CSCI 5541: Natural Language Processing

Lecture 20: Reasoning

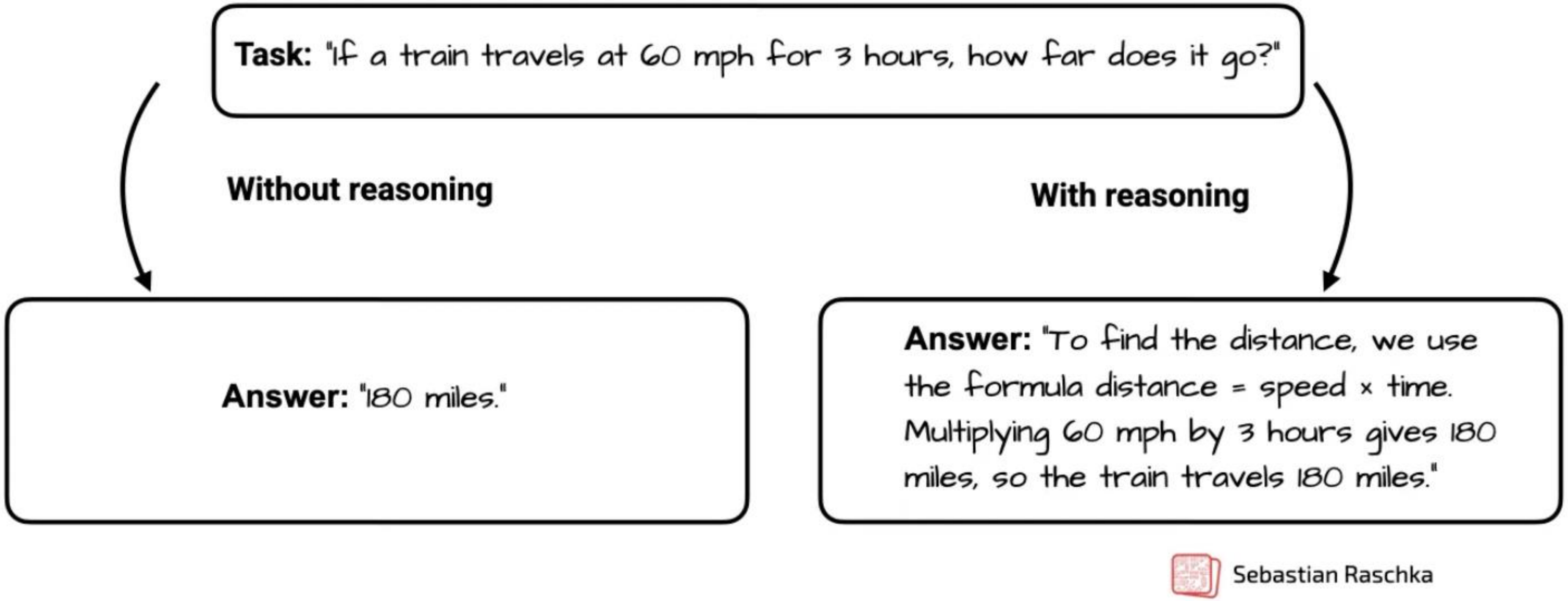
Slides borrowed from Sean Welleck (CMU), and Sebastian Raschka (UW-Madison)

University of Minnesota
Driven to Discover®

Announcements (4/22)

- Presentations (see canvas/slack after class for more details)
  - Location (Shepherd Drone Lab)
  - Poster Format and Printing Information
  - Presentation Dates
    - ✓ Group B → April 29
    - ✓ Group A → May 1
  - All students ***must*** show up for both days to give feedback to other students
- HW5 Due Today (4/22)
- HW6 Out (Due May 5)

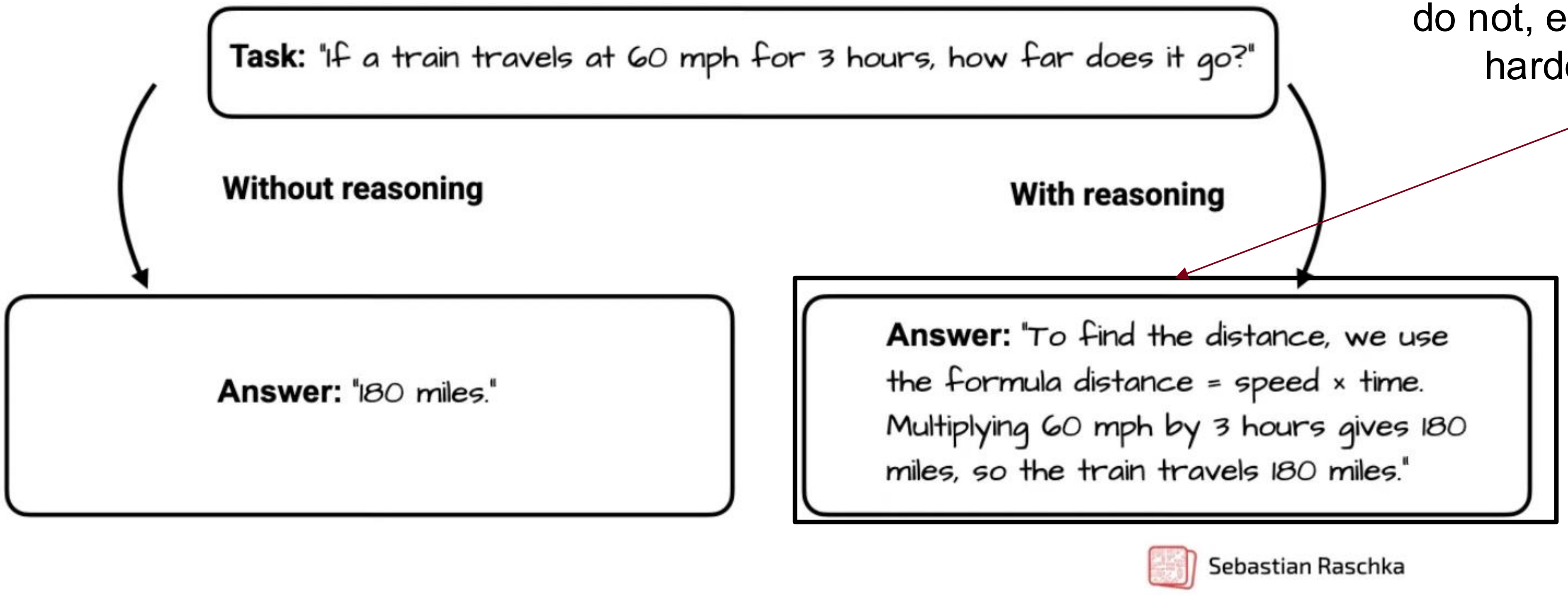
What is Reasoning



The State of LLM Reasoning Model Inference (Raschka, 2025)

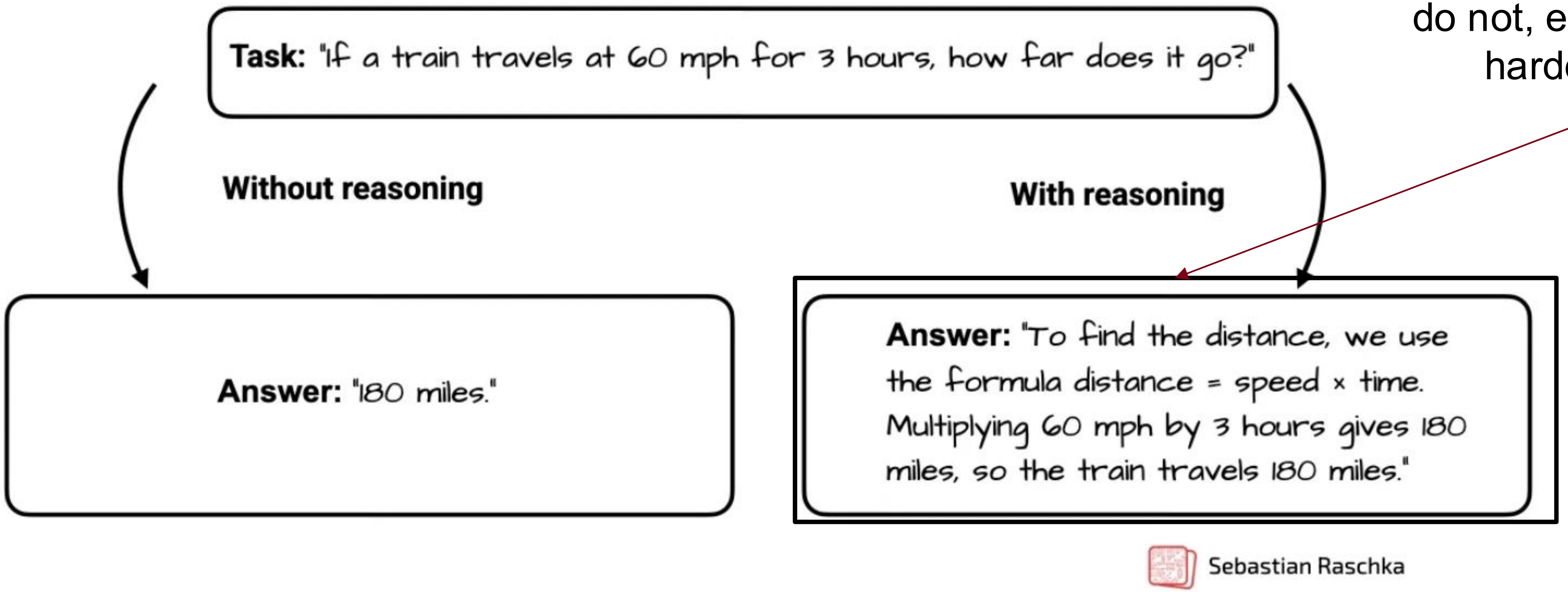
What is Reasoning

Models that use intermediate steps to answer questions oftentimes perform better than those which do not, especially on harder tasks



The State of LLM Reasoning Model Inference (Raschka, 2025)

What is Reasoning



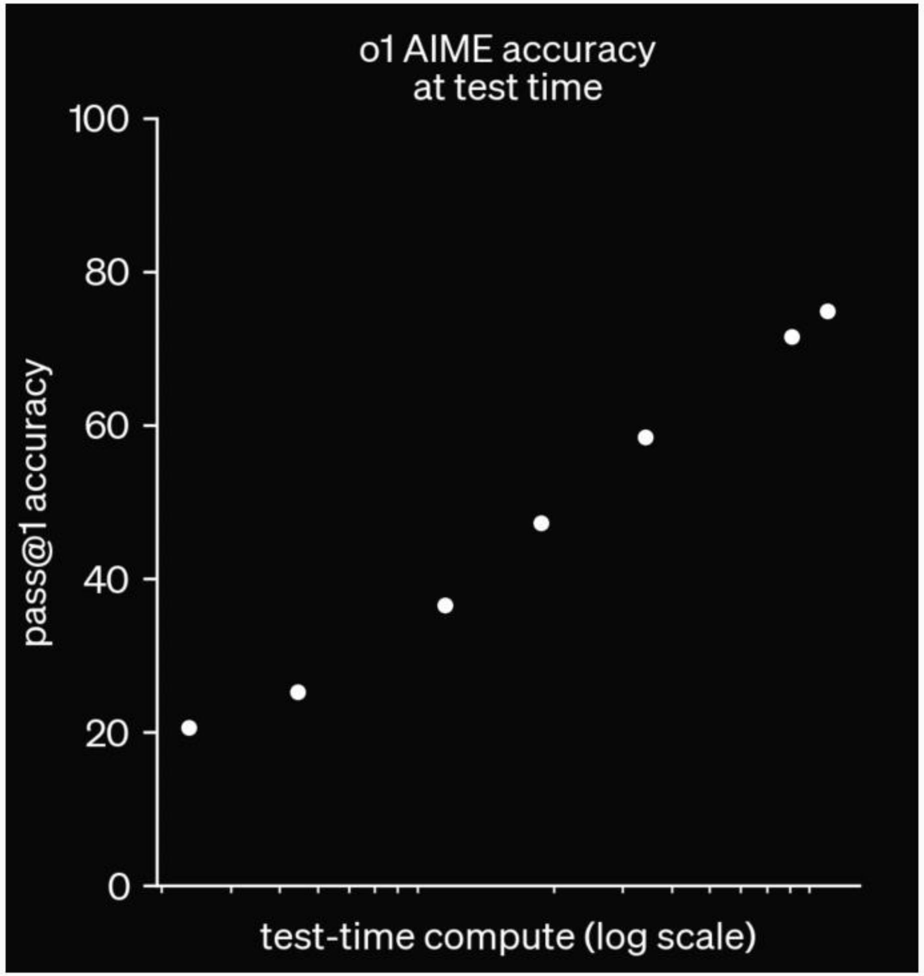
Models that use intermediate steps to answer questions oftentimes perform better than those which do not, especially on harder tasks

How can we get models to reason more in order to get to the right solution?

The State of LLM Reasoning Model Inference (Raschka, 2025)

What is Reasoning

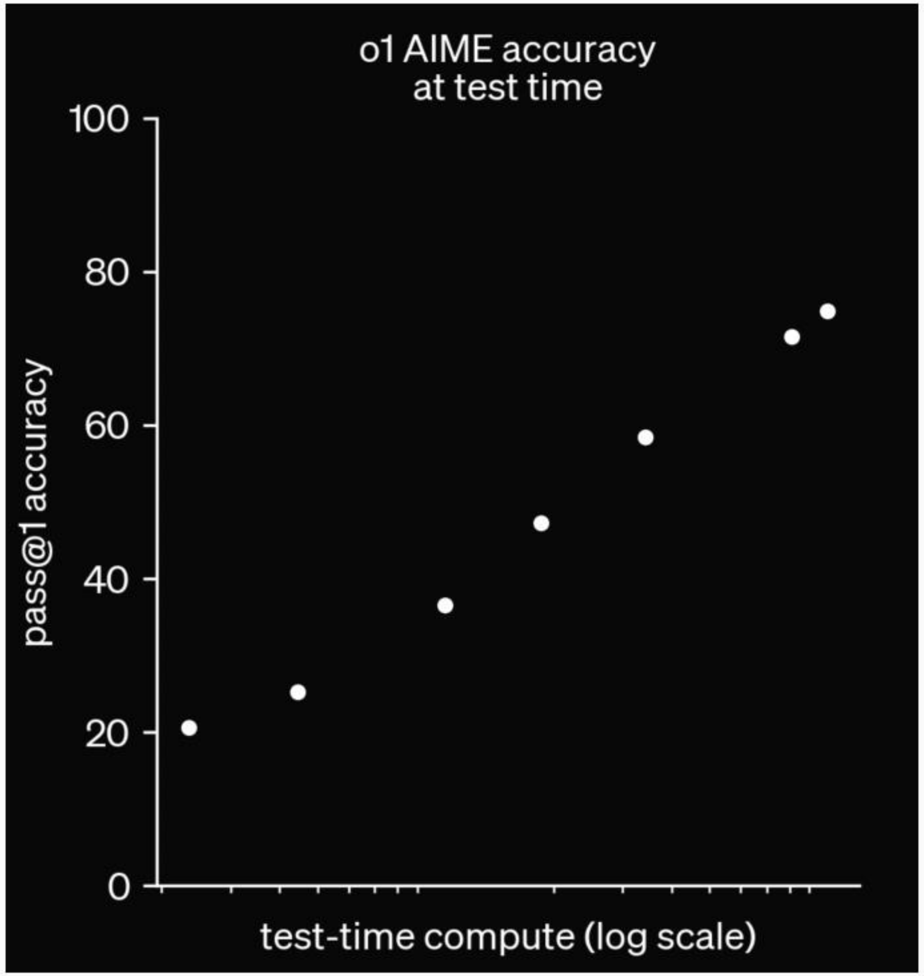
- In a more general sense, reasoning increases the number of generated tokens to improve model performance
- We examine several approaches that achieve significant performance improvement by increasing this 'reasoning' (i.e. increasing the number of tokens generated)



$$\text{Compute} \propto \text{Model size} \times \text{Generated tokens}$$

What is Reasoning

- In a more general sense, reasoning increases the number of generated tokens to improve model performance
- We examine several approaches that achieve significant performance improvement by increasing this ‘reasoning’ (i.e. increasing the number of tokens generated)



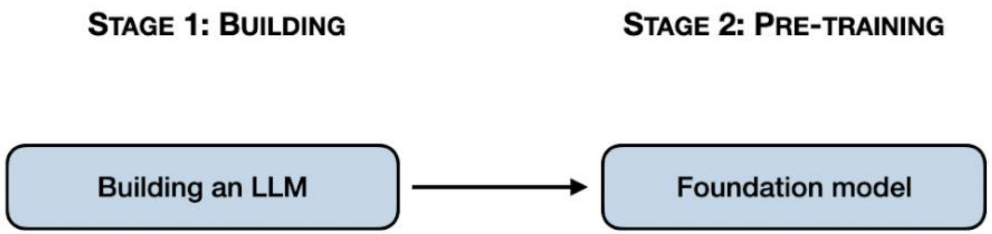
$$\text{Compute} \propto \text{Model size} \times \text{Generated tokens}$$

