

LLM based agents + Math-
small LLM rStar

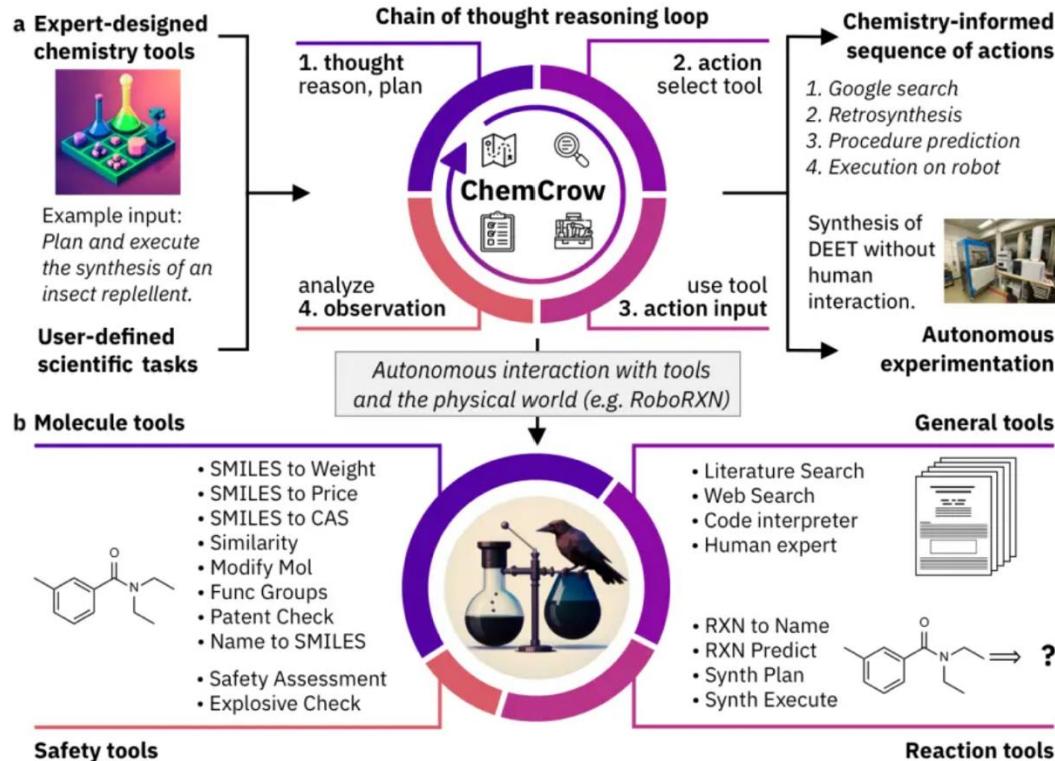
A Review of Prominent Paradigms for LLM-Based Agents: Tool Use (Including RAG), Planning, and Feedback Learning

First Half

Name: Swakshar Deb (swd9tc)

Introduction

LLM based agent in Chemistry applications



LLM agent in the Embodied Environment

You are in the middle of a room. Looking quickly around you, you see a drawer 2, a shelf 5, a drawer 1, a shelf 4, a sidetable 1, a drawer 5, a shelf 6, a shelf 1, a shelf 9, a cabinet 2, a sofa 1, a cabinet 1, a shelf 3, a cabinet 3, a drawer 3, a shelf 11, a shelf 2, a shelf 10, a dresser 1, a shelf 12, a garbagecan 1, a armchair 1, a cabinet 4, a shelf 7, a shelf 8, a safe 1, and a drawer 4.

Your task is to: put some vase in safe.

> go to shelf 6

You arrive at loc 4. On the shelf 6, you see a vase 2.

> take vase 2 from shelf 6

You pick up the vase 2 from the shelf 6.

> go to safe 1

You arrive at loc 3. The safe 1 is closed.

> open safe 1

You open the safe 1. The safe 1 is open. In it, you see a keychain 3.

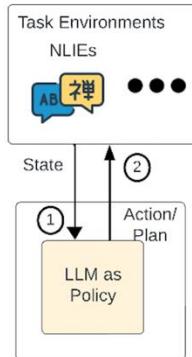
> put vase 2 in/on safe 1

You won!

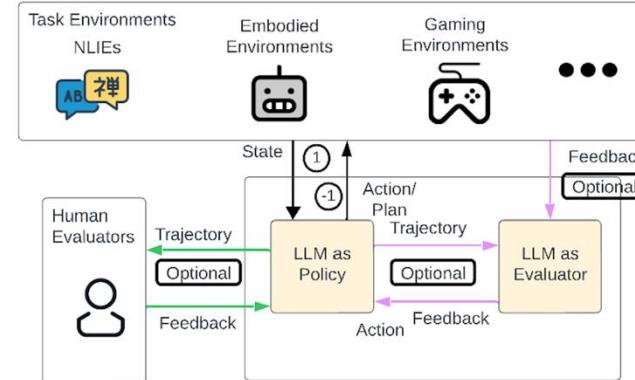
Align text with real world



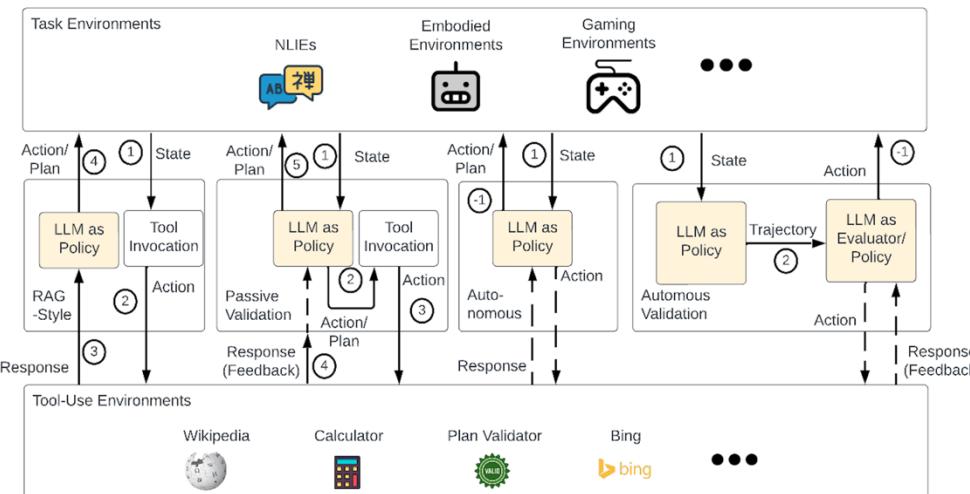
Various Type of LLM Based Agents



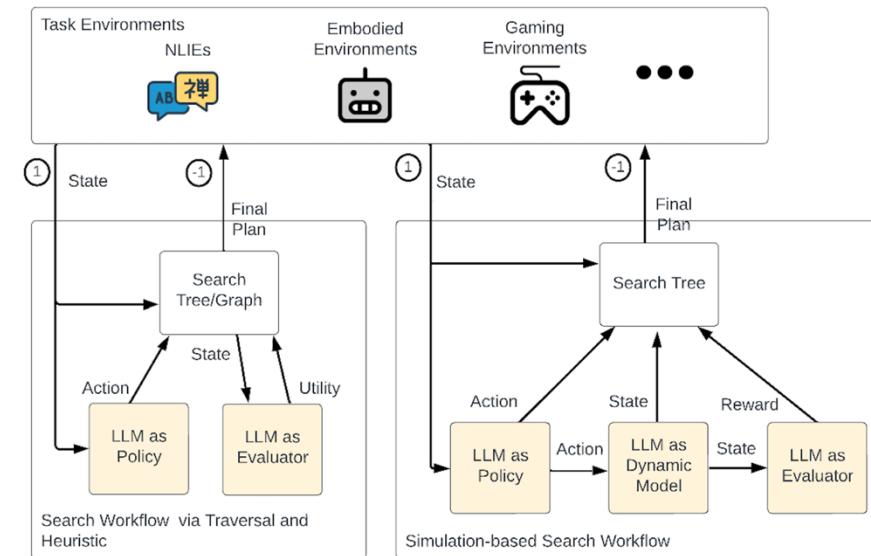
(a) Base Workflows.



(b) Workflows for Feedback Learning.



(c) Workflows for Tool Use, with validation types categorized under both Tool Use and Feedback Learning paradigms.



(d) Search Workflows for Planning.

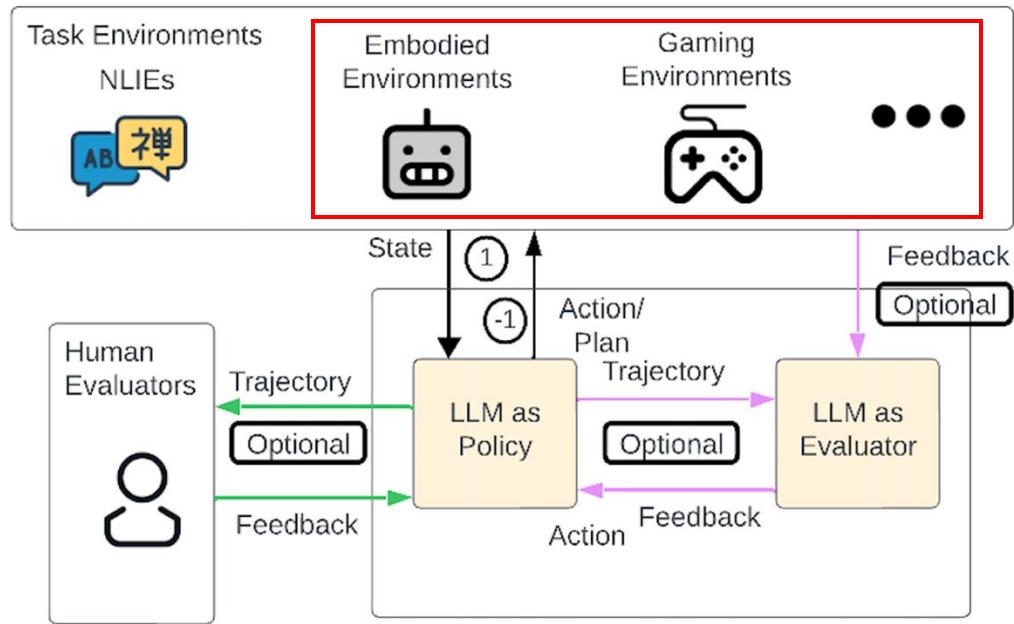
Task Environments

- Summary of different types of environments

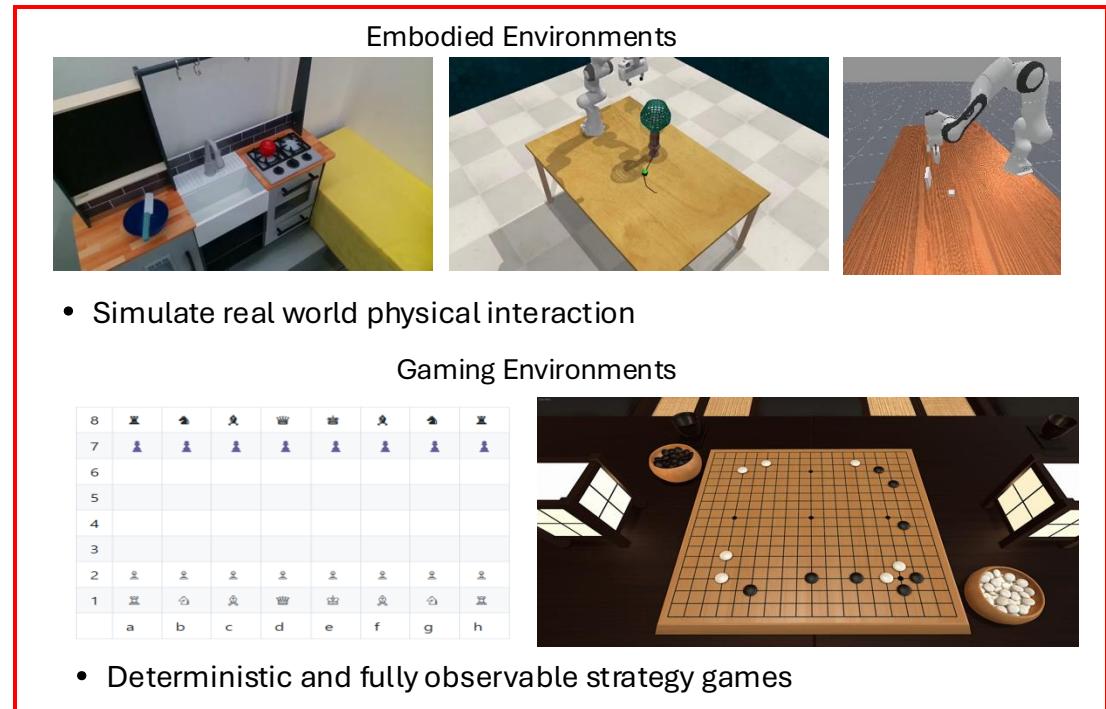
Env Types	Entities Interacted With by Agent	Action Properties	Examples of Action Instances	Examples of Env
Game Environments	Virtual game elements (objects, avatars, other characters), and possibly other players or game narratives	Discrete, Executable, Deterministic	Move(Right)	BlocksWorld (Valmeekam et al., 2022), CrossWords (Yao et al., 2023a)
Embodied Environments	Physical world (through sensors and actuators)	Discrete, Executable, Deterministic	Pick_Up[Object]	AlfWorld (Shridhar et al., 2021), VirtualHome (Puig et al., 2018), Minecraft (Fan et al., 2022)
Web Environments	Virtual web elements	Discrete, Executable, Deterministic	search(3 ounce bright citrus), click(Buy Now)	Webshop (Yao et al., 2022), WebArena (Zhou et al., 2024b), AppWorld (Trivedi et al., 2024)
NLIEs	Humans (through conversation or text)	Free-form, Stochastic	The answer is Answer, Finish[Answer]	GSM8K Cobbe et al. (2021) , HotpotQA (Yang et al., 2018)

