COSI-230B: Natural Language Annotation for Machine Learning

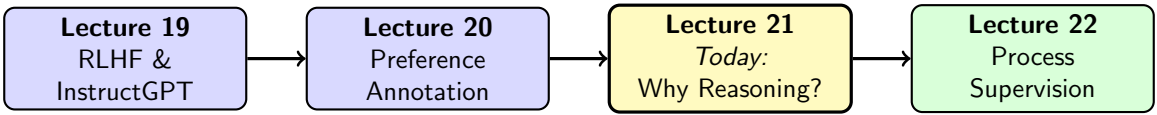
Lecture 21: From RLHF to Reasoning Tuning — Why the Paradigm Shifted

Jin Zhao

Brandeis University

April 22, 2026

Where We Are in the Course



Today's question

RLHF gave us ChatGPT. So why did the field pivot to **reasoning tuning** (o1, DeepSeek-R1, Claude thinking)?
What changed, and what does this mean for **annotation**?

Today's Agenda

Part I: The Limits of RLHF

- Recap: what RLHF solved
- Where RLHF gets stuck: reasoning
- The "one reward at the end" problem

Part II: The Reasoning Idea

- "Thought" as tokens
- Chain-of-thought (CoT)
- A bit of math: marginalizing over thoughts

Part III: Learning to Reason

- Outcome reward model (ORM)
- Process reward model (PRM)
- RL on reasoning trajectories

Part IV: Two New Scaling Laws

- Train-time scaling (RL)
- Test-time scaling (more thinking)
- The o1 milestone
- Bridge to Lecture 22

Goal: Understand *why* the field needed a new paradigm beyond RLHF — and what that means for how we collect data.

Part I

The Limits of RLHF

Quick Recap: What RLHF Does

The three-step recipe (from Lecture 19)

1. **SFT**: imitate human-written demonstrations
2. **Reward Model**: learn from human *pairwise preferences*
3. **RL**: optimize the policy against the reward model

The RLHF objective (from Lecture 19)

$$\max_{\theta} \; \mathbb{E}_{x,y\sim\pi_\theta}\big[r_\phi(x,y)\big] \; - \; \beta \cdot \mathrm{KL}\big(\pi_\theta \,\|\, \pi_{\mathsf{SFT}}\big)$$

What RLHF is good at

Tone, style, helpfulness, safety, instruction following
$\Rightarrow$ "Level-1 conversational AI."

The Wall: RLHF Struggles on Hard Reasoning

Where RLHF plateaus

Tasks that require **many correct steps in a row**:

- Competition math (AIME, Olympiad)
- Multi-step code debugging
- Scientific derivations, proofs
- Planning and long-horizon agents

Symptom: the model sounds confident and fluent — but the answer is *wrong*.

Why?

Helpfulness and correctness are *different* problems.
RLHF optimizes for "sounds like a good answer",
not "is actually a correct chain of deductions."

A Concrete Failure

Prompt: *"What is the 100th term of the arithmetic sequence* $6, 10, 14, 18, \ldots$ *?"*

RLHF-aligned LLM

"The 100th term of the sequence is **396**."

Confident. Fluent. Wrong.
(Correct: $6 + 99 \cdot 4 = 402$)

What we want

Step 1. Common difference $d = 10 - 6 = 4$.
Step 2. Formula: $a_n = a_1 + (n-1)d$.
Step 3. $a_{100} = 6 + 99 \cdot 4 = 402$.

Structured. Step-by-step. Verifiable.

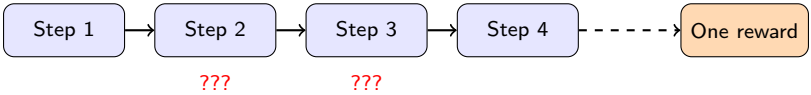
The insight

We need a model that **works things out**, not one that just *produces an answer shape*.

The Core RLHF Problem: One Reward at the End

In RLHF, the reward model scores the *whole response*:

$$r_\phi(x, y) \quad \text{where } y = (y_1, y_2, \ldots, y_T)$$



The credit assignment problem:

- If the final answer is wrong — *which step was to blame?*
- If the final answer is right — *were all the steps actually valid?*
- The model may luck into the right answer via **invalid reasoning**.

