non-deterministic nodes in the search tree.

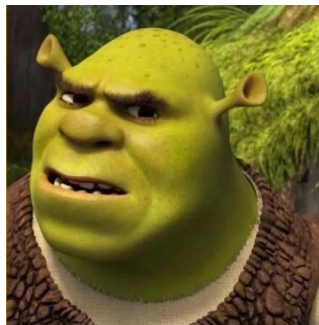
2 The Poker Game Tree

A game tree is used to reason about future states of the game. The expected reward in each state is computed



Kurt Driessens

What was AI like then?

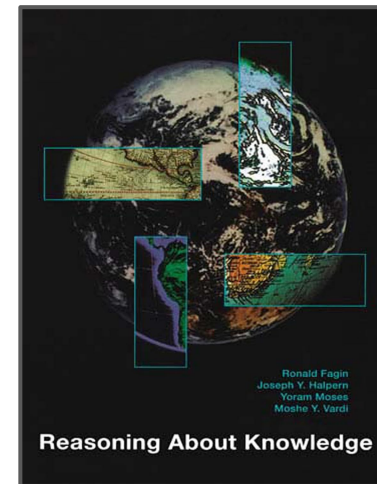


Reasoning with Symbolic AI

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

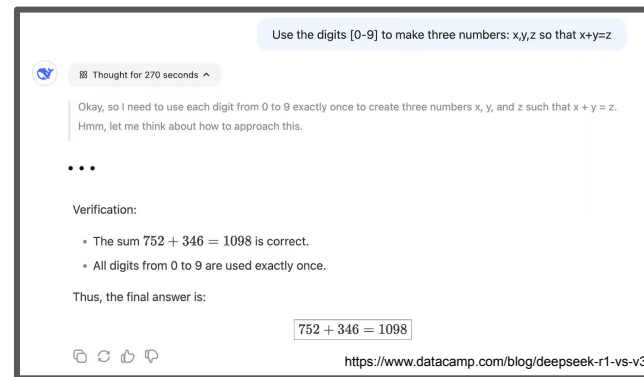


30 years



Reasoning with Transformers

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



Questions for this talk:



1. Do symbolic reasoning algorithms still have a purpose in the age of agentic AI?
2. Can we put “LLM probability” into the symbolic reasoning?
3. Can we put “symbolic reasoning” into the LLM?

What did I do this Spring?

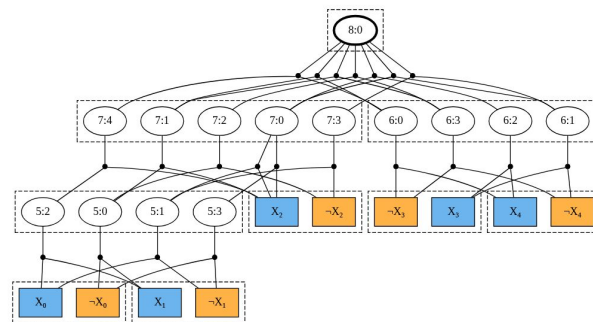
Tuesday 3/10: Submitted 30 pages of pure theory to SAT conference, defining a [data structure for logical reasoning](#) with nicer properties – a model counter

Wednesday 3/11: Gave the latex source to Claude Code, got a working implementation by EOD. Passes all tests, gives correct model counts,

Friday 3/13: Became faster than SoTA decision diagram compilers.

\$\$\$\$\$\$\$\$\$\$\$\$

Today: [Fastest model counter in the world](#)



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 458,609 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 neurosymbolic AI!

What did I do this Summer?

L	CD	FO	SFO	$\wedge C$	$\wedge BC$	$\vee C$	$\vee BC$	$\neg C$
NNF	✓	○	✓	✓	✓	✓	✓	✓
DNNF	✓	✓	✓	○	○	✓	✓	○
d-NNF	✓	○	✓	✓	✓	✓	✓	✓
s-NNF	✓	○	✓	✓	✓	✓	✓	✓
f-NNF	✓	○	✓	•	•	•	•	✓
d-DNNF	✓	○	○	○	○	○	○	?
sd-DNNF	✓	○	○	○	○	○	○	?

Darwiche and Marquis, A Knowledge Compilation Map, JAIR 2002

2001: Open problem: can you efficiently negate a tractable d-DNNF circuit?

2010-2026: I think about it on and off, and get nowhere

August 2026: AI helps prove: Negation cannot be done efficiently! An exponential gap!

Why? It happened because of neurosymbolic AI!

- Trained against a **symbolic** verifier: Lean in the loss
- **Symbolic** guardrails at inference: resample until Lean certificate is accepted

13,578 LOC

```
def NotClosedUnderNegation : Prop := ∀ d M : ℕ, ∃ C : Circuit ℕ, C.IsDeterministicDNNF ∧  
  ∀ D : Circuit ℕ, D.IsDNNF → D.Computes (fun v ↦ !C.eval v C.output) → M * C.size ^ d < D.size
```

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.
- A \$200-per-month Claude Code subscription uses \$5,000 in compute, subsidized by Anthropic [forbes]
- Can the people at OpenAI who posted these proofs even read Lean?

Scale and brute force compensate for this lack of understanding.

Why did neurosymbolic AI “win”?

The philosophy was always right.

Because of the overwhelming gravitational pull of

Compositionality
Abstraction
Soundness

*Neurosymbolic data, loss functions,
and inference are vastly more
powerful because of these
properties!*

Questions for this talk:



1. Do symbolic reasoning algorithms still have a purpose in the age of agentic AI?
2. **Can we put “LLM probability” into the symbolic reasoning?**
3. Can we put “symbolic reasoning” into the LLM?

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

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 =  $x_{12} x_{13} x_{14}$   
    smallest_group = (total_students +  $x_{25} x_{26}$ ) //  $x_{27}$   
    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 =  $X_{12} X_{13} X_{14}$   
    smallest_group = (total_students  $Z_{24} X_{25} X_{26}$ )  $Z_{27} X_{28}$   
    return smallest_group
```

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

**This is a
probabilistic
program!!**

This probabilistic program approximates LLM distribution around a sample

```
def compute_answer():
```

```
    total_students =  $x_{12}$   $x_{13}$   $x_{14}$ 
```

```
    smallest_group = (total_students
```

```
         $z_{24}$ 
```

```
         $x_{25}$   $x_{26}$ 
```

```
    )
```

```
         $z_{27}$ 
```

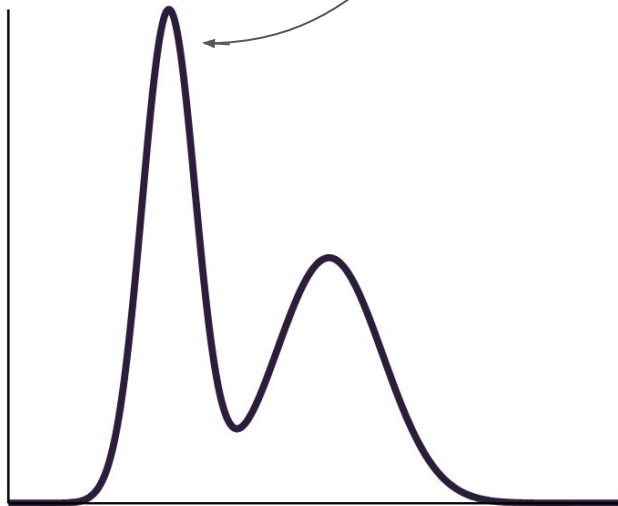
```
         $x_{28}$ 
```