Feed Back Based Environment

Feed-Back Based Environments



- Gaming and Embodied environments come under feed-back based environments



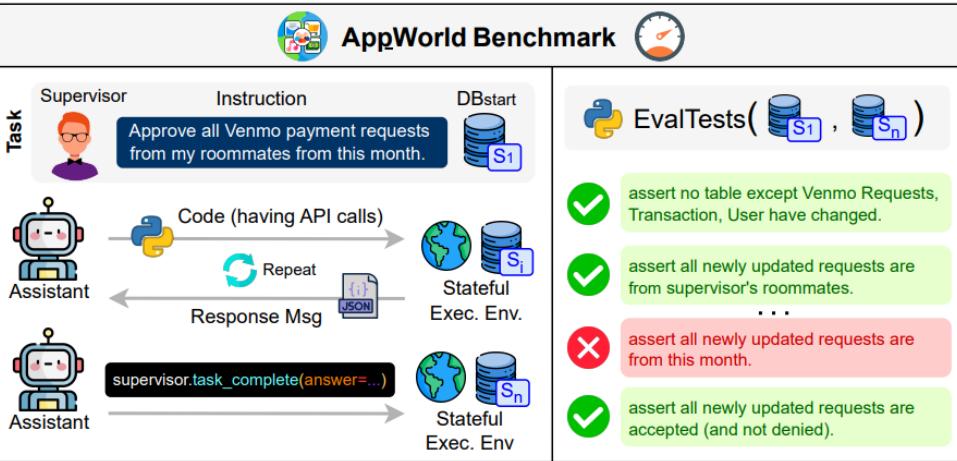
Env Types	Entities Interacted With by Agent	Action Properties	Examples of Action Instances	Examples of Env
Game Environments	Virtual game elements (objects, avatars, other characters), and possibly other players or game narratives	Discrete, Executable, Deterministic	Move(Right)	BlocksWorld (Valmeekam et al., 2022), CrossWords (Yao et al., 2023a)
Embodied Environments	Physical world (through sensors and actuators)	Discrete, Executable, Deterministic	Pick_Up[Object]	AlfWorld (Shridhar et al., 2021), VirtualHome (Puig et al., 2018), Minecraft (Fan et al., 2022)

Task Environments

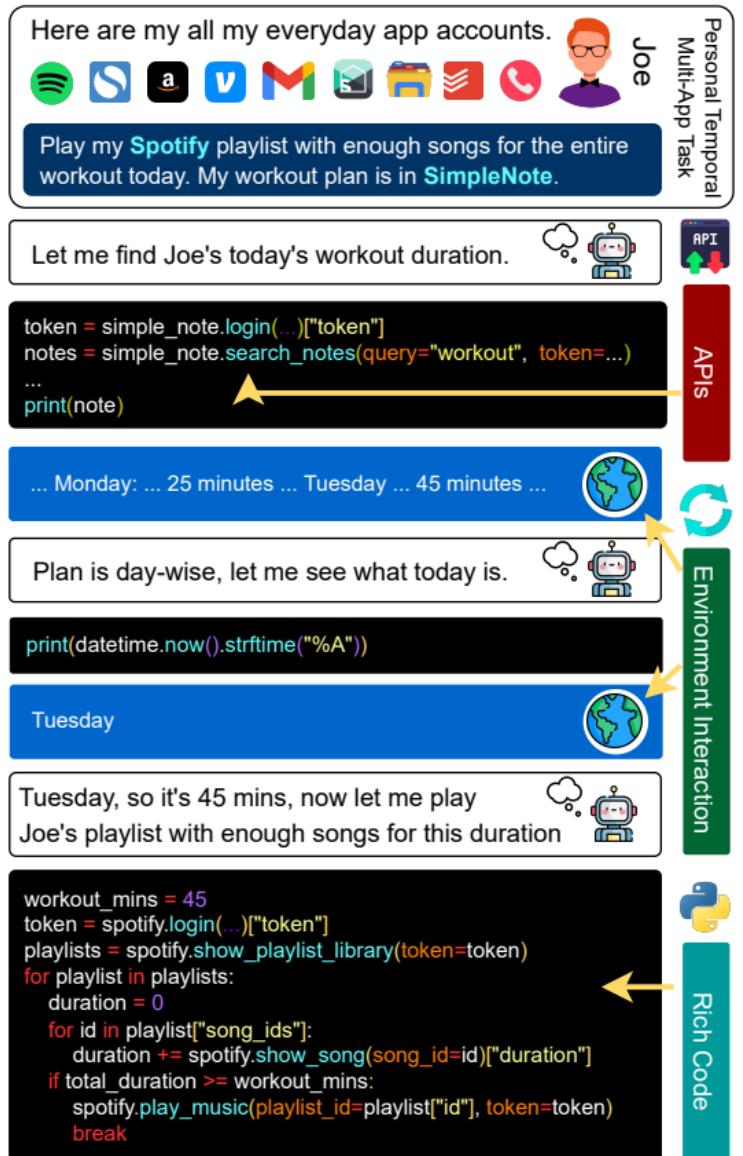
- Summary of different types of environments

Env Types	Entities Interacted With by Agent	Action Properties	Examples of Action Instances	Examples of Env
Game Environments	Virtual game elements (objects, avatars, other characters), and possibly other players or game narratives	Discrete, Executable, Deterministic	Move(Right)	BlocksWorld (Valmeekam et al., 2022), CrossWords (Yao et al., 2023a)
Embodied Environments	Physical world (through sensors and actuators)	Discrete, Executable, Deterministic	Pick_Up[Object]	AlfWorld (Shridhar et al., 2021), VirtualHome (Puig et al., 2018), Minecraft (Fan et al., 2022)
Web Environments	Virtual web elements	Discrete, Executable, Deterministic	search(3 ounce bright citrus), click(Buy Now)	Webshop (Yao et al., 2022), WebArena (Zhou et al., 2024b), AppWorld (Trivedi et al., 2024)
NLIEs	Humans (through conversation or text)	Free-form, Stochastic	The answer is Answer, Finish[Answer]	GSM8K Cobbe et al. (2021) , HotpotQA (Yang et al., 2018)

Web-Based Environment



Env Types	Entities Interacted With by Agent	Action Properties	Examples of Action Instances	Examples of Env
Game Environments	Virtual game elements (objects, avatars, other characters), and possibly other players or game narratives	Discrete, Executable, Deterministic	Move(Right)	BlocksWorld (Valmeekam et al., 2022), CrossWords (Yao et al., 2023a)
Embodied Environments	Physical world (through sensors and actuators)	Discrete, Executable, Deterministic	Pick_Up[Object]	AlfWorld (Shridhar et al., 2021), VirtualHome (Puig et al., 2018), Minecraft (Fan et al., 2022)
Web Environments	Virtual web elements	Discrete, Executable, Deterministic	search(3 ounce bright citrus), click(Buy Now)	Webshop (Yao et al., 2022), WebArena (Zhou et al., 2024b), AppWorld (Trivedi et al., 2024)
NLIEs	Humans (through conversation or text)	Free-form, Stochastic	The answer is Answer, Finish[Answer]	GSM8K Cobbe et al. (2021), HotpotQA (Yang et al., 2018)



Natural Language Instruction Environment (NLIE)

Env Types	Action Properties	Examples of Action Instances	Examples of Env	
Game Environments	Virtual game elements (objects, avatars, other characters), and possibly other players or game narratives	Discrete, Executable, Deterministic	Move(Right)	BlocksWorld (Valmeekam et al., 2022), CrossWords (Yao et al., 2023a)
Embodied Environments	Physical world (through sensors and actuators)	Discrete, Executable, Deterministic	Pick_Up[Object]	AlfWorld (Shridhar et al., 2021), VirtualHome (Puig et al., 2018), Minecraft (Fan et al., 2022)
Web Environments	Virtual web elements	Discrete, Executable, Deterministic	search(3 ounce bright citrus), click(Buy Now)	Webshop (Yao et al., 2022), WebArena (Zhou et al., 2024b), AppWorld (Trivedi et al., 2024)
NLIEs	Humans (through conversation or text)	Free-form, Stochastic	The answer is Answer, Finish[Answer]	GSM8K Cobbe et al. (2021), HotpotQA (Yang et al., 2018)

Natural Language Instruction Environment (NLIE)

- Single Step NLIE: **No intermediate state (S_i)**
 - Single step decision making process
 - Examples: Chain of Thought, Tree of Thought
- Multi-step NLIE: **Generate intermediate state (S_i)**
 - Reformulate QA task as sub-questions where each sub-question is an intermediate state
 - Example: Reasoning via planning (RAP)

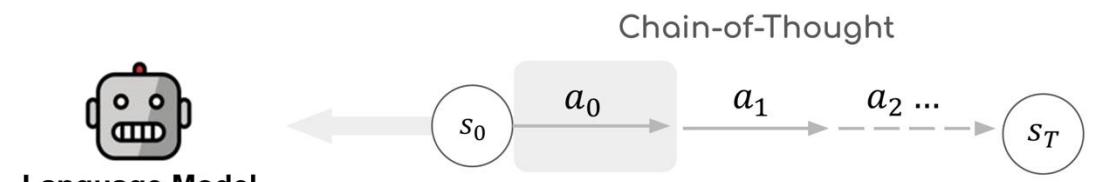


Figure: Single Step NLIE

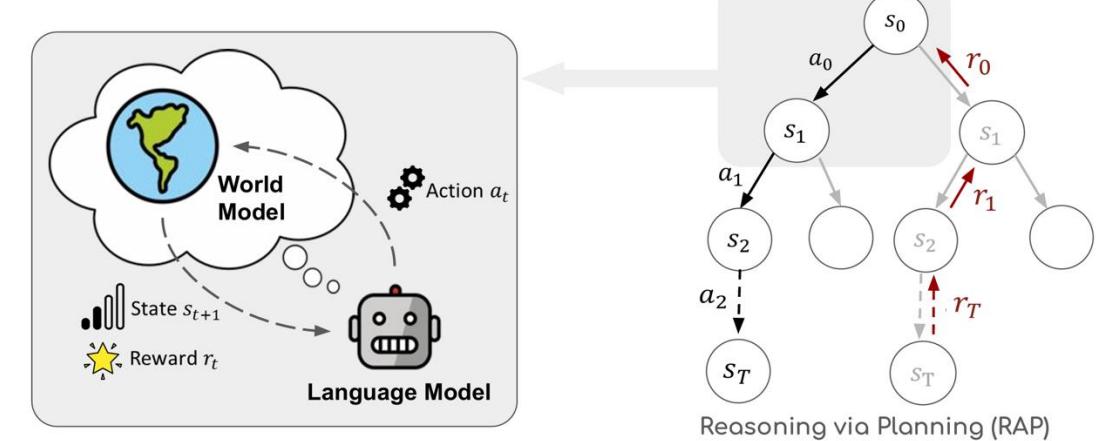


Figure: Multi-step NLIE

Single Step NLIE: Chain of Thought

Math Word Problems (free response)

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Math Word Problems (multiple choice)

Q: How many keystrokes are needed to type the numbers from 1 to 500?
Answer Choices: (a) 1156 (b) 1392 (c) 1480 (d) 1562 (e) 1788

A: There are 9 one-digit numbers from 1 to 9. There are 90 two-digit numbers from 10 to 99. There are 401 three-digit numbers from 100 to 500. $9 + 90(2) + 401(3) = 1392$. The answer is (b).