Different Kinds of Reasoning

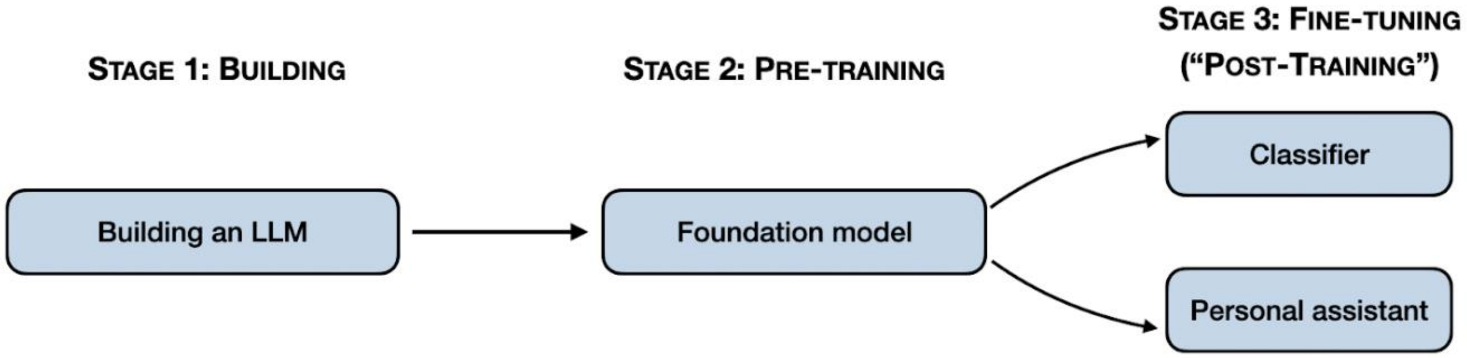
STAGE 1: BUILDING

Building an LLM

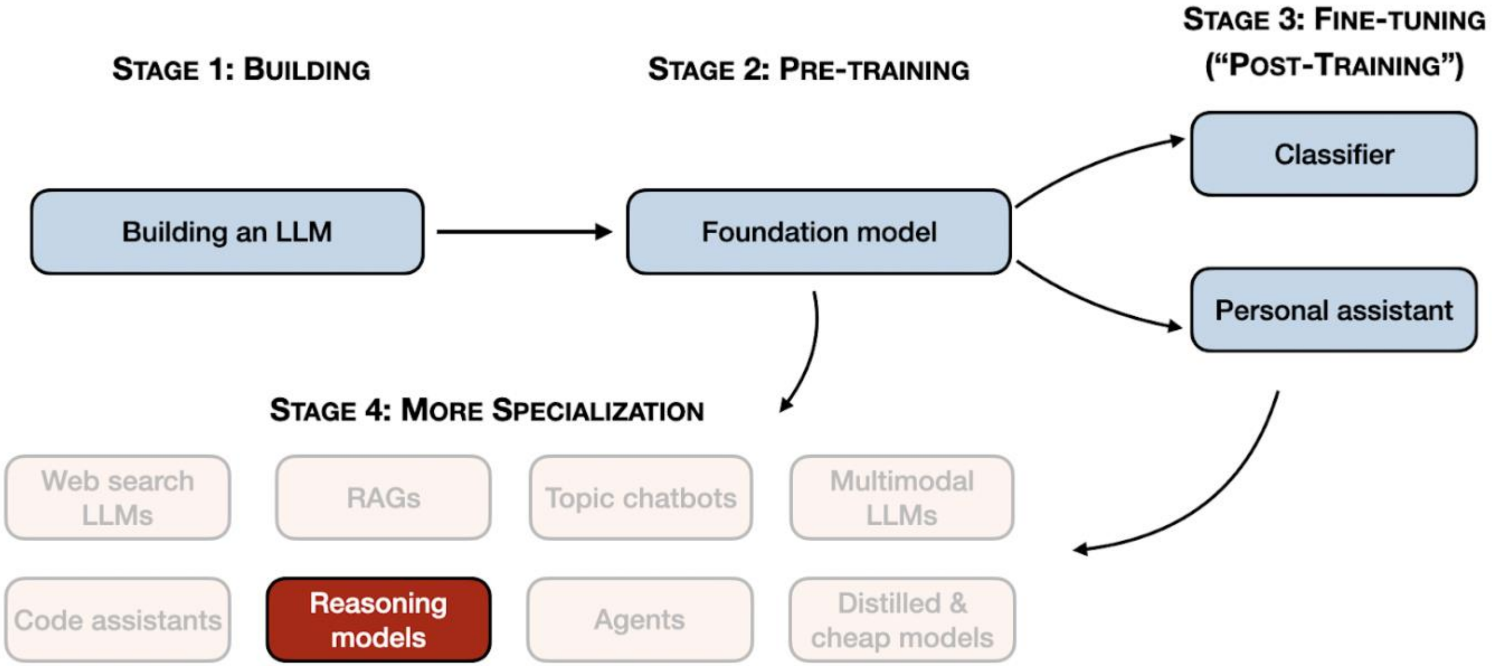
Different Kinds of Reasoning



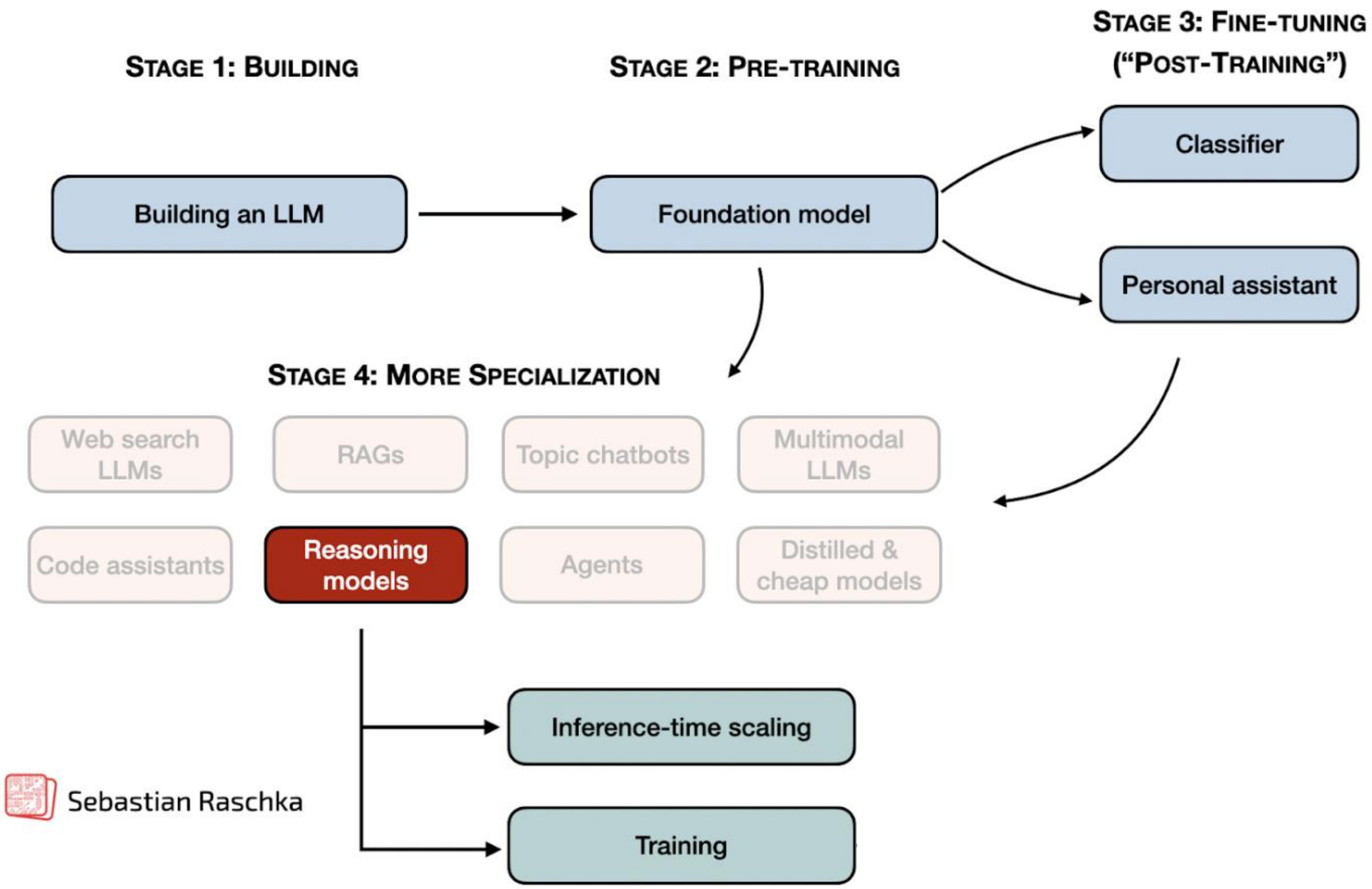
Different Kinds of Reasoning



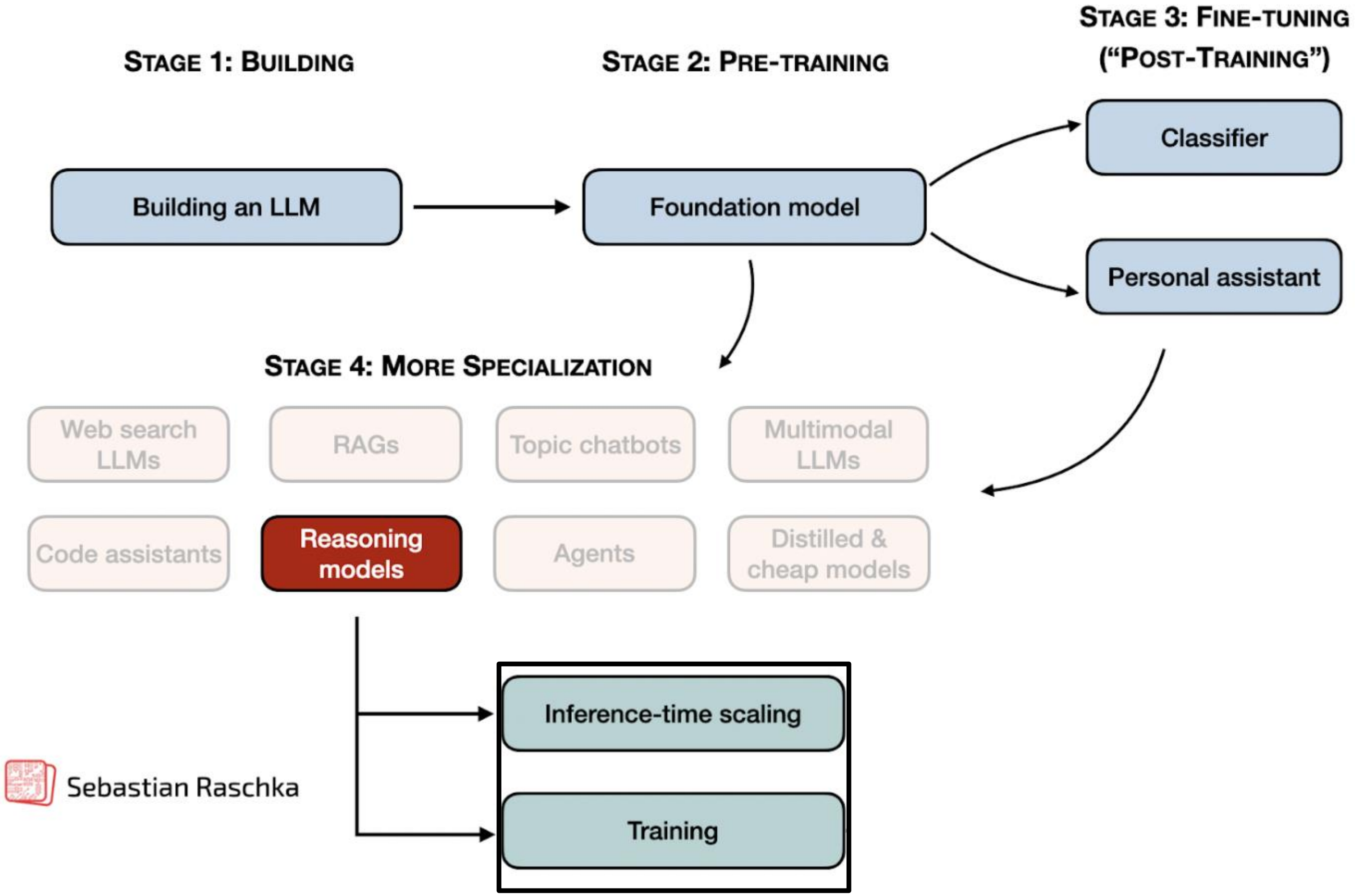
Different Kinds of Reasoning



Different Kinds of Reasoning



Different Kinds of Reasoning

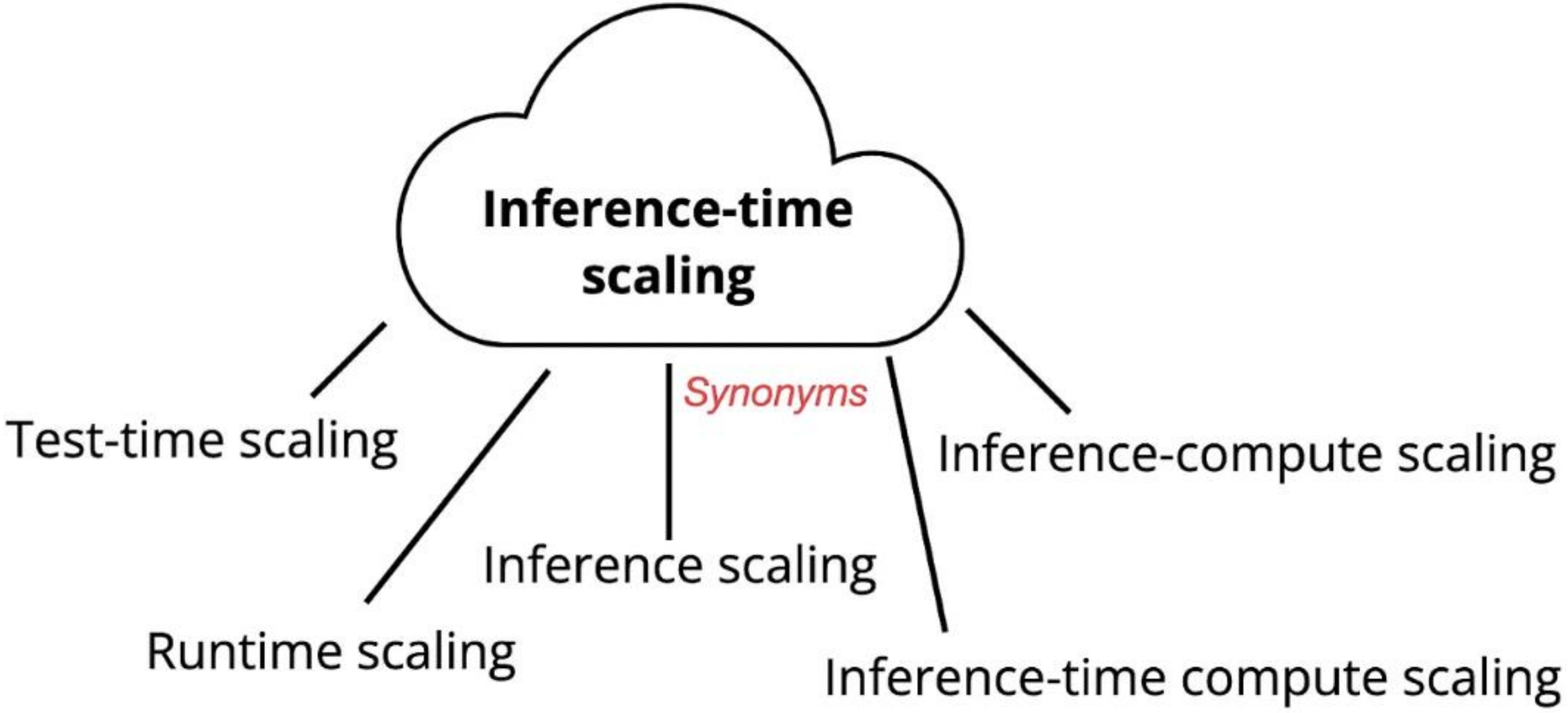


Different Kinds of Reasoning

- Test-time Scaling
  - Parallel
  - Tree-Search
  - Refinement
- Training Reasoners
  - Revisiting RLHF & PPO
  - From PPO to GRPO
  - RLHF to RLVR
  - Distillation from reasoning models
  - DeepSeek deepdive
- Latent Space Reasoning
  - Methods (Inner Thinking/Latent Reasoning/CoCoNut)

Test-Time Scaling

Test-Time Scaling



Test-Time Scaling (Prompting)

Regular prompting

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

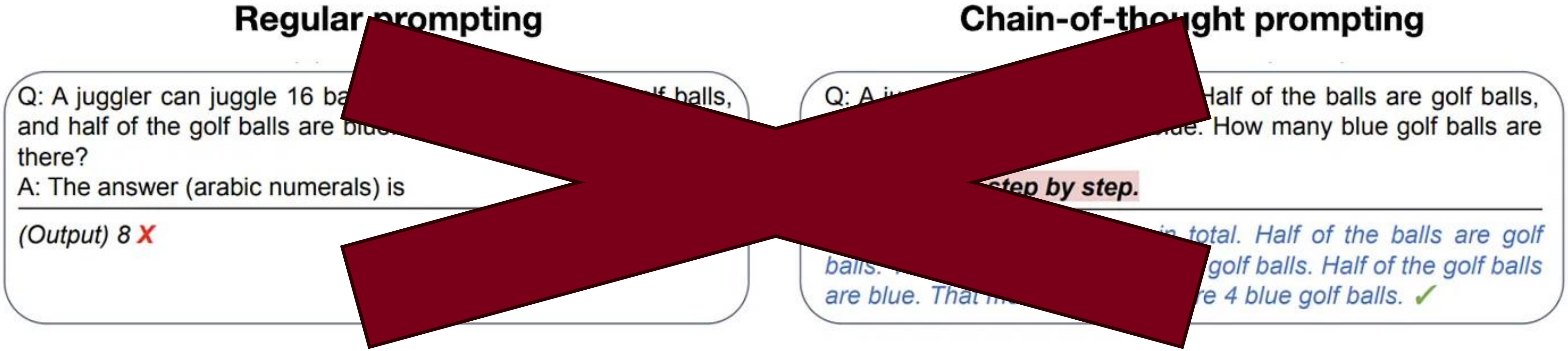
(Output) 8 ✗

Chain-of-thought prompting

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Test-Time Scaling (Prompting)

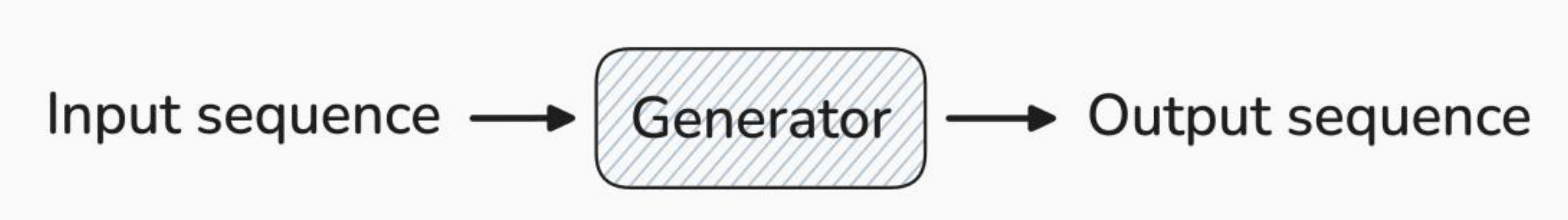


Good initial step for better results – but not as scalable. We have less control over increasing the number of generated tokens.

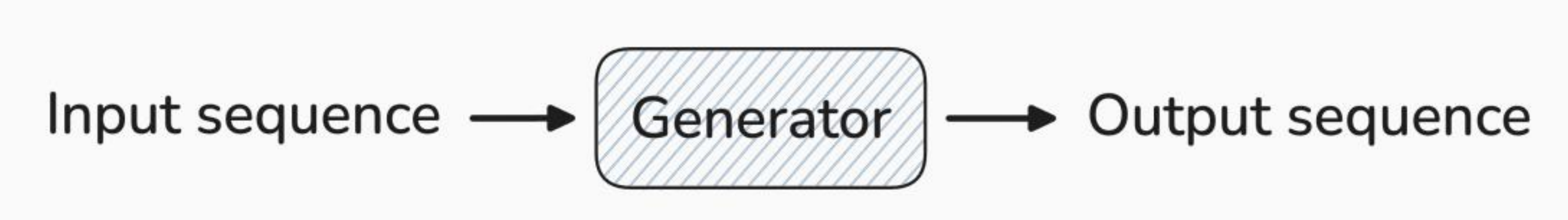
Test-Time Scaling Overview

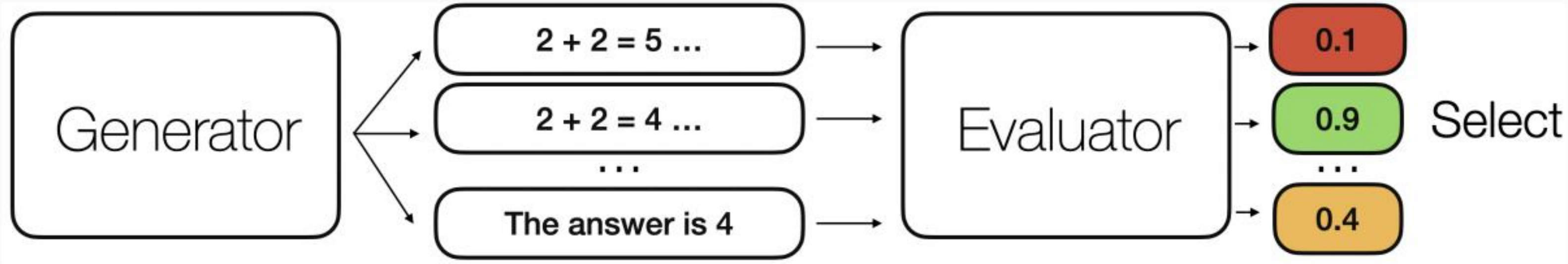
- Test-Time Scaling Approaches
  - Parallel
  - Tree Search
  - Refinement

Test-Time Scaling



Test-Time Scaling

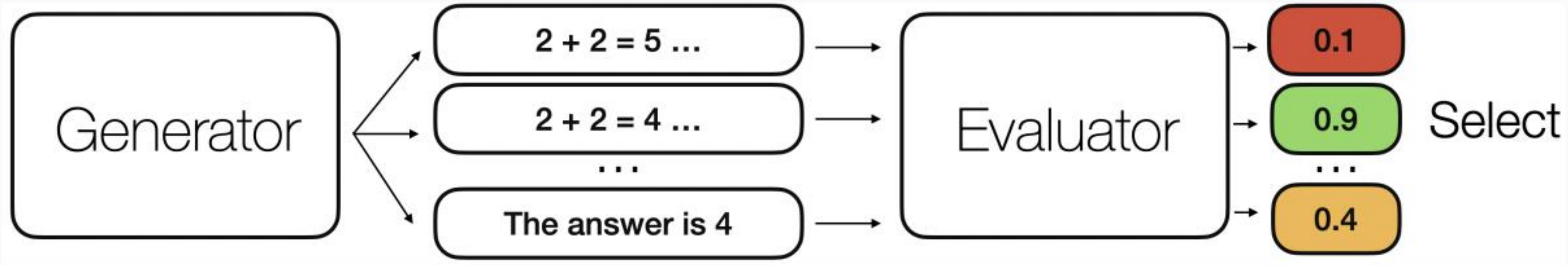




- Example: call API multiple times, select the best sequence with a separate model

Test-Time Scaling

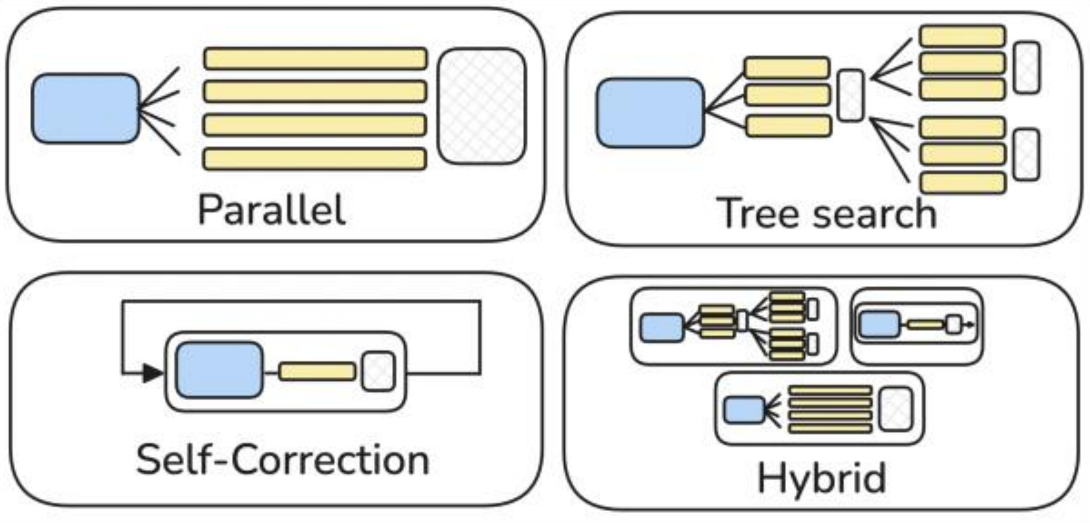
Meta-generator: Strategies for calling a generator multiple times



- Example: call API multiple times, select the best sequence with a separate model

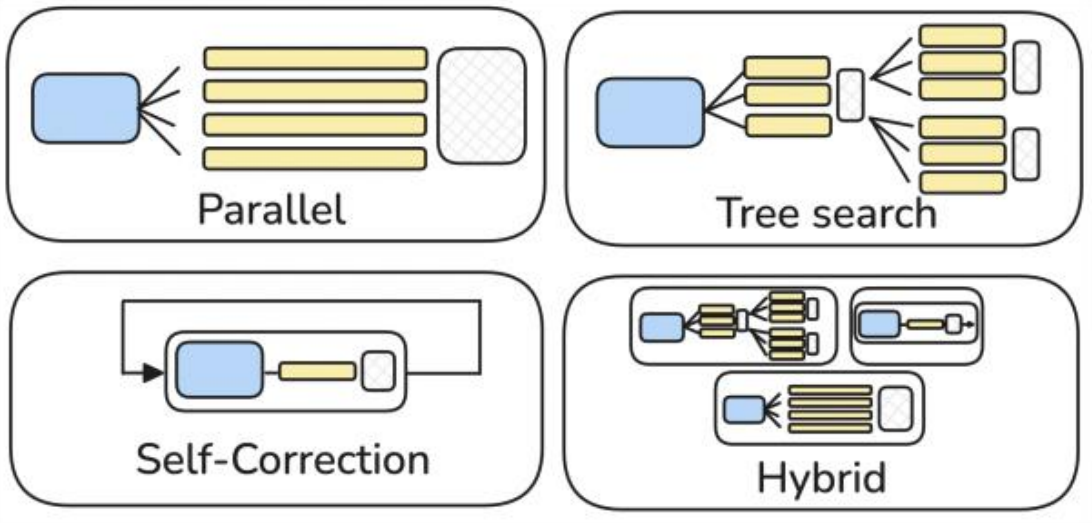
Test-Time Scaling Approaches

- Strategies
  - Parallel
  - Tree search
  - Refinement/self-correction

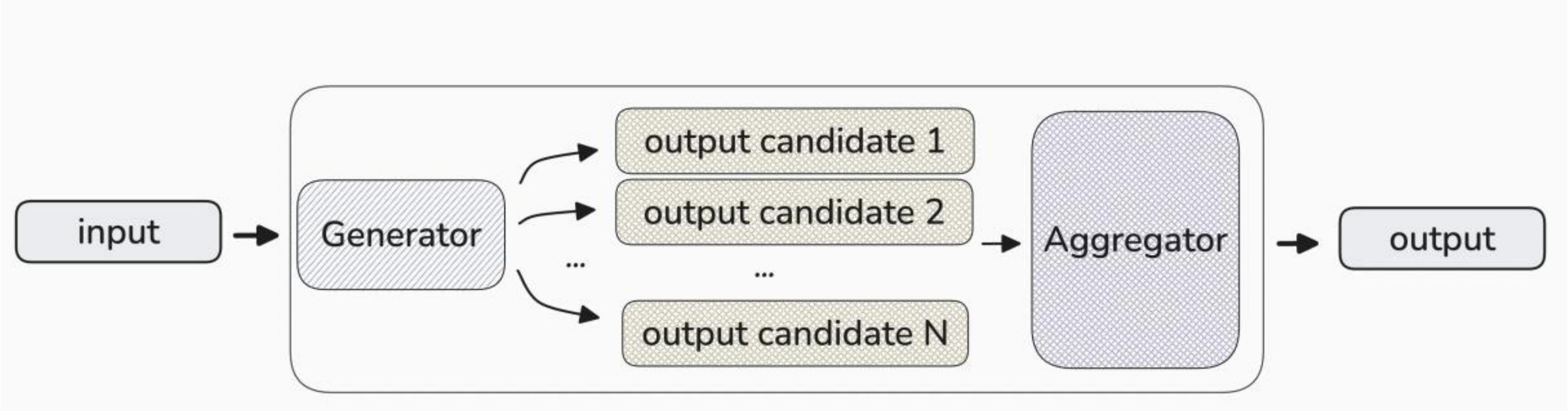


Test-Time Scaling Approaches

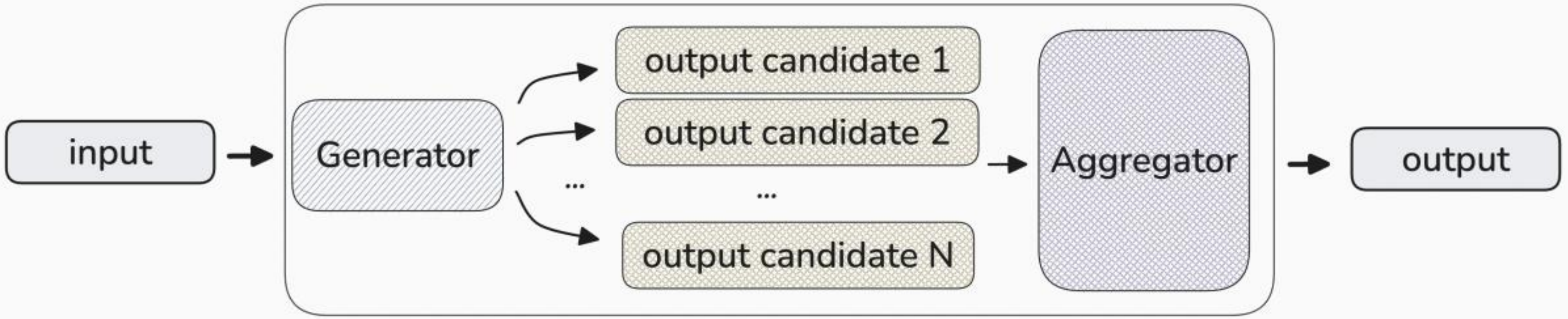
- Strategies
  - Parallel
  - Tree search
  - Refinement/self-correction



Parallel Generation



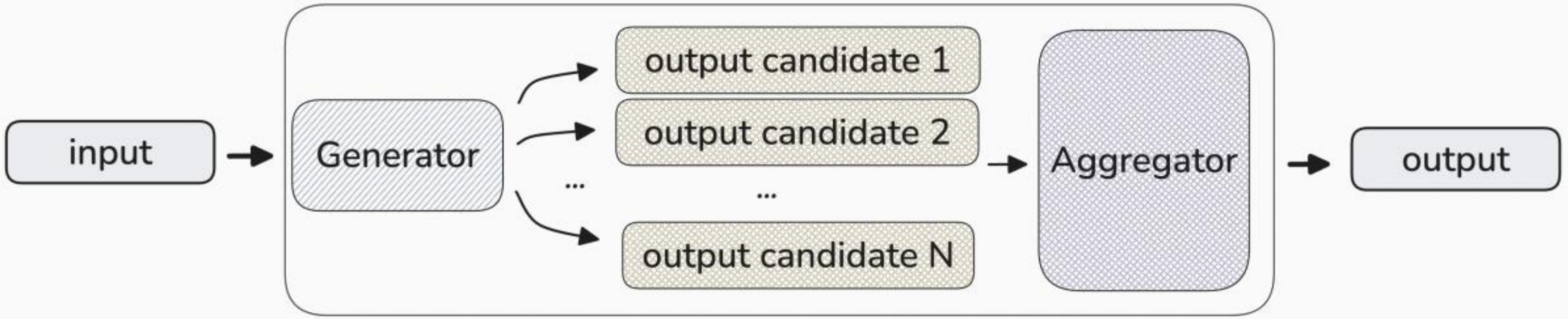
Parallel Generation



- Generate candidates:

$$\{y^{(1)}, \ldots, y^{(N)}\} \sim G(\cdot|x)$$

Parallel Generation



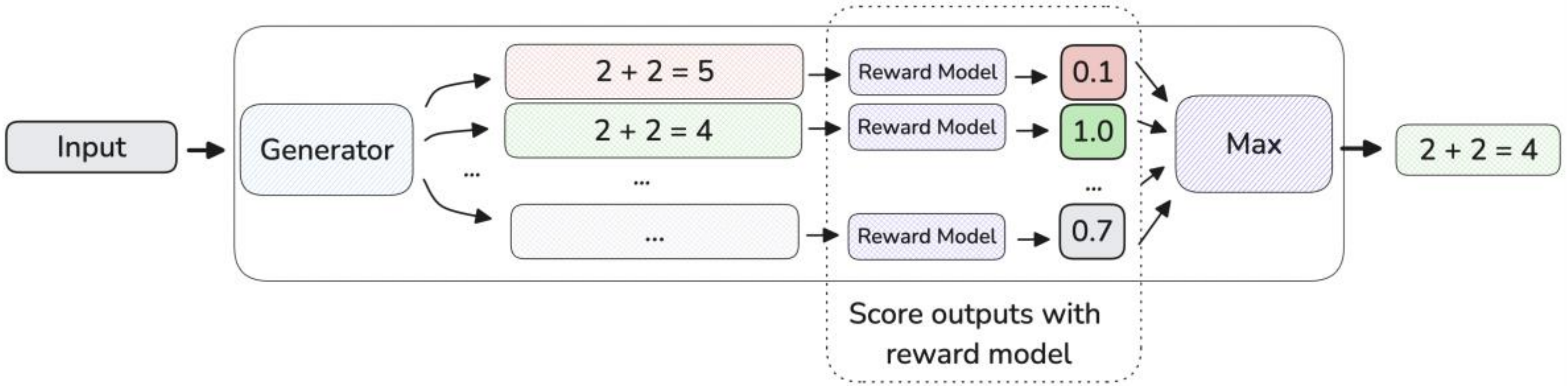
- Generate candidates:

$$\{y^{(1)}, \ldots, y^{(N)}\} \sim G(\cdot|x)$$

- Aggregate:

$$y = h(y^{(1)}, \ldots, y^{(N)})$$

Parallel Generation (Best-of-N)

