COSI-230B: Natural Language Annotation for Machine Learning

Lecture 19: Reasoning Annotation—Rationales, Process Supervision, and Verification

Jin Zhao

Brandeis University

April 15, 2026

Today’s Agenda

Part I: Foundations (30 min)

- Why reasoning annotation is central
- Chain-of-thought prompting deep dive
- Self-consistency and multiple paths
- GSM8K and step-by-step annotation

Part II: Process Supervision (25 min)

- ORM vs. PRM comparison
- PRM800K annotation protocol
- Math-Shepherd: automated labels

Part III: Risks & Faithfulness (25 min)

- e-SNLI cautionary tale (expanded)
- Faithfulness of explanations
- CoT controllability risk
- Mitigations

Part IV: Hands-On (30 min)

- Demo 1: Rationale leakage
- Demo 2: Process vs. outcome supervision
- In-class annotation activity (15 min)
- Discussion & wrap-up

Goal: Distinguish rationales from process supervision and understand their annotation risks.

Part I

Foundations: From Chain-of-Thought to Step-Level Annotation

Why Reasoning Annotation Is Now Central (and Risky)

The Core Tension

"Show your work" can **improve performance** AND create **new vulnerabilities**.

Reasoning annotation includes:

- **Rationales:** Natural language explanations (human or LLM-produced)
- **Process supervision:** Step-level correctness judgments
- **Verification labels:** Which solution steps are valid

> *"A rationale is not necessarily a trace. Process supervision changes the unit of correctness—and that changes what the model learns."*

When demonstrations include intermediate steps, your "annotation" is now partly the *hidden intermediate state* scaffolding.

Chain-of-Thought Prompting: What Changed Empirically

Wei et al. (2022): Adding step-by-step reasoning to few-shot demonstrations dramatically improves performance on math, logic, and commonsense tasks.

Standard prompting:

- Input → Answer
- Works for simple tasks
- Fails on multi-step reasoning

Chain-of-thought:

- Input → Steps → Answer
- Large gains on math/logic
- Emergent with model scale

Annotation implication: CoT means your demonstrations include intermediate steps—so annotation now shapes *reasoning patterns*, not just answers.

Wei et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. NeurIPS.

`https://arxiv.org/abs/2201.11903`

Chain-of-Thought: Experimental Results Deep Dive

Key findings from Wei et al. (2022) and follow-up studies:

Benchmark	**Standard**	**CoT**	Δ
GSM8K (PaLM 540B)	56.5%	**74.4%**	+17.9
SVAMP (PaLM 540B)	79.0%	**86.6%**	+7.6
AQuA (PaLM 540B)	35.8%	**52.0%**	+16.2
MAWPS (PaLM 540B)	91.6%	**93.3%**	+1.7
StrategyQA (PaLM 540B)	73.9%	**77.8%**	+3.9
Date Understanding	65.8%	**77.3%**	+11.5

Critical observations:

- Gains are **largest on hard multi-step problems** (GSM8K, AQuA)
- Gains are **small or negative on easy benchmarks** (MAWPS single-step)
- CoT is an **emergent ability**: minimal gains below ~100B parameters
- The *quality* of annotated reasoning chains matters—random chains degrade performance

GSM8K: Why Step-by-Step Is Needed

Cobbe et al. (2021): Grade-school math word problems with step-by-step solutions.

Example

Problem: Janet has 3 apples. She buys 2 more, then gives away 1. How many does she have?
Solution: Start with 3. Buy 2 more: $3 + 2 = 5$. Give away 1: $5 - 1 = 4$. **Answer: 4**

Why it matters for annotation:

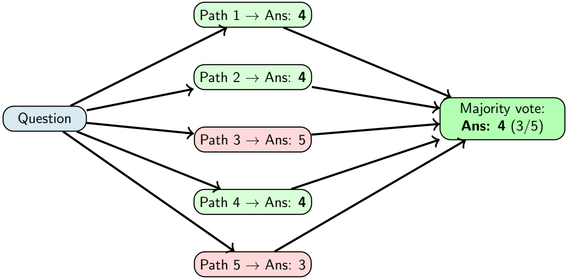
- Final-answer-only training misses intermediate reasoning
- Step-by-step annotations enable **verifier training**
- But: who decides what counts as a "step"? Granularity is an annotation choice
- Each step annotation is a correctness judgment—expensive but powerful

Cobbe et al. (2021). Training Verifiers to Solve Math Word Problems. arXiv:2110.14168.