CSQA (commonsense)

Q: Sammy wanted to go to where the people were. Where might he go?
Options: (a) race track (b) populated areas (c) desert (d) apartment (e) roadblock

A: The answer must be a place with a lot of people. Race tracks, desert, apartments, and roadblocks don't have a lot of people, but populated areas do. So the answer is (b).

StrategyQA

Q: Yes or no: Would a pear sink in water?

A: The density of a pear is about 0.6 g/cm³, which is less than water. Thus, a pear would float. So the answer is no.

Date Understanding

Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?

A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.

Sports Understanding

Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."

A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.

SayCan (Instructing a robot)

Human: How would you bring me something that isn't a fruit?

Explanation: the user wants something to eat that isn't a fruit. An energy bar is not a fruit, so I will bring the user an energy bar.

Plan: 1. find(energy bar) 2. pick(energy bar) 3. find(user) 4. put(energy bar) 5. done().

Last Letter Concatenation

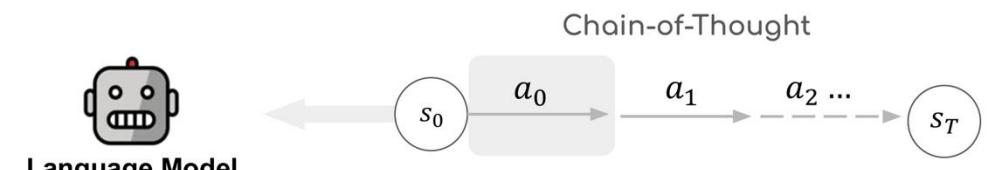
Q: Take the last letters of the words in "Lady Gaga" and concatenate them.

A: The last letter of "Lady" is "y". The last letter of "Gaga" is "a". Concatenating them is "ya". So the answer is ya.

Coin Flip (state tracking)

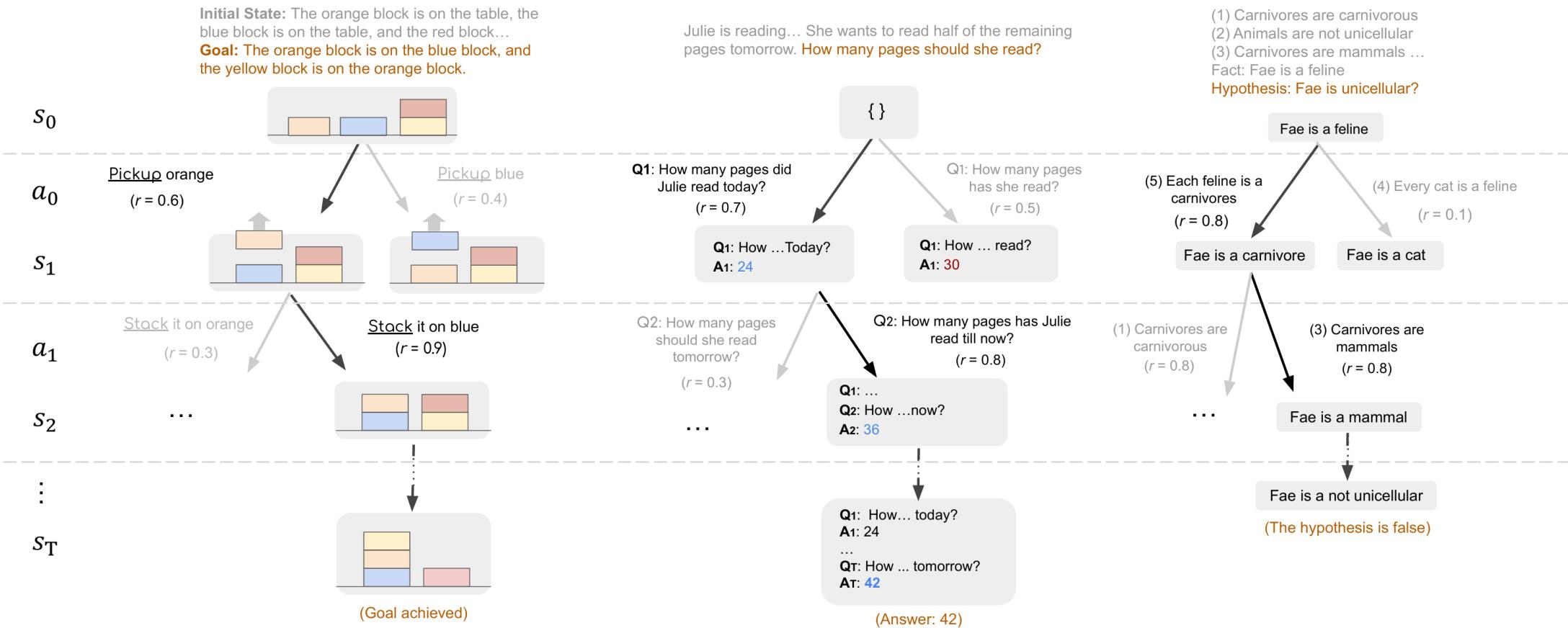
Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.



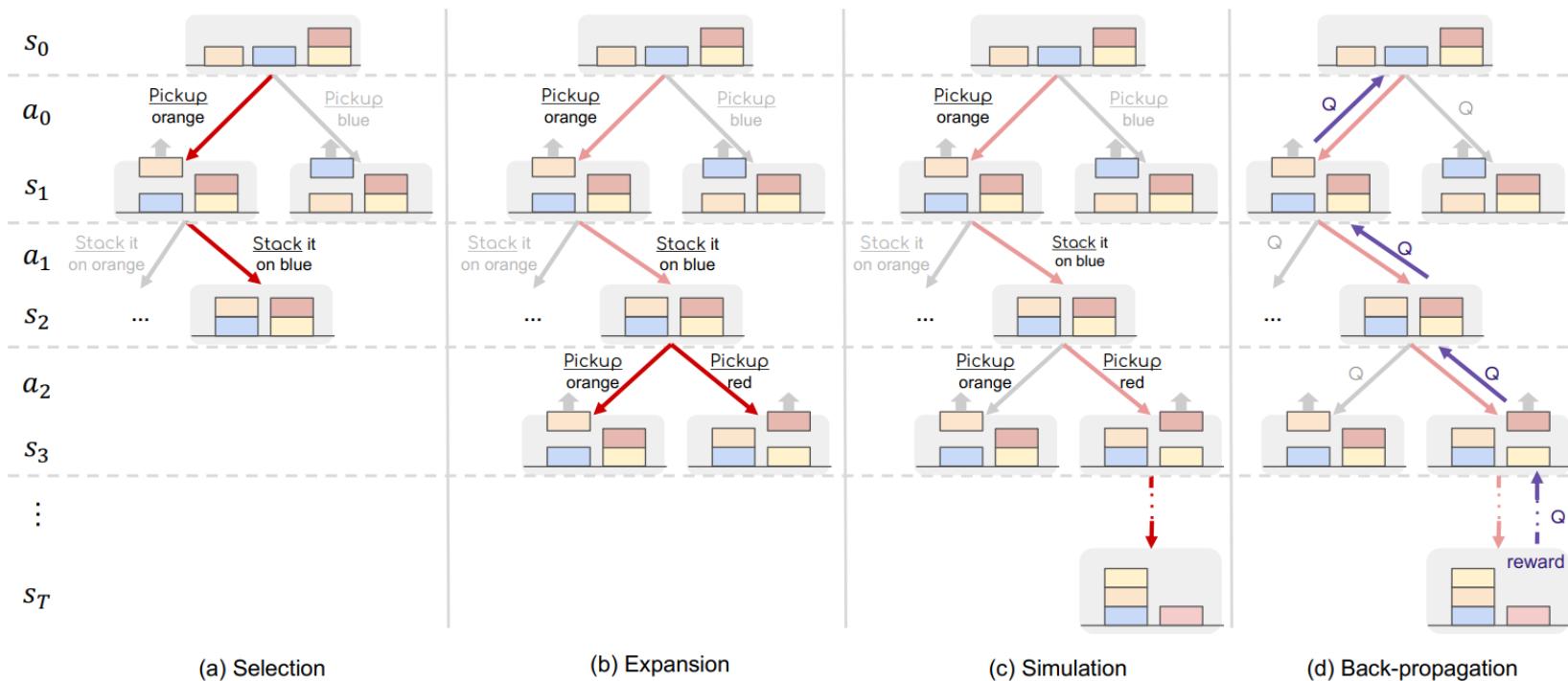
- Question (prompt) is the initial state, S_0
- Answer is the final state, S_T
- No state transition

Multi-Step NLIE: RAP



Multi-Step NLIE: RAP

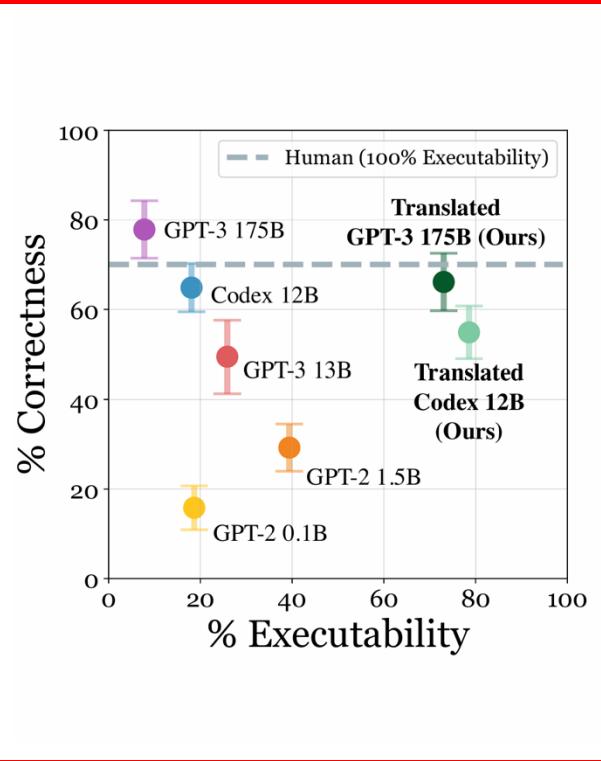
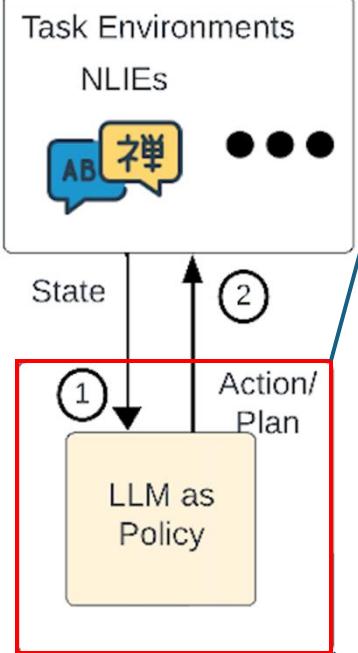
- RAP planning with Monte Carlo Tree Search
- Used LLM to generate different actions



LLM Profile Roles

	Prompting	Example Works	Example Prompts (in Appendix)
glm _{actor}	Few-shot	ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), MultiTool-CoT (Inaba et al., 2023)	Table 7, 8
glm _{planner}	Zero-shot	Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022)	Table 5
	Few-shot	DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022)	
glm _{evaluator}	Few-shot	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)	Table 10, 11
glm _{dynamic}	Few-shot	RAP (Hao et al., 2023)	Table 14

LLM Policy (Planner)



