

CoTs as Tractable Probabilistic Programs

Kyle Richardson^{*,+}, Yu Feng^{^,+}, Poorva Garg[`], Junyan Cheng[”],
Guy Van den Brock[`] and Dan Roth[^] (+ equal contribution)



Allen Institute for AI^{*}, University of Pennsylvania[^], Dartmouth[”], UCLA[`]

Motivation

Chain-of-thought (CoT) in LLMs

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

“Think step-by-step”

LLM CoT Reasoning Trace:

The members of The Police were musicians. s1
Only law enforcement officers perform lawful arrests of other people. s2
Based on this, the answer is no. (answer: no) s3

Reasoning trace

- CoT traces appear throughout LLM development yet are modeled in task specific ways.
- Lack of a unified formalism makes it difficult to relate different research areas, study formally.

This work

- Propose** modeling CoT traces as a probabilistic programs, inferences as structured queries over programs.
- Develop** a simple discrete probabilistic programming language for CoT that supports tractable probabilistic and logical inference.

CoTs as probabilistic programs (PP)

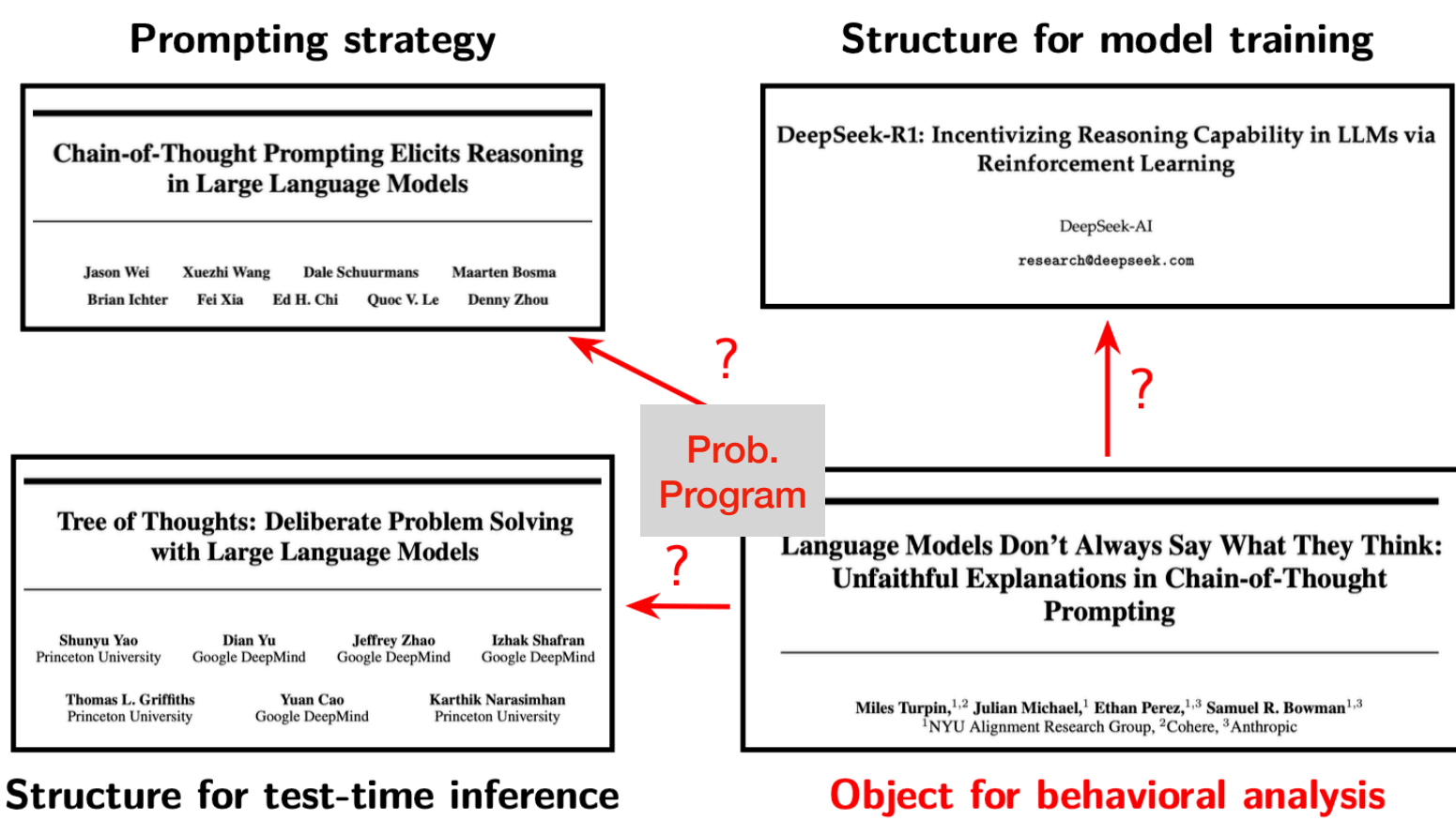
(A) Chain-of-thought

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
Only law enforcement officers perform lawful arrests of other people. s2
Based on this, the answer is no. (answer: no) s3
step deps. s1, s2 lm cond. likelihood: 0.53 verbal confidence: 0.8