The diagnosis

Outcome-level feedback is too sparse for multi-step reasoning.

Two Axes Where RLHF Falls Short

1. Supervision is too coarse

Humans rank final answers, not intermediate steps.

$\Rightarrow$ Model learns *what looks good*, not *how to think*.

2. Compute is frozen at inference

A trained RLHF model runs in one forward pass.

$\Rightarrow$ Hard problems get the same "budget" as easy ones.

The two responses that define modern reasoning models

1. Supervise the **steps**, not just the answer $\rightarrow$ *process supervision*
2. Let the model **think longer** when a problem is hard $\rightarrow$ *test-time scaling*

The rest of today's lecture is about these two ideas.

Part II

The Reasoning Idea

"Let the model show its work"

The Key Shift: "Thought" Is Just More Tokens

Old picture

Question $x \longrightarrow$ Answer y

New picture

Question $x \longrightarrow$ **Reasoning** $z \longrightarrow$ Answer y

z is called a **thought** or **rationale**:

- It is a sequence of tokens produced *before* the final answer
- It is generated by the same model, with the same next-token objective
- It turns a single forward pass into a **multi-step computation**

Why this is powerful

The model now has **scratch space**. The transformer computes over its own generated tokens — giving it *more effective depth* than its architecture alone.

Chain-of-Thought (CoT) Prompting

Wei et al. (2022): just adding "Let's think step by step" can dramatically improve math performance.

Standard prompting

Q: Roger has 5 tennis balls. He buys 2 more cans of 3 balls each. How many balls now?

A: 11.

(Often wrong on harder versions.)

Chain-of-thought prompting

Q: Same question.

A: Roger starts with 5. Two cans of 3 balls $= 6$ balls. $5 + 6 = 11$. The answer is **11**.

(Shows the steps $\Rightarrow$ more reliable.)

What CoT teaches us

Reasoning ability was *latent* in LLMs — we just had to let the model **spend tokens** on the problem.

A Bit of Math: Marginalizing Over Thoughts

The probability of the final answer is a sum over all possible reasoning paths:

Answer as a marginal

$$P(y \mid x) = \sum_{z} \underbrace{P(y \mid x, z)}_{\text{answer given thought}} \cdot \underbrace{P(z \mid x)}_{\text{thought given question}}$$

Plain English:

- There are **many** valid reasoning paths z leading to the same answer y
- A good model should put probability mass on *good* paths
- CoT just makes z **explicit** in the output sequence

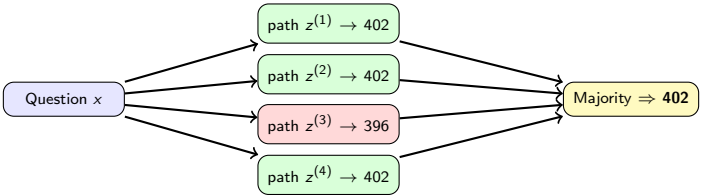
Two ways to exploit this

- **Test-time:** sample many z's, aggregate their y's (self-consistency, majority vote)
- **Train-time:** reward the model for finding z's that lead to correct y's

Self-Consistency: Many Paths, One Answer

Sample K reasoning paths; take a **majority vote** over answers:

$$\hat{y} \;=\; \arg\max_{y} \sum_{k=1}^{K} \mathbb{1}\big[\,\text{answer}(z^{(k)}) = y\,\big]$$



- Wrong paths tend to produce *different* wrong answers
- Right paths tend to **converge** on the same right answer
- No extra training — just more **test-time compute**

Beyond Linear Chains

CoT is just the beginning. The field generalized "thinking" to:

Tree-of-Thoughts

Branch, evaluate partial paths, backtrack.

Like search in game-playing.

Least-to-Most

Decompose into sub-problems first; solve in order.

Like divide-and-conquer.

ReAct

Interleave *thoughts* with *actions* (tool calls, searches).