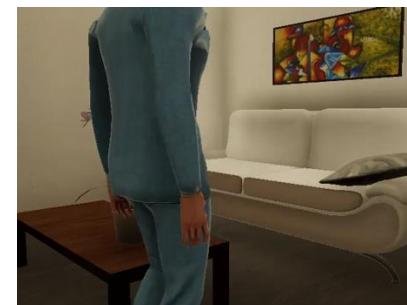
	Human	GPT-3 175B	Codex 12B	Translated Codex 12B
Task: Throw away paper	Task: Throw away paper	Task: Throw away paper	Task: Brush teeth	Task: Brush teeth
Step 1: Walk to home office	Step 1: Walk to home office	Step 1: Walk to bathroom	Step 1: Walk to bathroom	Step 1: Walk to bathroom
Step 2: Walk to wastebasket	Step 2: Walk to desk	Step 2: Walk to sink	Step 2: Open door	Step 2: Open trashcan
Step 3: Find wastebasket	Step 3: Find desk	Step 3: Find toothbrush	Step 3: Walk to sink	Step 3: Walk to trashcan
Step 4: Grab wastebasket	Step 4: Turn to desk	Step 4: Pick up toothbrush	Step 4: Put pot on sink	Step 4: Put brush on toothbrush
Step 5: Walk to desk	Step 5: Find chair	Step 5: Put toothbrush in mouth	Step 5: Turn to toothpaste	Step 5: Turn to toothpaste
Step 6: Drop paper in wastebasket	Step 6: Sit on chair	Step 6: Move brush around mouth for two minutes	Step 6: Put toothpaste on toothbrush	Step 6: Put teeth on toothbrush



Browsing internet



Empty dishwasher



Organize closet

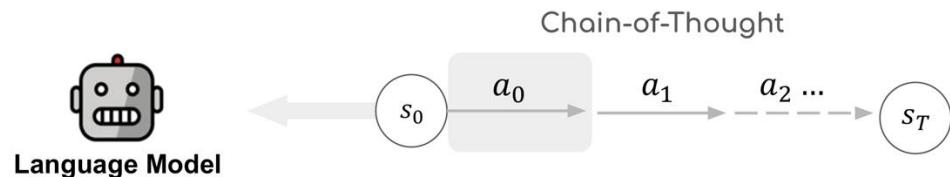


Wash face



Take off shoes

LLM Policy (Planner)



- An LLM planner under NLIE-QA, implemented by zero-shot Chain-of-Thought (CoT).
- Each **step** is considered as an **action**.

Prompting	Example Works	Example Prompts (in Appendix)	
glm _{actor}	Few-shot	ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), MultiTool-CoT (Inaba et al., 2023)	Table 7, 8
glm _{planner}	Zero-shot	Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022)	Table 5
	Few-shot	DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022)	
glm _{evaluator}	Few-shot	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)	Table 10, 11
glm _{dynamic}	Few-shot	RAP (Hao et al., 2023)	Table 14

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

A: Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step.

Plan:

Step 1: Calculate the total number of students who enrolled in contemporary and jazz dance.

Step 2: Calculate the total number of students who enrolled in hiphop dance.

Step 3: Calculate the percentage of students who enrolled in hiphop dance.

Solution:

Step 1: 20% of 20 students is 4 students. 25% of the remaining 16 students is 4 students. So, a total of 8 students enrolled in contemporary and jazz dance.

Step 2: The remaining 12 students enrolled in hip-hop dance.

Step 3: The percentage of students who enrolled in hip-hop dance is $12/20 = 60\%$.

Therefore, the answer (arabic numerals) is

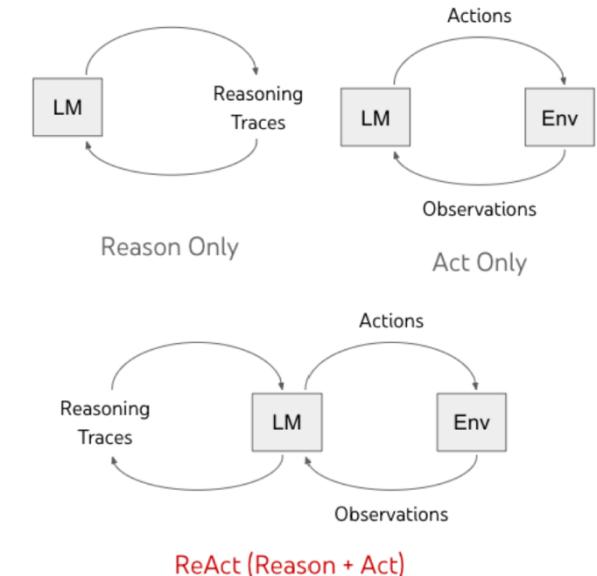
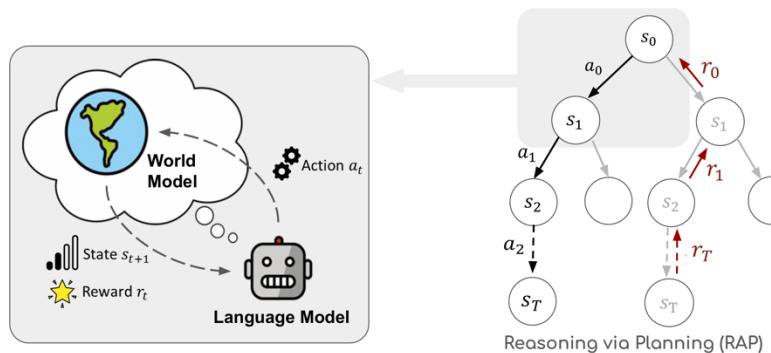
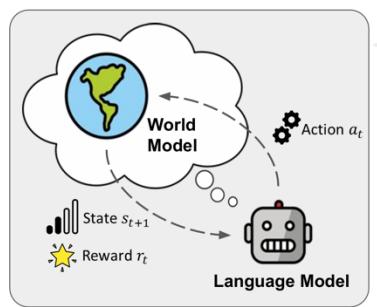
60%

LLM Policy (Actor)

Prompting	Example Works	Example Prompts (in Appendix)	
glm _{actor}	Few-shot MultiTool-CoT (Inaba et al., 2023)	ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), Table 7, 8	
glm _{planner}	Zero-shot	Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022)	
	Few-shot	DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022)	
glm _{evaluator}	Few-shot	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)	
glm _{dynamic}	Few-shot	RAP (Hao et al., 2023)	Table 14

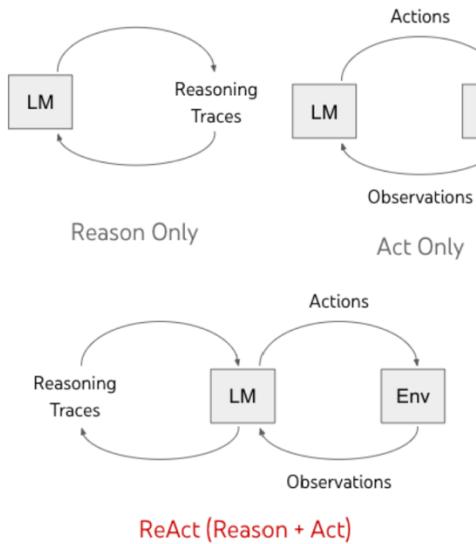
LLM Policy (Actor)

- Directly maps a state to a single action
- Early prompting frameworks for language generation tasks such as Chain-of-Thought
- For embodied tasks, ReAct employ actor

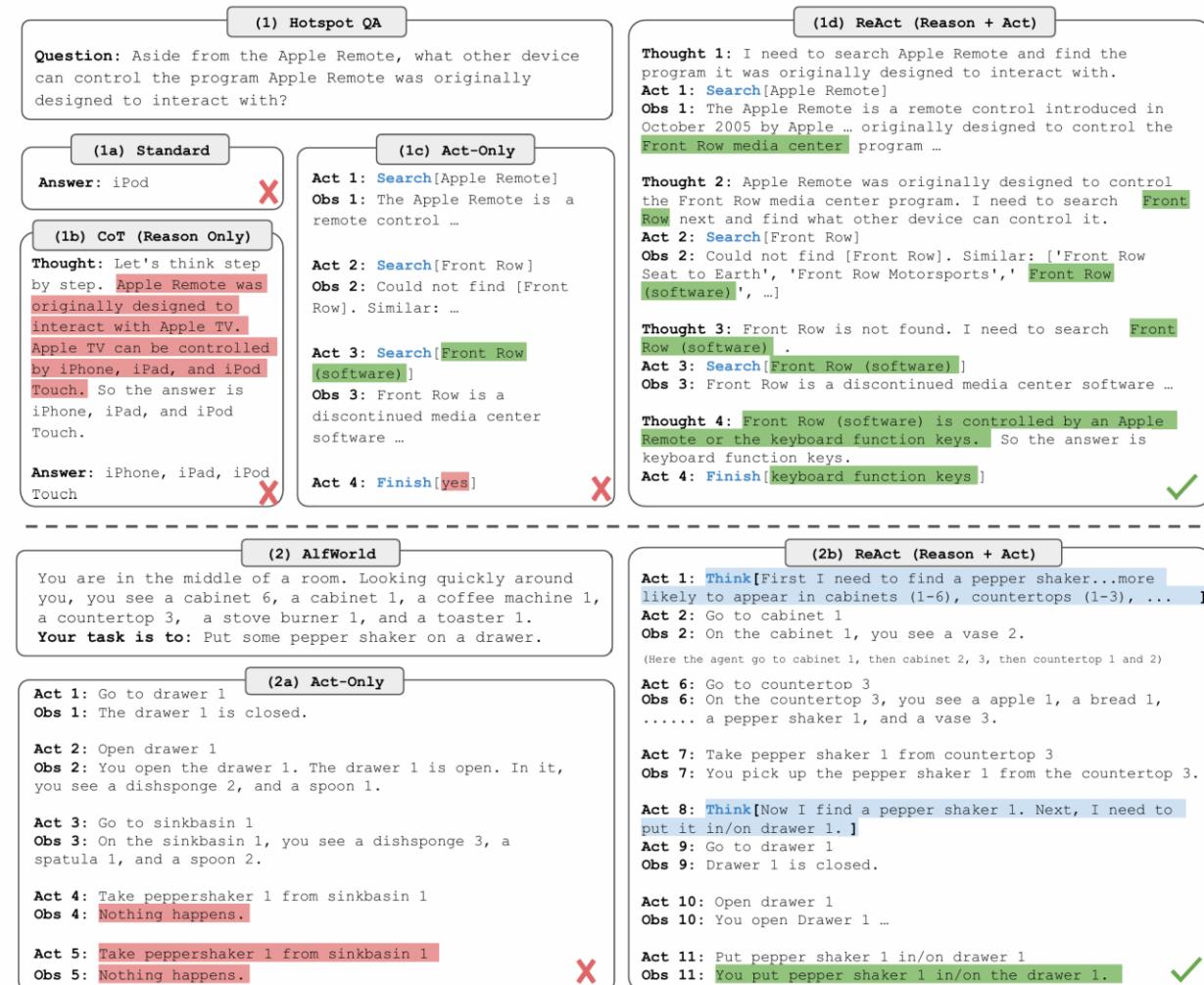


Prompting	Example Works	Example Prompts (in Appendix)
$\text{glm}_{\text{actor}}$	Few-shot ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), MultiTool-CoT (Inaba et al., 2023)	Table 7, 8

LLM Policy (Actor): ReAct



- CoT Prone to **hallucination**
- Act-Only model **unable to reason**
- ReAct uses both **reasoning and interaction** with the world model



LLM Policy (Actor)

Use the following tools at your disposal and answer some questions:

- «Calculator»
- «Chemical reaction predictor»
- «Molar mass list»

Q: Find the amount of H₂CO₃ formed on combining 2 moles of CO₂ and 2 moles of H₂O.

A: First, use the chemical reaction predictor to determine the chemical equation for the reaction.

Reactants: 2CO₂, 2H₂O

Products: ?H₂CO₃

«Chemical reaction predictor»



Second, use the molar mass list to determine the molar mass of H₂CO₃.

H₂CO₃ «Molar mass list» 62 g/mol

Third, use the calculator to calculate the amount of 2 moles of H₂CO₃. $2 \times 62 =$

«Calculator» 124

Therefore, 124 g of H₂CO₃ is formed.

LLM Policy (Actor)

Question: VIVA Media AG changed its name in 2004. What does their new acronym stand for?

