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

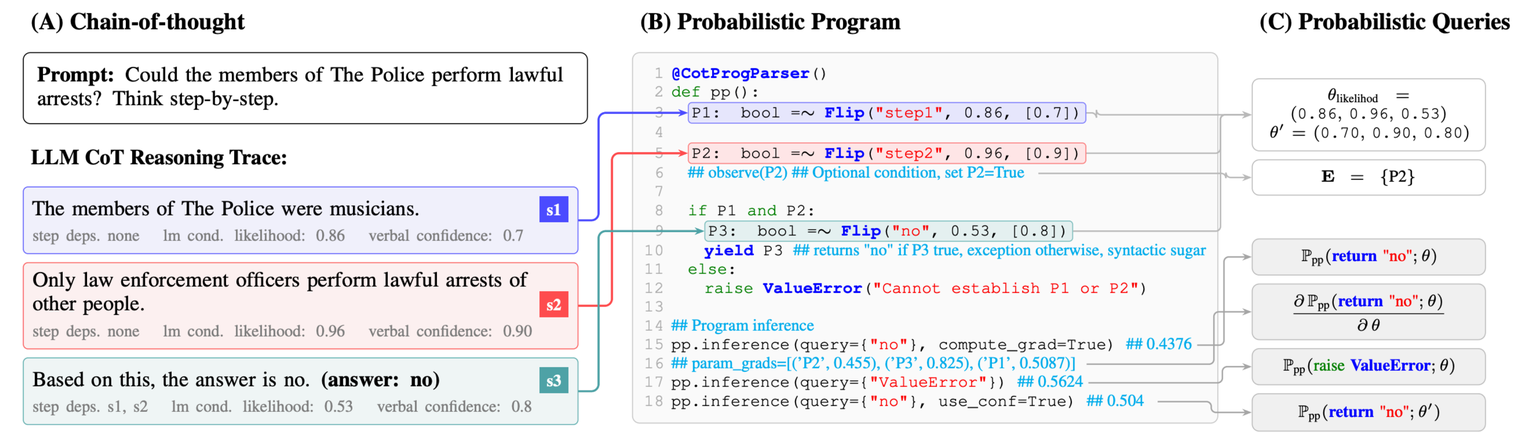
Reasoning trace

- 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.

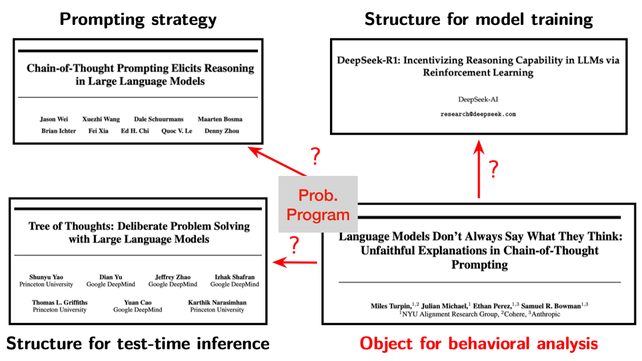
This work

- **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)



The many uses of CoT



- **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?

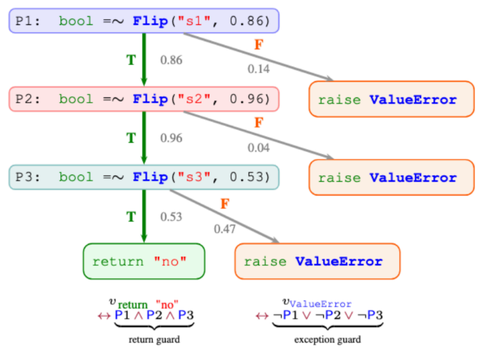
(a) **Core syntax of Copper and grounding in CoT**

CoT Concept	Syntax	Program Vars
Steps are probabilistic Booleans	`P =~ Flip(<str>, <p>)`	$\mathcal{V}_{\text{dec}}$
Step dependencies are controls	`if <bool condition>:` `  <new fact or answer branch v>`	guards(v)

Reasoning Errors are Exceptions	`if <bool condition>:`
	 `  <new fact or answer branch v>`
	 `else:`
	 `  raise ValueError(<error>)`
		$\mathcal{V}_{\neg\text{dec}}$

Answers are returns	`return <answer output>`	$\mathcal{V}_{\neg\text{dec}}$
Conditions are observes	`observe(<bool condition e>)`	evidence
Chains/answers can be unique	$A_1,..,A_n$ `=~ Categorical(<p1, ..>)`	$\mathcal{V}_{\text{dec}}$
Step scores can be aggregated	`P =~ Factor(<str>,` $f_m(p_1,\ldots,p_m)$`)`	$\mathcal{V}_{\text{dec}}$

(b) **Reachability semantics**



(c) **Probabilistic query computation**

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

(d) **Boolean program encoding**

$$\mathsf{F} := \bigwedge_{v \in \mathcal{V}_{\neg\text{dec}}} \left(v \leftrightarrow \bigvee_{g \in \text{guards}(v)} g \right) \quad \mathsf{E} := \bigwedge_{e \in \text{evidence}} e$$