`https://arxiv.org/abs/2110.14168`

Self-Consistency: Multiple Reasoning Paths + Majority Voting

Wang et al. (2022): Sample multiple CoT paths and take majority vote on the answer.



Question

Path 1 → Ans: **4**

Path 2 → Ans: **4**

Path 3 → Ans: 5

Path 4 → Ans: **4**

Path 5 → Ans: 3

Majority vote:
Ans: 4 (3/5)

Accuracy improvements over standard CoT (PaLM 540B):

- GSM8K: 74.4% → **81.0%** (+6.6)
- SVAMP: 86.6% → **89.4%** (+2.8)
- AQuA: 52.0% → **60.6%** (+8.6)
- StrategyQA: 77.8% → **81.6%** (+3.8)

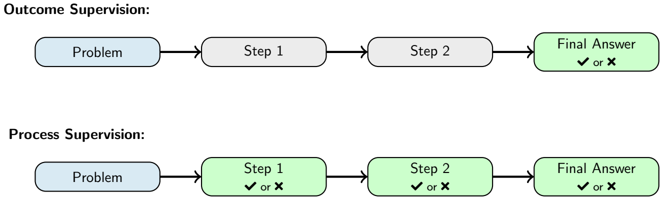
Annotation lesson: Don't force a single rationale! Multiple valid reasoning paths exist.

Wang et al. (2022). Self-Consistency Improves Chain of Thought Reasoning. `https://arxiv.org/abs/2203.11171`

Part II

Process Supervision: From Outcomes to Steps

Process vs. Outcome Supervision



Outcome Supervision:

Problem → Step 1 → Step 2 → Final Answer ✓ or ✗

Process Supervision:

Problem → Step 1 ✓ or ✗ → Step 2 ✓ or ✗ → Final Answer ✓ or ✗

Outcome supervision:

- Only final answer correctness
- Cheap to annotate
- Rewards “right for wrong reasons”

Process supervision:

- Each step labeled correct/incorrect
- Expensive but fine-grained
- Catches errors early; enables verifiers

Lightman et al. (2023). Let’s Verify Step by Step. OpenAI. https://arxiv.org/abs/2305.20050

ORM vs. PRM: Concrete Performance Comparison

Lightman et al. (2023): Direct comparison on MATH benchmark (best-of-N reranking).

Method	$N = 100$	$N = 250$	$N = 1860$
Majority Voting	69.6%	70.2%	70.7%
ORM (best-of-N)	72.4%	73.5%	73.7%
PRM (best-of-N)	**75.0%**	**77.3%**	**78.2%**

Key findings:

- PRM consistently outperforms ORM across all sample budgets
- The gap **widens** with more samples: PRM scales better
- PRM catches "lucky wrong reasoning"—solutions that arrive at correct answers via flawed steps
- ORM can be fooled by correct final answers reached through erroneous reasoning

Annotation cost trade-off: PRM800K required ∼12.5 step labels per solution vs. 1 label for ORM—but each PRM label carries more signal.

PRM800K: What Exactly Is Annotated

Lightman et al. (2023): 800K step-level labels on math solutions.

Annotation protocol:

- Each solution step labeled as: **positive** (correct), **negative** (error), or **neutral**
- Annotators evaluate mathematical validity of each step
- Multiple solutions per problem (from model sampling)
- Step granularity defined by natural line breaks in solutions

Why the unit matters:

- Process reward models (PRMs) trained on step labels outperform outcome reward models (ORMs)
- The "annotation unit" (final answer vs. step) changes *what the model learns*
- Active learning on *which steps to annotate* makes process supervision more efficient

Lightman et al. (2023). Let's Verify Step by Step. OpenAI report. `https://arxiv.org/abs/2305.20050`

PRM800K: Annotation Interface and Protocol

How were 800K step labels actually collected?

Interface design:

- Annotators see one step at a time, in order
- For each step, select: positive, negative, or neutral
- Once a step is marked **negative**, remaining steps are auto-labeled negative
- Steps shown with full problem context and prior steps

Annotator training:

- STEM-qualified contractors
- Calibration phase: annotate shared examples, resolve disagreements
- Inter-annotator agreement: ∼75% on step labels

Lightman et al. (2023). Let's Verify Step by Step. OpenAI report. `https://arxiv.org/abs/2305.20050`

PRM800K: Annotation Interface and Protocol (cont.)