The many uses of CoT



(B) Probabilistic Program

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

(C) Probabilistic Queries

$\theta_{\text{likelihood}} = (0.86, 0.96, 0.53)$
 $\theta' = (0.70, 0.90, 0.80)$
 $E = \{P2\}$
 $\mathbb{P}_{\text{pp}}(\text{return "no"}; \theta)$
 $\frac{\partial \mathbb{P}_{\text{pp}}(\text{return "no"}; \theta)}{\partial \theta}$
 $\mathbb{P}_{\text{pp}}(\text{raise ValueError}; \theta)$
 $\mathbb{P}_{\text{pp}}(\text{return "no"}; \theta')$

- Making a CoT trace a PP:** Steps are stochastic assignments, step dependencies are control flow, answers are return values, extra constraints are observes, reasoning errors are exceptions.
- CoT inference as structured program queries:** Likelihoods as marginal queries, sensitivities as gradients and step conditionals, robustness as weight perturbations.

What properties would a language for CoT need to have?

(a) Core syntax of Copper and grounding in CoT		
CoT Concept	Syntax	Program Vars
Steps are probabilistic Booleans	1 P ~ Flip(<str>, <p>)	$\mathcal{V}_{\text{dec}}$
Step dependencies are controls	1 if <bool condition>: 2 <new fact or answer branch v>	guards(v)
Reasoning Errors are Exceptions	1 if <bool condition>: 2 <new fact or answer branch v> 3 else: 4 raise ValueError(<error>)	$\mathcal{V}_{\text{dec}}$
Answers are returns	1 return <answer output>	$\mathcal{V}_{\text{dec}}$
Conditions are observes	1 observe(<bool condition e>)	evidence
Chains/answers can be unique	1 A ₁ , ..., A _n ~ Categorical(<p1, ...>)	$\mathcal{V}_{\text{dec}}$
Step scores can be aggregated	1 P ~ Factor(<str>, f _m (p ₁ , ..., p _m))	$\mathcal{V}_{\text{dec}}$

(b) Reachability semantics		
P1: bool ~ Flip("s1", 0.86)	T 0.86, F 0.14	
P2: bool ~ Flip("s2", 0.96)	T 0.96, F 0.04	raise ValueError
P3: bool ~ Flip("s3", 0.53)	T 0.53, F 0.47	raise ValueError
return "no"		return guard

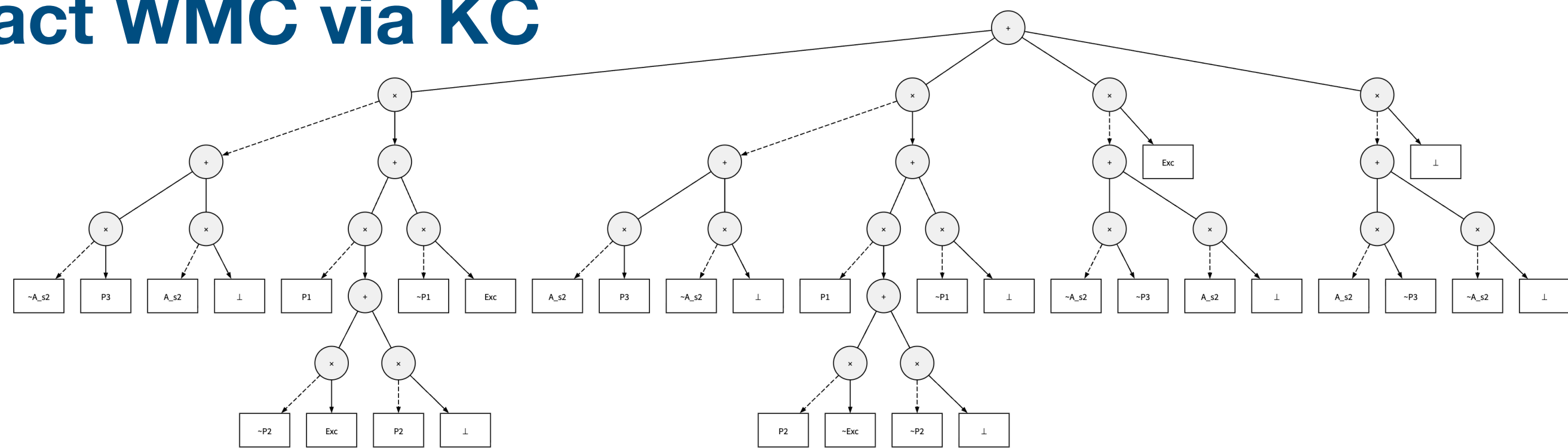
(c) Probabilistic query computation		
$\mathbb{P}_{\text{F}}(q \mid E; \theta) = \frac{\text{WMC}(\text{F} \wedge E \wedge q; \theta)}{\text{WMC}(\text{F} \wedge E; \theta)}$		

(d) Boolean program encoding		
$\text{F} := \bigwedge_{v \in \mathcal{V}_{\text{dec}}} (v \leftrightarrow \bigvee_{g \in \text{guards}(v)} g)$	$\text{E} := \bigwedge_{e \in \text{evidence}} e$	