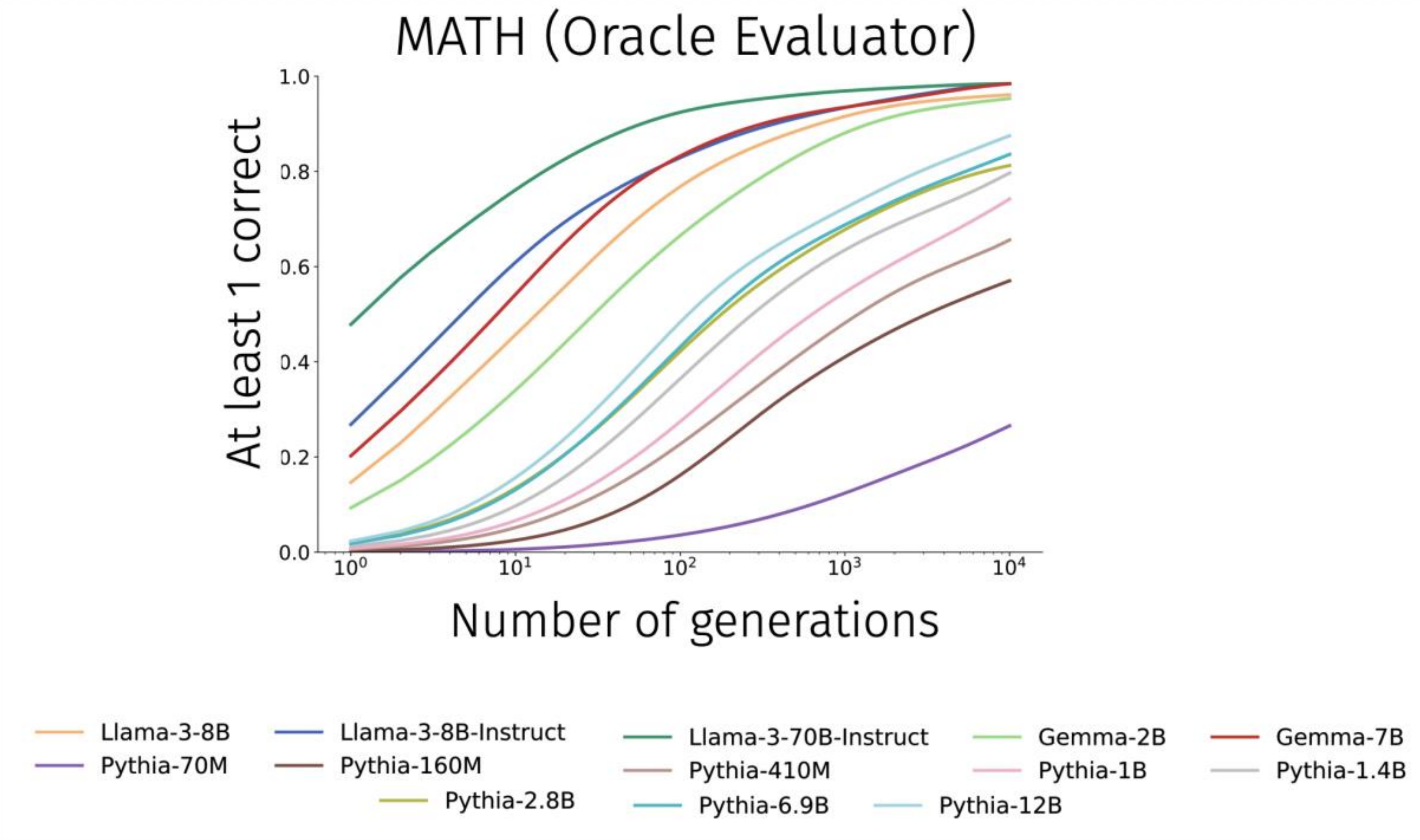


$$\underset{\{y^{(1)},\ldots,y^{(N)}\}}{\arg\max} \underbrace{v(y)}_{\text{reward model}}$$

Stiennon, et. al (2020), Nakano et. al (2022)

Parallel Generation (Best-of-N)

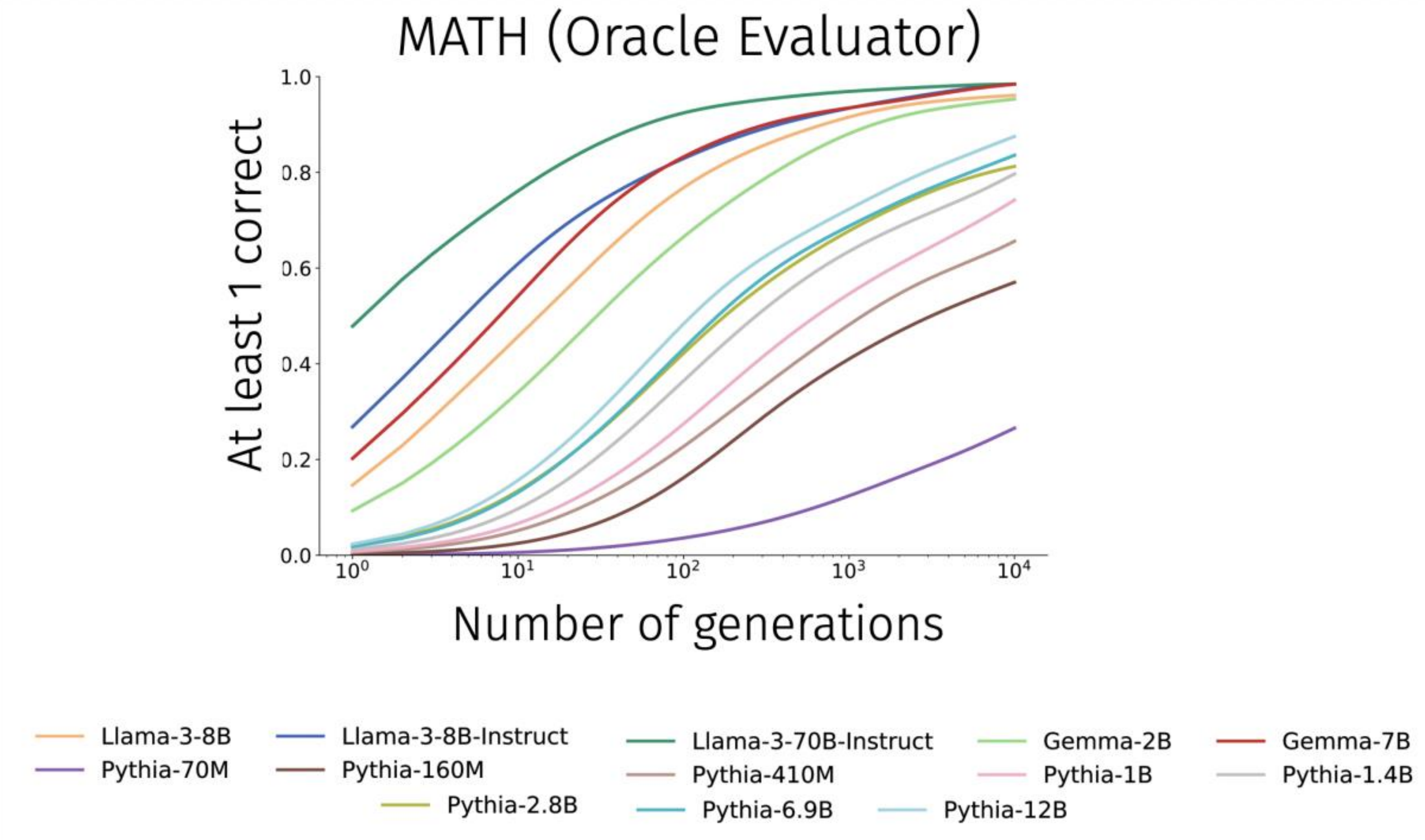
What if we had a perfect reward model $v^*(y)$?



Parallel Generation (Best-of-N)

Q: What is a 'perfect reward model'?

What if we had a perfect reward model $v^*(y)$?

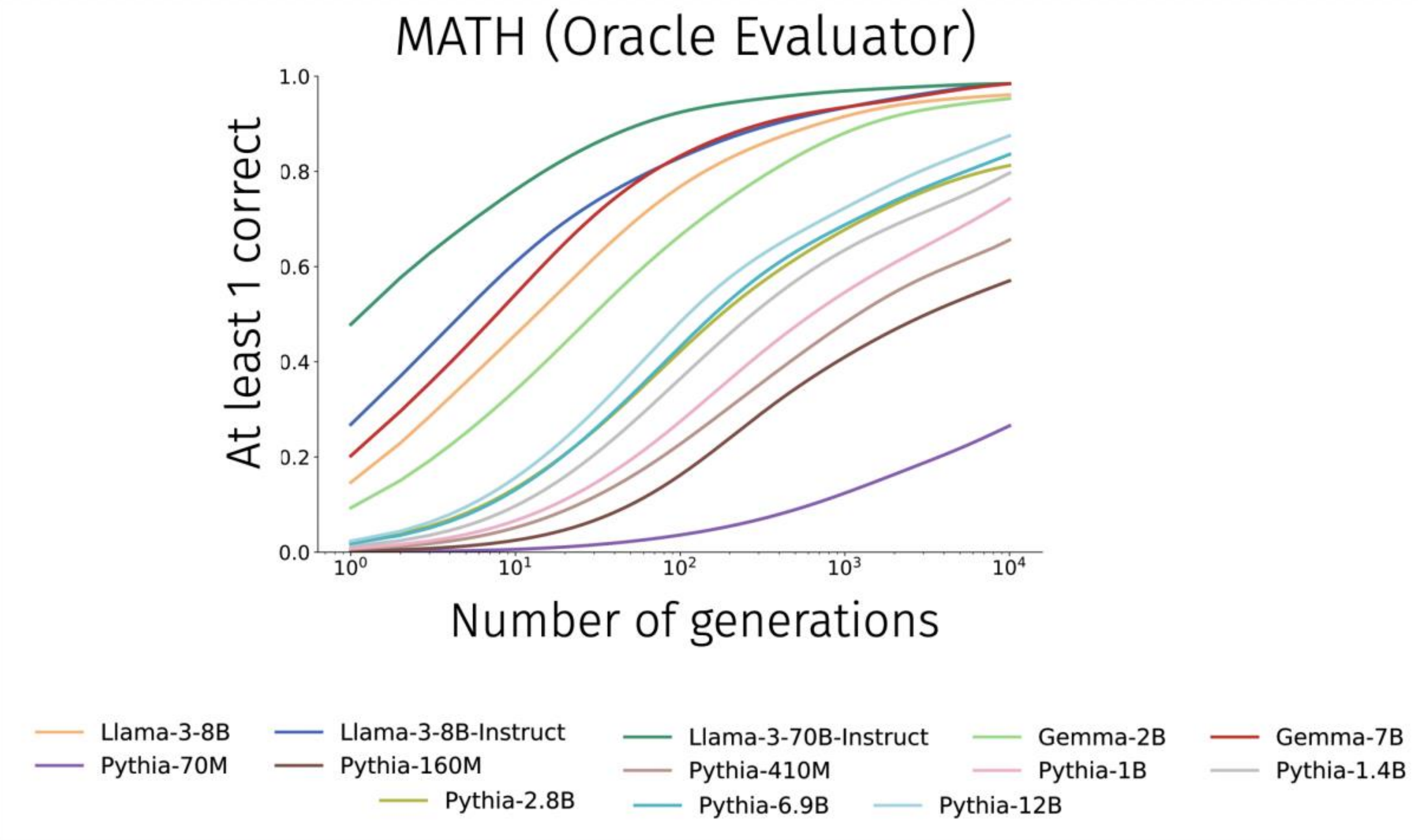


Parallel Generation (Best-of-N)

Q: What is a 'perfect reward model'?

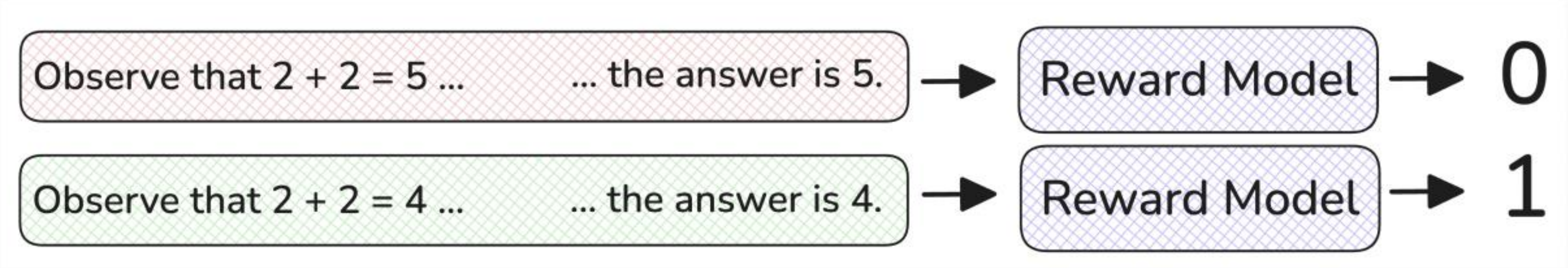
A: A 'perfect reward model' allows us to always select the correct answer if it is given.

What if we had a perfect reward model $v^*(y)$?



Best of N: Finding a reward model in practice

Learned reward model $v(y) \to [0, 1] \approx R(y)$:

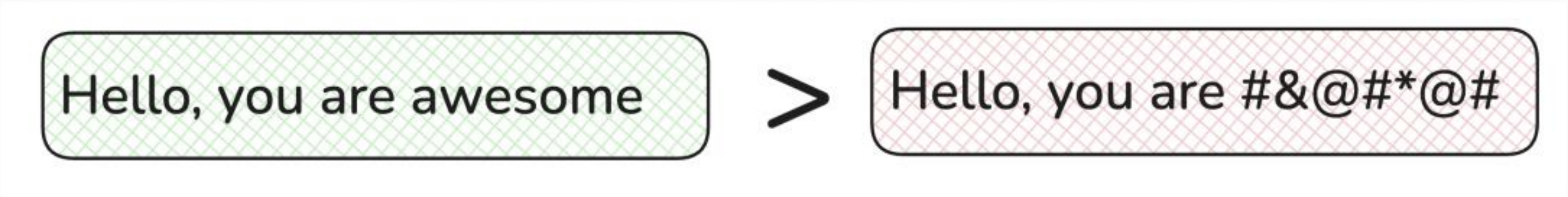


Train reward model with correct and incorrect examples.[2]

Cobbe et. al (2021)

Best of N: Finding a reward model in practice

Learned reward model $v(y) \rightarrow [0, 1] \approx R(y)$:



Train reward model with preference data.[2]

Stiennon et. al (2020)

Why Best of N?

Why Best-of-N?

- Approximates maximum (true) reward:

$$\text{Best-of-}N = \underset{y \in \{y^{(1)},\dots,y^{(N)}\}}{\arg\max}\; v(y)$$

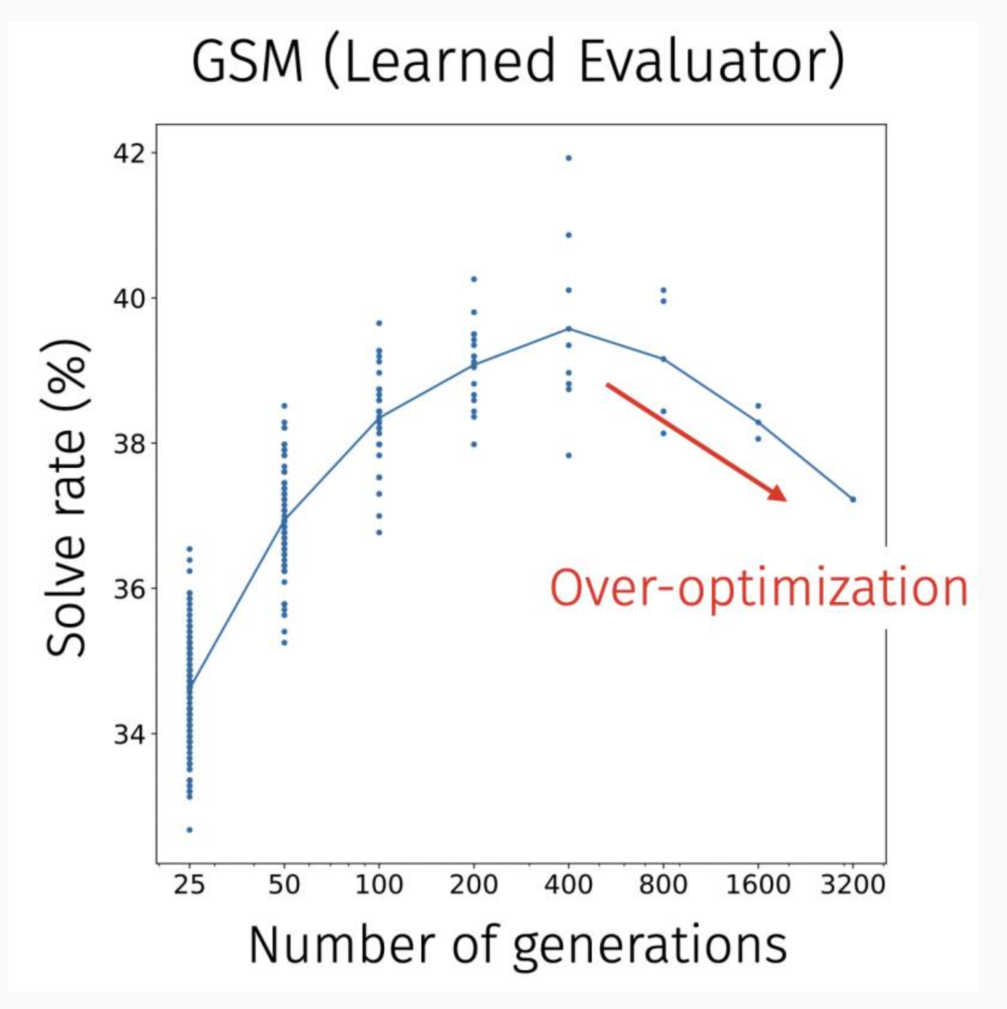
$$\approx \underset{y}{\arg\max}\, v(y) \qquad (1)$$

$$\approx \underset{y}{\arg\max}\, R(y) \qquad (2)$$

(1) gets better as number of generations N increases!

(2) Suffers from imperfect reward model, aka "over-optimization"

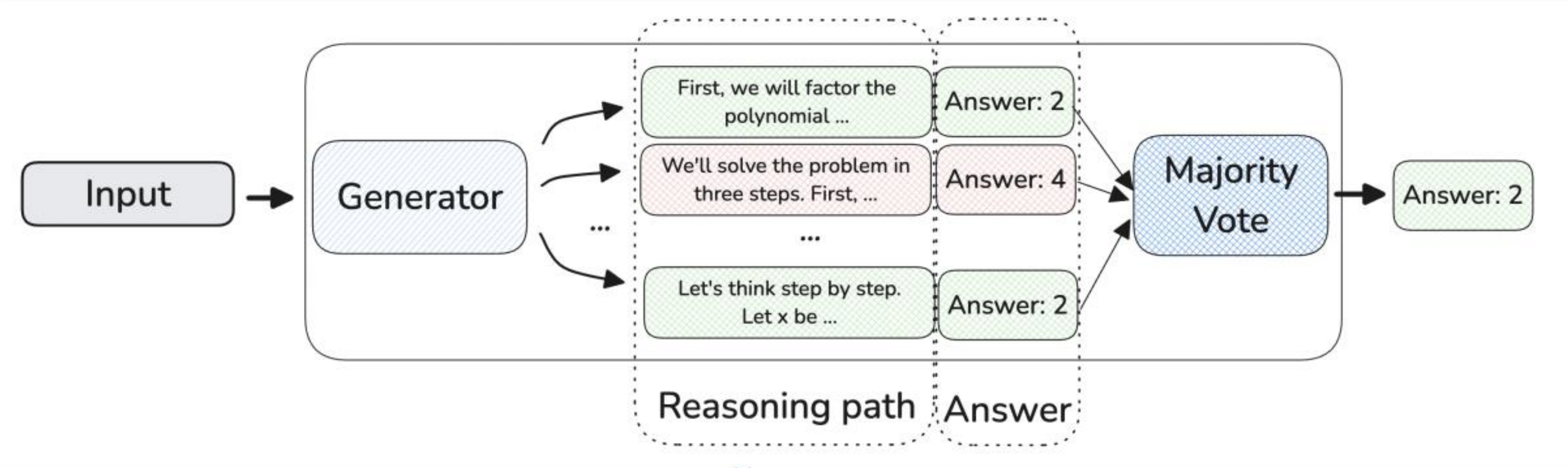
Best of N (example)



Cobbe et. al (2020)

Parallel Generation (Voting)

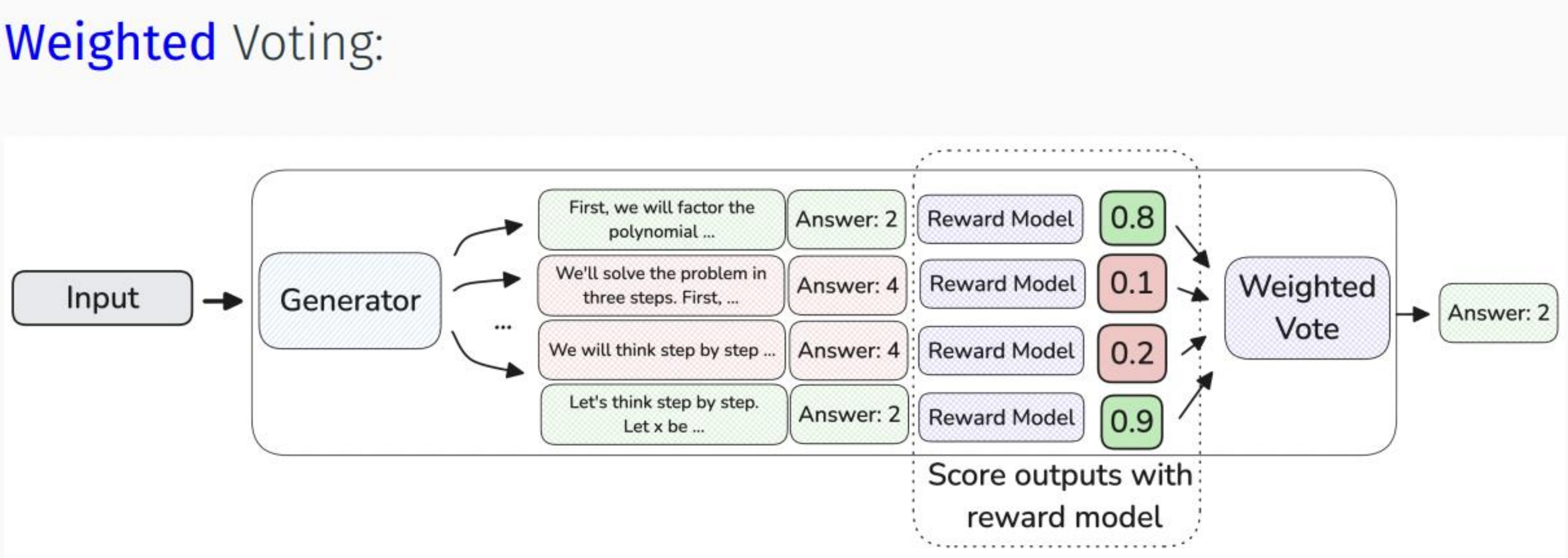
Voting aggregation:[4]



$$\arg\max_{a} \sum_{i=1}^{N} 1\{y^{(i)} = a\},$$

Wang et. al (2023)

Parallel Generation (Weighted Voting)

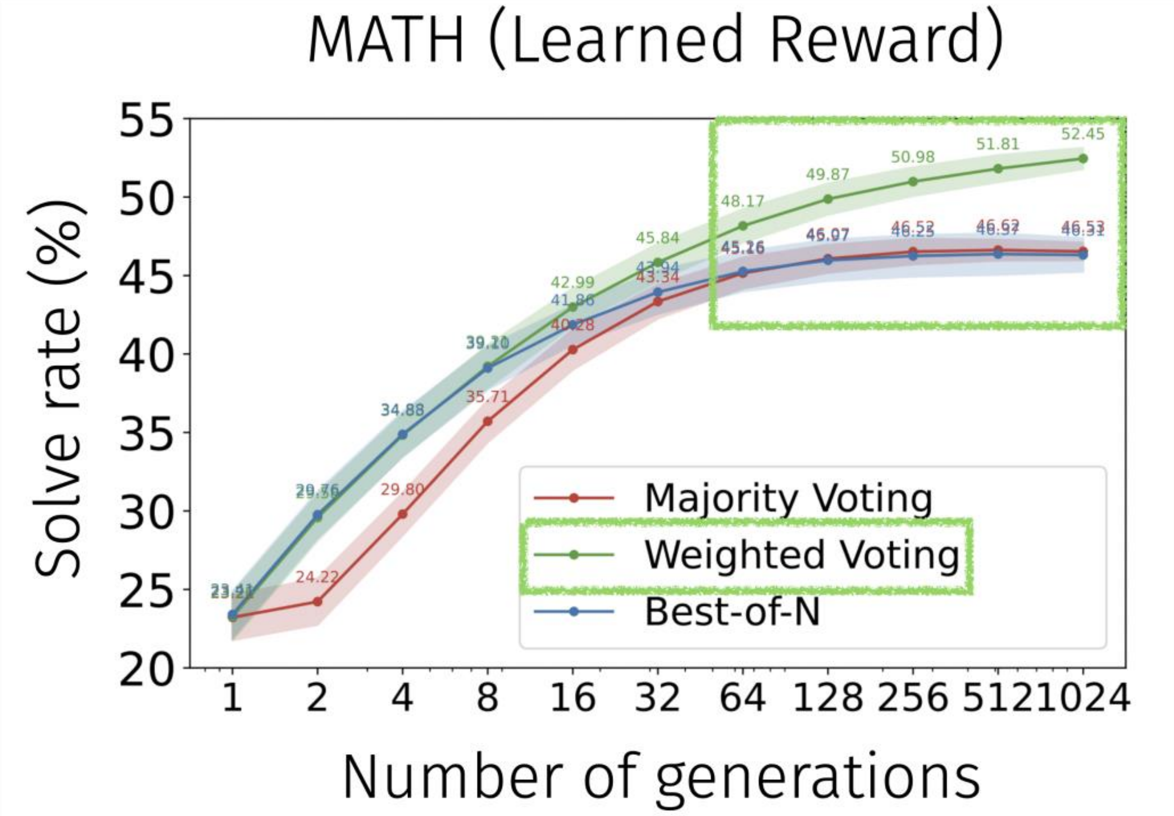


$$\arg\max_{a} \sum_{i=1}^{N} \underbrace{v(y^{(i)})}_{\text{reward model}} \cdot \mathbb{1}\{y^{(i)} = a\},$$

Li et. al (2023)

Parallel Generation (Weighted Voting)

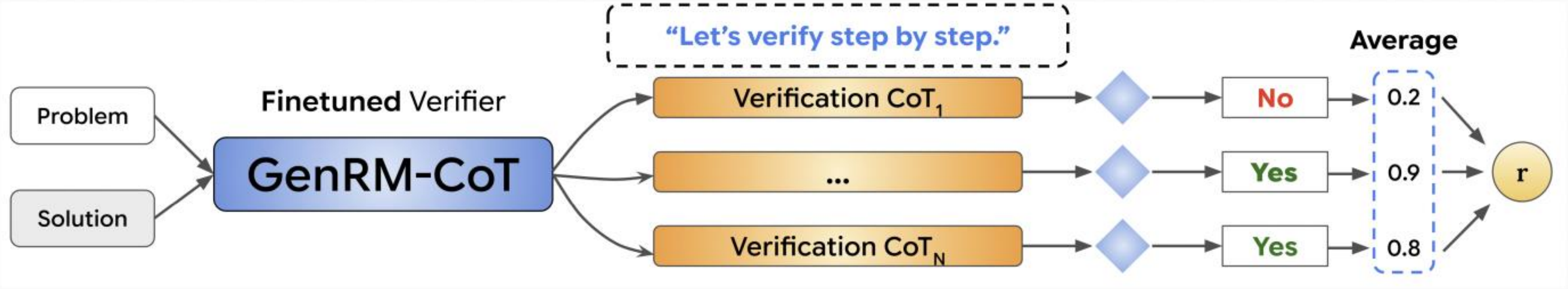
Can outperform Best-of-*N*, e.g.:[6]



[Sun et al., 2024] Easy-to-Hard Generalization: Scalable Alignment Beyond Human Supervision.