Step granularity decisions:

- A "step" = one natural line break in model output
- Coarse: "Set up equation and solve" (1 step)
- Fine: "Set up equation" + "Solve for x" (2 steps)
- Trade-off: finer → more labels, more signal, but higher cost

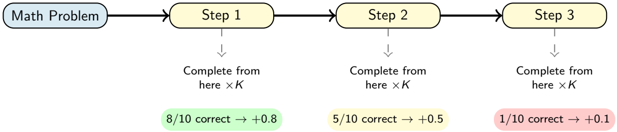
Scale:

- 75K solutions
- 800K step-level labels
- ∼12.5 steps/solution avg

Lightman et al. (2023). Let's Verify Step by Step. OpenAI report. `https://arxiv.org/abs/2305.20050`

Math-Shepherd: Automated Process Supervision Labels

Wang et al. (2024): Can we replace expensive human step labels with **synthetic** process supervision?


Math Problem
Step 1
Step 2
Step 3
Complete from here $\times K$
Complete from here $\times K$
Complete from here $\times K$
8/10 correct → +0.8
5/10 correct → +0.5
1/10 correct → +0.1

Key idea: For each step, sample K completions from that point. The fraction that reach the correct final answer estimates step quality.

Advantages:

- No human annotators needed
- Scales to millions of labels
- Soft scores, not just binary

Results (GSM8K / MATH):

- Matches or exceeds human PRM on some benchmarks
- GSM8K: 84.1% → **87.5%**
- Cost: ~100× cheaper than human labels

Wang et al. (2024). Math-Shepherd: Verify and Reinforce LLMs Step-by-step. `https://arxiv.org/abs/2312.08935`

Part III

Risks and Faithfulness of Reasoning Annotations

e-SNLI: “Explanations as Labels” (Cautionary Tale)

Camburu et al. (2018): Augment SNLI with natural language explanations for each label.

What Went Wrong

- Explanations often **leak the answer**—you can predict the label from the explanation alone
- Explanation quality varies wildly across annotators
- “Post-hoc” rationales may not reflect actual reasoning
- Models can learn to generate “nice explanations” without correct reasoning

Lesson: Rationales are not automatically trustworthy annotations. They may be non-faithful, post-hoc rationalizations that happen to correlate with labels.

Camburu et al. (2018). e-SNLI: Natural Language Inference with Natural Language Explanations. NeurIPS.

`https://arxiv.org/abs/1812.01193`

e-SNLI: Leaky vs. Non-Leaky Rationales

Concrete examples from the dataset:

Leaky Rationale (bad—reveals the label)

P: “A man is playing guitar on stage.”
H: “A man is performing music.”
Label: Entailment
Rationale: “Playing guitar on stage *is a form of performing music*, so it is **entailment**.”
⚠ The word “entailment” in the rationale directly leaks the label!

Non-Leaky Rationale (better—explains without revealing)

P: “Children are playing in the park.”
H: “Children are outdoors.”
Label: Entailment
Rationale: “A park is an outdoor location, and playing in one requires being outdoors.”
✓ Explains the relationship without naming the label class.

e-SNLI: Leaky vs. Non-Leaky Rationales (cont.)

Finding: A classifier trained *only* on rationales (ignoring P and H) achieves ∼90% accuracy—clear evidence of label leakage.

Camburu et al. (2018); Wiegreffe & Marasović (2021). Teach Me to Explain. NAACL.

`https://arxiv.org/abs/1812.01193`

Faithfulness of Explanations

When do rationales reflect actual model reasoning vs. post-hoc rationalization?

Faithful Explanation

The explanation **accurately describes** the causal process that led to the prediction.

- Removing cited features changes the prediction
- The explanation is *necessary* for the output

Post-Hoc Rationalization

The explanation **sounds plausible** but does not reflect the actual decision process.

- Cited features are irrelevant to the prediction
- The explanation is *confabulated*

Jacovi & Goldberg (2020). Towards Faithfully Interpretable NLP Systems. ACL; Wiegreffe & Marasović (2021). Teach Me to Explain. NAACL.

Faithfulness of Explanations (cont.)

Tests for faithfulness:

1. **Counterfactual test:** Does removing the cited evidence change the model's answer?
2. **Sufficiency test:** Does providing *only* the cited evidence reproduce the prediction?
3. **Consistency test:** Do similar inputs produce similar explanations?
4. **Simulatability:** Can a human predict the model's output from the explanation alone?