Boolean guard semantics

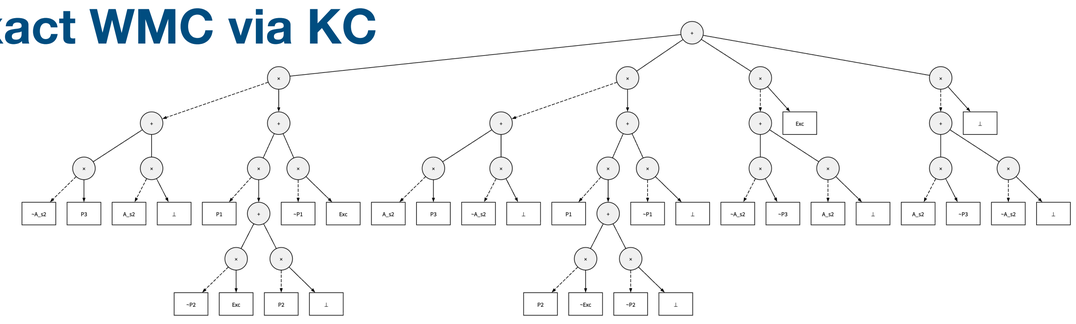
```
@CotProgParser()
def pp():
  P1: bool =~ Flip("s1", 0.86)
  P2: bool =~ Flip("s2", 0.96)
  if P1 and P2:
    P3: bool =~ Flip("s3", 0.53)
    if P3: return "no"
    else: raise ValueError
  else:
    raise ValueError
```

$$v_{\texttt{return "no"}} \leftrightarrow \underbrace{\text{P1} \wedge \text{P2} \wedge \text{P3}}_{\text{return guard}}$$

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



What can we do with this language?

Formal analysis of LLM likelihood

Parameterize step scores via LLM

$k-$**answer program** F_a

```
## Answer 1
P1,1   =~ Flip(s1,1, θ1,1) ...
P1,n1  =~ Flip(s1,n1, θ1,n1) ...
## Answer k
Pk,1   =~ Flip(sk,1, θk,1) ...
Pk,nk  =~ Flip(sk,nk, θk,nk)
if P1,1 and ... and P1,n1:
    return a1 ...
elif Pk,1 and ... and Pk,nk:
    return ak
else:
    raise ValueError('Error')
```

$k-$**chain program** F_c

```
## Trace 1
P1,1   =~ Flip(s1,1, θ1,1) ...
P1,n1  =~ Flip(s1,n1, θ1,n1) ...
## Trace k
Pk,1   =~ Flip(sk,1, θk,1) ...
Pk,nk  =~ Flip(sk,nk, θk,nk)
A1, .. Ak, ¬A =~ Categorical(
prod(θ1,:), ... , prod(θk,:)
1 - (prod(θ1,:) + .. + prod(θk,:))
)
if A1 or ... or Ak:
    return a
```

LLM CoT likelihood

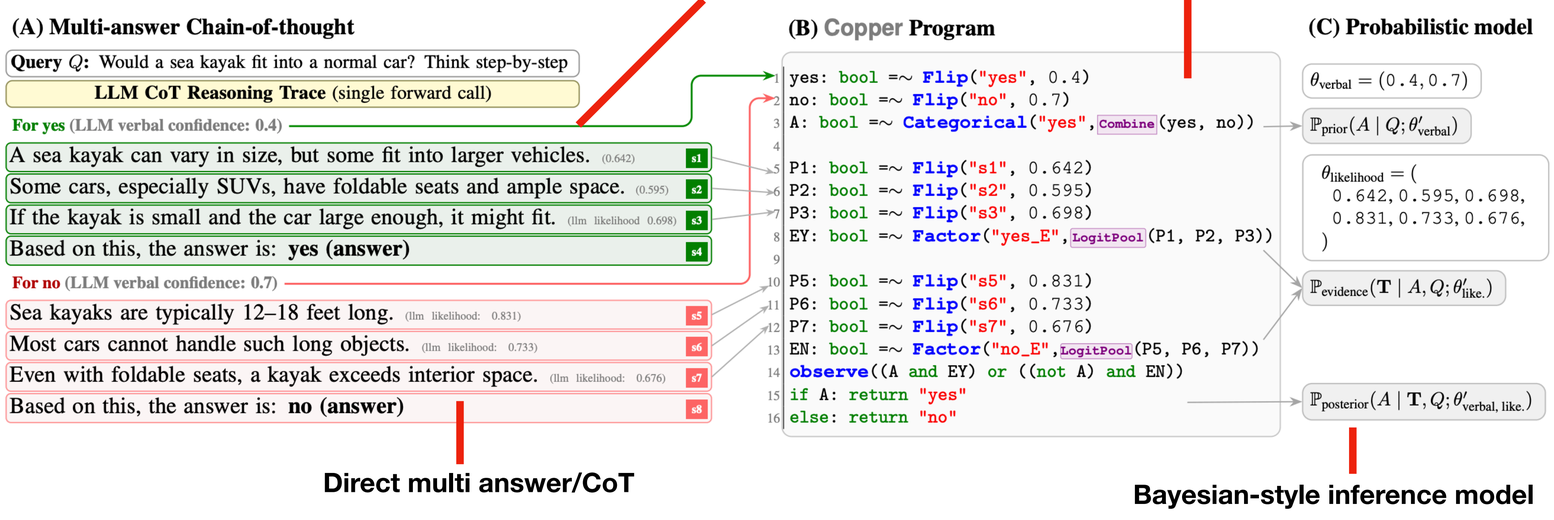
Standard “latent trace” model of CoT

- 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.

Derive new test-time inference and prompting strategies

Answer before reasoning steps

Mixes LLM likelihood + verbal estimation



- 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.