Score aggregation

P1: bool ~ Flip("s1", 0.642)
P2: bool ~ Flip("s2", 0.595)
P3: bool ~ Flip("s3", 0.698)
EY: bool ~ Factor("yes_E", LogitPool(P1, P2, P3))

Exact WMC via KC



Boolean guard semantics

```
1 @CotProgParser()
2 def pp():
3     P1: bool = ~ Flip("s1", 0.86)
4     P2: bool = ~ Flip("s2", 0.96)
5     if P1 and P2:
6         P3: bool = ~ Flip("s3", 0.53)
7         if P3: return "no"
8     else: raise ValueError
9 else:
10    raise ValueError
```

$\text{Vreturn "no"} \leftrightarrow P1 \wedge P2 \wedge P3$
return guard

What can we do with this language?

Formal analysis of LLM likelihood

Parameterize step scores via LLM

k-answer program F _a	k-chain program F _c
1 # Answer 1 2 P1,1 ~ Flip(s1,1, θ1,1) ... 3 P1,n1 ~ Flip(s1,n1, θ1,n1) ... 4 # Answer k 5 Pk,1 ~ Flip(sk,1, θk,1) ... 6 Pk,nk ~ Flip(sk,nk, θk,nk) ... 7 if P1,1 and ... and P1,n1: 8 return a1 ... 9 elif Pk,1 and ... and Pk,nk: 10 return ak 11 else: 12 raise ValueError("Error")	1 # Trace 1 2 P1,1 ~ Flip(s1,1, θ1,1) ... 3 P1,n1 ~ Flip(s1,n1, θ1,n1) ... 4 # Trace k 5 Pk,1 ~ Flip(sk,1, θk,1) ... 6 Pk,nk ~ Flip(sk,nk, θk,nk) ... 7 A1, ..., Ak,1 ~ A ~ Categorical(8 prod(θ1,1), ..., prod(θk,1)) 9 1 - (prod(θ1,1) + ... + prod(θk,1)) 10 if A1 or ... or Ak: 11 return a 12

LLM CoT likelihood

Standard “latent trace” model of CoT

Derive new test-time inference and prompting strategies

Answer before reasoning steps

Mixes LLM likelihood + verbal estimation

(A) Multi-answer Chain-of-thought

Query Q: Would a sea kayak fit into a normal car? Think step-by-step

LLM CoT Reasoning Trace (single forward call)

For yes (LLM verbal confidence: 0.4)

A sea kayak can vary in size, but some fit into larger vehicles. (0.642) s1
Some cars, especially SUVs, have foldable seats and ample space. (0.595) s2
If the kayak is small and the car large enough, it might fit. (lm likelihood 0.698) s3
Based on this, the answer is: yes (answer) s4

For no (LLM verbal confidence: 0.7)

Sea kayaks are typically 12–18 feet long. (lm likelihood: 0.831) s5
Most cars cannot handle such long objects. (lm likelihood: 0.733) s6
Even with foldable seats, a kayak exceeds interior space. (lm likelihood: 0.676) s7
Based on this, the answer is: no (answer) s8

Direct multi answer/CoT

(B) Copper Program

```
yes: bool = ~ Flip("yes", 0.4)
no: bool = ~ Flip("no", 0.7)
A: bool = ~ Categorical("yes", Combine(yes, no))

P1: bool = ~ Flip("s1", 0.642)
P2: bool = ~ Flip("s2", 0.595)
P3: bool = ~ Flip("s3", 0.698)
EY: bool = Factor("yes_E", LogitPool(P1, P2, P3))

P5: bool = ~ Flip("s5", 0.831)
P6: bool = ~ Flip("s6", 0.733)
P7: bool = ~ Flip("s7", 0.676)
EN: bool = Factor("no_E", LogitPool(P5, P6, P7))
observe((A and EY) or ((not A) and EN))
if A: return "yes"
else: return "no"
```

Bayesian-style inference model

(C) Probabilistic model

$\theta_{\text{verbal}} = (0.4, 0.7)$
 $\mathbb{P}_{\text{prior}}(A \mid Q; \theta'_{\text{verbal}})$
 $\theta_{\text{likelihood}} = (0.642, 0.595, 0.698, 0.831, 0.733, 0.676,)$
 $\mathbb{P}_{\text{evidence}}(T \mid A, Q; \theta'_{\text{like}})$
 $\mathbb{P}_{\text{posterior}}(A \mid T, Q; \theta'_{\text{verbal, like}})$

- Distill CoT traces to PPs that capture exact LLM likelihood inference, provides deeper view into semantics, help derive new bounds.
- Identify structural properties that reveal the limitations of using LLM likelihood for ranking CoT traces, helps to rule out possible prompting strategies from first principles.

- As an alternative to expensive test-time inference strategies, we explore variants of CoT that directly generate multiple CoTs, perform structured inference over direct LLM signals.
- Study a particular variant, **Bayesian multi-answer CoT**, that aggregates step likelihoods via a Bayesian factorization, devise new robustness testing techniques for comparing against standard CoT.