```
    )
```

```
    return smallest_group
```

The LLM distribution
conditioned on the
prompt

$p(y|x)$



This probabilistic program approximates LLM distribution around a sample

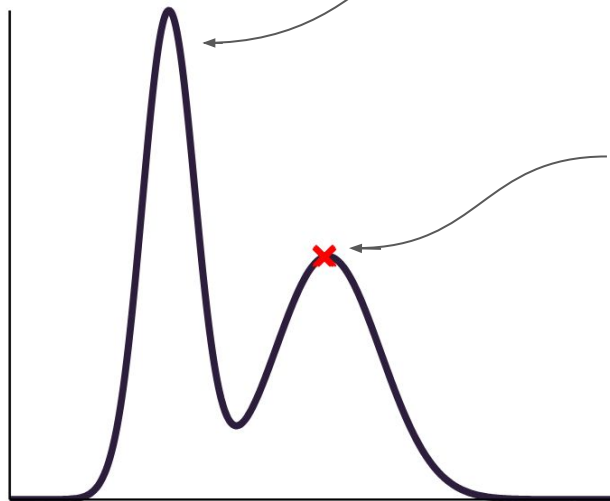
```
def compute_answer():
```

```
    total_students =  $x_{12}$   $x_{13}$   $x_{14}$ 
```

```
    smallest_group = (total_students  $z_{24}$   $x_{25}$   $x_{26}$  )  $z_{27}$   $x_{28}$ 
```

```
    return smallest_group
```

$$p(\mathbf{y}|\mathbf{x})$$



The LLM distribution
conditioned on the
prompt

The sample

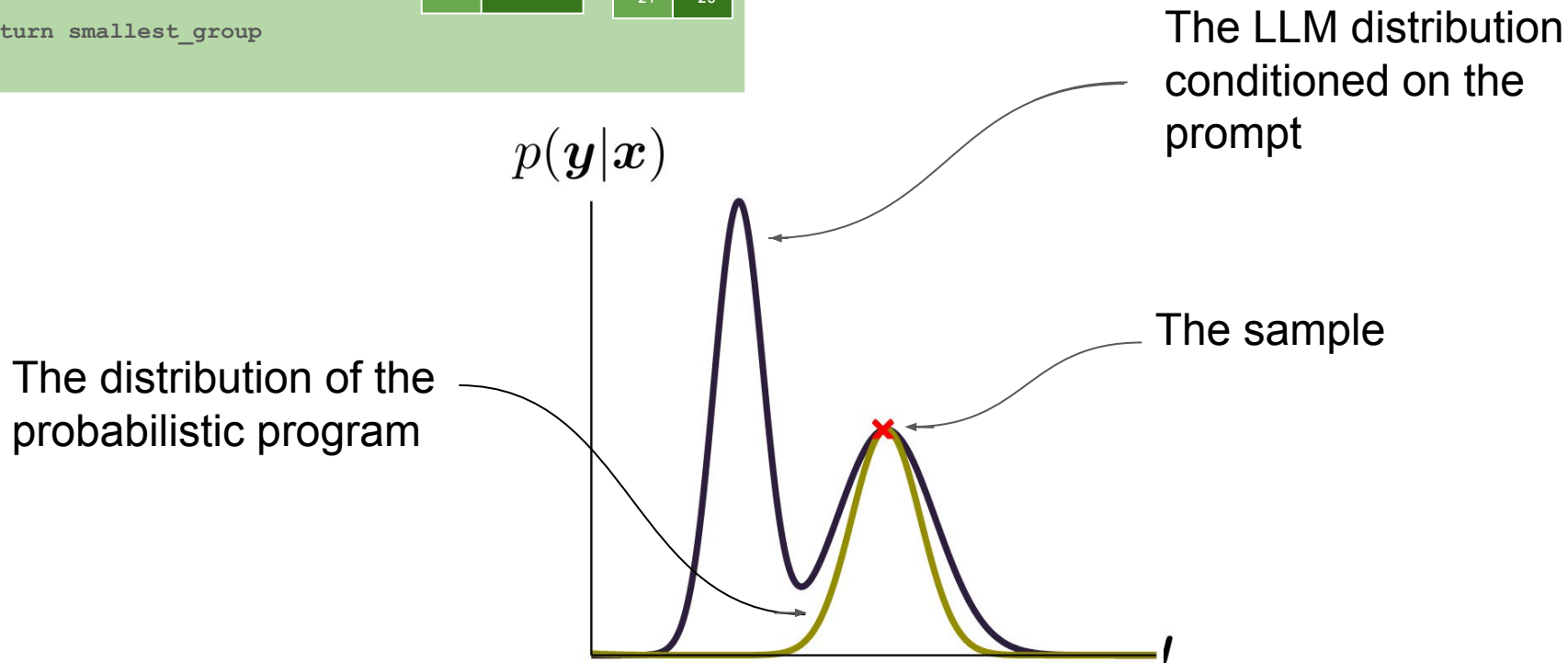
This probabilistic program approximates LLM distribution around a sample

```
def compute_answer():
```

```
    total_students =  $x_{12}$   $x_{13}$   $x_{14}$ 
```

```
    smallest_group = (total_students  $z_{24}$   $x_{25}$   $x_{26}$ )  $z_{27}$   $x_{28}$ 
```

```
    return smallest_group
```



This probabilistic program approximates LLM distribution around a sample

```
def compute_answer():
```

```
    total_students =  $x_{12}$   $x_{13}$   $x_{14}$ 
```

```
    smallest_group = (total_students  $z_{24}$   $x_{25}$   $x_{26}$ )  $z_{27}$   $x_{28}$ 
```

```
    return smallest_group
```

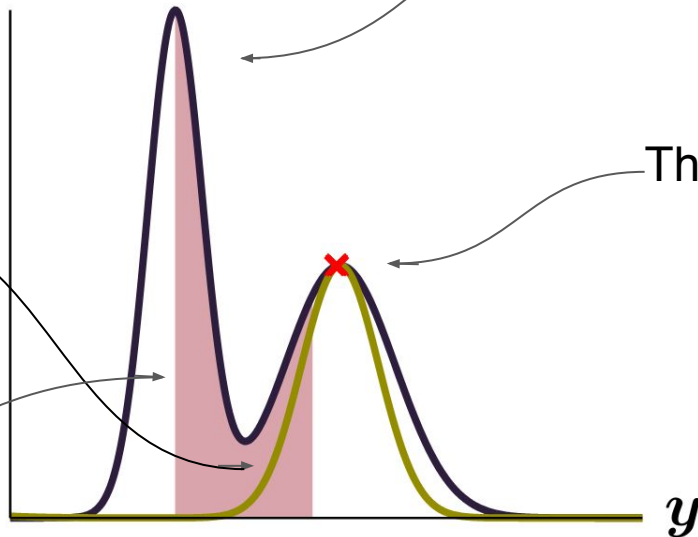
The LLM distribution
conditioned on the
prompt

The sample

The distribution of the
probabilistic program

The space of the correct
programs

$$p(y|x)$$



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 =  $x_{12} \ x_{13} \ x_{14}$ 
```

```
    smallest_group = (total_students  $z_{24} \ x_{25} \ x_{26}$ )  $z_{27} \ x_{28}$ 
```

```
    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 =  $x_{12} \ x_{13} \ x_{14}$ 
```

```
    smallest_group = (total_students
```

```
         $z_{24}$   $x_{25} \ x_{26}$ 
```

```
    )  $z_{27} \ x_{28}$ 
```

```
    return smallest_group
```

```
def compute_answer():
```

```
...
```

```
def compute_answer():
```

```
    total_students = 200
```

```
    smallest_group = (total_students - 20) // 3
```

```
    return smallest_group
```

```
...
```

```
def compute_answer():
```