Parallel Generation

Improve the reward model:



Parallel generation *in the reward model too*[8]

Active area of research!

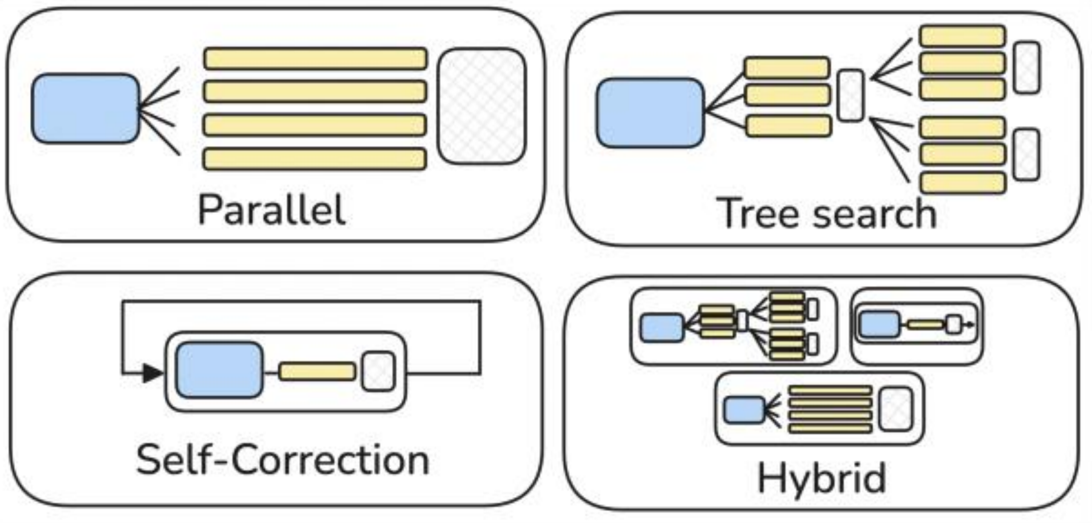
Zhang et. al (2024)

Parallel Generation

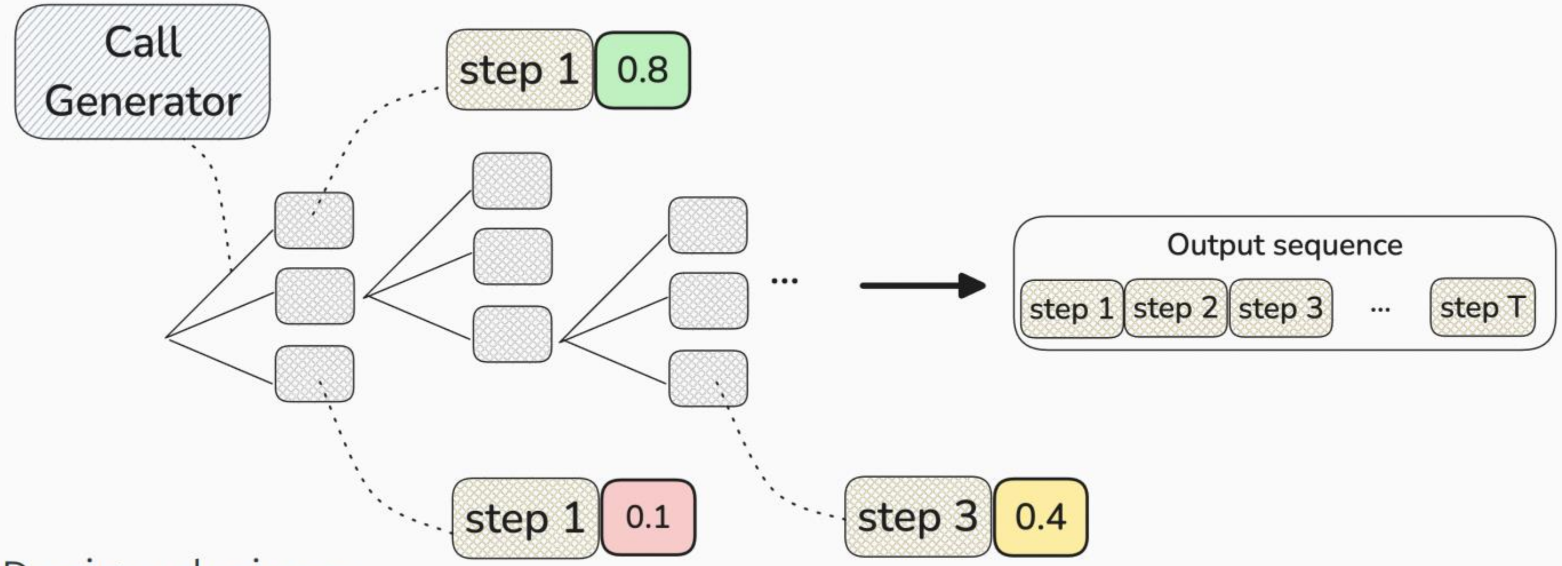
- Parallel
  - Explores output space by generating full sequences
  - Large performance gains in practice
  - Bounded by the quality of the evaluator and generator
- Insight: only uses the verifier at the end (full sequence outputs)
  - Next: How can intermediate evaluation improve on this approach

Test-Time Scaling Approaches

- Strategies
  - Parallel
  - Tree search
  - Refinement/self-correction



Tree Search

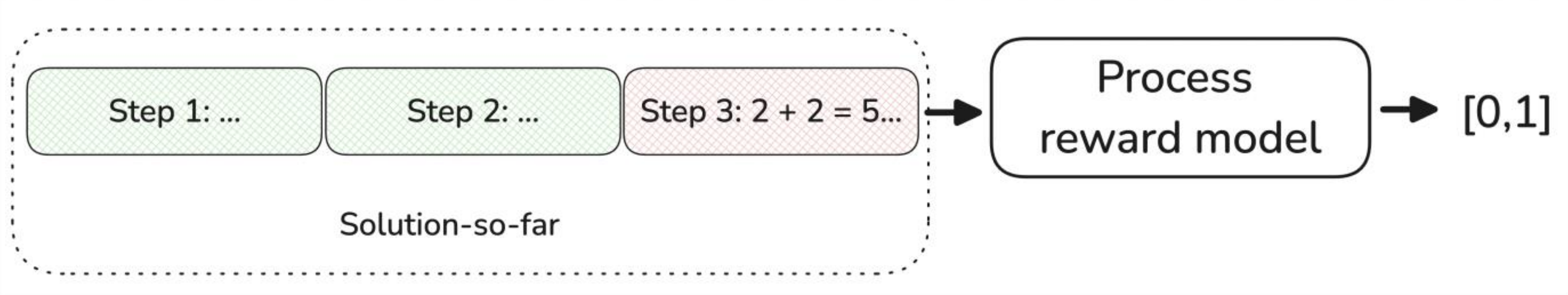


Design choices:

- States s
- Transitions $s \to s'$
- Scores $v(s)$
- Strategy (breadth-first, depth-first, ...)

Tree Search

1. Scores: “process reward model (PRM)”[9]

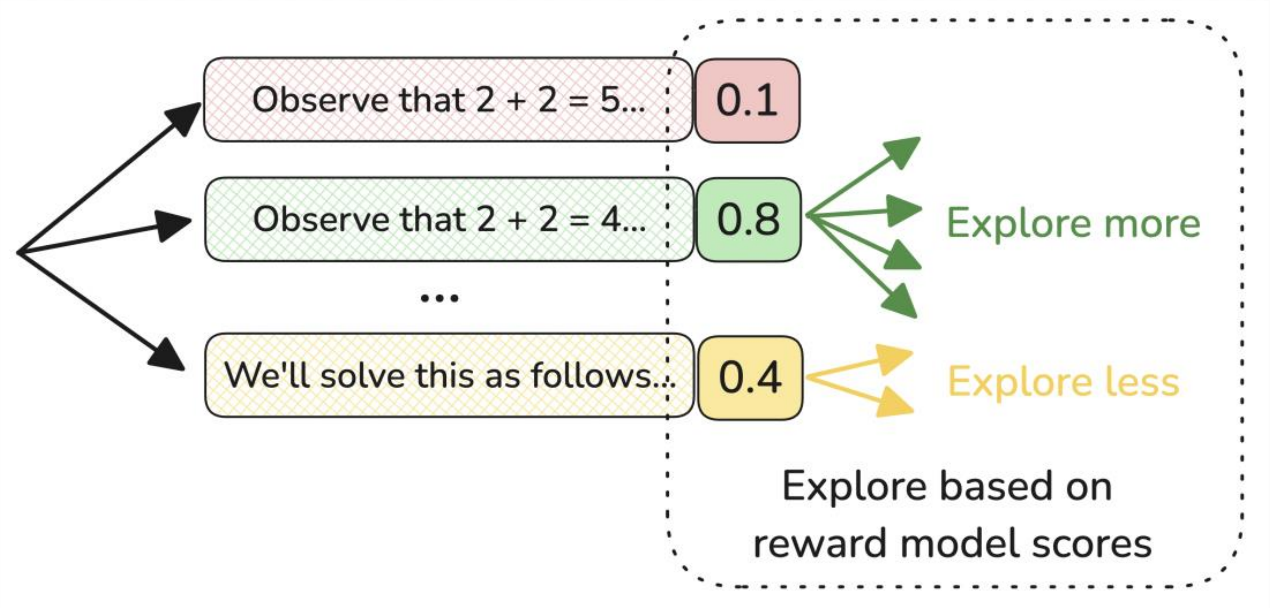


$$v(x, s_1, s_2, \ldots, s_t) \rightarrow [0, 1]$$

[Uesato et al., 2022, Lightman et al., 2024, Wang et al., 2024a]

Tree Search

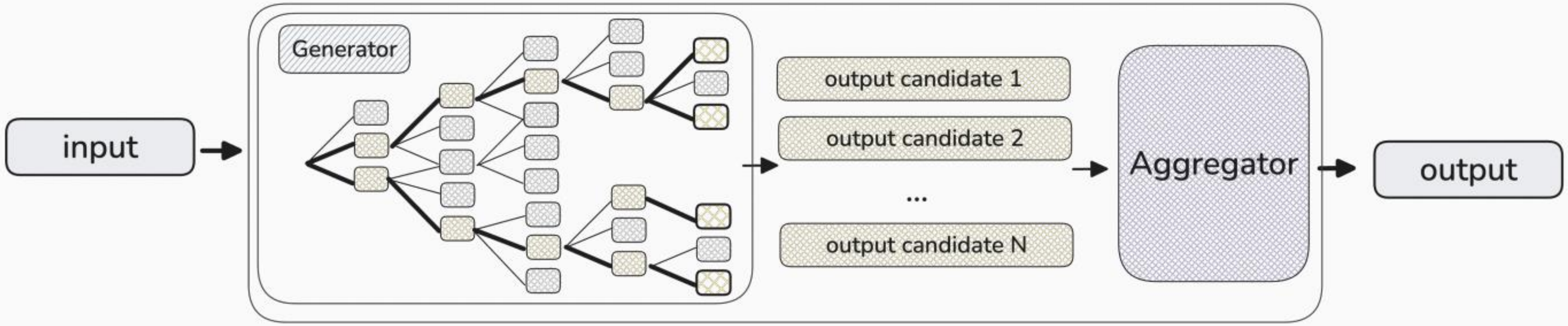
2. *Reward Balanced Search (Rebase)*[10]



$$\text{explore}_i = \text{Round}\left(\text{Budget}\frac{\exp\left(v(s_i)/\tau\right)}{\sum_j \exp\left(v(s_j)/\tau\right)}\right), \qquad (3)$$

[Wu et al., 2024b] Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference.

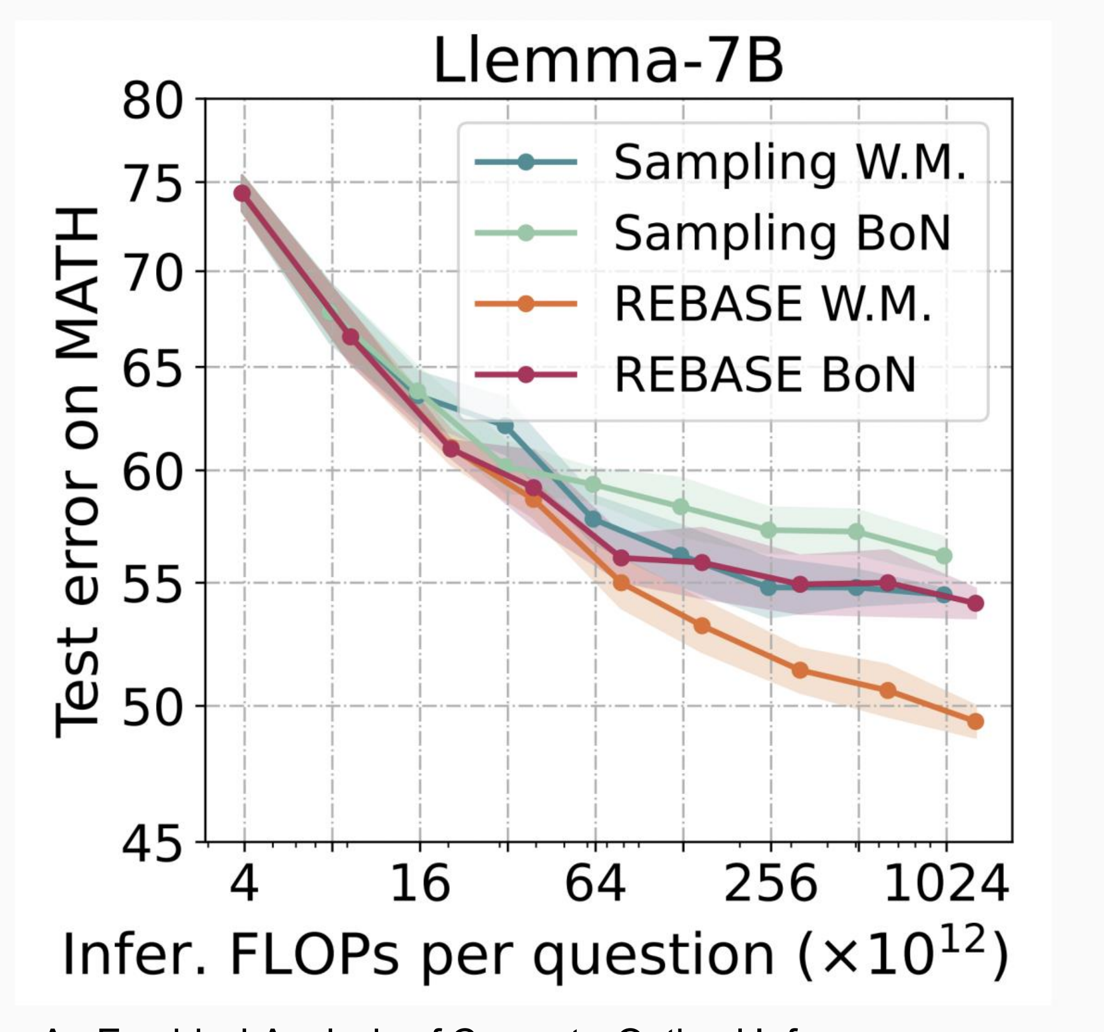
Tree Search



Run tree search to get candidates for aggregation (e.g., voting).

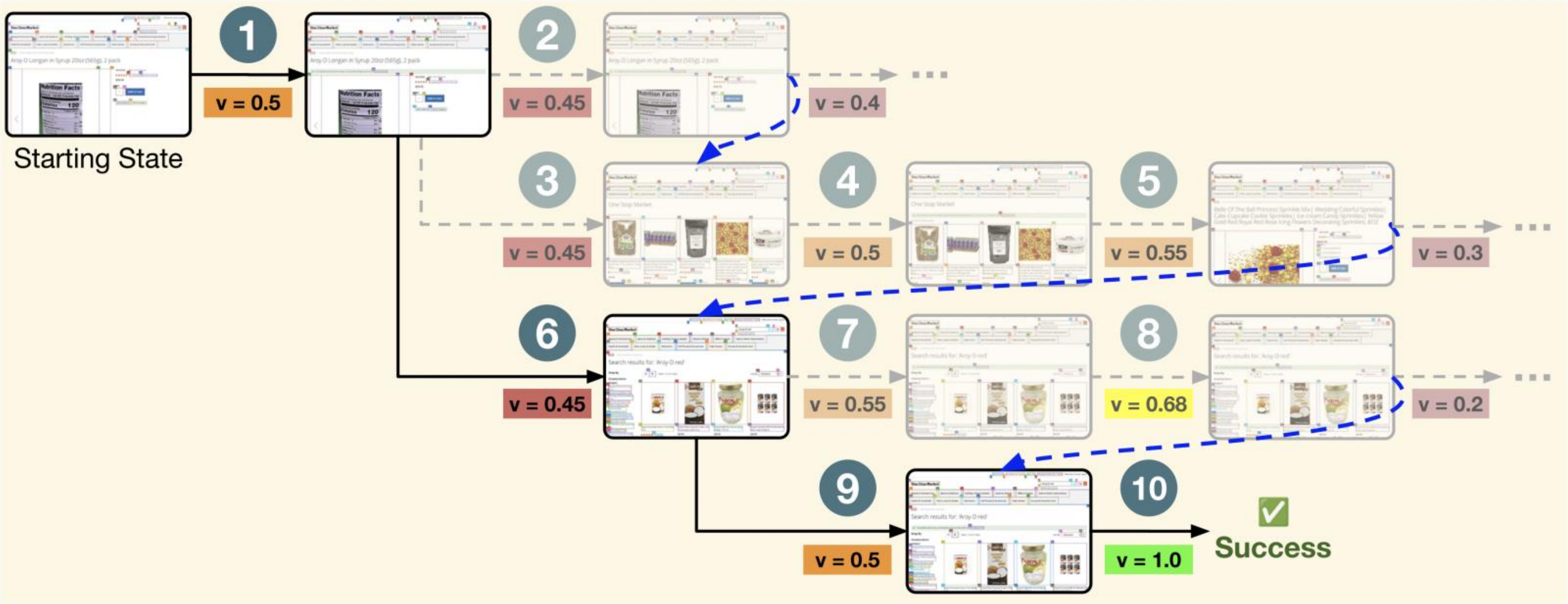
- **Key idea:** Leverages scores on *intermediate* states
  - Backtracking
  - Exploration

Tree Search



[Wu et al., 2024b] Inference Scaling Laws: An Empirical Analysis of Compute-Optimal Inference.

Tree Search (Example)



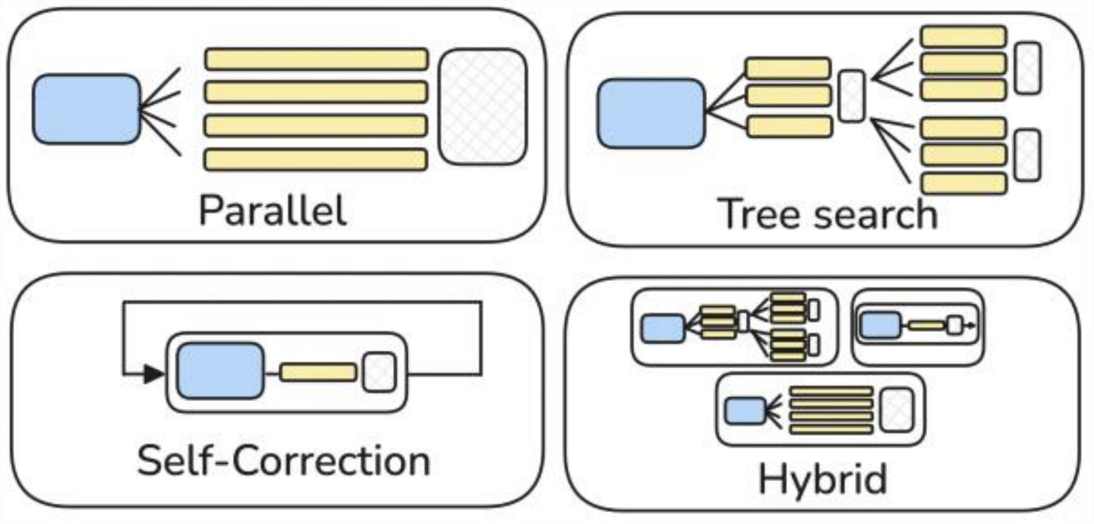
Best-first search in web agents [Koh et al., 2024]

Tree Search

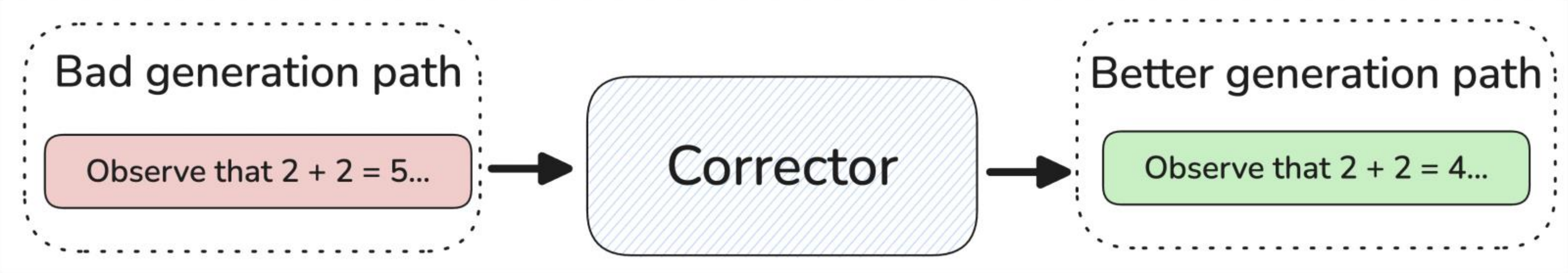
- Can backtrack and explore using intermediate scores
- Requires a suitable environment and value function
  - Decomposition into states
  - Good reward signal

Refinement/self-correction

- Strategies
  - Parallel
  - Tree search
  - Refinement/self-correction



Refinement/self-correction

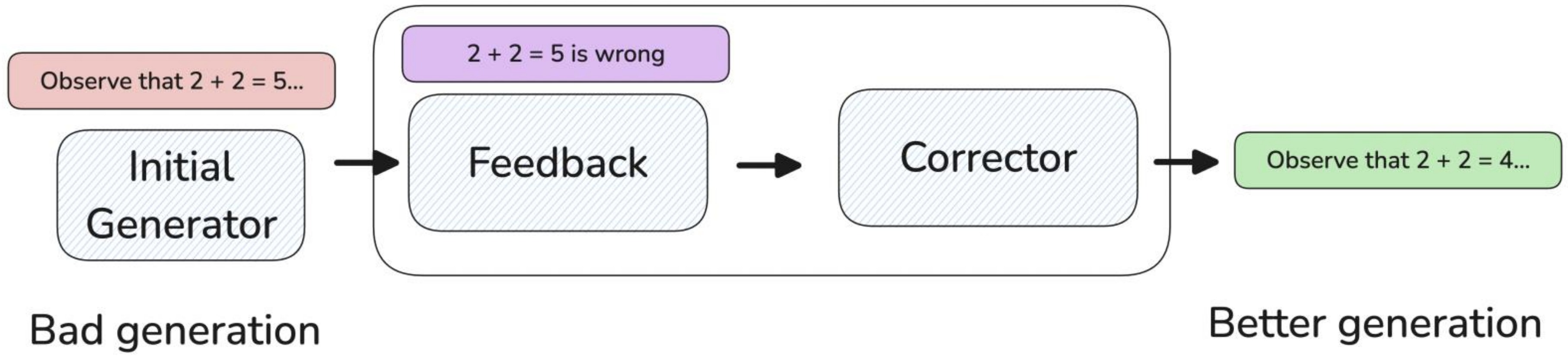


Improve a generation

Repeat:

- $y^{(i+1)} \sim g(x, y^{(i)})$

Refinement/self-correction



Improve a generation using feedback

Repeat:

- $y^{(i+1)} \sim g(x, y^{(i)}, F(y^{(i)}))$

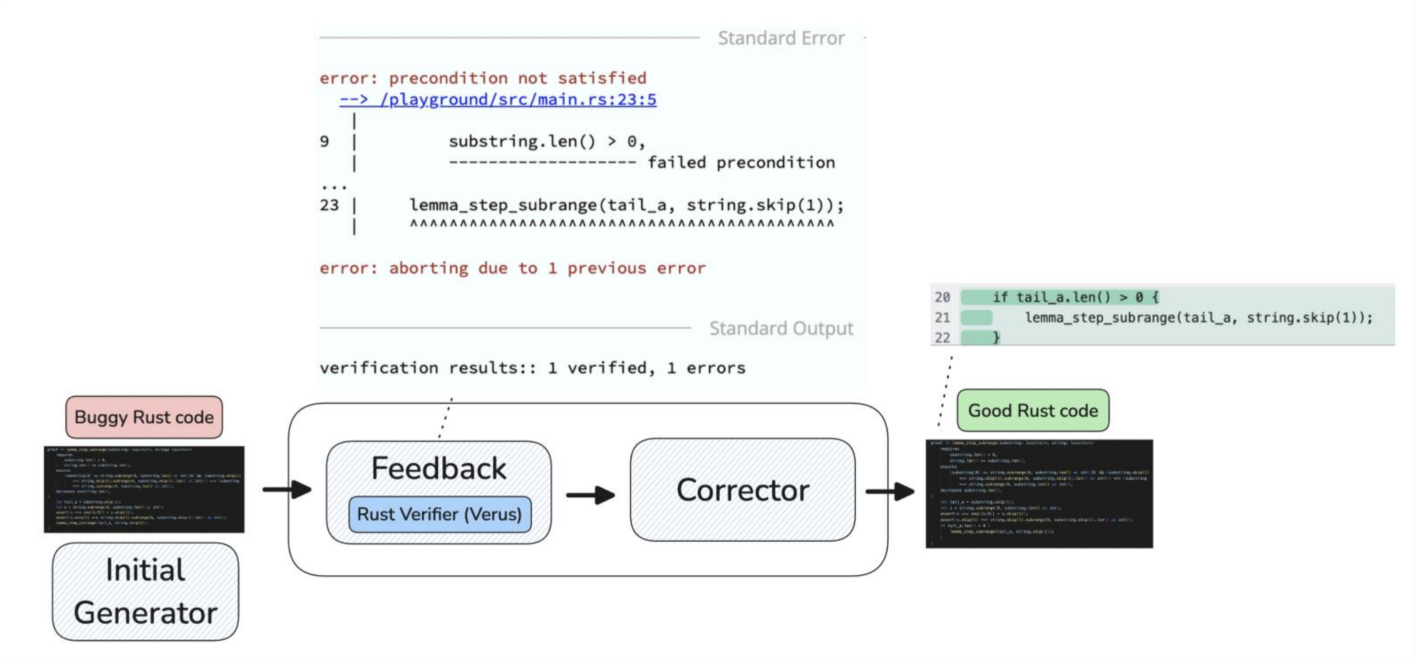
Refinement/self-correction

In practice, the quality and source of feedback is crucial:

- ❑ Extrinsic: external information at inference time
- ❑ Intrinsic: no external information at inference time

Refinement (Extrinsic)

1. **Extrinsic**: external feedback

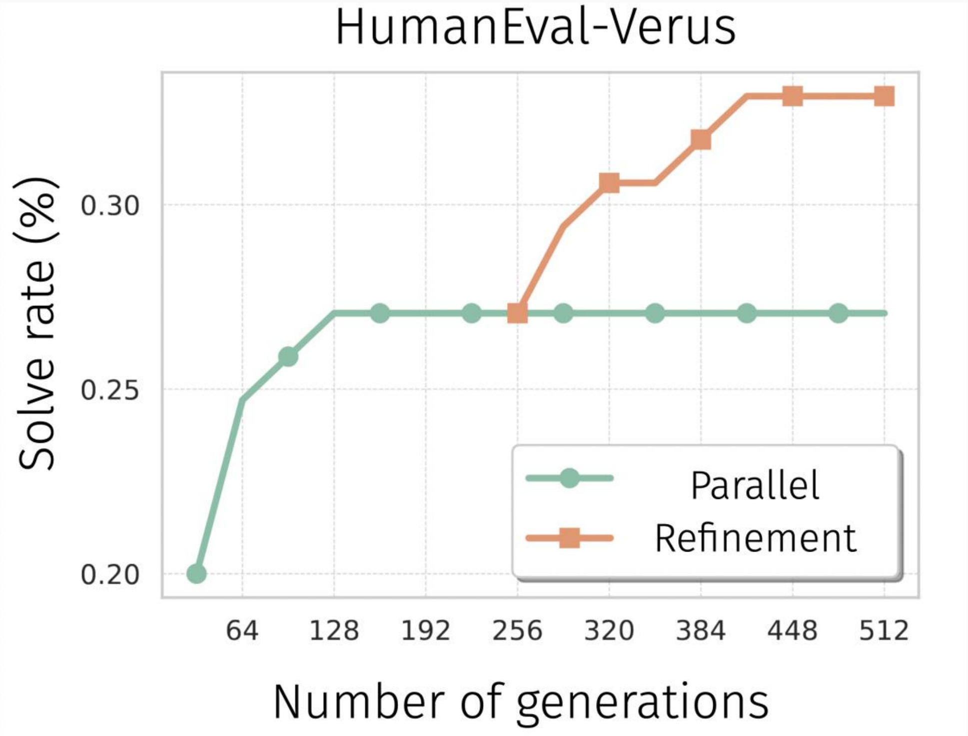


Feedback: external program verifier[12]

[12] [Aggarwal et al., 2024], *AlphaVerus*. P. Aggarwal, B. Parno, S. Welleck.

Refinement (Extrinsic)

1. **Extrinsic**: external feedback



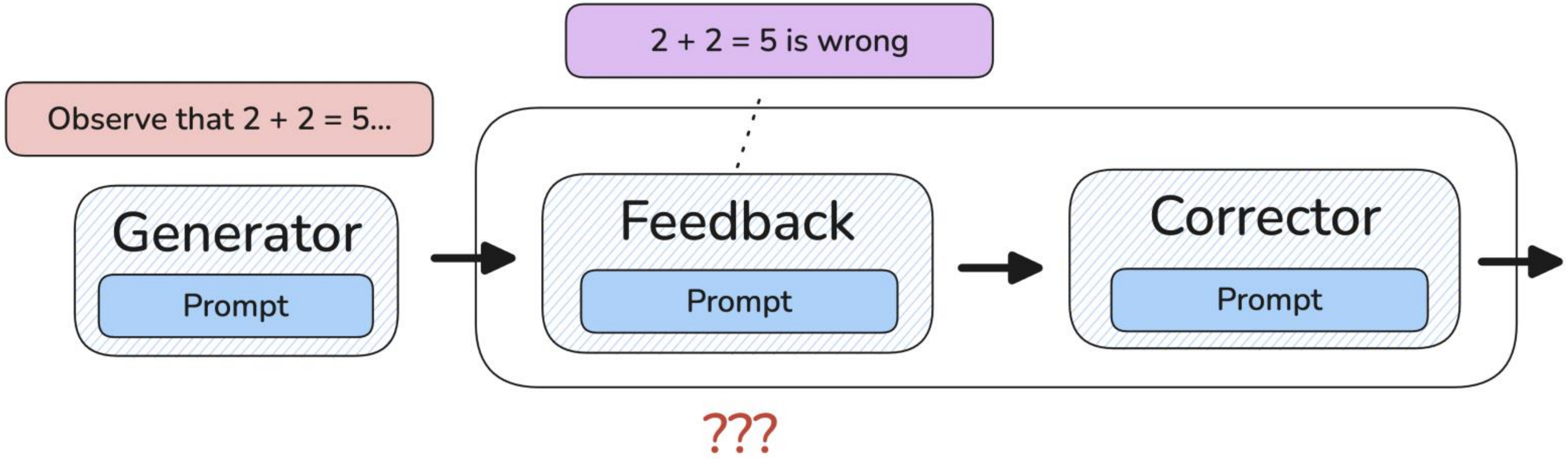
AlphaVerus. P. Aggarwal, B. Parno, S. Welleck.

Refinement (Extrinsic)

- Extrinsic: External Feedback
- Succuess cases
  - Verifiers [Aggarwal et al., 2024]
  - Code interpreters [Chen et al., 2024]
  - Retrievers [Asai et al., 2024]
  - ...
- Intuition: adds new information that allows detection and localization of erros

Refinement (Intrinsic)

2. **Intrinsic**: Re-prompt the same model:



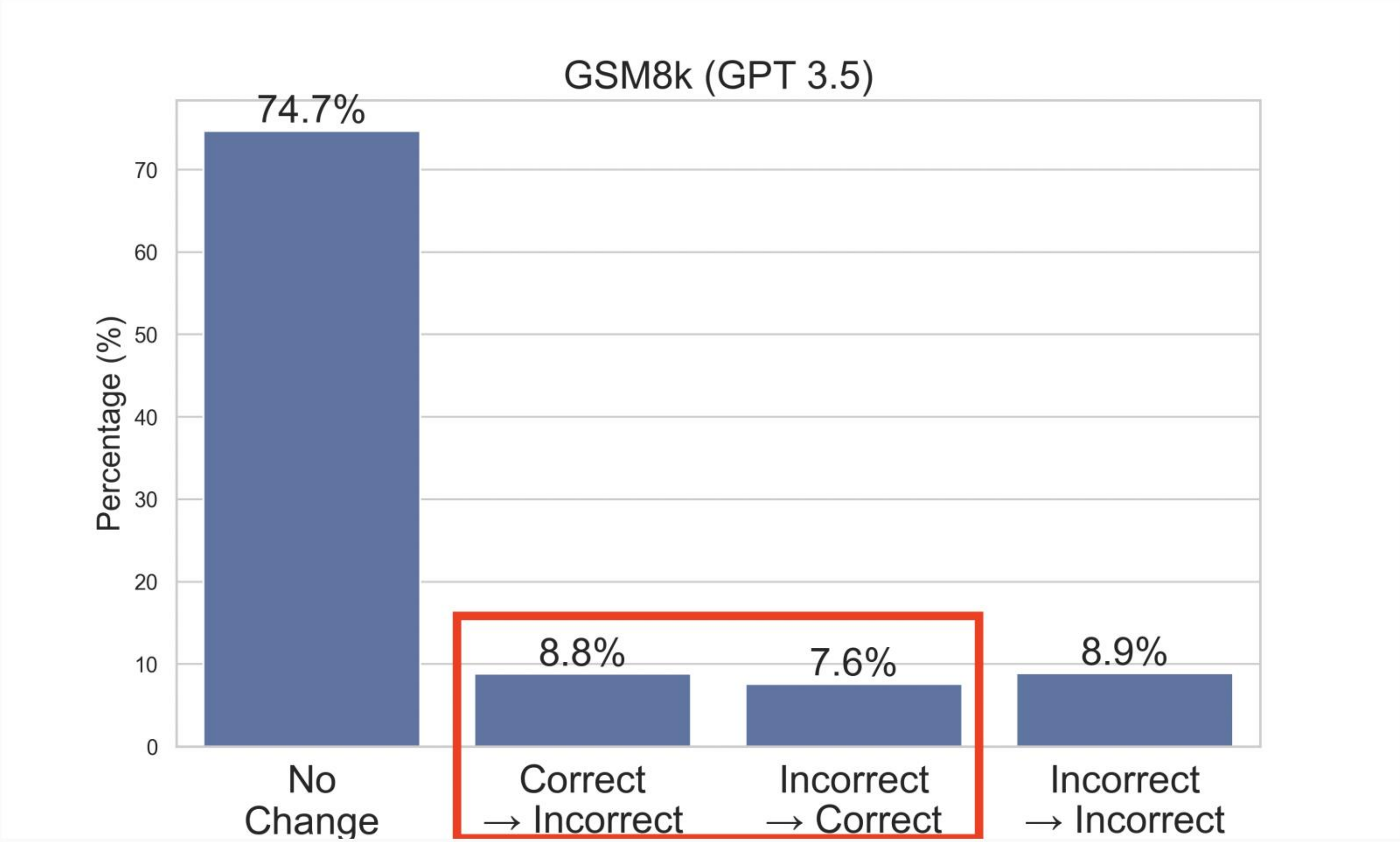
Re-prompt a single LLM, e.g. [Madaan et al., 2023]

Refinement (Intrinsic)

Mixed results

- ❑ Easy to evaluate tasks: positive [Wang et al., 2024]
- ❑ Mathematical reasoning: mixed [Huang et al., 2024]

Refinement (Intrinsic)



Takeaway: feedback is too noisy From [Huang et al., 2024]

Refinement

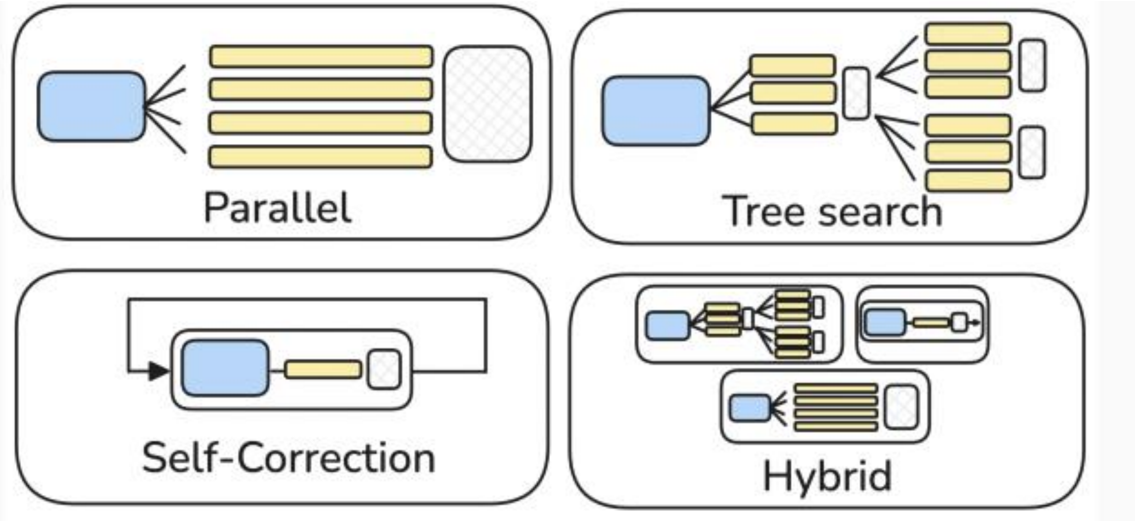
Refinement / self-correction

- Extrinsic
  - Positive results for environments that detect or localize errors
- Intrinsic
  - Mixed results, depends on difficulty of verification

Training Reasoners

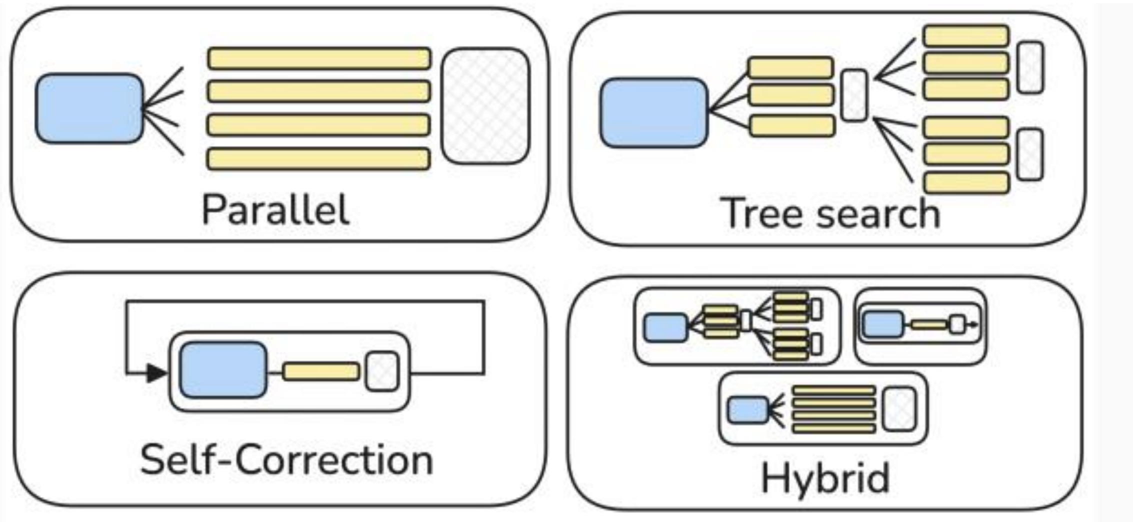
Latent Space Reasoning

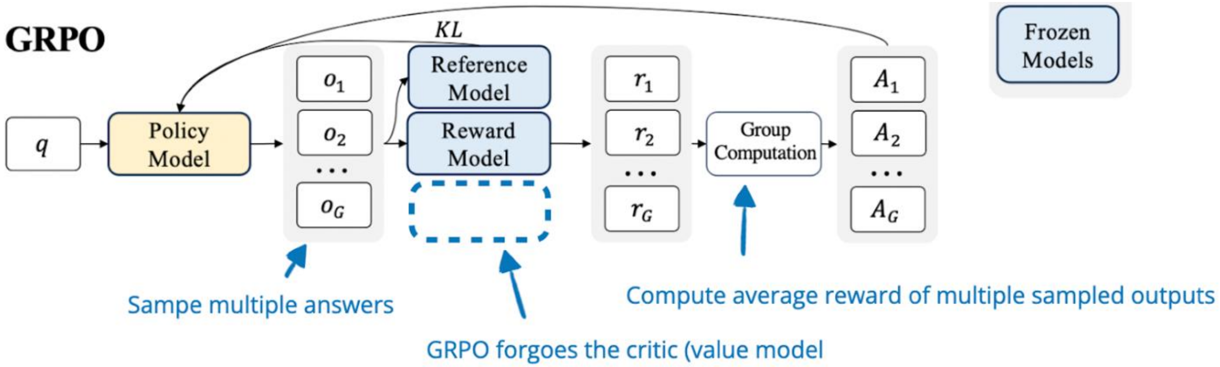
- Test-time strategies rely on already trained models in order to get improvements with more token generation



Latent Space Reasoning

- Test-time strategies rely on already trained models in order to get improvements with more token generation
- What if we just trained the models to reason directly?





Training Reasoners

- ❑ Revisiting RLHF & PPO
- ❑ From PPO to GRPO
- ❑ RLHF to RLVR
- ❑ Distillation from Reasoning Models
- ❑ DeepSeek deepdive