Like an agent loop.

But a limitation remains

These are all **prompting tricks**. They help a frozen model think better — they do not *teach* it to think better.

Next: can we **train** *models to reason, not just prompt them?*

Part III

Learning to Reason

"Reward the steps, not just the answer"

Why Reinforcement Learning Fits Reasoning

In math / code / logic, we have something RLHF didn't have: a **cheap correctness check**.

- Math: compare to the ground-truth answer
- Code: run the unit tests
- Proofs: run the proof checker

The RL loop for reasoning

1. Sample many reasoning trajectories for a prompt
2. Keep the ones that reach the correct answer (trial & error)
3. Train the model to prefer those trajectories

The big shift

No labeler writes the rationale. The model **generates** candidates; an automatic check selects the good ones — "the shift from human annotation to LLM-driven search."

Two Kinds of Reward Models

Outcome Reward Model (ORM)

Scores the **final answer**: $r_\phi^{\text{ORM}}(x, y)$

- Easy to label (check the answer)
- Credit assignment is hard
- Inherits RLHF's sparsity

Process Reward Model (PRM)

Scores **each step**: $r_\phi^{\text{PRM}}(x, z_{1:t})$

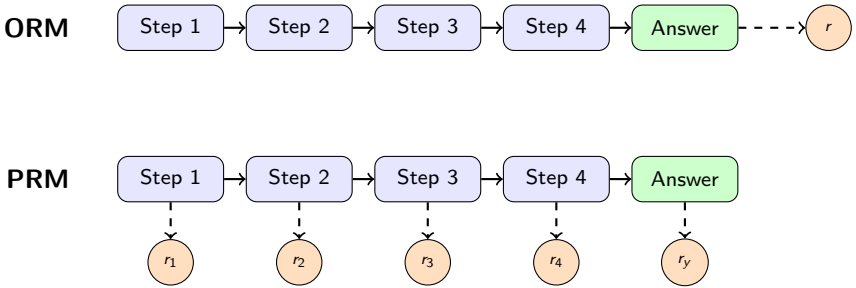
- Dense, step-level signal
- Labels expensive (or automated)
- Much easier to debug

In one line

ORM asks *"is the answer right?"* — PRM asks *"is* **each step** *right?"*

Lecture 22 is all about how PRMs are annotated. Today: what they're for.

ORM vs. PRM, Visually



The payoff of PRM

If Step 3 is wrong, we can **punish Step 3 directly** — not dilute the signal across the whole trajectory.

The Reasoning RL Objective (Light Math)

Just like RLHF, we maximize expected reward with a KL anchor —
but now the trajectory **includes reasoning tokens** z:

Outcome-based version

$$\max_{\theta} \; \mathbb{E}_{x,\,(z,y)\sim\pi_\theta}\Big[r_\phi^{\mathrm{ORM}}(x,y)\Big] \;-\; \beta \cdot \mathrm{KL}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$$

Process-based version

$$\max_{\theta} \; \mathbb{E}_{x,\,(z,y)\sim\pi_\theta}\Big[\sum_t r_\phi^{\mathrm{PRM}}(x, z_{1:t})\Big] \;-\; \beta \cdot \mathrm{KL}(\pi_\theta \,\|\, \pi_{\mathrm{ref}})$$

Key differences from Lecture 19's RLHF:

- y is now a final answer *after reasoning*, not the whole response
- PRM gives a reward **per step** — denser credit assignment
- The reward often comes from a **verifier**, not just human preferences

Two Algorithms You'll Hear About

DPO (Direct Preference Optimization)

Skip the explicit reward model. Train directly on preference pairs:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma\Big(\beta \log \frac{\pi_\theta(y^+)}{\pi_{\text{ref}}(y^+)} - \beta \log \frac{\pi_\theta(y^-)}{\pi_{\text{ref}}(y^-)}\Big)$$

- Simpler, no RL loop
- Popular for chat alignment

GRPO (Group Relative Policy Optimization)

Sample a **group** of answers per prompt; use their relative reward as the advantage.