Thought:

Let's think step by step. VIVA Media AG changed its name in 2004. The new acronym must stand for the new name they adopted. To find the answer, we need to research the new name of VIVA Media AG after 2004.

Action: Finish[Further research is needed to determine the new acronym.]

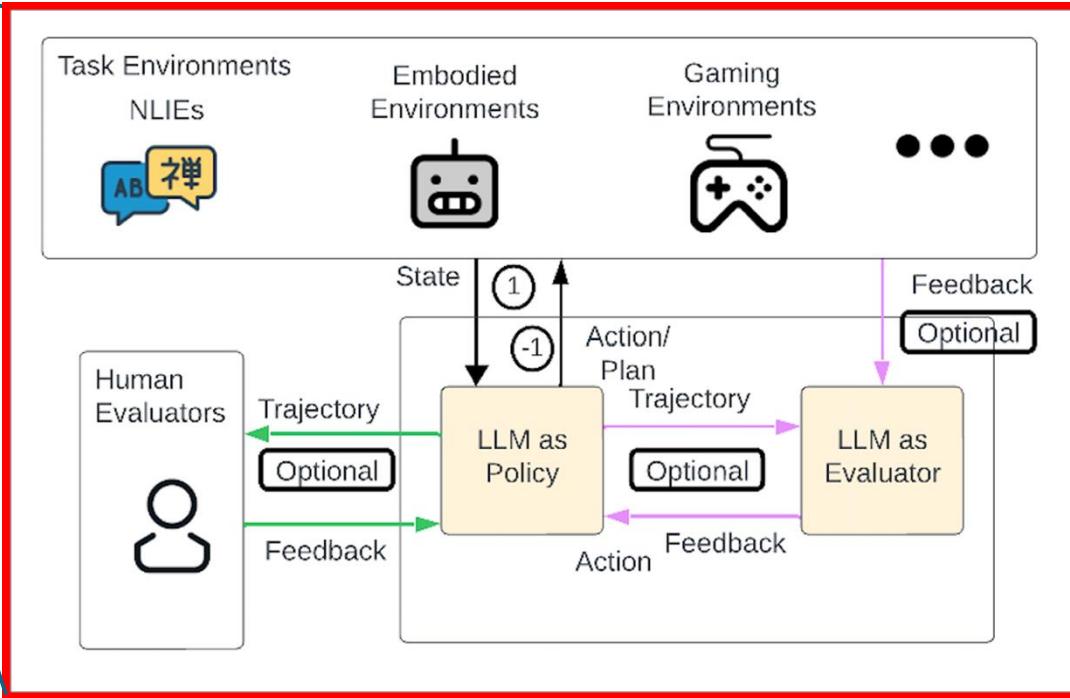
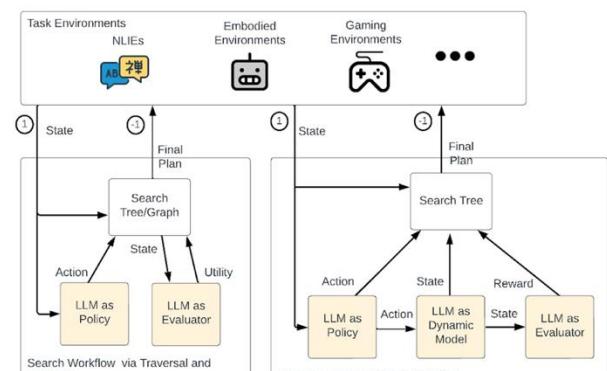
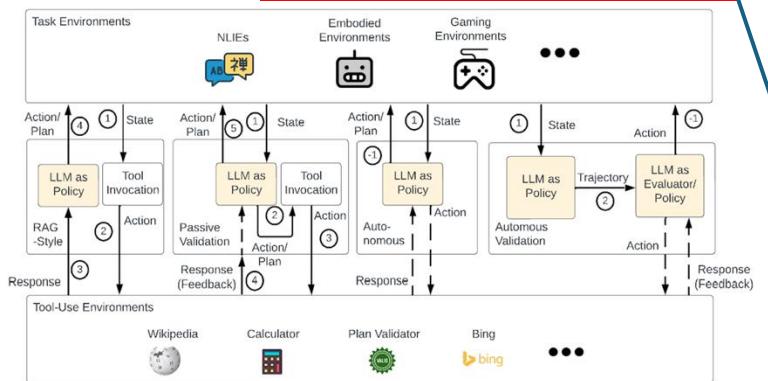
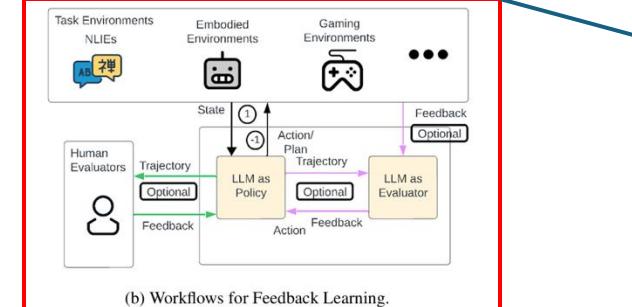
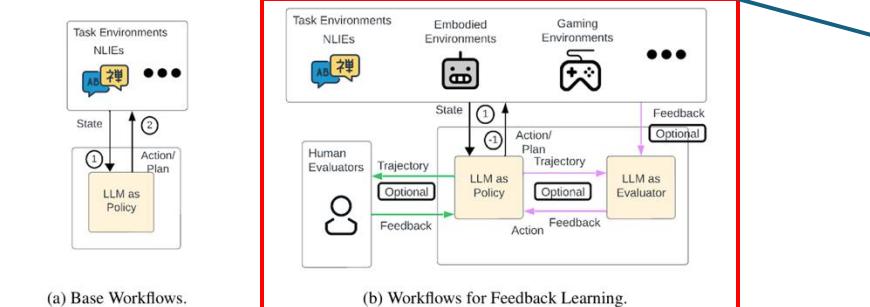
Action:

Finish[Further research is needed to determine the new acronym.]

LLM as Evaluator

Prompting	Example Works	Example Prompts (in Appendix)	
glm _{actor}	Few-shot	ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), MultiTool-CoT (Inaba et al., 2023)	Table 7, 8
glm _{planner}	Zero-shot	Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022)	Table 5
	Few-shot	DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022)	
glm _{evaluator}	Few-shot	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)	Table 10, 11
glm _{dynamic}	Few-shot	RAP (Hao et al., 2023)	Table 14

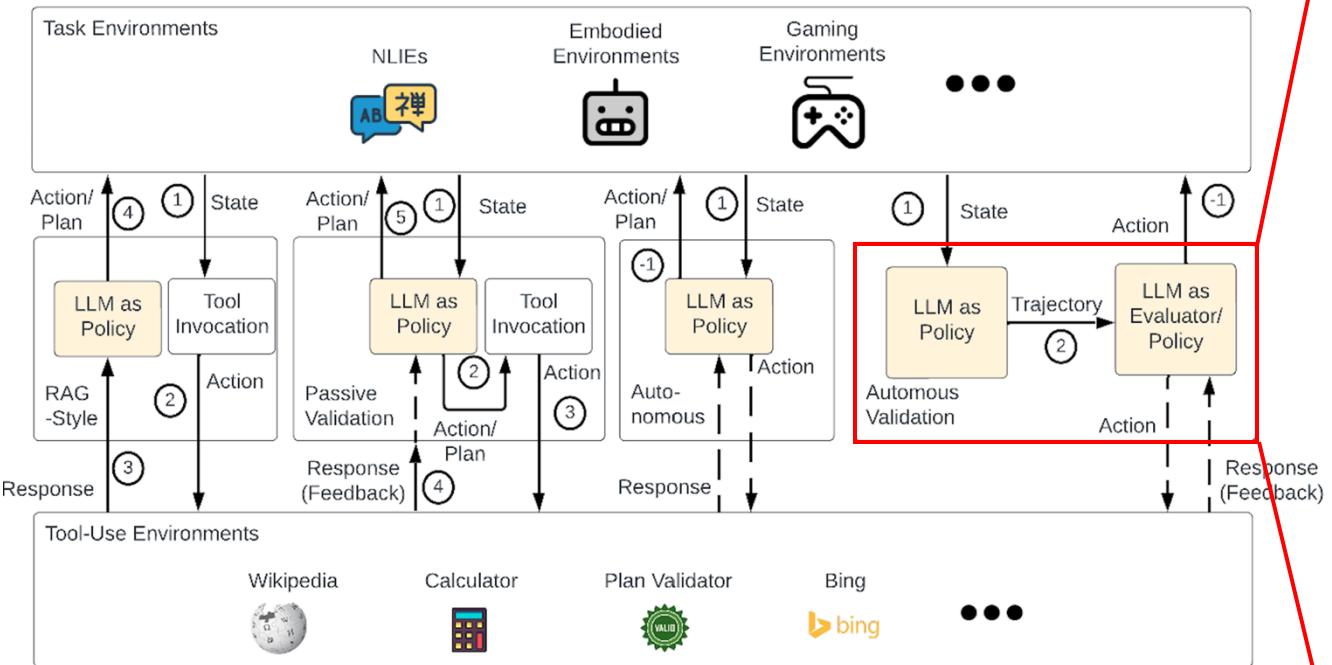
LLM as Evaluator



General Flow diagram of LLM with feedback loop

Prompting	Example Works	Example Prompts (in Appendix)
glm _{actor}	Few-shot ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), MultiTool-CoT (Inaba et al., 2023)	Table 7, 8
glm _{planner}	Zero-shot Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022)	Table 5
glm _{evaluator}	Few-shot DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022) Few-shot RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)	Table 10, 11
glm _{dynamic}	Few-shot RAP (Hao et al., 2023)	Table 14

LLM as Evaluator



{Few-shot Demonstrations Omitted for Brevity}

Question: Serianna is a band of what genre that combines elements of heavy metal and hardcore punk?

Proposed Answer: Let's think step by step. Serianna is a band of metalcore genre. Metalcore is a subgenre of heavy metal and hardcore punk. So Serianna is a band of heavy metal and hardcore punk. So the answer is: heavy metal and hardcore punk.

1. Plausibility:

The question asks for the genre that combines elements of heavy metal and hardcore punk, and the answer is "heavy metal and hardcore punk", simply repeat the question. So it's not plausible.

2. Truthfulness:

Let's search the question in google:

> **Search Query:** Serianna is a band of what genre that combines elements of heavy metal and hardcore punk?

> Evidence:

[Metalcore - Wikipedia] Metalcore is a fusion music genre that combines elements of extreme metal and hardcore punk.

The evidence suggests that metalcore is a genre that combines elements of extreme metal and hardcore punk, as stated in the proposed answer.

Let's search the proposed answer in google:

> **Search Query:** Serianna is a band of metalcore genre.

> Evidence:

[Serianna - Wikipedia] Serianna was a metalcore band from Madison, Wisconsin. The band formed in 2006...

The evidence suggests Serianna is a metalcore band, the proposed answer is correct about this.

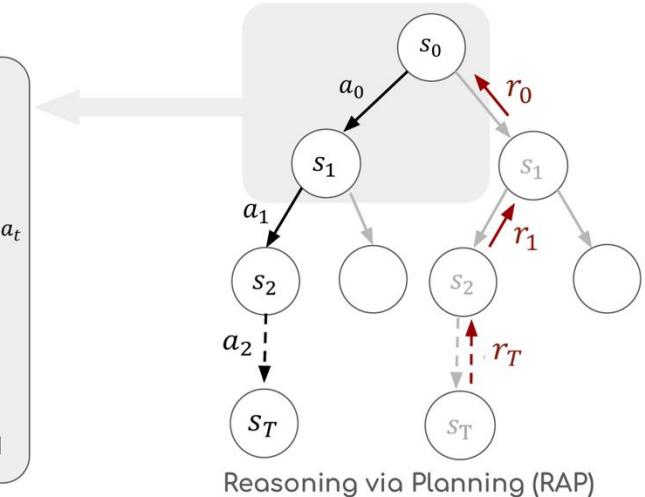
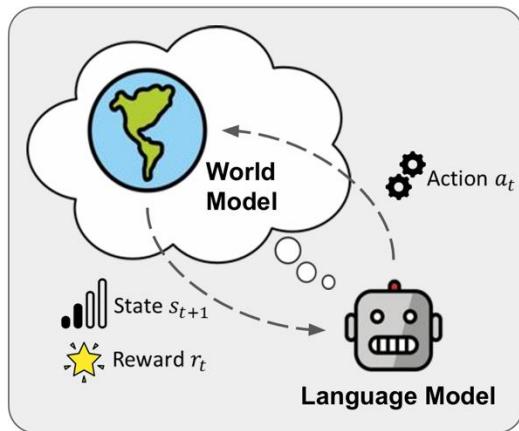
Above all, the proposed answer correctly identifies that Serianna is a band of the metalcore genre, which combines elements of heavy metal and hardcore punk. However, the final answer is not plausible since it just lists the genres that metalcore combines.