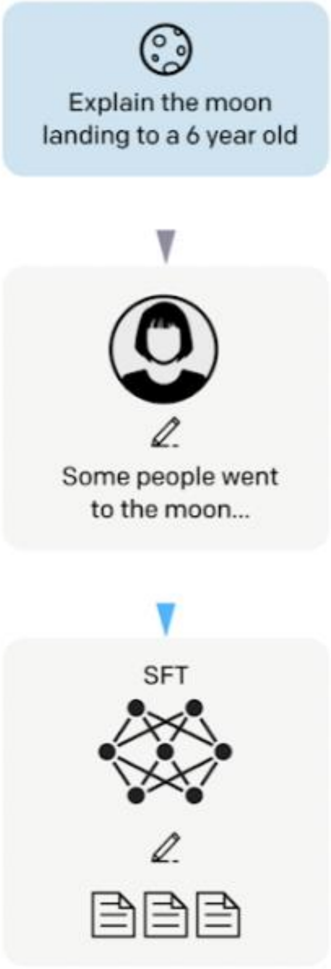
Revisiting RLHF & PPO

RLHF Step 1

Sample prompt

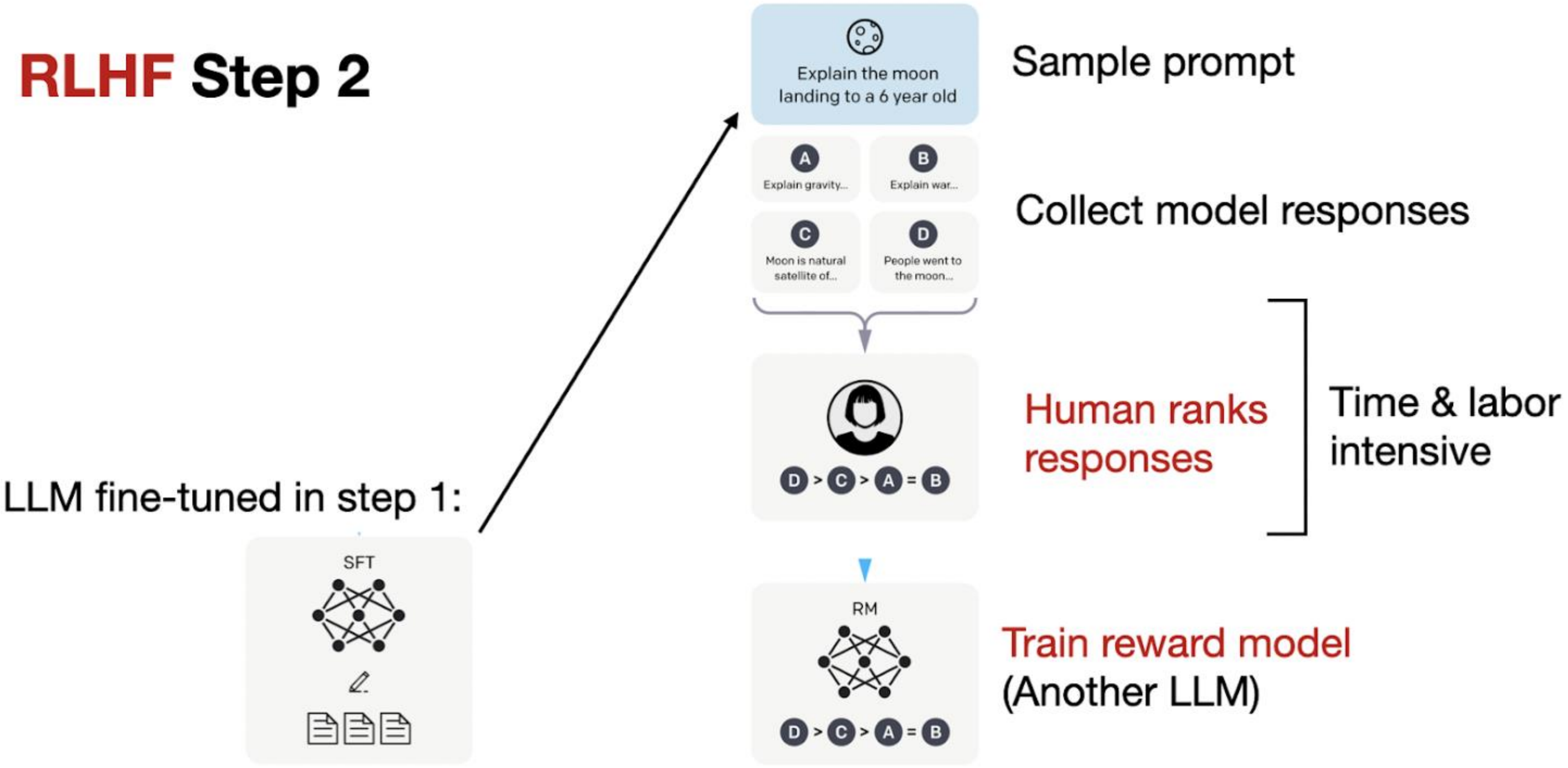
Human writes response

Supervised fine-tuning
of pre-trained LLM

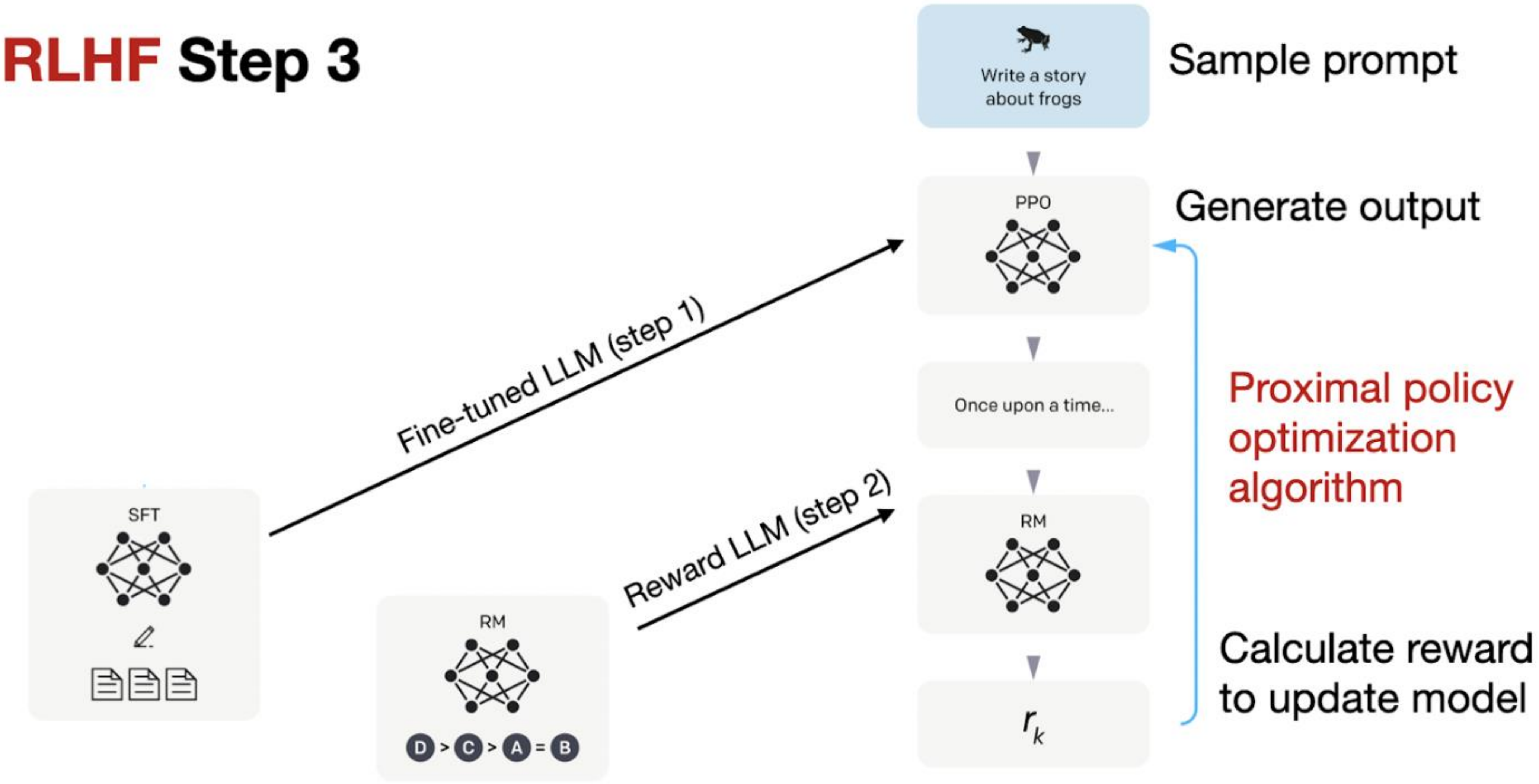


Time & labor intensive

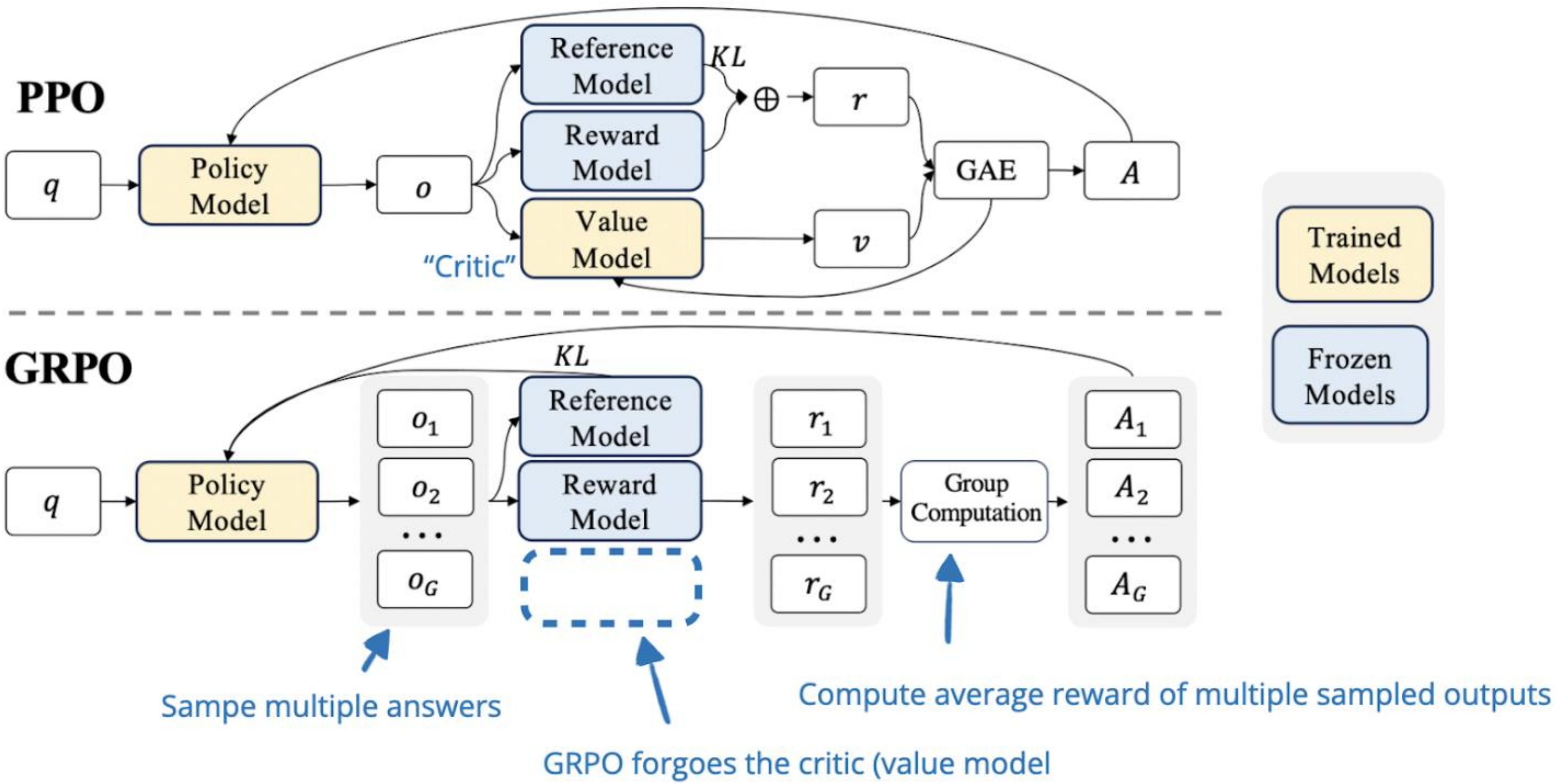
Revisiting RLHF & PPO



Revisiting RLHF & PPO



From PPO to GRPO



Guo et al., (2025)

RLHF to RLVR

RLHF with **PPO**

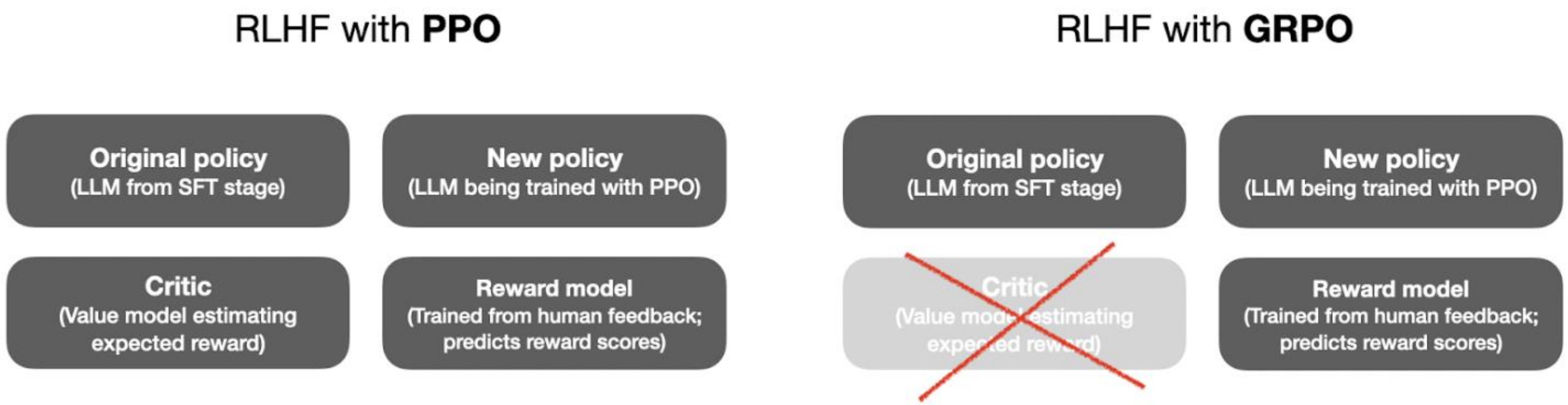
Original policy
(LLM from SFT stage)

New policy
(LLM being trained with PPO)

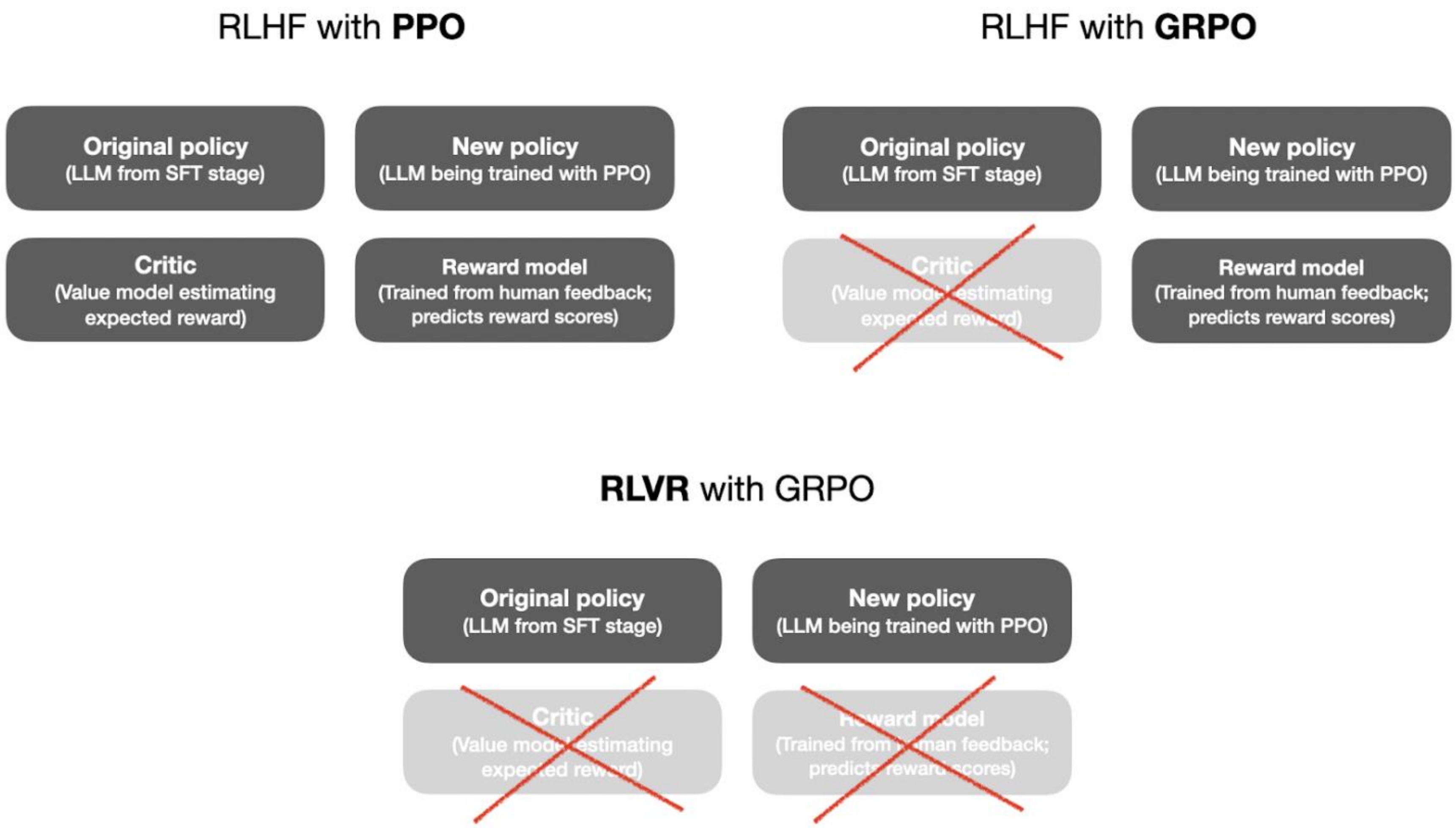
Critic
(Value model estimating expected reward)

Reward model
(Trained from human feedback; predicts reward scores)

RLHF to RLVR


RLHF with PPO
Original policy
(LLM from SFT stage)
New policy
(LLM being trained with PPO)
Critic
(Value model estimating expected reward)
Reward model
(Trained from human feedback; predicts reward scores)
RLHF with GRPO
Original policy
(LLM from SFT stage)
New policy
(LLM being trained with PPO)
~~Critic~~
~~(Value model estimating expected reward)~~
Reward model
(Trained from human feedback; predicts reward scores)

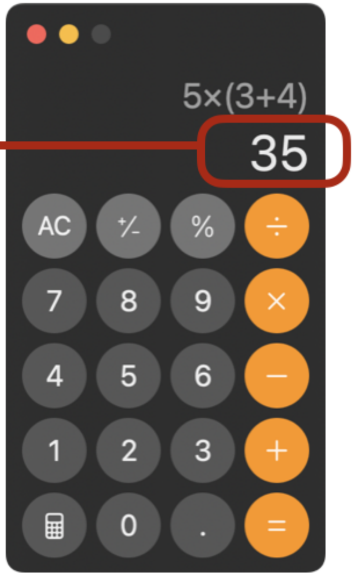
RLHF to RLVR


RLHF with PPO
Original policy
(LLM from SFT stage)
New policy
(LLM being trained with PPO)
Critic
(Value model estimating
expected reward)
Reward model
(Trained from human feedback;
predicts reward scores)
RLHF with GRPO
Original policy
(LLM from SFT stage)
New policy
(LLM being trained with PPO)
Critic
Reward model
(Trained from human feedback;
predicts reward scores)
RLVR with GRPO
Original policy
(LLM from SFT stage)
New policy
(LLM being trained with PPO)
Critic

RLHF to RLVR (RL with Verifiable Rewards)

Input: Solve $5 \times (3 + 4)$

Verifier output:



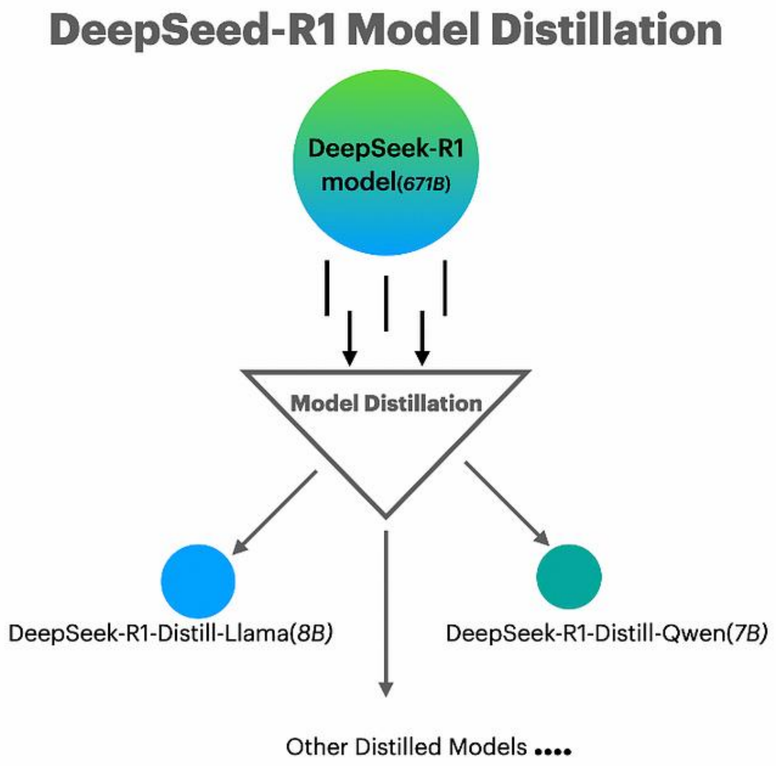
LLM output:

$$5 \times (3 + 4) = \\ 5 \times 7 = \\ \boxed{= 35}$$

Correct

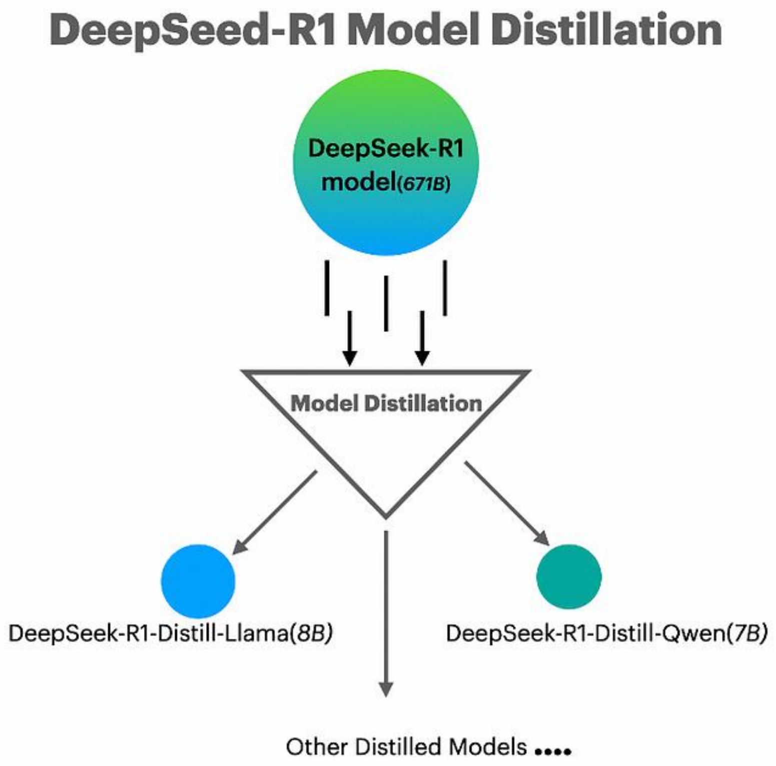
Distillation from Reasoning Models

1. Train a large, very capable reasoning language model



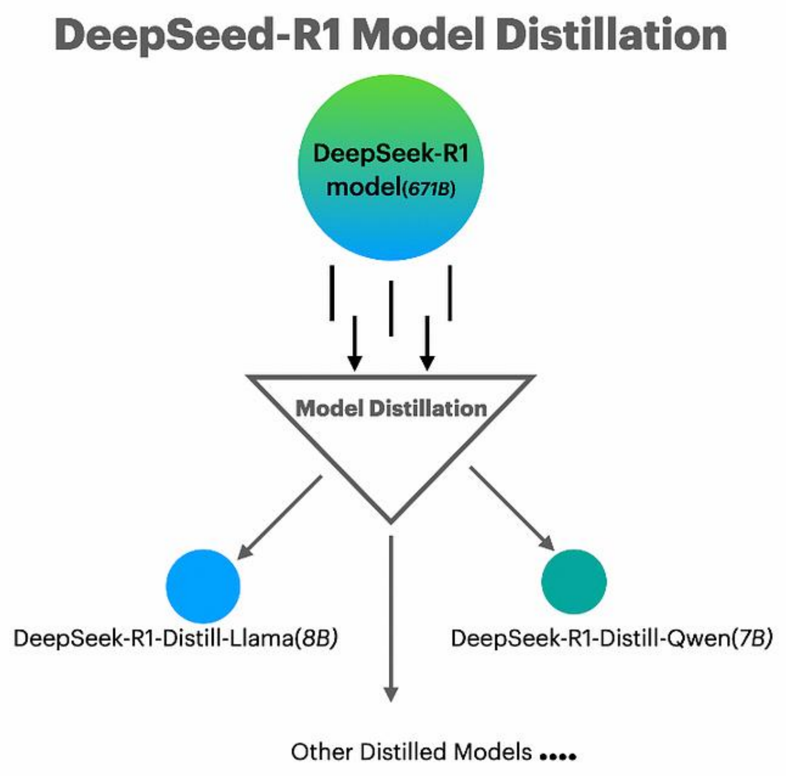
Distillation from Reasoning Models

1. Train a large, very capable reasoning language model
2. Get a number of outputs from this reasoning model (i.e. curate a reasoning dataset



Distillation from Reasoning Models

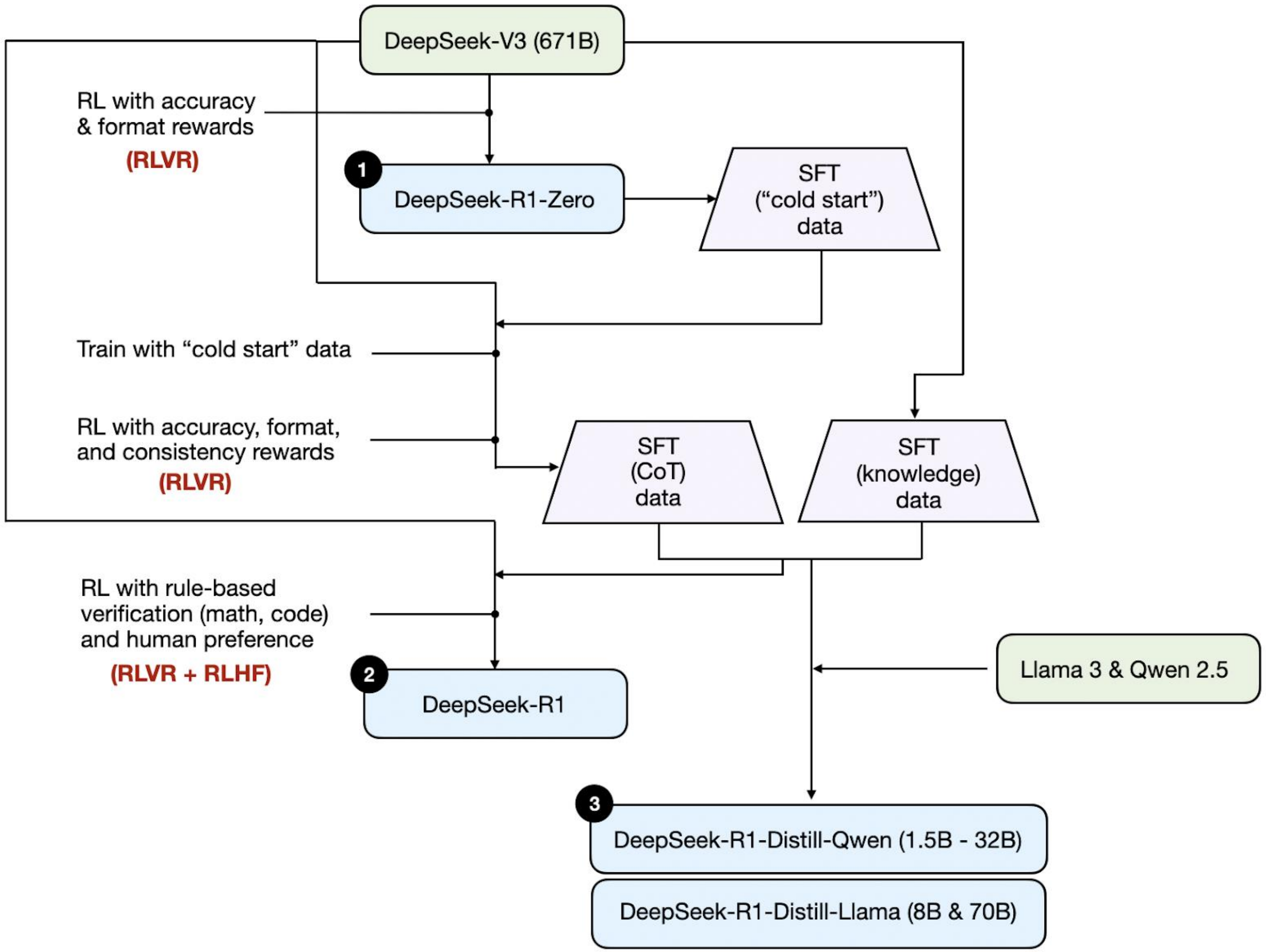
1. Train a large, very capable reasoning language model
2. Get a number of outputs from this reasoning model (i.e. curate a reasoning dataset)
3. Train smaller language models with SFT on this reasoning dataset



Training (DeepSeek R1 deepdive)

DeepSeek R1

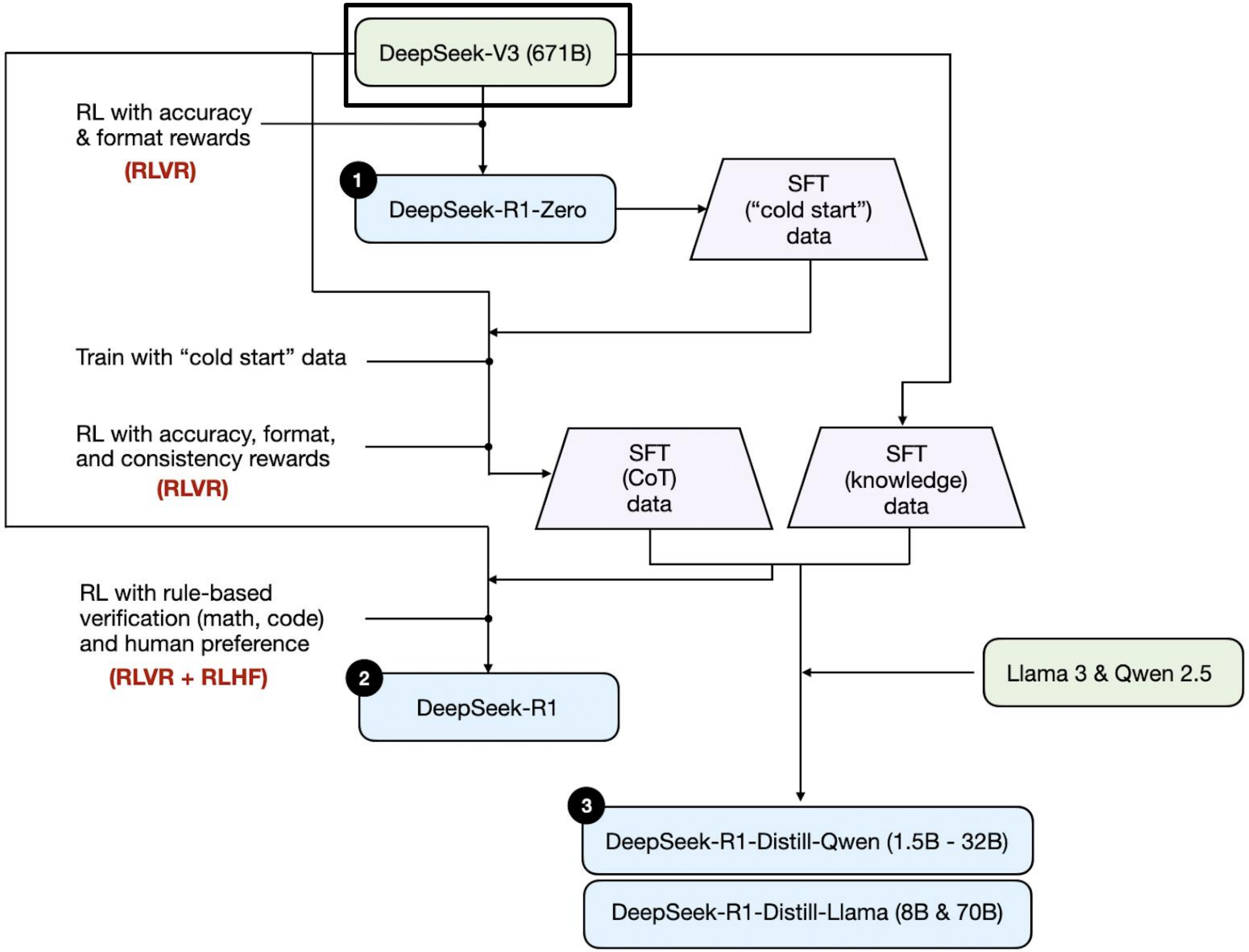
DeepSeek R1 was one of the first efforts to open source models trained explicitly to reason