Jacovi & Goldberg (2020). Towards Faithfully Interpretable NLP Systems. ACL; Wiegreffe & Marasović (2021). Teach Me to Explain. NAACL.

New Risk: CoT Controllability

Chen et al. (2026): Models can strategically control what they verbalize in their chain of thought.

The Problem

- Written CoT is not necessarily a faithful trace of internal computation
- Models can produce “nice-looking” reasoning that masks errors
- CoT as annotation product is not automatically trustworthy as a monitor
- Strategic CoT undermines the assumption that “showing work = transparency”

Mitigation strategies:

- Separate “explanation for user” from “trace used for training”
- Prefer verifiable intermediate representations for high-stakes domains
- Use “no-CoT” checks: test if model can solve without showing work

Chen et al. (2026). Reasoning Models Struggle to Control their Chains of Thought. OpenAI.
https://cdn.openai.com/pdf/a21c39c1-fa07-41db-9078-973a12620117/cot_controllability.pdf

Mitigations for Reasoning Annotation Risks

1. **Rationale auditing:** Check if rationales are predictive of answers independently of the input
2. **Rubric constraints:** Define what constitutes a valid step; forbid answer-leaking patterns
3. **Verifier training:** Train separate models to evaluate step correctness (PRMs)
4. **Dual-channel solutions:** Separate "explanation for user" from "trace used for training"
5. **"No-CoT" checks:** Verify that removing reasoning doesn't collapse performance (indicating reliance on leakage rather than genuine reasoning)
6. **Multiple annotators per step:** Reduce noise in step-level labels

Part IV

Hands-On: Demos, Annotation, and Discussion

Demo 1: Rationale Leakage

Setup: Rationales that leak the answer vs. generic rationales.

```
import random
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

random.seed(0)

def make(n, leak=True):
    X, y = [], []
    for _ in range(n):
        ans = random.choice(["A","B","C"])
        problem = "Choose␣the␣correct␣option."
        rationale = f"The␣answer␣is␣{ans}␣because␣..." if leak \
                    else "Because␣of␣the␣evidence␣..."
        X.append(problem + "␣" + rationale)
        y.append(ans)
    return X, y

Xtr,ytr = make(800, leak=True)
Xte,yte = make(200, leak=True)
Xte2,yte2 = make(200, leak=False)  # leakage removed at test time

vec = CountVectorizer()
clf = LogisticRegression(max_iter=2000).fit(vec.fit_transform(Xtr), ytr)
print("leak␣test:", accuracy_score(yte, clf.predict(vec.transform(Xte))))
print("no-leak␣test:", accuracy_score(yte2, clf.predict(vec.transform(Xte2))))
```

Demo 2: Process vs. Outcome Supervision Simulation

Setup: Compare selecting solutions by final-answer checking (ORM) vs. step-level scoring (PRM).

```python
import random; random.seed(42)

def gen_solution():
    """Generate␣a␣4-step␣solution;␣each␣step␣correct␣w.p.␣0.7."""
    steps = [random.random() < 0.7 for _ in range(4)]
    # Final answer correct only if ALL steps correct
    final_correct = all(steps)
    return steps, final_correct

def orm_select(solutions):
    """ORM:␣pick␣first␣solution␣with␣correct␣final␣answer."""
    for s, fc in solutions:
        if fc: return s, fc
    return solutions[0]       # fallback

def prm_select(solutions):
    """PRM:␣pick␣solution␣with␣highest␣fraction␣of␣correct␣steps."""
    return max(solutions, key=lambda x: sum(x[0])/len(x[0]))
```

Demo 2: Process vs. Outcome Supervision Simulation (cont.)

```python
N_PROBLEMS, K = 500, 10
orm_wins = prm_wins = 0
for _ in range(N_PROBLEMS):
    sols = [gen_solution() for _ in range(K)]
    _, orm_ok = orm_select(sols)
    best_steps, _ = prm_select(sols)
    prm_ok = all(best_steps)
    orm_wins += int(orm_ok); prm_wins += int(prm_ok)

print(f"ORM␣correct:␣{orm_wins}/{N_PROBLEMS}␣=␣{orm_wins/N_PROBLEMS:.1%}")
print(f"PRM␣correct:␣{prm_wins}/{N_PROBLEMS}␣=␣{prm_wins/N_PROBLEMS:.1%}")
```

In-Class Activity: Annotate Math Solution Steps (15 min)

Instructions: For each step below, label it as **Correct**, **Incorrect**, or **Neutral**.