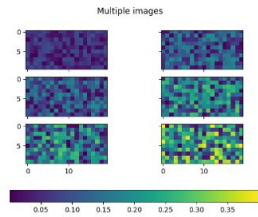
```
...
```



Sample from probabilistic program

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

Plot2Code

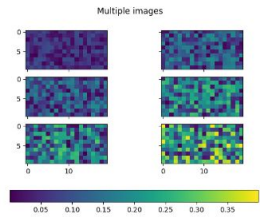


ground-truth

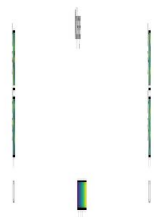


LLM sample

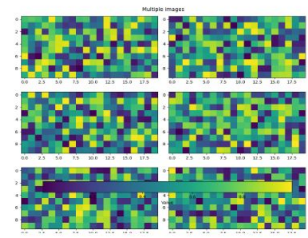
Plot2Code



ground-truth

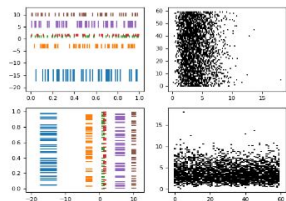


LLM sample

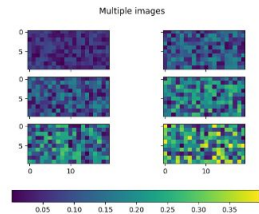


PPoT sample

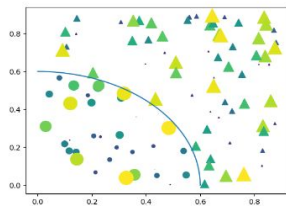
Plot2Code



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

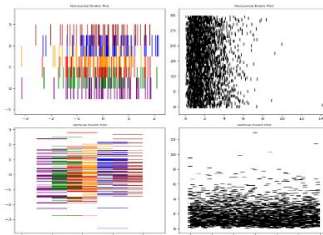
LLM sample



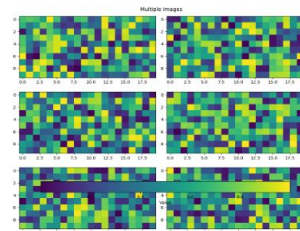
LLM sample

ERROR

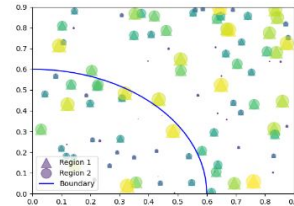
LLM sample



PPoT sample



PPoT sample



PPoT sample

Probabilistic Programs-of-thought

Best-of-n performance:

1.5% - 7% boost

for the same amount of GPU compute

PPoT samples

LLM samples

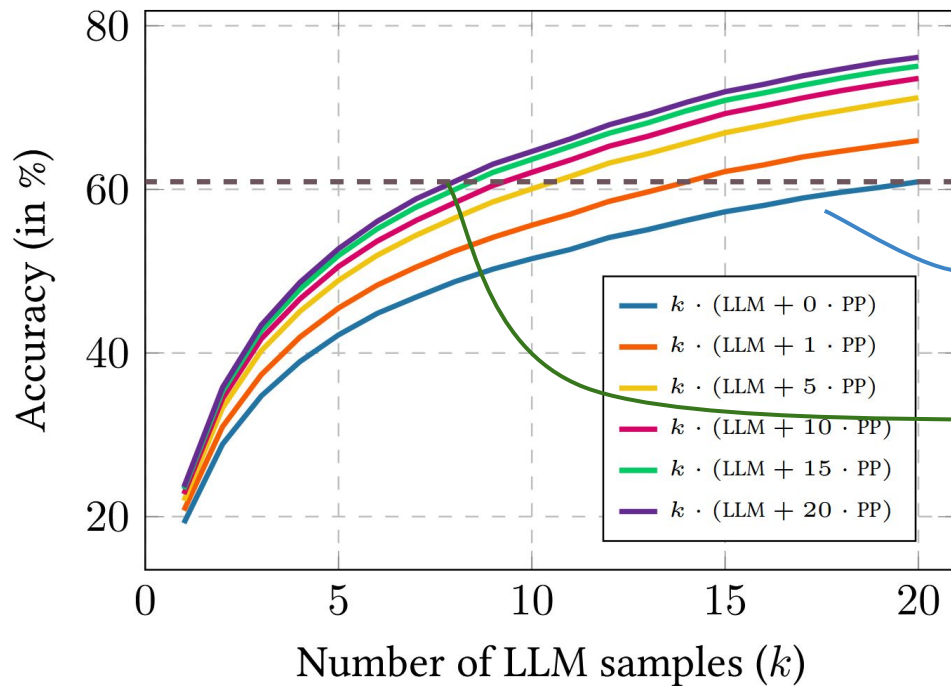
math reasoning

program inversion

plot generation

	GSM8k			CRUXEval			Plot2Code	
	Qwen2.5 Coder			Qwen2.5 VL			Qwen2.5 VL	
$k \cdot (\text{LLM} + m \cdot \text{PP})$	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



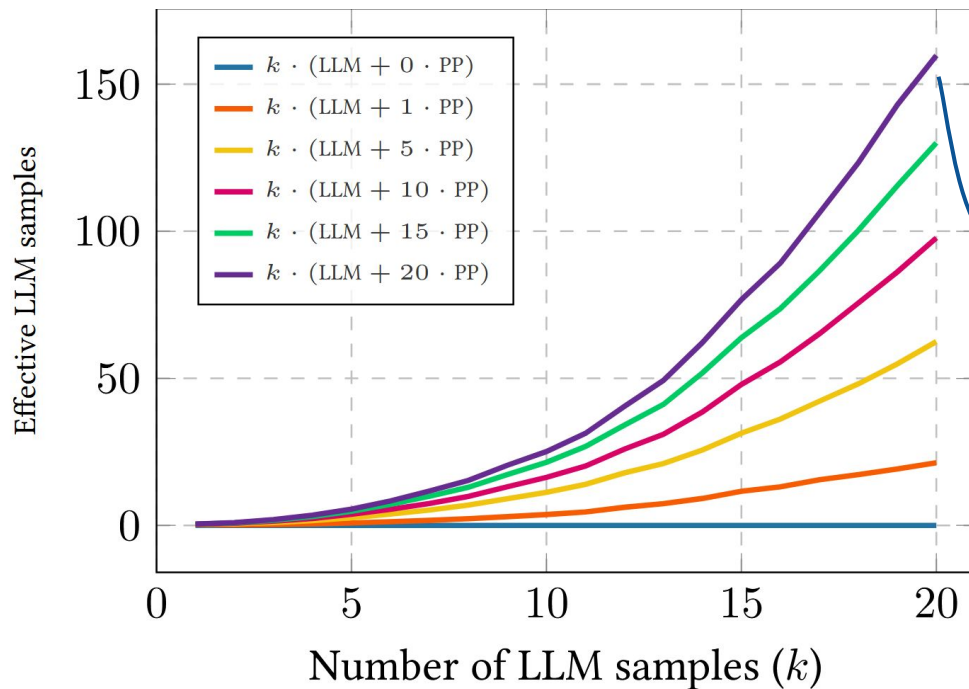
How much can we gain from PPoT?

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



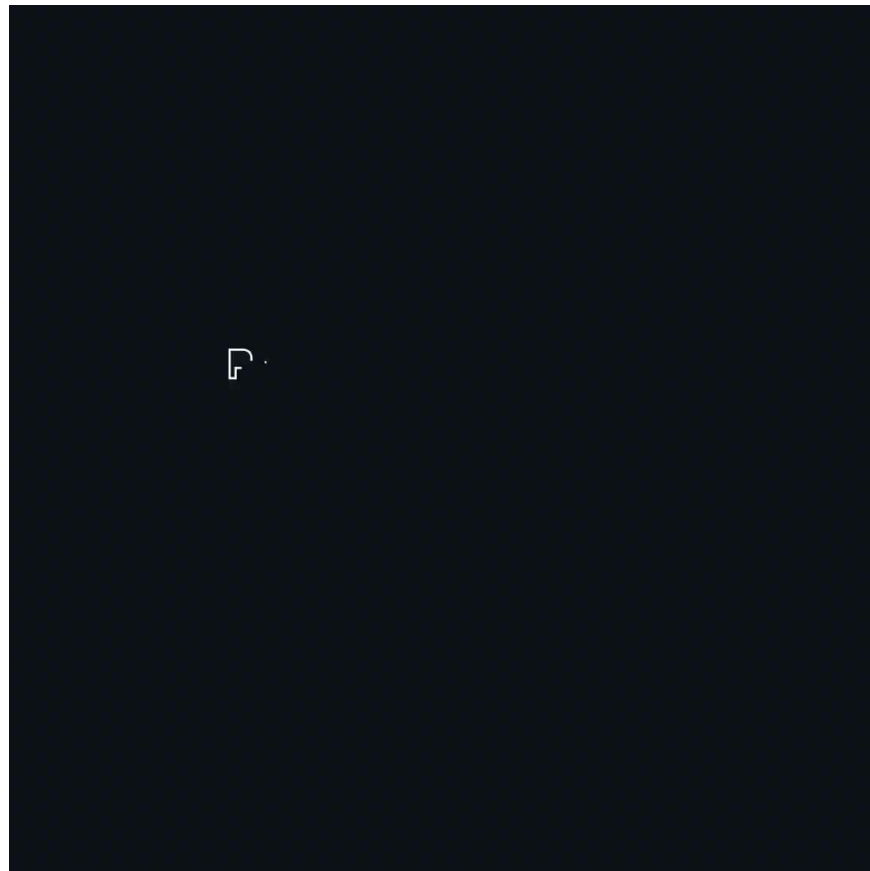
How much can we gain from PPOt?