- No separate value network
- Used in DeepSeek-R1 / DeepSeek-Math
- Well-suited to verifiable rewards

Big picture

PPO, DPO, GRPO are all **implementation choices**. The real story is *what* they optimize: RLHF optimizes human preference on answers; **reasoning RL** optimizes verifiable correctness on trajectories.

Part IV

Two New Scaling Laws

"More training compute *and* more thinking compute"

The Old Scaling Law

The classic recipe

Better models = more data + more parameters + more pretraining FLOPs.

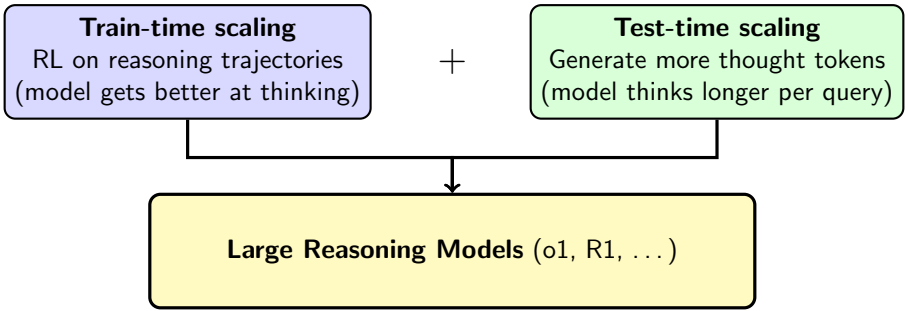
- This worked from GPT-2 $\rightarrow$ GPT-3 $\rightarrow$ GPT-4
- But returns are **diminishing**: $10\times$ compute no longer gives $10\times$ better models
- High-quality text data is increasingly **exhausted**

The practical question (circa 2024)

If we cannot easily keep scaling pretraining, where does the next leap come from?

Answer: two new axes to scale.

Two New Axes to Scale

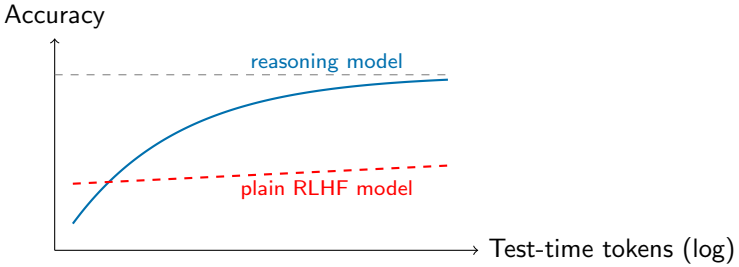


Both axes use the same primitive

"Thought tokens." Generating them during training grows capability; generating more of them at test time grows accuracy.

Test-Time Scaling: Spend Tokens to Gain Accuracy

Intuition: for the same model, more "thinking" at inference gives a better answer.



- RLHF model **plateaus** — answers in one shot, more tokens don't help
- Reasoning model **keeps improving** as it thinks longer
- This is the **test-time scaling law**: search, verification, majority vote are all ways to spend test-time compute

PRM-Guided Search at Test Time

Given a PRM, we can **search** over reasoning paths, not just sample one:

Majority vote

Sample K chains, vote on the answer.
No PRM needed.

Beam / best-of-N

Keep top-K partial paths by PRM score.

Tree search (MCTS)

Expand promising branches; backtrack on bad ones.

The unified picture

A PRM doubles as:

- a **training signal** (Part III)
- a **search heuristic at test time** (this slide)

The same annotation effort pays off twice.

The Milestone: OpenAI o1 (Sept. 2024)

Why o1 mattered

First widely-used model to **openly "think before answering"** — with long internal chains of thought that noticeably improve hard reasoning.

What o1 demonstrated:

- Large-scale RL on reasoning traces is a real new training regime
- Test-time thinking yields real accuracy gains on math / code / science
- "Level 1 conversational AI" $\rightarrow$ "Level 2 reasoning AI"

What followed (all very recent):

- DeepSeek-R1 (open weights; GRPO-based)
- Claude's extended / thinking modes
- Open-source reproductions (OpenR, LLaMA-Berry, Journey Learning, ...)