LLM as Dynamic Model

Prompting	Example Works	Example Prompts (in Appendix)
glm _{actor}	Few-shot MultiTool-CoT (Inaba et al., 2023)	ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), Table 7, 8
glm _{planner}	Zero-shot	Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022) Table 5
	Few-shot	DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022)
glm _{evaluator}	Few-shot	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024) Table 10, 11
glm _{dynamic}	Few-shot	RAP (Hao et al., 2023) Table 14

LLM as Dynamic Model

- Describe changes to the environment.
- Part of a world model that predict next state from current state and action.



	Prompting	Example Works	Example Prompts (in Appendix)
$\text{glm}_{\text{actor}}$	Few-shot	ReAct (Yao et al., 2023b), Reflexion (Shinn et al., 2023), RAP (Hao et al., 2023), MultiTool-CoT (Inaba et al., 2023)	Table 7, 8
$\text{glm}_{\text{planner}}$	Zero-shot	Plan-and-Solve (Wang et al., 2023a), LLM Planner (Huang et al., 2022)	Table 5
$\text{glm}_{\text{evaluator}}$	Few-shot	DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022)	
$\text{glm}_{\text{evaluator}}$	Few-shot	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)	Table 10, 11
$\text{glm}_{\text{dynamic}}$	Few-shot	RAP (Hao et al., 2023)	Table 14

LLM as Dynamic Model

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question: ".

Question 1: Weng earns \$12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?

Question 1.1: How much does Weng earn per minute?

Answer 1.1: Since Weng earns \$12 an hour for babysitting, she earns $\$12 / 60 = \0.2 per minute. The answer is 0.2.

Question 1.2: Now we can answer the question: How much did she earn?

Answer 1.2: Working 50 minutes, she earned $\$0.2 \times 50 = \10 . The answer is 10.

A Review of Prominent Paradigms for LLM-Based Agents: Tool Use (Including RAG), Planning, and Feedback Learning

2nd Half Presented By: Md. Mahir Ashhab (ftm2nu)

Overview of LLM-Profiled Roles (LMPRs)

The LMPR concept abstracts the internal mechanisms of LLM agents into three roles:

- **glm_{policy}** – the decision-making component, generating actions or plans
- **glm_{eval}** – an evaluator role, providing feedback or scoring candidate actions/states
- **glm_{dynamic}** – a world model, predicting state transitions

These roles serve as

- primitives for designing **universal workflows** that are agnostic to specific tasks or domains
- suitable across NLEs (natural language interaction environments), decision-making tasks, and embodied simulations.

Types	Subtypes	Universal LMPRs	Used For	Related Frameworks
Base	glm _{actor}	glm _{actor}	/	ReAct (Yao et al., 2023b), CoT (Wei et al., 2022)
	glm _{planner}	glm _{planner}	Planning	Huang et al. (2022) , DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022), Plan-and-solve (Wang et al., 2023a), OPEx (Shi et al., 2024a)
Tool-Use	RAG-Style (Passive)	glm _{policy}	Tool Use	RAG (Lewis et al., 2020 ; Shi et al., 2024b ; Wang et al., 2024b ; Zhang et al., 2024)
	Passive Validation	Tool Use, Feedback Learning	glm _{policy}	Guan et al. (2023)
	Autonomous	glm _{policy}	Tool-Use	MultiTool-CoT (Inaba et al., 2023), ReAct (Yao et al., 2023b), Active RAG Jiang et al. (2023)
Search	Autonomous Validation	glm _{policy} , glm _{eval}	Tool Use, Feedback Learning	CRITIC (Gou et al., 2024)
	Traversal & Heuristic	glm _{policy} , glm _{eval}	Planning	Tree-of-Thoughts (ToT) (Yao et al., 2023a), Tree-BeamSearch (Xie et al., 2023), Boost-of-Thoughts (Chen et al., 2024a), Graph-of-Thoughts (Besta et al., 2024), Tree-of-Traversal (Markowitz et al., 2024)
	Simulation-based (MCTS)	glm _{policy} , glm _{eval} , glm _{dynamic}	Planning	RAP (Hao et al., 2023), LLM-MCTS (Zhao et al., 2023), Wan et al. (2024), AgentQ (Putta et al., 2024), Chen et al. (2024b), Yu et al. (2023), Sprueill et al. (2023)
Feedback Learning	from glm _{eval} only	glm _{policy} , glm _{eval}	Feedback Learning	Reflexion (Shinn et al., 2023), Self-refine (Madaan et al., 2023), TextGrad (Yuksekgonul et al., 2024)
	from glm _{eval} & Task Env	glm _{policy} , glm _{eval}	Feedback Learning	Reflexion (Shinn et al., 2023)
	from Humans	glm _{policy}	Feedback Learning	CRITIC (Gou et al., 2024)

Table 3: Universal Workflows of LLM-Based Agents.

Base Workflows

Base workflows - simplest LLM-agent designs, involving direct execution of policy outputs:

- **Planners ($\text{glm}_{\text{planner}}$)**: Generate full plans upfront (e.g., Plan-and-Solve, DEPS, OPEx)
- **Actors ($\text{glm}_{\text{actor}}$)**: Produce immediate next-step actions (e.g., Chain-of-Thought, ReAct)

These workflows are particularly prominent in

- NLEs where interaction is often a single-step decision (e.g., QA tasks)
- sequential interactions with environments like ALFWorld.
- Chain-of-Thought prompting (Wei et al., 2022) is considered a base workflow
 - because the action space is static
 - no explicit feedback loop exists.



Example Prompt for Planner workflows

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

A: Let's first understand the problem and devise a plan to solve the problem.
Then, let's carry out the plan and solve the problem step by step.

Plan:

Step 1: Calculate the total number of students who enrolled in contemporary and jazz dance.
Step 2: Calculate the total number of students who enrolled in hiphop dance.
Step 3: Calculate the percentage of students who enrolled in hiphop dance.

Solution:

Step 1: 20% of 20 students is 4 students. 25% of the remaining 16 students is 4 students. So, a total of 8 students enrolled in contemporary and jazz dance.

Step 2: The remaining 12 students enrolled in hip-hop dance.

Step 3: The percentage of students who enrolled in hip-hop dance is $12/20 = 60\%$.

Therefore, the answer (arabic numerals) is

60%

Table 5: An LLM planner under NLIE-QA, implemented by zero-shot CoT. The CoT trigger is highlighted in red font. The content in the shadow box is generated by LLM. Source from Wang et al. (2023a)

Example Prompt for LLM Actor workflows

Interact with a household to solve a task. Here are two examples.

{Demo 1}

{Demo 2}

Here is the task: You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 5, a cabinet 4, a cabinet 3, a cabinet 2, a cabinet 1, a coffeemachine 1, a countertop 3, a countertop 2, a countertop 1, a drawer 3, a drawer 2, a drawer 1, a fridge 1, a garbagecan 1, a microwave 1, a shelf 3, a shelf 2, a shelf 1, a sinkbasin 1, a stoveburner 4, a stoveburner 3, a stoveburner 2, a stoveburner 1, and a toaster 1.

Your task is to: put a cool tomato in microwave.

>

think: To solve the task, I need to find a tomato, then cool it with the fridge, and finally put it in the microwave.

OK.

>

think: First, I need to find a tomato. A tomato is more likely to be found on the countertop (1-3), cabinet (1-6), drawer (1-3), shelf (1-3), or the sinkbasin. I can start by checking the countertop 1.

OK. >

go to countertop 1

On the countertop 1, you see a dishponge 1, and a mug 2. ...

Table 6: An LLM actor within base workflow under the Alfred World benchmark, implemented by few-shot CoT.
Source from (Yao et al., 2023b).

Tool-Use Workflows

Tool-use workflows

- expand the agent's capabilities by
 - incorporating external systems (retrievers, validators, calculators, etc.) into the decision loop.
- Categorized into
 - **Passive**
 - **Autonomous**

Passive Tool Use

- **RAG-Style (Retrieval-Augmented Generation):**
 - Tools are used to retrieve information before generation.
 - $\text{glm}_{\text{policy}}$ remains unaware of tools during generation.
 - Example: classic RAG, RePlug, Multi-Task Embedder.
- **Passive Validation:**
 - Tools validate outputs from $\text{glm}_{\text{policy}}$ after generation.
 - Does not affect the generation process in real time (e.g., Guan et al., 2023).

Autonomous Tool Use

Autonomous Tool Use

- The LLM itself decides
 - When and how to invoke tools
 - Necessitating tool-awareness in the prompt or memory.

Three types of usage

- **In-Generation Triggers:** Tools are invoked based on detected token patterns during generation (e.g., MultiTool-CoT; see Appendix Table 7).
- **Reasoning-Acting Strategy:** Inspired by ReAct (Yao et al., 2023b), each reasoning-action loop prompts the agent explicitly, enabling tighter integration.
- **Confidence-Based Invocation:** Tools are used if the LLM's generation confidence (e.g., token probability) falls below a threshold (e.g., Active RAG).
 - However, this method lacks tool specificity.

LLM actor within Tool- Use Workflow: In- generation Triggers

Use the following tools at your disposal and answer some questions:

- «Calculator»
- «Chemical reaction predictor»
- «Molar mass list»

Q: Find the amount of H₂CO₃ formed on combining 2 moles of CO₂ and 2 moles of H₂O.

A: First, use the chemical reaction predictor to determine the chemical equation for the reaction.

Reactants: 2CO₂, 2H₂O

Products: ?H₂CO₃

«Chemical reaction predictor»



Second, use the molar mass list to determine the molar mass of H₂CO₃.
H₂CO₃ **«Molar mass list»** 62 g/mol

Third, use the calculator to calculate the amount of 2 moles of H₂CO₃. $2 \times 62 =$
«Calculator» 124

Therefore, 124 g of H₂CO₃ is formed.

Q: Find the amount of Calcium hydroxide that is required to react with 2 moles of Carbon dioxide to form 2 moles of Calcium carbonate along with 2 moles of Water.

A:

Table 7: An LLM actor within tool-use workflow. In-generation triggers are highlighted in red font. Source from MultiTool-CoT (Inaba et al., 2023)

LLM actor within Tool- Use Workflow: Reasoning Acting Strategy

Solve a question answering task by having a Thought, then Finish with your answer. Thought can reason about the current situation. Finish[answer] returns the answer and finishes the task.
Here are some examples:

Question 1: What is the elevation range for the area that the eastern sector of the Colorado orogeny extends into?

Thought: Let's think step by step. The eastern sector of Colorado orogeny extends into the High Plains. High Plains rise in elevation from around 1,800 to 7,000 ft, so the answer is 1,800 to 7,000 ft.

Action: Finish[1,800 to 7,000 ft]

...

Question 6: Were Pavel Urysohn and Leonid Levin known for the same type of work?

Thought: Let's think step by step. Pavel Urysohn is a mathematician. Leonid Levin is a mathematician and computer scientist. So Pavel Urysohn and Leonid Levin have the same type of work.

Action: Finish[Yes]

END OF EXAMPLES)

Question: VIVA Media AG changed its name in 2004. What does their new acronym stand for?

Thought:

Let's think step by step. VIVA Media AG changed its name in 2004. The new acronym must stand for the new name they adopted. To find the answer, we need to research the new name of VIVA Media AG after 2004. Action: Finish[Further research is needed to determine the new acronym.]

Action:

Finish[Further research is needed to determine the new acronym.]

Table 8: An LLM actor within tool-use workflow for the HotPot-QA benchmark, implemented by few-Shot CoT₄₃
Source from ReAct (Yao et al., 2023b).