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

Qwen 2.5 Coder on GSM8k

PPoT Conclusions

1. LLM comes with uncertainty
2. Pushing it forward through the symbolic program lets us reason about it
3. We can do better than prompting + LLM sampling
4. Programs of thought were always meant to be **probabilistic programs of thought**



What if the reasoning is in English, not code?

Can we still put LLM probability into the reasoning?

(A) Chain-of-thought

Prompt: Could the members of The Police perform lawful arrests? Think step-by-step.

LLM CoT Reasoning Trace:

The members of The Police were musicians. s1

step deps. none lm cond. likelihood: 0.86 verbal confidence: 0.7

Only law enforcement officers perform lawful arrests of other people. s2

step deps. none lm cond. likelihood: 0.96 verbal confidence: 0.90

Based on this, the answer is no. **(answer: no)** s3

step deps. s1, s2 lm cond. likelihood: 0.53 verbal confidence: 0.8

(B) Probabilistic Program

```
1 @CotProgParser()
2 def pp():
3   P1: bool = ~ Flip("step1", 0.86, [0.7])
4
5   P2: bool = ~ Flip("step2", 0.96, [0.9])
6   ## observe(P2) ## Optional condition, set P2=True
7
8   if P1 and P2:
9     P3: bool = ~ Flip("no", 0.53, [0.8])
10    yield P3 ## returns "no" if P3 true, exception otherwise, syntactic sugar
11  else:
12    raise ValueError("Cannot establish P1 or P2")
13
14 ## Program inference
15 pp.inference(query={"no"}, compute_grad=True) ## 0.4376
16 ## param_grads=[('P2', 0.455), ('P3', 0.825), ('P1', 0.5087)]
17 pp.inference(query={"ValueError"}) ## 0.5624
18 pp.inference(query={"no"}, use_conf=True) ## 0.504
```

(C) Probabilistic Queries

$\theta_{\text{likelihood}} = (0.86, 0.96, 0.53)$
 $\theta' = (0.70, 0.90, 0.80)$

$E = \{P2\}$

$\mathbb{P}_{\text{pp}}(\text{return "no"; } \theta)$

$\frac{\partial \mathbb{P}_{\text{pp}}(\text{return "no"; } \theta)}{\partial \theta}$

$\mathbb{P}_{\text{pp}}(\text{raise ValueError; } \theta)$

$\mathbb{P}_{\text{pp}}(\text{return "no"; } \theta')$

- What is the probability that the answer is “no”?
- If the answer is “yes”, which reasoning step is most likely wrong?

Questions for this talk:

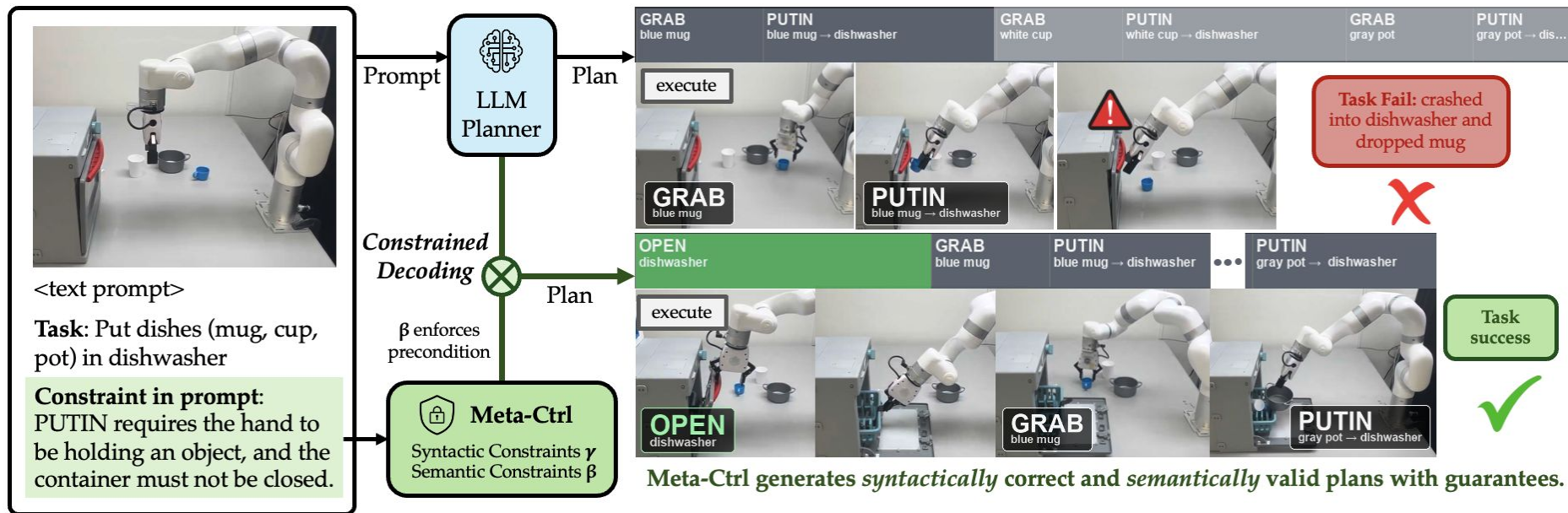


1. Do symbolic reasoning algorithms still have a purpose in the age of agentic AI?
2. Can we put “LLM probability” into the symbolic reasoning?
3. **Can we put “symbolic reasoning” into the LLM?**

LLM plans break the rules stated in the prompt



Probabilistic constrained decoding for robot planning



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...