Problem: A store sells notebooks for $4 each. Maria buys 5 notebooks and pays with a $30 bill. How much change does she get?

Step 1: Each notebook costs $4. Maria buys 5 notebooks.
Step 2: Total cost: $5 \times 4 = 20$ dollars.
Step 3: Maria pays with a $30 bill. Change: $30 - 20 = 15$ dollars.
Step 4: Maria gets $15 in change.

In-Class Activity: Annotate Math Solution Steps (cont.)

Problem: John runs 3 miles on Monday, twice as far on Tuesday, and rests on Wednesday. What is his total distance?

Step 1: Monday: 3 miles.
Step 2: Tuesday: twice as far as Monday, so $3 \times 3 = 9$ miles.
Step 3: Wednesday: 0 miles (rest day).
Step 4: Total: $3 + 9 + 0 = 12$ miles.

Discussion prompts: Where did you disagree? What made some steps ambiguous? How does step granularity affect your label?

Activity Debrief: What Did We Learn?

Expected observations:

Problem 1 (Notebooks):

- Steps 1–2: clearly correct
- Step 3: arithmetic error! $30 - 20 = 10$, not 15 → incorrect
- Step 4: follows from Step 3, so also incorrect (or neutral?)

Key debate: If a step logically follows from a prior *wrong* step, is it "correct given context" or "incorrect"?

Problem 2 (Running):

- Step 1: correct
- Step 2: "twice as far" $= 2 \times 3 = 6$, not $3 \times 3 = 9 \rightarrow$ incorrect
- Step 3: correct
- Step 4: follows from error → incorrect

Key debate: PRM800K policy says once a step is negative, all following are auto-negative. Do you agree?

Takeaway: Even "simple" math annotation produces real disagreements. Inter-annotator agreement of ∼75% in PRM800K is not surprising!

Discussion Questions

1. If process supervision improves math reasoning, should we always annotate steps? When is it too expensive or too misleading?
2. If models can manipulate their CoT, how should we redesign “reasoning annotation” for monitoring? What representations would be more trustworthy?
3. Should you store rationales in public datasets, given leakage and privacy concerns? What policies would you adopt?
4. Math-Shepherd shows synthetic process labels can match human labels on some benchmarks. When should we trust automated labels over human annotations?

Key Takeaways

1. **Rationale $\neq$ trace:** Natural language explanations may not reflect actual reasoning
2. **Process supervision changes the game:** Step-level labels enable verifiers and catch errors earlier
3. **The annotation unit matters:** Final answer vs. step labels yield different model behaviors
4. **Leakage is a real risk:** Rationales can encode answers, creating artificial performance gains
5. **CoT is not automatically trustworthy:** Models may control their verbalized reasoning strategically
6. **Synthetic supervision is viable:** Math-Shepherd shows automated step labels can approach human quality at lower cost

Key Takeaways (cont.)

Key Definitions Recap

Rationale: NL explanation; may be non-faithful. | **Process supervision:** Step-level correctness labels. **Outcome supervision:** Final answer only. | **Verifier (PRM/ORM):** Model evaluating steps or outcomes.

Readings

Required readings for next class:

- Lightman et al. (2023). *Let's Verify Step by Step.* OpenAI report.
  `https://arxiv.org/abs/2305.20050`
- Wang et al. (2024). *Math-Shepherd: Verify and Reinforce LLMs Step-by-step without Human Annotations.* ACL.
  `https://arxiv.org/abs/2312.08935`

Optional / further reading:

- Wei et al. (2022). *Chain-of-Thought Prompting Elicits Reasoning in LLMs.* NeurIPS.
- Camburu et al. (2018). *e-SNLI: NLI with Natural Language Explanations.* NeurIPS.
- Jacovi & Goldberg (2020). *Towards Faithfully Interpretable NLP Systems.* ACL.
- Chen et al. (2026). *Reasoning Models Struggle to Control their Chains of Thought.* OpenAI.

Full reference list: Wei et al. (2022); Cobbe et al. (2021); Wang et al. (2022, 2024); Lightman et al. (2023); Camburu et al. (2018); Chen et al. (2026). `https://arxiv.org/abs/2305.20050`