Guo et al., (2025)

Base Model

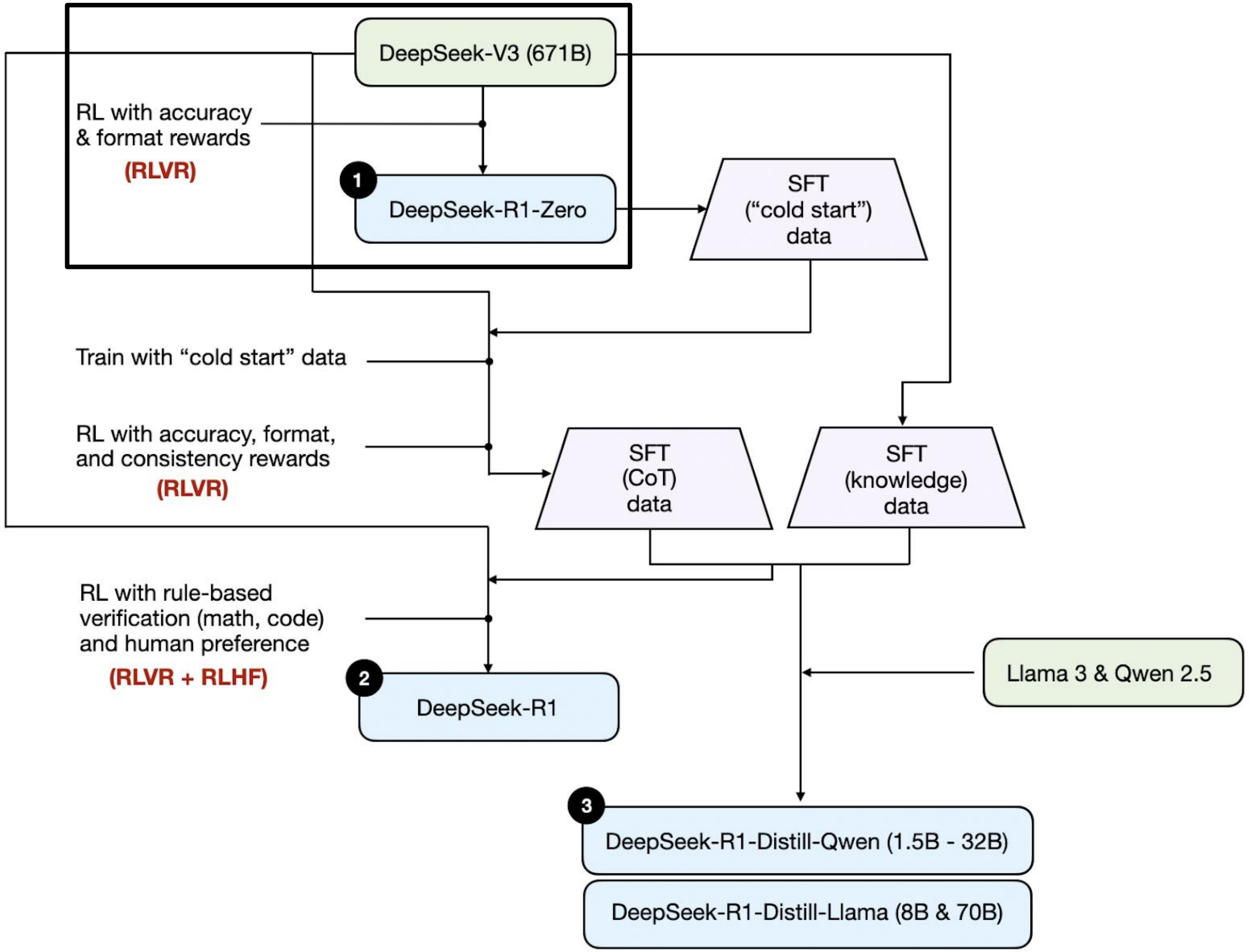
Base Model from pretraining



Guo et al., (2025)

DeepSeek-R1-Zero

- DeepSeek-R1-Zero is trained with RLVR
- No SFT was applied, only RL

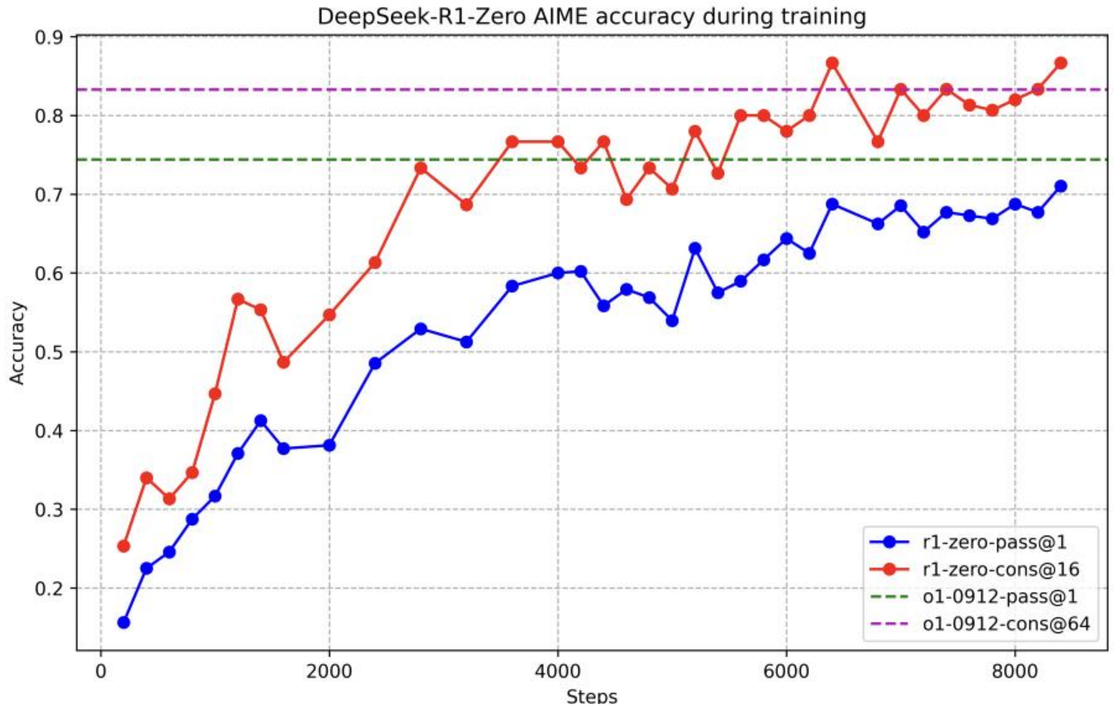


Guo et al., (2025)

DeepSeek R1-Zero (Rewards)

- **Accuracy rewards**: The accuracy reward model evaluates whether the response is correct. For example, in the case of math problems with deterministic results, the model is required to provide the final answer in a specified format (e.g., within a box), enabling reliable rule-based verification of correctness. Similarly, for LeetCode problems, a compiler can be used to generate feedback based on predefined test cases.
- **Format rewards**: In addition to the accuracy reward model, we employ a format reward model that enforces the model to put its thinking process between ‘<think>’ and ‘</think>’ tags.

DeepSeek R1-Zero (Performance)

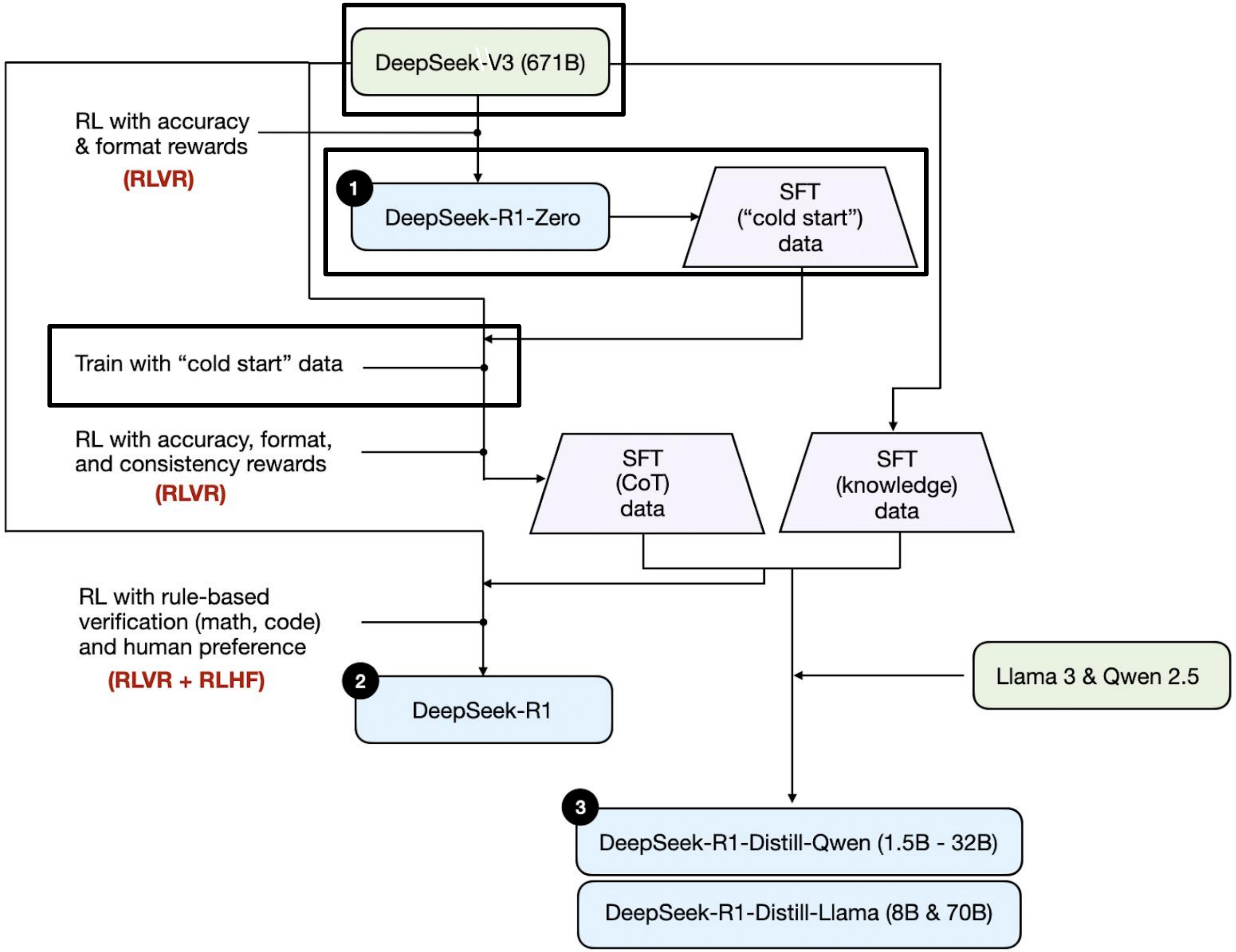


Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	1820
OpenAI-o1-0912	74.4	83.3	94.8	77.3	63.4	1843
DeepSeek-R1-Zero	71.0	86.7	95.9	73.3	50.0	1444

Guo et al., (2025)

DeepSeek-R1

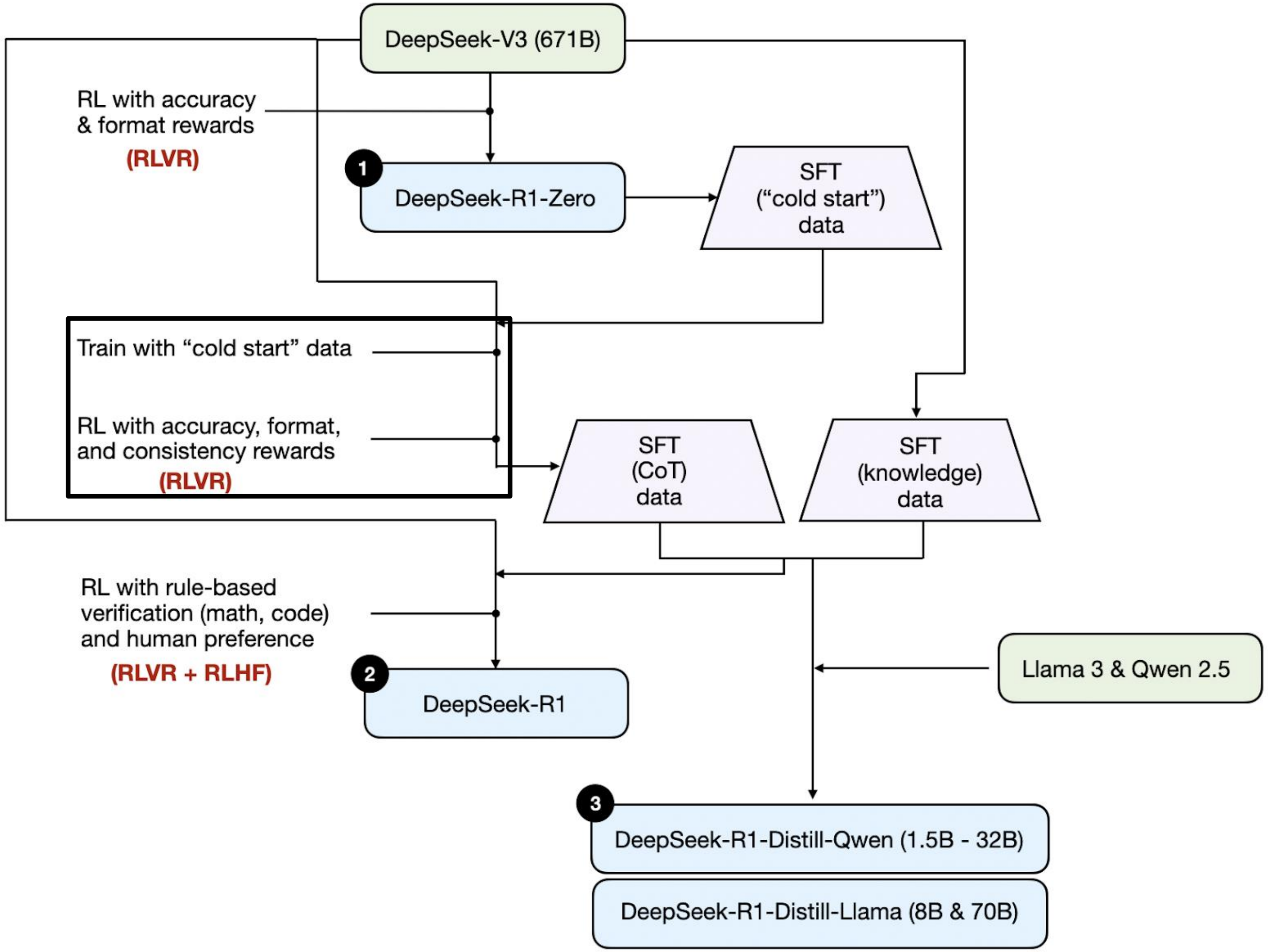
- Collect cold start data from R1-Zero using CoT, and altered prompts
- Use human annotated post-processing on this data
- Use SFT on base model with this post-processed dataset



Guo et al., (2025)

DeepSeek-R1

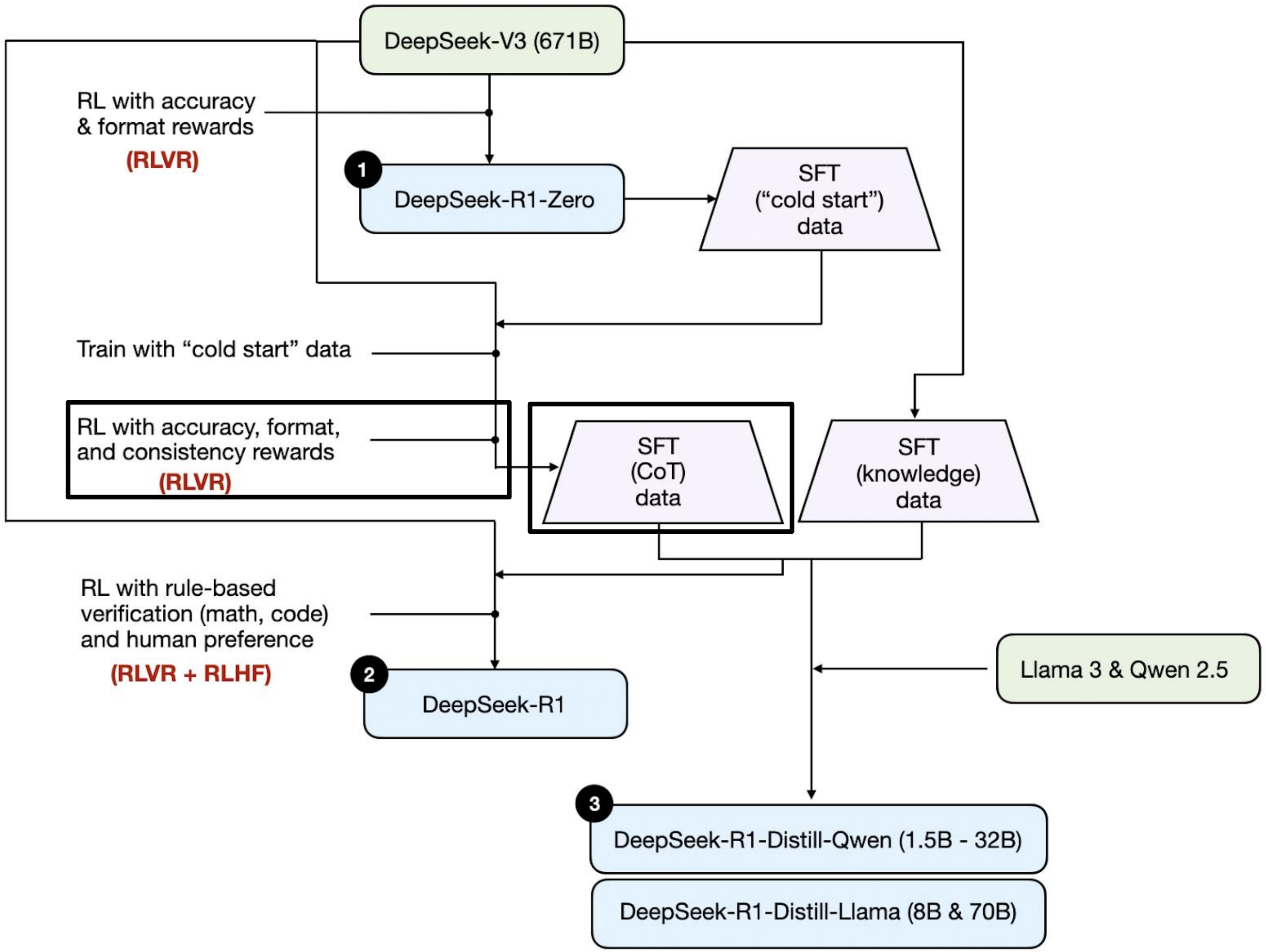
- Apply RLVR to model trained with SFT on "cold start" data
- Use same accuracy, format rewards as was used for R1-Zero
- Add consistency reward to prevent language-switching



Guo et al., (2025)

DeepSeek-R1

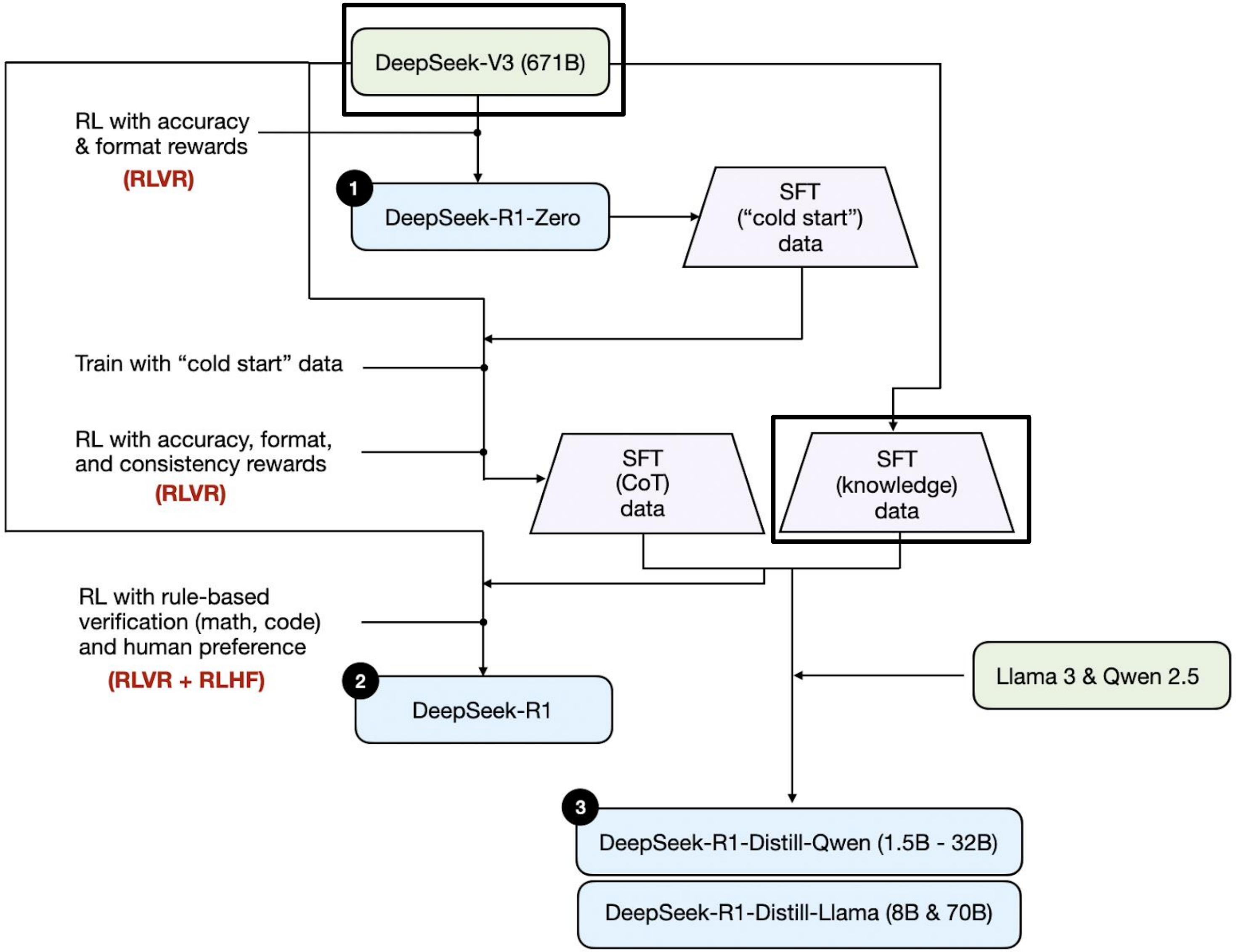
- Using model from previous slide, curate a reasoning dataset (CoT data)
- Sample from this model only when correct and apply simple reward based filters



Guo et al., (2025)

DeepSeek-R1

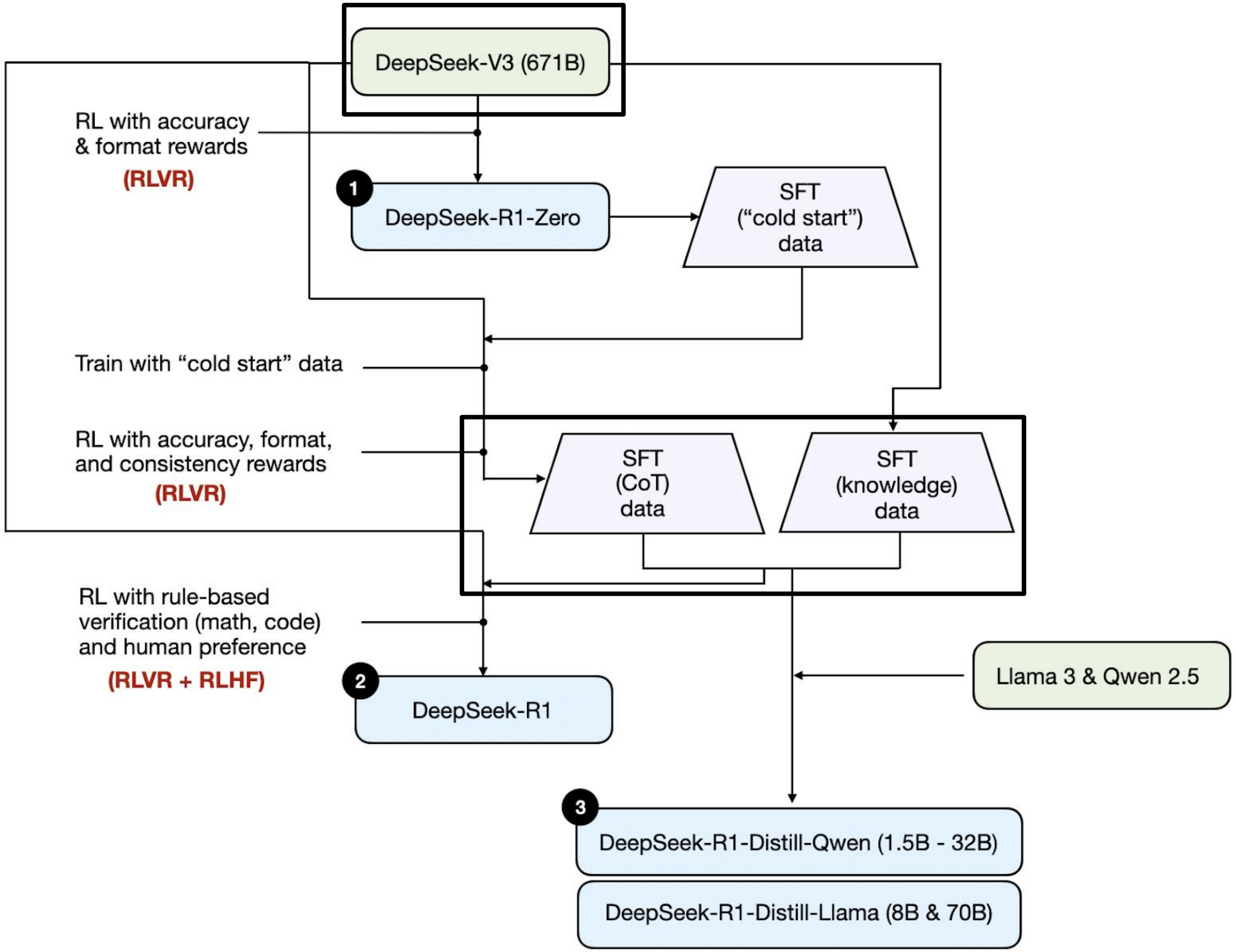
- ❑ Sample non-reasoning data from the base model
- ❑ Sample for tasks related to writing, factual QA, self-cognition, and translation, etc.