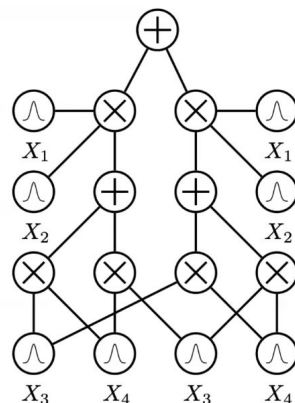
A 'tractable digital twin'

Reasoning about all Future Tokens: Constraints

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

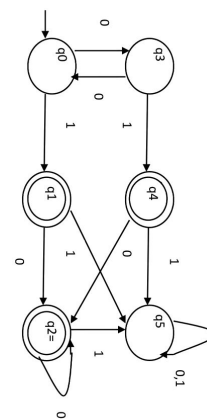
Abusing Bayes rule,

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



weights/circuit p

X



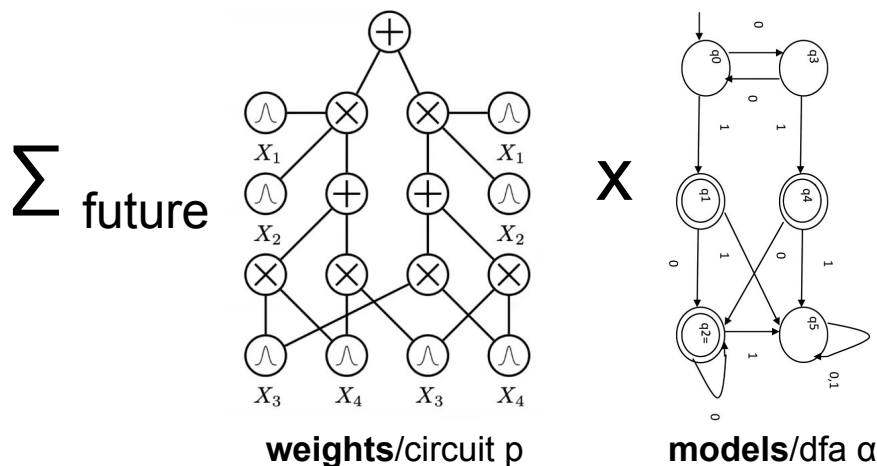
models/dfa α

Reasoning about all Future Tokens: Constraints

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

Abusing Bayes rule,

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

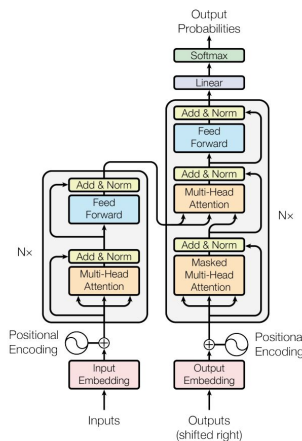


Reasoning about all Future Tokens: Constraints

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

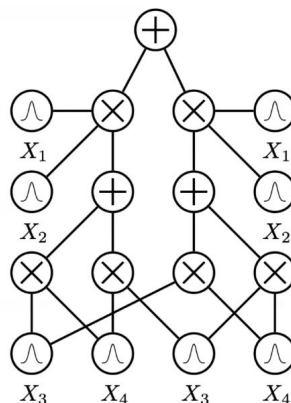
Abusing Bayes rule,

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

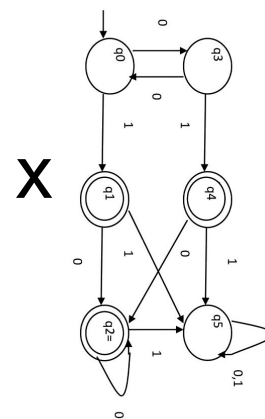


transformer p

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



weights/circuit p



models/dfa α

Probabilistic Circuit Language Model

How did we train a probabilistic circuit?

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



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



	<i>K</i>	<i>L</i>	<i>K&L</i>
<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**

JSON Tool Calling

calls must follow the tool's JSON schema

User: What's the weather like in NYC today?
Tools: described by JSON schemas

A correct call

✓ `{"name": "get weather", "location": "NYC"}`

Two kinds of mistakes

```
X {"name": "get weather", "location": {}} ...
```

Never closes before the token budget runs out

X {"name": "get_time", "location": "NYC"}

Valid JSON, but the wrong function or arguments

xLAM benchmark, Llama-3.1-8B, 16 samples

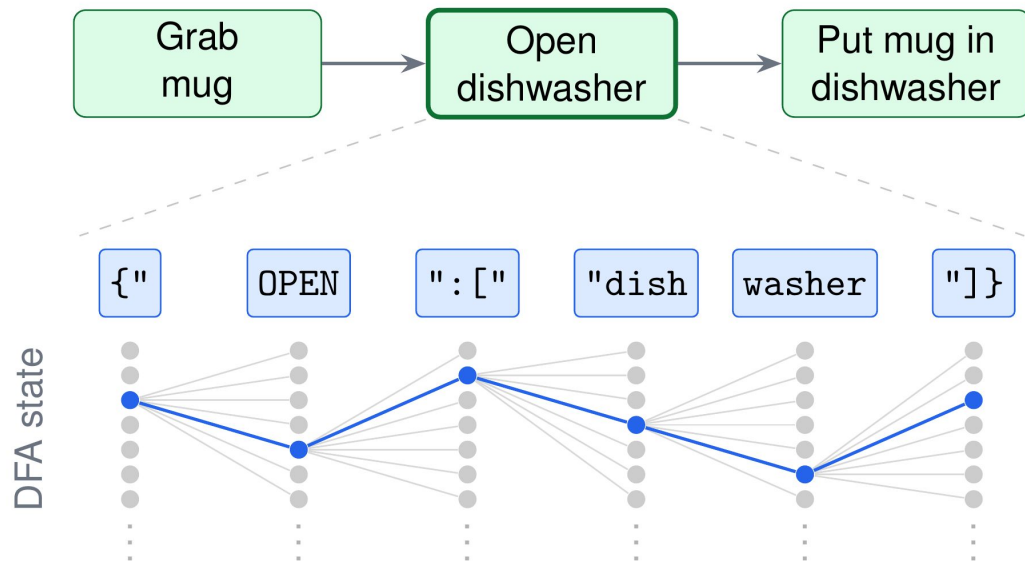
	Valid JSON	Correct calls
Local masking	95.4%	69.7%
Ctrl-G proposal	100%	71.6%
Exact (rejection)	100%	$\approx 72\%$

Local masking is what JSON modes do today
(Outlines, XGrammar, OpenAI API)

Sequential Monte Carlo removes the digital twin's bias, for a few more samples

Can we write the robot's rules as a DFA?

The rules constrain sequences of actions



Rules translated into tokens:

350 million

DFA states

107 TB

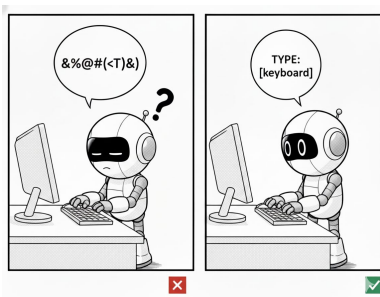
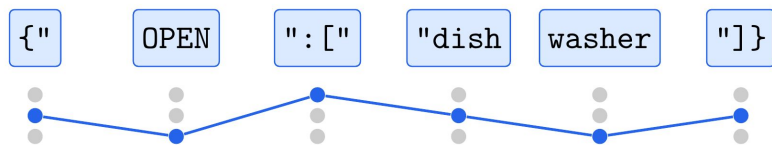
of dynamic programming
tables

But the LLM writes each action as about 6 tokens

Instead, reason at two levels

Syntax, over tokens

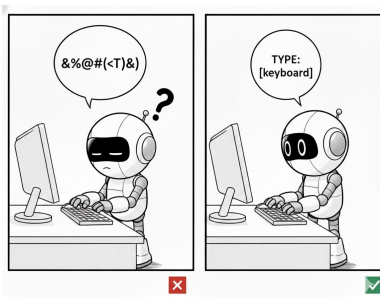
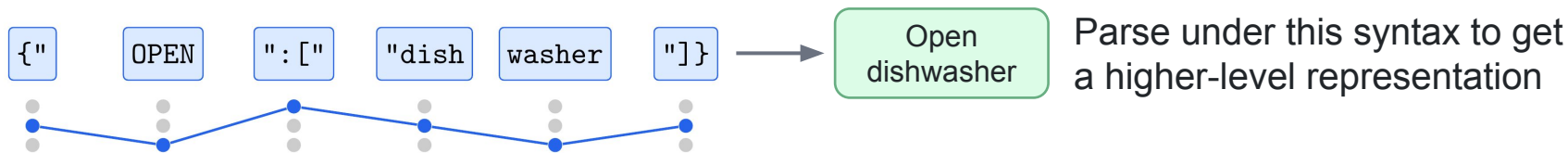
Is it valid JSON?