Autonomous Validation

- A hybrid workflow
- $\text{glm}_{\text{policy}}$ generates output
- glmeval autonomously decides whether to invoke tools for validation (e.g., CRITIC).
- This setup overlaps with feedback learning, revealing workflow entanglement.
- **Remark:** These validation workflows are often feedback-learning in disguise—reinforcing the broader claim that paradigms are not mutually exclusive but structurally linked.

Types	Subtypes	Universal LMPRs	Used For	Related Frameworks
Base	glm _{actor}	glm _{actor}	/	ReAct (Yao et al., 2023b), CoT (Wei et al., 2022)
	glm _{planner}	glm _{planner}	Planning	Huang et al. (2022) , DEPS (Wang et al., 2023b), Planner-Actor-Reporter (Dasgupta et al., 2022), Plan-and-solve (Wang et al., 2023a), OPEx (Shi et al., 2024a)
Tool-Use	RAG-Style (Passive)	glm _{policy}	Tool Use	RAG (Lewis et al., 2020 ; Shi et al., 2024b ; Wang et al., 2024b ; Zhang et al., 2024)
	Passive Validation	Tool Use, Feedback Learning	glm _{policy}	Guan et al. (2023)
	Autonomous	glm _{policy}	Tool-Use	MultiTool-CoT (Inaba et al., 2023), ReAct (Yao et al., 2023b), Active RAG Jiang et al. (2023)
Search	Autonomous Validation	glm _{policy} , glm _{eval}	Tool Use, Feedback Learning	CRITIC (Gou et al., 2024)
	Traversal & Heuristic	glm _{policy} , glm _{eval}	Planning	Tree-of-Thoughts (ToT) (Yao et al., 2023a), Tree-BeamSearch (Xie et al., 2023), Boost-of-Thoughts (Chen et al., 2024a), Graph-of-Thoughts (Besta et al., 2024), Tree-of-Traversal (Markowitz et al., 2024)
	Simulation-based (MCTS)	glm _{policy} , glm _{eval} , glm _{dynamic}	Planning	RAP (Hao et al., 2023), LLM-MCTS (Zhao et al., 2023), Wan et al. (2024), AgentQ (Putta et al., 2024), Chen et al. (2024b), Yu et al. (2023), Sprueill et al. (2023)
Feedback Learning	from glm _{eval} only	glm _{policy} , glm _{eval}	Feedback Learning	Reflexion (Shinn et al., 2023), Self-refine (Madaan et al., 2023), TextGrad (Yuksekgonul et al., 2024)
	from glm _{eval} & Task Env	glm _{policy} , glm _{eval}	Feedback Learning	Reflexion (Shinn et al., 2023)
	from Humans	glm _{policy}	Feedback Learning	CRITIC (Gou et al., 2024)

Table 3: Universal Workflows of LLM-Based Agents.

Search Workflows

Search workflows provide

- exploration capabilities for complex or long-horizon tasks
- addressing the limitations of greedy planning.

Traversal and Heuristic-Based Search

Traversal and Heuristic-Based Search

- Nodes are generated by $\text{glm}_{\text{policy}}$ and stored in a tree or graph (e.g., Tree-of-Thoughts, Tree-Beam Search).
- glm_{eval} selects nodes to expand using scoring or classification
- This setup supports
 - depth-first, breadth-first, or beam search.
 - Beam search retains top-N paths based on glm_{eval} scores, enabling broader exploration with limited compute.



LLM evaluator: Traversal and Heuristic- Based Search

Given a question and some sub-questions, determine whether the last sub-question is useful to answer the question. Output 'Yes' or 'No', and a reason.

Question 1: Four years ago, Kody was only half as old as Mohamed. If Mohamed is currently twice as 30 years old, how old is Kody?

Question 1.1: How old is Mohamed?

Question 1.2: How old was Mohamed four years ago?

New question 1.3: How old was Kody four years ago?

Is the new question useful? Yes. We need the answer to calculate how old is Kody now.

...

Question 5: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

New question 5.1: Now we can answer the question: How much in dollars does she make every day at the farmers' market?

Is the new question useful?

Table 10: An LLM evaluator within simulation-based search workflow for NLIE-QA, implemented by few-shot CoT. It assesses the usefulness of new sub-questions in solving the original question. Source from [Hao et al. \(2023\)](#)



LLM evaluator: Traversal and Heuristic- Based Search

Evaluate if given numbers can reach 24 (sure/likely/impossible)
10 14
$10 + 14 = 24$
sure
...
1 3 3
$1 * 3 * 3 = 9$
$(1 + 3) * 3 = 12$
1 3 3 are all too small
impossible
11, 12

Table 11: An LLM evaluator within Tree-of-Thought Workflow under Game 24, implemented by few-Shot CoT prompting. The LLM is profiled for multi-class classification. Source from ToT (Yao et al., 2023a).

Simulation-Based Search (MCTS)

Monte Carlo Tree Search (MCTS) introduces probabilistic simulation into the planning process:

- Nodes are expanded based on accumulated statistics (e.g., average rewards).
- The simulation phase involves glm_{policy} (action proposal), glm_{eval} (evaluation), and $glm_{dynamic}$ (state transition).
- Only the root action is executed, minimizing risk from speculative branches.
- Key examples include RAP, LLM-MCTS, and AgentQ.

LLM actor: Simulation -Based Search

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question:"

Question 1: James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?

Question 1.1: How many pages does he write every week?

Answer 1.1: James writes a 3-page letter to 2 different friends twice a week, so he writes $3 * 2 * 2 = 12$ pages every week. The answer is 12.

Question 1.2: How many weeks are there in a year?

Answer 1.2: There are 52 weeks in a year. The answer is 52.

Question 1.3: Now we can answer the question: How many pages does he write a year?

Answer 1.3: James writes 12 pages every week, so he writes $12 * 52 = 624$ pages a year. The answer is 624.

...

Question 5: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Question 5.1:

How many eggs does Janet have left after eating three for breakfast and baking muffins with four?

Table 9: An LLM actor for the GSM8K benchmark. Source from [Hao et al. \(2023\)](#).



LLM- Profiled Dynamic Model

Given a question, please decompose it into sub-questions. For each sub-question, please answer it in a complete sentence, ending with "The answer is". When the original question is answerable, please start the subquestion with "Now we can answer the question: ".

Question 1: Weng earns \$12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?

Question 1.1: How much does Weng earn per minute?

Answer 1.1: Since Weng earns \$12 an hour for babysitting, she earns $\$12 / 60 = \0.2 per minute. The answer is 0.2.

Question 1.2: Now we can answer the question: How much did she earn?

Answer 1.2: Working 50 minutes, she earned $\$0.2 \times 50 = \10 . The answer is 10.

...

Question 5: Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Question 5.1: How many eggs does Janet have left after eating three for breakfast and using four for muffins?

Answer 5.1:

Table 14: An LLM-Profiled Dynamic Model.

Feedback-Learning Workflows

Feedback-learning workflows close the loop by allowing agents to refine their decisions using feedback. Three main feedback sources exist:

- **Internal (glm_{eval})**
 - Provides reflection (e.g., Self-Refine, Reflexion).
 - Often produces free-form textual feedback
- **Task Environment:**
 - Rewards or state changes inform learning (e.g., Reflexion, where failures induce “self-reflection”).
- **External Tools or Humans:**
 - Either used directly or mediated via glm_{eval} (e.g., CRITIC, Guan et al.).

LLM evaluator within Feedback- Learning Workflow

[Few-shot Demonstrations Omitted for Brevity]

Question: Serianna is a band of what genre that combines elements of heavy metal and hardcore punk?

Proposed Answer: Let's think step by step. Serianna is a band of metalcore genre. Metalcore is a subgenre of heavy metal and hardcore punk. So Serianna is a band of heavy metal and hardcore punk. So the answer is: heavy metal and hardcore punk.

1. Plausibility:

The question asks for the genre that combines elements of heavy metal and hardcore punk, and the answer is "heavy metal and hardcore punk", simply repeat the question. So it's not plausible.

2. Truthfulness: Let's search the question in google:

> Search Query: Serianna is a band of what genre that combines elements of heavy metal and hardcore punk?

> Evidence:

[Metalcore - Wikipedia] Metalcore is a fusion music genre that combines elements of extreme metal and hardcore punk.

The evidence suggests that metalcore is a genre that combines elements of extreme metal and hardcore punk, as stated in the proposed answer.

Let's search the proposed answer in google:

> Search Query: Serianna is a band of metalcore genre.

> Evidence:

[Serianna - Wikipedia] Serianna was a metalcore band from Madison, Wisconsin. The band formed in 2006...

The evidence suggests Serianna is a metalcore band, the proposed answer is correct about this.

Above all, the proposed answer correctly identifies that Serianna is a band of the metalcore genre, which combines elements of heavy metal and hardcore punk. However, the final answer is not plausible since it just lists the genres that metalcore combines.

Table 13: An LLM evaluator within the Feedback-Learning workflow (feedback from tools). In-generation triggers are highlighted in red font, and tool-generated content is highlighted in green font. Source from Gou et al. (2024).

Comparative Discussions

Plan Generation Approaches

- **glm_{planner} (Base Workflow):** Greedy, static planning. Prone to failures in long-horizon tasks due to lack of exploration.
- **Search Workflows:** Explore alternatives, backtrack, and improve robustness. MCTS offers dynamic adaptability by discarding invalid subtrees.

glm_{actor} Usage Variants

- **Task Actions (Base/Feedback):** Immediate execution
- **Planning Actions (Search):** Tree expansion
- **Tool Actions (Tool-Use):** Trigger or execute tools autonomously

glm_{eval} Functional Differences

- In **feedback learning**, outputs are free-form and reused to regenerate decisions.
- In **search**, outputs are numerical or classification scores for node selection.

Types of LLM-Profiled Evaluators According to Task Formulation and Feedback Types

Task Formulation	Feedback Types	Applicable Workflows	Example Works
Text Generation	Free-form reflection	Feedback-learning workflows	Self-Refine (Madaan et al., 2023), Reflexion (Shinn et al., 2023), CRITIC (Gou et al., 2024)
Binary/Multi-class Classification	Discrete values	Search workflows	RAP (Hao et al., 2023), Tree-BeamSearch (Xie et al., 2023), ToT (Yao et al., 2023a), Koh et al. (2024)
Binary Classification	Continuous values (logits)	Search workflow via MCTS	RAP (Hao et al., 2023)
Multi-choice QA	Choices of top-N actions	Search workflows via traversal and heuristic	ToT (Yao et al., 2023a)

Table 4: Types of LLM-Profiled Evaluators According to Task Formulation and Feedback Types

Limitations and Future Directions

Limitation 1: Unified Workflows for Base + Tool Use

- While ReAct unifies base and tool workflows using alternating reasoning-acting steps, task-specific dependencies persist.
- In QA, prompt templates are fixed.
- In embodied tasks, decision sequencing is dynamic.

Thus, a fully general unified workflow remains elusive.

Limitations and Future Directions

Limitation 2: No Universal Tool Use Framework

- Despite the theoretical appeal, tool use today remains highly specialized (e.g., calculators for math, retrievers for QA).
- Developing declarative tool interfaces and profiling strategies will be essential for general-purpose tool integration.

Conclusion

This work is especially valuable for researchers aiming to:

- Compare frameworks in a task-agnostic manner
- Combine paradigms to create more powerful hybrid agents
- Understand design trade-offs based on the interaction of $\text{glm}_{\text{policy}}$, glm_{eval} , and $\text{glm}_{\text{dynamic}}$

rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking

Presenters-

Rishov Paul (vst2hb)

Radowan Mahmud Redoy (snf4za)

Rishov Paul (vst2hb)

Problem Statement

- LLMs struggle with complex multi-step reasoning tasks (e.g., Olympiad-level problems).
- **System 1 vs System 2 Reasoning:**
 - **System 1:** Generating complete solutions in a single inference. Fast, error-prone reasoning.
 - **System 2:** Deliberate, slower, deeper reasoning (human-like reasoning).
- **Problem:** Single-step reasoning results in errors, requiring more robust models to handle step-by-step thinking.

Motivation

- High-quality math reasoning data is scarce.
- **Synthesizing math data** faces challenges in distinguishing correct and erroneous reasoning steps.
- Eliminating low-quality data is difficult without accurate feedback.
- **Human labeling** for step-by-step feedback is resource-intensive and hard to scale.
- Distill-based data synthesis, like GPT-4-distilled CoT data, has diminishing returns and **cannot surpass teacher model capabilities**.