Guo et al., (2025)

DeepSeek-R1

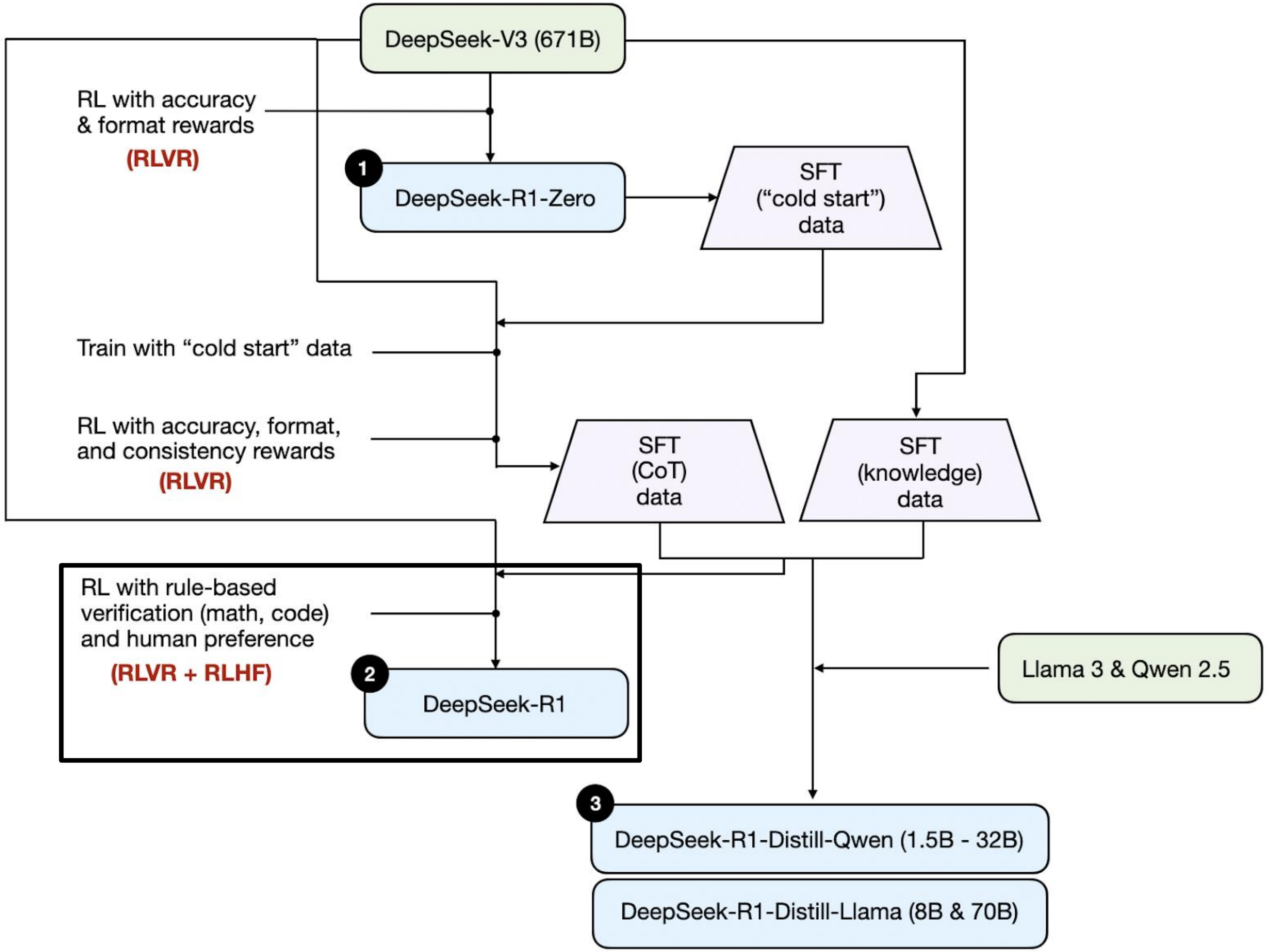
- ❑ Using the two curated datasets, perform SFT on the base model
- ❑ Combined, this is a dataset of around 800K generated, rather than human created, instances



Guo et al., (2025)

DeepSeek-R1

- After applying SFT to the base model, combine RLVR and RLHF methods to produce the final model (DeepSeek-R1)



Guo et al., (2025)

DeepSeek-R1

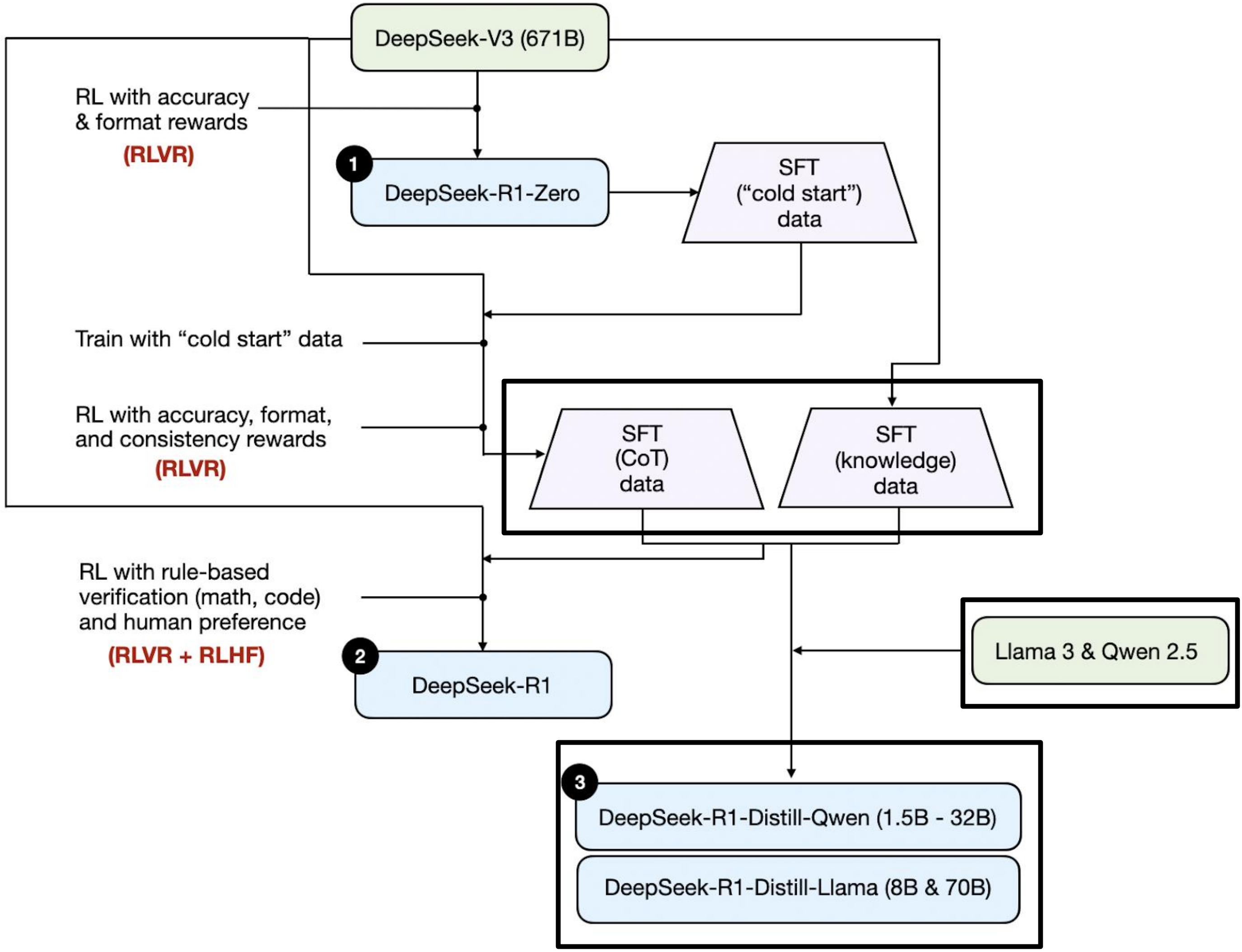
- Very strong model performance
- Note that the base model is a MoE model
- Strongest open source result up to that point

	Benchmark (Metric)	Claude-3.5-Sonnet-1022	GPT-4o 0513	DeepSeek V3	OpenAI o1-mini	OpenAI o1-1217	DeepSeek R1
	Architecture	-	-	MoE	-	-	MoE
	# Activated Params	-	-	37B	-	-	37B
	# Total Params	-	-	671B	-	-	671B
English	MMLU (Pass@1)	88.3	87.2	88.5	85.2	**91.8**	90.8
	MMLU-Redux (EM)	88.9	88.0	89.1	86.7	-	**92.9**
	MMLU-Pro (EM)	78.0	72.6	75.9	80.3	-	**84.0**
	DROP (3-shot F1)	88.3	83.7	91.6	83.9	90.2	**92.2**
	IF-Eval (Prompt Strict)	**86.5**	84.3	86.1	84.8	-	83.3
	GPQA Diamond (Pass@1)	65.0	49.9	59.1	60.0	**75.7**	71.5
	SimpleQA (Correct)	28.4	38.2	24.9	7.0	**47.0**	30.1
	FRAMES (Acc.)	72.5	80.5	73.3	76.9	-	**82.5**
	AlpacaEval2.0 (LC-winrate)	52.0	51.1	70.0	57.8	-	**87.6**
	ArenaHard (GPT-4-1106)	85.2	80.4	85.5	92.0	-	**92.3**
Code	LiveCodeBench (Pass@1-COT)	38.9	32.9	36.2	53.8	63.4	**65.9**
	Codeforces (Percentile)	20.3	23.6	58.7	93.4	**96.6**	96.3
	Codeforces (Rating)	717	759	1134	1820	**2061**	2029
	SWE Verified (Resolved)	**50.8**	38.8	42.0	41.6	48.9	49.2
	Aider-Polyglot (Acc.)	45.3	16.0	49.6	32.9	**61.7**	53.3
Math	AIME 2024 (Pass@1)	16.0	9.3	39.2	63.6	79.2	**79.8**
	MATH-500 (Pass@1)	78.3	74.6	90.2	90.0	96.4	**97.3**
	CNMO 2024 (Pass@1)	13.1	10.8	43.2	67.6	-	**78.8**
Chinese	CLUEWSC (EM)	85.4	87.9	90.9	89.9	-	**92.8**
	C-Eval (EM)	76.7	76.0	86.5	68.9	-	**91.8**
	C-SimpleQA (Correct)	55.4	58.7	**68.0**	40.3	-	63.7

Guo et al., (2025)

DeepSeek-R1-{Qwen, Llama} (*B)

- Apply SFT to Llama3 and Qwen2.5 models using the ~800k SFT data from the pipeline
- Note: ***Only*** SFT is applied at this point, no RL is used



Guo et al., (2025)

DeepSeek-R1-{Qwen, Llama} (*B)

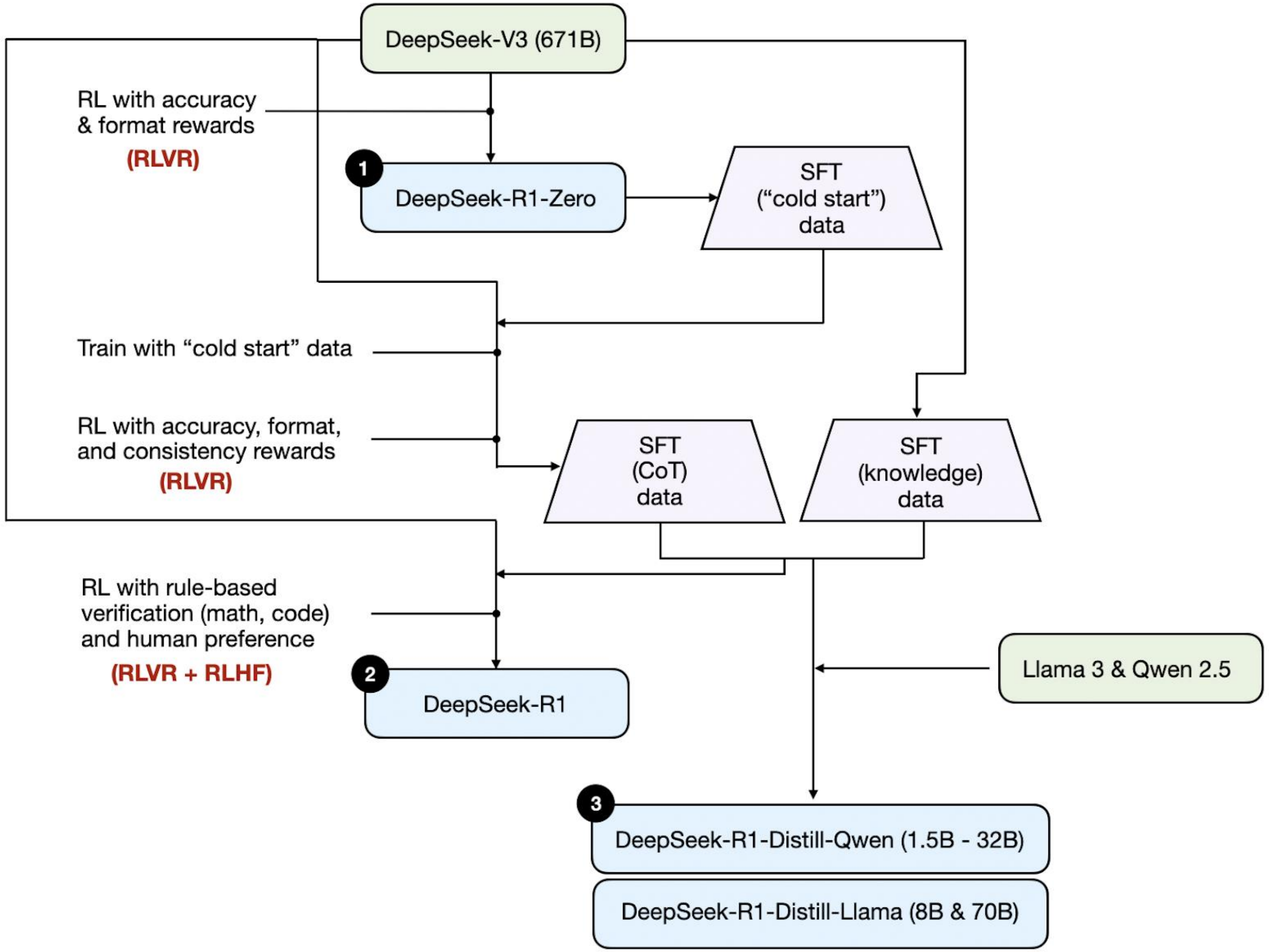
- Even without any reinforcement learning, SFT with a reasoning dataset is sufficient to achieve very good performance with these other open Small Language Models (SLMs)
- Opens possibility of improving SLMs by curating better reasoning datasets

Model	AIME 2024		MATH-500	GPQA Diamond	LiveCode Bench	CodeForces
	pass@1	cons@64	pass@1	pass@1	pass@1	rating
GPT-4o-0513	9.3	13.4	74.6	49.9	32.9	759
Claude-3.5-Sonnet-1022	16.0	26.7	78.3	65.0	38.9	717
OpenAI-o1-mini	63.6	80.0	90.0	60.0	53.8	**1820**
QwQ-32B-Preview	50.0	60.0	90.6	54.5	41.9	1316
DeepSeek-R1-Distill-Qwen-1.5B	28.9	52.7	83.9	33.8	16.9	954
DeepSeek-R1-Distill-Qwen-7B	55.5	83.3	92.8	49.1	37.6	1189
DeepSeek-R1-Distill-Qwen-14B	69.7	80.0	93.9	59.1	53.1	1481
DeepSeek-R1-Distill-Qwen-32B	**72.6**	83.3	94.3	62.1	57.2	1691
DeepSeek-R1-Distill-Llama-8B	50.4	80.0	89.1	49.0	39.6	1205
DeepSeek-R1-Distill-Llama-70B	70.0	**86.7**	**94.5**	**65.2**	**57.5**	1633

Guo et al., (2025)

DeepSeek Main Contributions (3 key areas)

1. Possible to train reasoning model with RL alone on math/coding (DeepSeek-R1-Zero)
2. It is possible to curate a reasoning dataset that enables SFT for reasoning models (DeepSeek-R1-{Qwen, Llama} (*B))
3. Open Sources RL + SFT solution which is competitive with closed-source models (DeepSeek-R1)



Guo et al., (2025)

Latent Space Reasoning

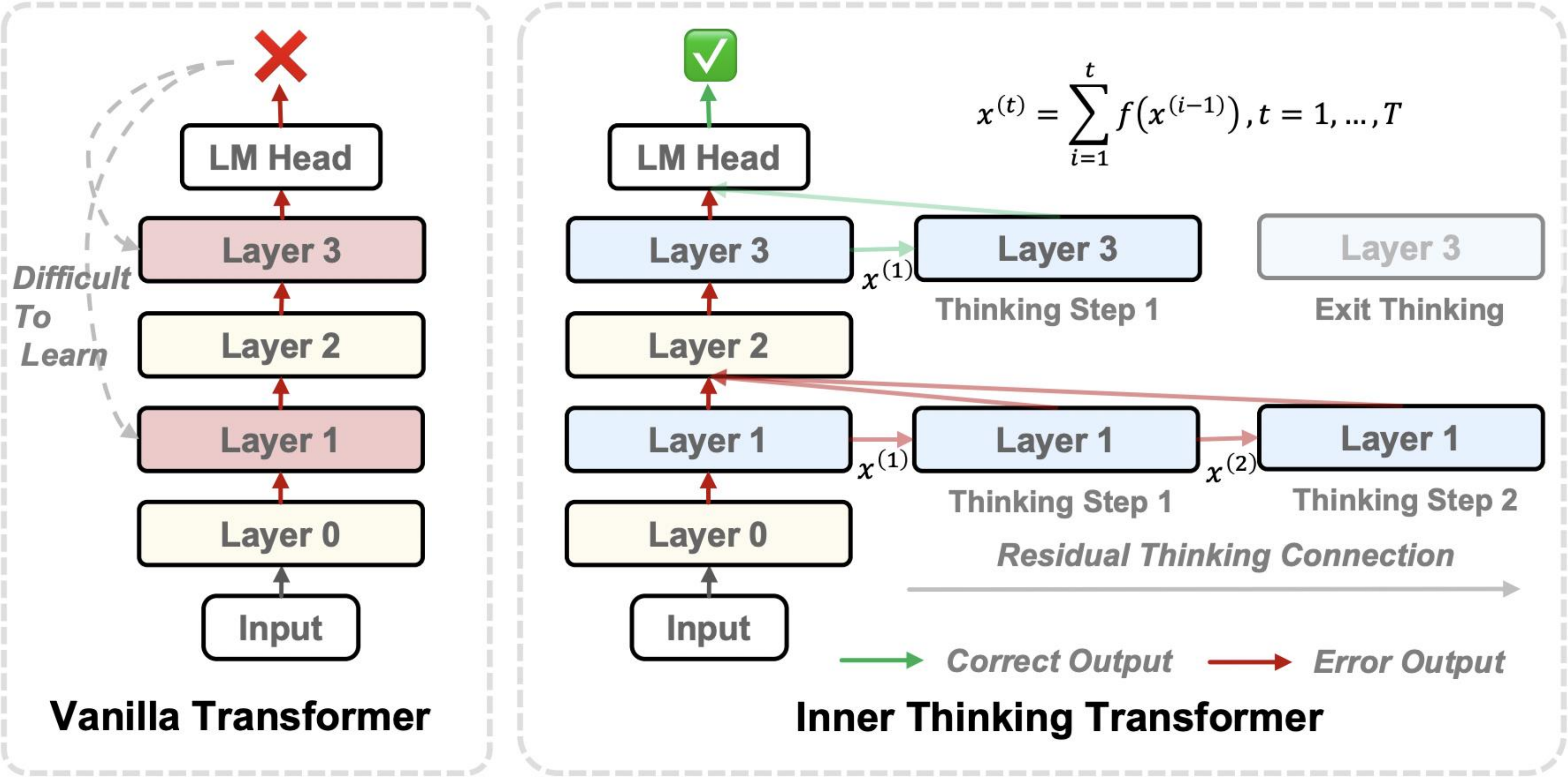
Latent Space Reasoning

- What if, instead of generating more tokens at inference, we just used more compute by altering the hidden states of the transformer itself

Latent Space Reasoning

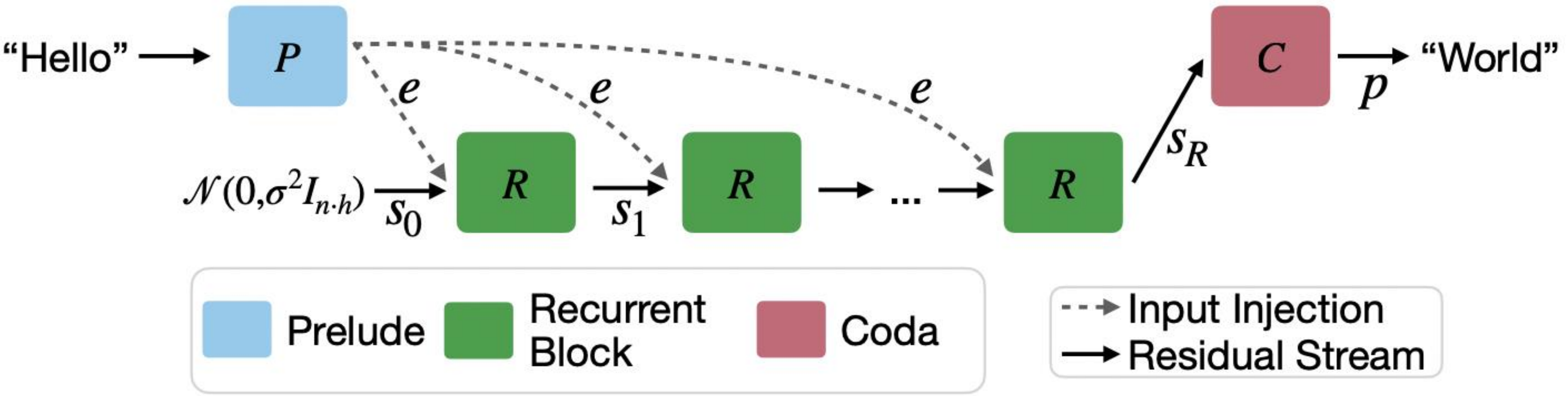
- ❑ What if, instead of generating more tokens at inference, we just used more compute by altering the hidden states of the transformer itself
- ❑ In analogical terms, less talking, more thinking

Methods (Inner Thinking Transformer)



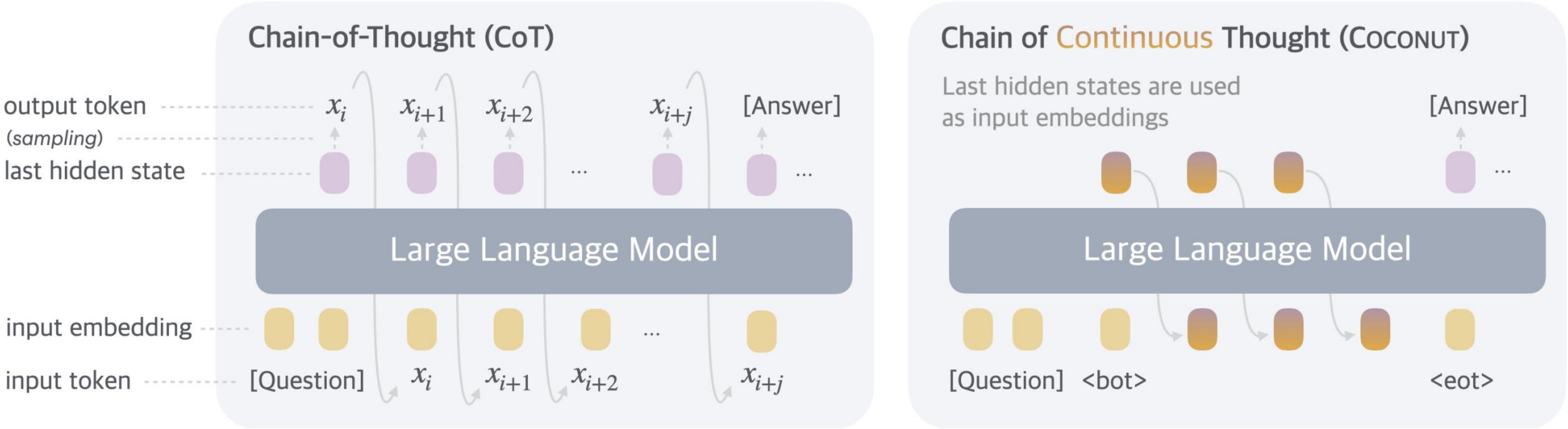
Chen et al., (2025)

Methods (Scaling up Test-Time Compute with Latent Reasoning)



Geiping et al., (2025)

Training Large Language Models to Reason in a Continuous Latent Space



Hao et al., (2024)