Instead, reason at two levels

Syntax, over tokens

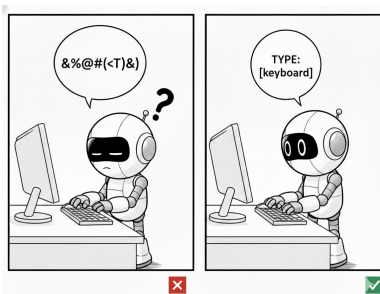
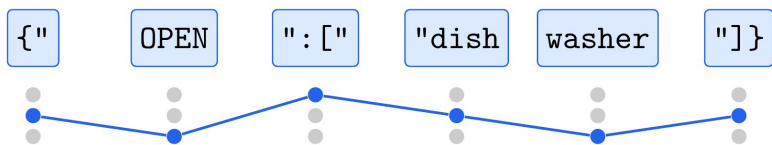
Is it valid JSON?



Instead, reason at two levels

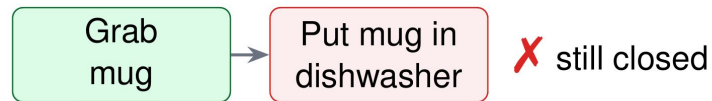
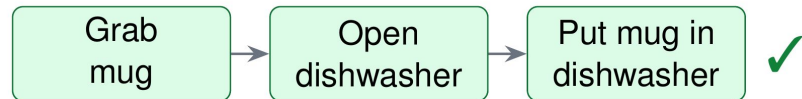
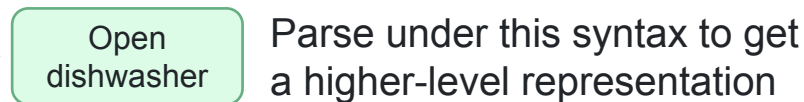
Syntax, over tokens

Is it valid JSON?



Semantics, over actions

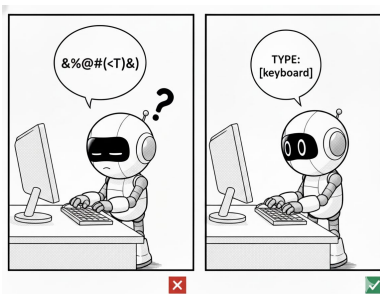
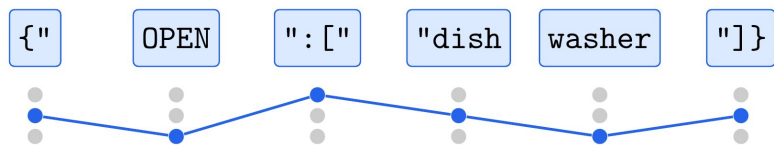
Which actions must come first?



Instead, reason at two levels

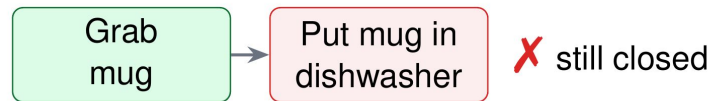
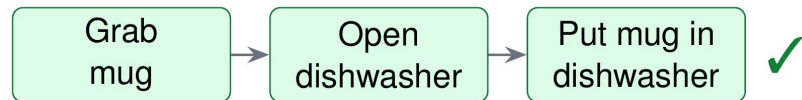
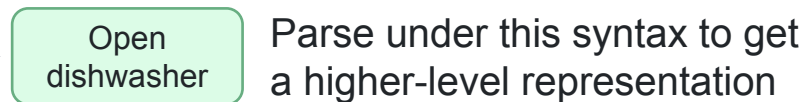
Syntax, over tokens

Is it valid JSON?



Semantics, over actions

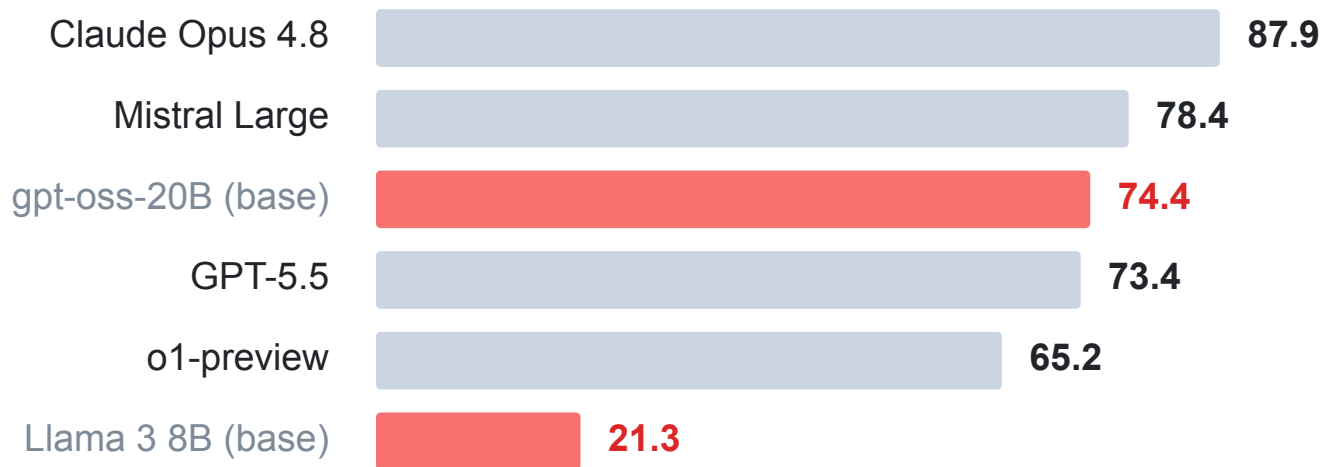
Which actions must come first?



4.6K + 76K = 81K states, 1.4 GB + 1.6 GB of dynamic programming tables

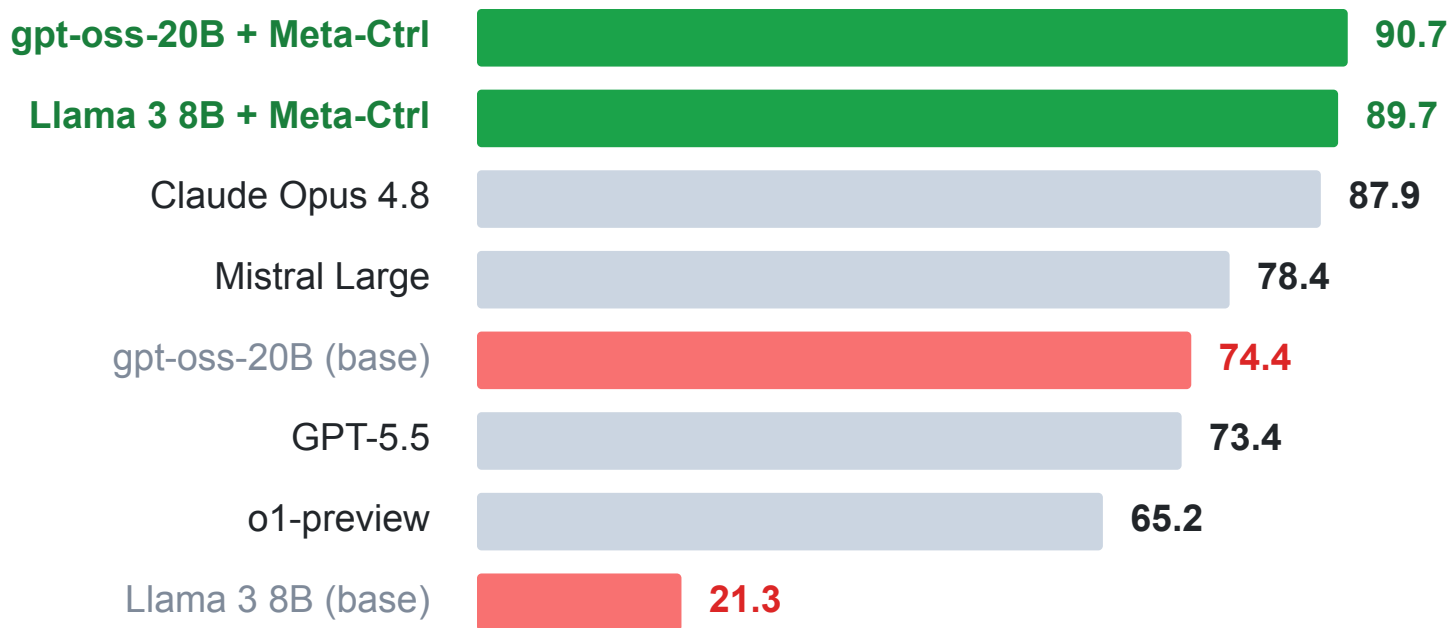
Small open models **don't do so well**

VirtualHome action sequencing: task success rate (%)