The survey's central claim: this is a genuinely new paradigm, not a bigger RLHF.

RLHF vs. Reasoning Tuning: Side by Side

	RLHF (Lectures 19–20)	**Reasoning Tuning (today)**
Goal	Helpful, honest, harmless chat	Correct multi-step reasoning
Output shape	Direct answer	Thought z, then answer y
Feedback signal	Human preferences over full answers	Verifiable correctness or step-level labels
Reward model	One score per answer (ORM)	Often a **PRM** scoring each step
Data source	Human labelers writing & ranking	LLM search + checkers (MCTS, verifiers)
Algorithm	PPO, DPO	PPO, GRPO, DPO on reasoning traces
Scaling knob	Pretraining FLOPs	Train-time RL + **test-time tokens**

Take-away

Reasoning tuning **reuses** RLHF's machinery — with *denser rewards*, *verifiable signals*, and a *new compute axis at inference time*.

What This Means for Annotation (Setup for Lecture 22)

Moving from RLHF to reasoning changes **what annotators do**:

RLHF annotation (Lecture 20)

- Rank whole answers
- Judge helpfulness / harmfulness
- One label per response

Reasoning annotation (Lecture 22)

- Judge individual reasoning steps
- "Correct / incorrect / neutral" per step
- Label 10–50 **steps** per response

New annotation challenges

- **Cost explodes** — which motivates automated process annotation (MCTS, verifiers)
- **Expertise matters** — labelers must understand the domain (math, code)
- **Rubrics matter** — what counts as a "step"? A valid deduction? A useful one?

Lecture 22 dives into PRM800K, Math-Shepherd, and the faithfulness pitfalls.

Risks and Open Problems

Reward hacking on reasoning

The model can produce reasoning *that looks good to the PRM* but is unfaithful to the real computation.

Faithfulness

Stated reasoning may not reflect *how* the model actually reached its answer. (Lecture 22.)

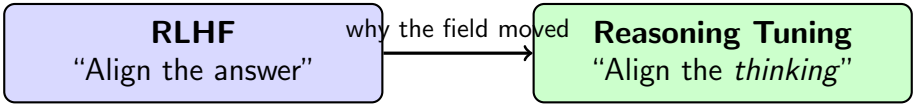
Verifiable-domain bias

Reasoning RL works best where an automatic check exists (math, code). **Open domains** — ethics, writing, medicine — lack such checks.

Cost and stability of PRMs

PRM data is expensive; MCTS simulation is compute-heavy; RL on long trajectories is unstable.

One-Sentence Summary



The idea in one sentence

RLHF optimizes the final answer against human preferences; reasoning tuning optimizes the **reasoning trajectory** against step-level or verifiable rewards — and then lets the model **think longer at test time**.

- CoT: let the model use scratch space
- PRM: reward the steps, not just the answer
- RL + search: train and infer with more "thinking compute"

Questions to Think About

1. Why doesn't RLHF, on its own, teach a model to do Olympiad math well?
2. If you had 100 annotator-hours, would you spend them on preference ranking or step-level labels? What would you be optimizing for?
3. Test-time scaling means inference is *more expensive*. What are the deployment trade-offs?
4. Reasoning RL works beautifully in math. Why is it harder in, say, medical advice or creative writing?
5. A PRM and a verifier both give step-level signal. How are they different?

Reading

Xu et al. (2025). "Towards Large Reasoning Models: A Survey of Reinforced Reasoning with Large Language Models." arXiv:2501.09686.

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

Lightman et al. (2023). "Let's Verify Step by Step." (PRM800K — *next lecture*.)

Next Time: Lecture 22

Reasoning Annotation: Rationales, Process Supervision, and Verification

- How is PRM data actually collected? (PRM800K protocol)
- Automated alternatives: Math-Shepherd, MCTS labeling
- Rationales vs. process supervision — a crucial distinction
- Faithfulness: when the stated reasoning *lies* about the real computation
- Hands-on annotation activity

Today gave you the **why**. *Next time: the* **how** *of annotation.*