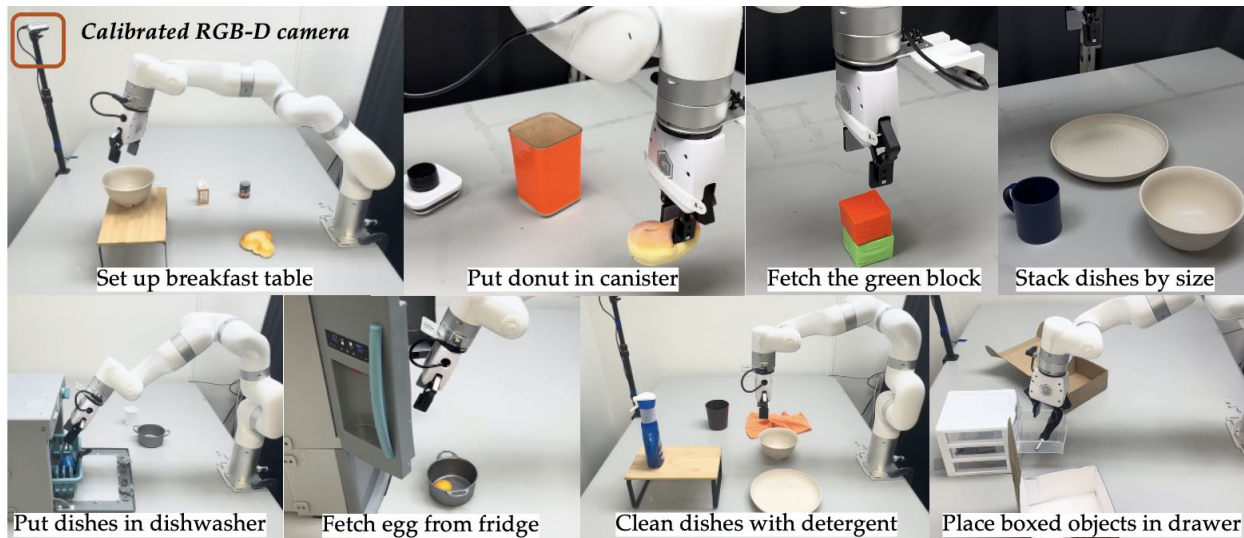


Small open models **top the leaderboard**

VirtualHome action sequencing: task success rate (%)



On a real robot, Meta-Ctrl makes every plan valid



Meta-Ctrl

Llama 3 8B

xArm 7, 20 runs

valid plan

executed

valid plan

executed

Donut in canister

20/20

16/20

2/20

1/20

Green block

20/20

18/20

1/20

1/20

Stack dishes

20/20

17/20

0/20

0/20

The remaining failures are
in perception and grasping

Ctrl-G & Meta-Ctrl Conclusions

1. Constraint α is guaranteed to be satisfied:

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

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

Ctrl-G & Meta-Ctrl Conclusions

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. Bayesian = goal-oriented
($\leftrightarrow$ masking)

Ctrl-G & Meta-Ctrl Conclusions

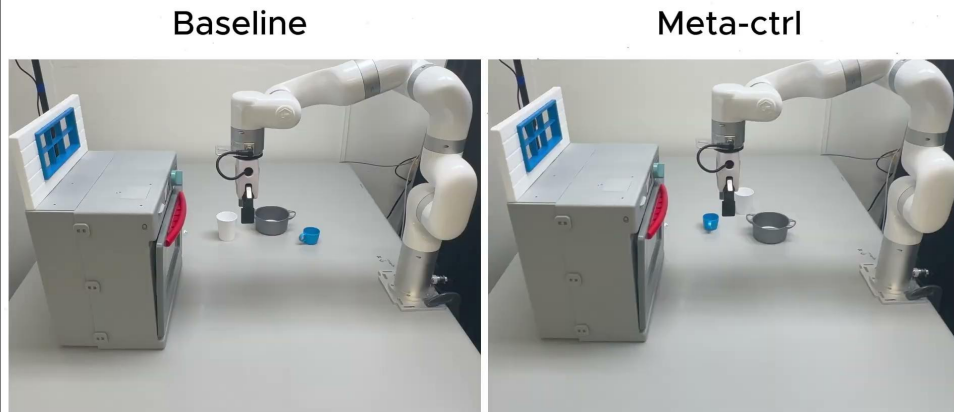
1. Constraint α is guaranteed to be satisfied:

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. Bayesian = goal-oriented
($\leftrightarrow$ masking)

3. Deployed on a real robot that now knows to *open the microwave before putting something in the microwave*



What if we lack a formal language for what we want?

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

Prefix: It's a pain ...

Constraint α : non-toxic

What if we lack a formal language for what we want?

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

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

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

in	0.1
to	0.8



Why not use RL to solve all your problems?

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
DPO	39.52

Lower **entropy** means
lower diversity

**DPO “thinks” that 99.9996% of all internet text is toxic.
Ad-Hoc Probabilistic Reasoning - broken**

Attribute Probability



0 (toxic)

1 (nontoxic)

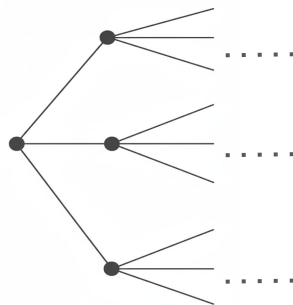
It's a pain

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



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

...

...

$EAP = 0.1$

future text

deal with

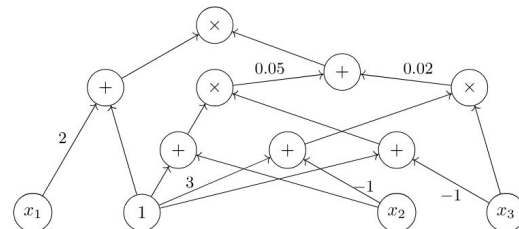
handle

...

...

$EAP = 0.8$

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 (↓)		Approach Type
	avg. max.	prob.	
GPT-2 Large Results			
GPT2	0.385	0.254	Baseline
DAPT ⁽¹⁾	0.428	0.360	Finetuning
GeDi ⁽²⁾	0.363	0.217	Decoding (Trained Guide)
FUDGE ⁽³⁾	0.302	0.371	Decoding (Trained Guide)
DExperts ⁽⁴⁾	0.314	0.128	Decoding (Trained Guide)
PPLM ⁽⁵⁾	0.520	0.518	Decoding (Logit Control)
MuCoLa ⁽⁶⁾	0.308	0.088	Decoding (Sampling)
PPO ⁽⁷⁾	0.218	0.044	RL
Quark ⁽⁸⁾	0.196	0.035	RL
DPO ⁽⁹⁾	0.180	0.026	RL
TRACE	0.163	0.016	Decoding (HMM Reasoning)
Gemma-2B Results			
Gemma-2B	0.359	0.23	Baseline
DPO ⁽⁹⁾	0.222	0.06	RL
TRACE	0.189	0.02	Decoding (HMM Reasoning)

State-of-the-art LLM Detoxification

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

State-of-the-art LLM

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

Method	Entropy (↑)
GPT2-large	52.06
DPO	39.52
TRACE	52.54

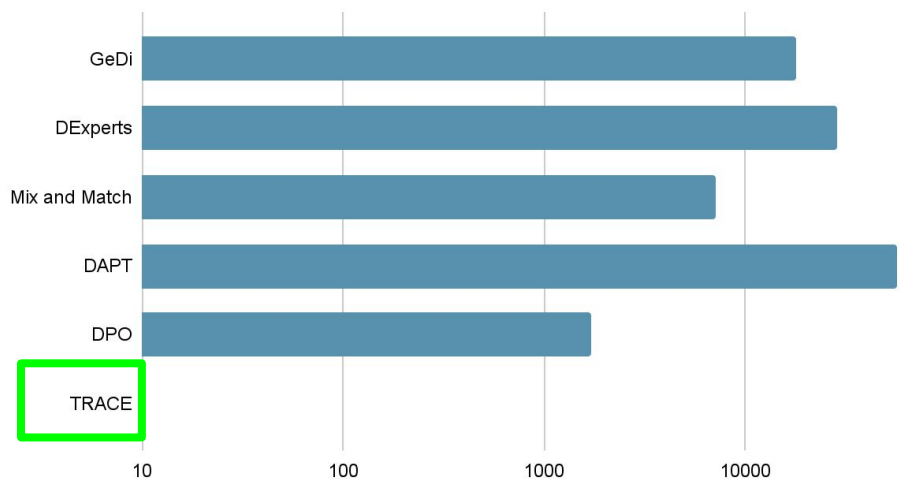
State-of-the-art LLM Detoxification

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)

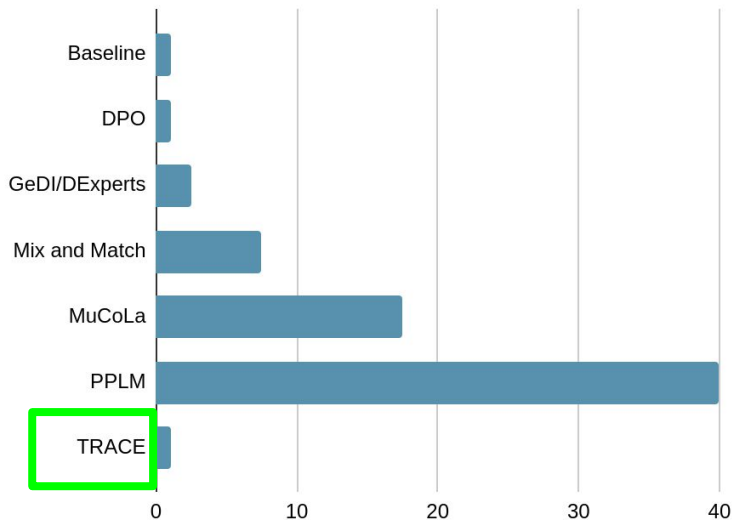
TRACE is Blazingly Fast

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

Training Time per Attribute (seconds)



Inference Time



Can a proof's style tell us if the answer is correct?

How often each word appears in the LLM's correct and incorrect MATH solutions

Qwen2.5-7B	Correct	Incorrect	Llama 3.1 8B	Correct	Incorrect
boxed	100%	68%	boxed	100%	56%
confirm, verify, confirms	14%	9.4%	Therefore	47%	27%
What, Please, ?	0.7%	2.1%	However	6.5%	19%
			question	1.6%	4.7%

- Correct solutions check their own work
- Incorrect ones drift off task (What, Please)

Can we control the generation toward correct answers?

Steering toward correct answers, no fine-tuning

Share of sampled solutions that are correct (%)

	MATH500	GPQA	HumanEval
Qwen2.5-7B	40.9	25.0	31.0
+ TRACE	44.6	28.0	32.1
Llama 3.1 8B	35.1	28.9	41.7
+ TRACE	37.9	31.4	42.9

- The LLM's weights never change
- Overhead is minimal: TRACE costs less than one LLM evaluation per token

Same ideas, completely different applications

Text

LLM

It's a pain

$\alpha = \text{non-toxic}$

[Yidou-Weng et al., ICML 2025](#)

Robots

LLM

GRAB mug OPEN

$\alpha = \text{all preconditions are satisfied}$

[Yidou-Weng et al., CoRL 2026](#)

Reasoning

LLM

Since 2x = 6 ,

$\alpha = \text{the final answer is correct}$

[Zhang et al., 2026](#)

Same ideas, completely different applications

Text

LLM

It's a pain

$\alpha = \text{non-toxic}$

[Yidou-Weng et al., ICML 2025](#)

Robots

LLM

GRAB mug OPEN

$\alpha = \text{all preconditions are satisfied}$

[Yidou-Weng et al., CoRL 2026](#)

Reasoning

LLM

Since 2x = 6 ,

$\alpha = \text{the final answer is correct}$

[Zhang et al., 2026](#)

Offline RL

trajectory model

s_1 a_1 r_1 s_2

$\alpha = \text{high reward and safe actions}$

[Liu et al., NeurIPS 2024](#)

Same ideas, completely different applications

Text

LLM

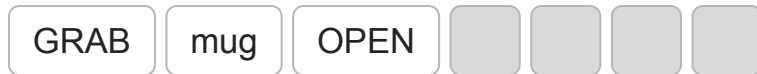


$\alpha = \text{non-toxic}$

[Yidou-Weng et al., ICML 2025](#)

Robots

LLM



$\alpha = \text{all preconditions are satisfied}$

[Yidou-Weng et al., CoRL 2026](#)

Reasoning

LLM



$\alpha = \text{the final answer is correct}$

[Zhang et al., 2026](#)

Offline RL

trajectory model



$\alpha = \text{high reward and safe actions}$

[Liu et al., NeurIPS 2024](#)

Images

diffusion model



$\alpha = \text{contains the given patch}$

[Liu et al., ICLR 2024](#)

Same ideas, completely different applications

Text

LLM

It's a pain

$\alpha = \text{non-toxic}$

[Yidou-Weng et al., ICML 2025](#)

Robots

LLM

GRAB mug OPEN

$\alpha = \text{all preconditions are satisfied}$

[Yidou-Weng et al., CoRL 2026](#)

Reasoning

LLM

Since 2x = 6 ,

$\alpha = \text{the final answer is correct}$

[Zhang et al., 2026](#)

Offline RL

trajectory model

s_1 a_1 r_1 s_2

$\alpha = \text{high reward and safe actions}$

[Liu et al., NeurIPS 2024](#)

Images

diffusion model

$\alpha = \text{contains the given patch}$

[Liu et al., ICLR 2024](#)

DNA

diffusion model

A C G T A C

$\alpha = \text{strong enhancer activity}$

[Xu et al., 2026](#)

Same ideas, completely different applications

Text LLM	It's a pain	$\alpha = \textit{non-toxic}$ Yidou-Weng et al., ICML 2025
Robots LLM	GRAB mug OPEN	$\alpha = \textit{all preconditions are satisfied}$ Yidou-Weng et al., CoRL 2026
Reasoning LLM	Since 2x = 6 ,	$\alpha = \textit{the final answer is correct}$ Zhang et al., 2026
Offline RL trajectory model	s₁ a₁ r₁ s₂	$\alpha = \textit{high reward and safe actions}$ Liu et al., NeurIPS 2024
Images diffusion model		$\alpha = \textit{contains the given patch}$ Liu et al., ICLR 2024
DNA diffusion model	A C G T A C	$\alpha = \textit{strong enhancer activity}$ Xu et al., 2026
Protein diffusion model	M K L E V D	$\alpha = \textit{scaffolds a binding motif}$ Xu et al., 2026

Same technique everywhere: Bayesian conditioning of a generative model on an outcome α

Questions for this talk:



1. Do symbolic reasoning algorithms still have a purpose in the age of agentic AI?
2. Can we put “LLM probability” into the symbolic reasoning?
3. Can we put “symbolic reasoning” into the LLM?



Thanks

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



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