Key Contributions of rStar-Math

- rStar-Math is a self-evolvable **System 2-style** reasoning approach.
- Achieves state-of-the-art math reasoning, outperforming **OpenAI o1** on challenging benchmarks with a 7B parameter model.
- Utilizes smaller language models (SLMs) with Monte Carlo Tree Search (MCTS) for **self-evolutionary data generation**.

Math Data Synthesis Challenges and Limitations

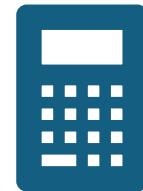
- **Current Advancements:** Math reasoning largely relies on curating high-quality CoT data using GPT-distilled models like GPT-4.
- **Limitations:**
 - Reasoning capabilities limited by the teacher LLM.
 - Problems unsolvable by the teacher LLM are excluded from training.
 - Error-prone intermediate steps in solvable problems are hard to detect.
- **Scaling Issues:**
 - Rejection sampling improves quality but doesn't guarantee correct reasoning steps.
 - Scaling CoT data results in diminishing returns (e.g., OpenMathInstruct-2 with only a **3.9%** boost despite **8x** dataset increase).

Scaling Test-time Compute for Math Reasoning



Scaling Laws:

New scaling methods allow LLMs to improve by generating multiple samples and selecting the best solution using reward models.



Challenges:

Open-source methods for scaling test-time computation show limited math reasoning gains.

Performance often limited by policy LLM or reward model issues.



rStar-Math Contribution:

Addresses limitations by iteratively evolving the policy LLM and reward model.

Achieves System 2-level reasoning performance, comparable to OpenAI o1.

MCTS Prompt for our policy SLM

MCTS prompt for next step generation:

<userl>:

{input question}

<assistant>: Let's think step by step and solve the problem with code. {trajectory of previous steps}

SFT Prompt Example:

<userl>:

How many numbers can you get by multiplying two or more distinct members of the set 1, 2, 3, 5, 11 together?

<assistant>: Let's think step by step and solve the problem with code.

<code>

Step 1: Create the set of numbers

```
numbers = {1, 2, 3, 5, 11}
```

<end_of_step>

Step 2: Define a function to generate all possible products of two or more distinct members of the set

```
from itertools import combinations
```

def generate_products(numbers):

products = set()

for r in range(2, len(numbers) + 1):

For combinations of 2, 3, 4, and 5 elements
for combo in combinations(numbers, r):

product = 1

for num in combo:

product *= num

products.add(product)

return products

<end_of_step>

Step 3: Calculate the products

```
unique_products = generate_products(numbers)
```

<end_of_step>

Step 4: Calculate the number of unique products

```
num_unique_products = len(unique_products)
```

<end_of_step>

Now print the final answer

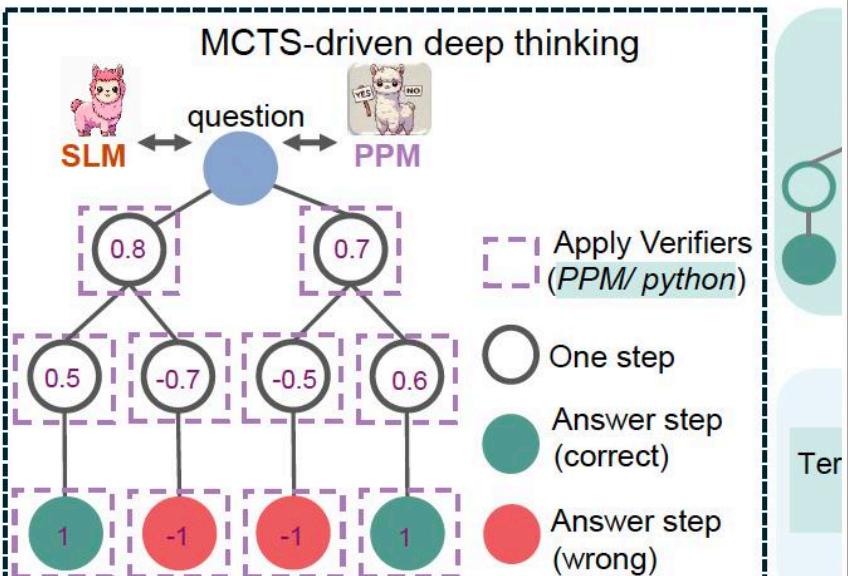
```
print(num_unique_products)
```

<end_of_code>

<output>15<end_of_output>

<answer>From the result, we can see that there are 15 unique products.

<end_of_answer>



(a) step-by-step verified reasoning trajectory

Figure 1

Reward Models and the Process Preference Model (PPM)

- **Reward models are essential but challenging to obtain for complex reasoning tasks.**
- **Step-Level Annotations:**
 - Collecting step-level annotations remains a significant obstacle.
 - Current approaches like MCTS struggle to generate precise reward scores, limiting performance.
- **rStar-Math Innovation:**
 - Introduces the Process Preference Model (PPM), eliminating the need for accurate step-level reward score annotations.
 - Enhances reasoning quality by leveraging iterative improvements.

Methodology Design Choices

Monte Carlo Tree Search for System 2 Reasoning

Goal: Train a math policy SLM and a process reward model (PRM) integrated with Monte Carlo Tree Search (MCTS) for System 2 reasoning.

- **Why MCTS?**

1. Simplifies Complex Problems: Breaks down complex math problems into multiple single-step tasks, easing the difficulty for the policy SLM.

- Compared to methods like Best-of-N or self-consistency, which require full solution generation in one inference.

2. Step-by-Step Training Data:

- MCTS naturally provides step-level training data for both the policy SLM and PRM.
- Standard MCTS rollout assigns Q-values to each step, avoiding the need for human annotations.

Key Challenges

Limited Capability of 7 B SLMs

- Smaller models struggle to solve complex tasks compared with GPT-4–class models.

Error-Prone Self-Generated Data

- Incorrect final answers are common; intermediate reasoning steps are often flawed or low-quality.

Sparse Success on Hard Problems

- Baseline SLMs solve relatively few challenging examples, reducing the diversity of useful training examples.

Cost-Quality Trade-off

- Exhaustive search or manual annotation is impractical; naive self-training risks amplifying model mistakes.

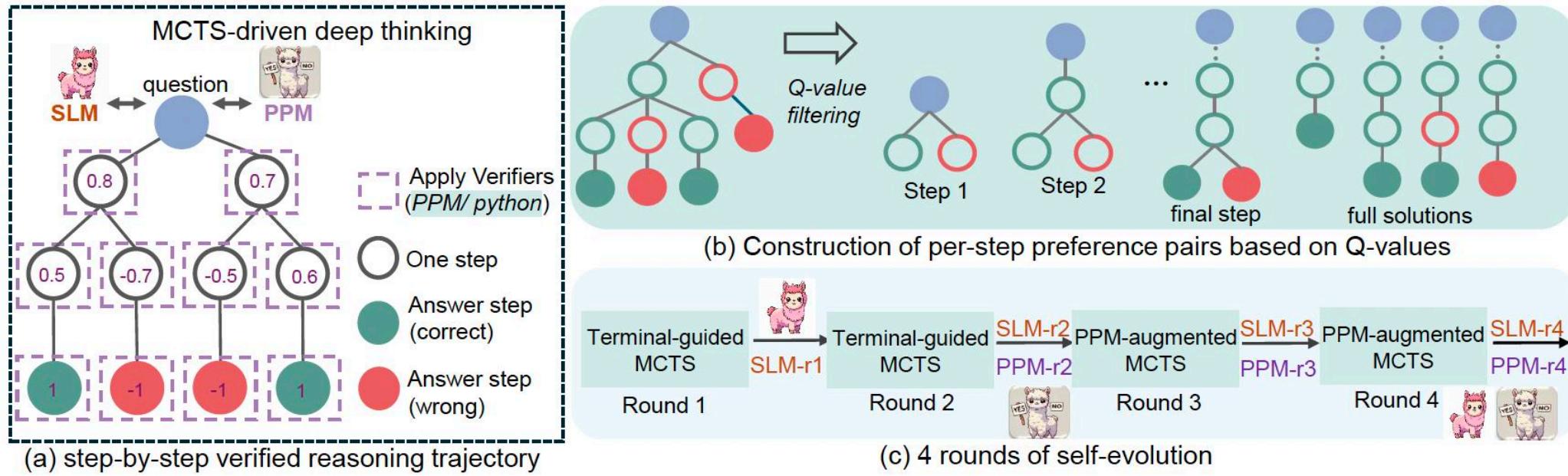


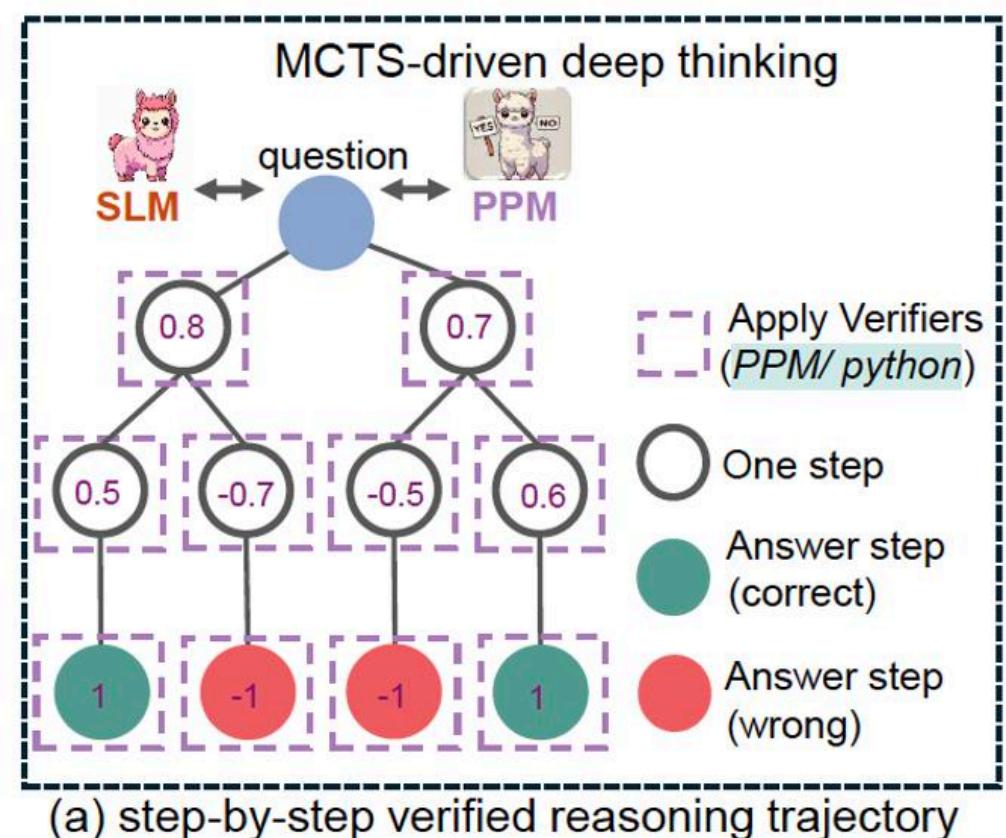
Figure 1: The overview of rStar-Math.

Proposed Approach

- **Dual 7 B SLMs:** policy model + process reward model (PRM)
- **Code-augmented CoT + MCTS:** verified, Q-valued reasoning traces
- **Four-Round Self-Evolution:** Iteratively upgrade both the policy SLM and the reward model, enabling the system to tackle progressively harder problems and yield higher-quality data.
- **Process Preference Model (PPM)** scores whole trajectories, no per-step labels
- **Compute-Efficient:** fits on 4×40 GB A100 GPUs — practical for most labs

Step-by-Step Verified Reasoning Trajectory

- **Objective:** Generate step-by-step solution trajectories annotated with per-step Q-values to evaluate reasoning quality.
- **Input:** Problem x and policy model M .
- **Search:** Run Monte Carlo Tree Search to build a reasoning tree.
- **Tree:** Root is question x ; children are intermediate steps; each step gets $Q(s_i)$; a root-to-leaf path forms trajectory $t = x \oslash s_1 \oslash \dots \oslash s_d$.
- **Extraction:** Collect all root-to-leaf paths as set $T = \{t_1, t_2, \dots, t_n\}$.
- **Filtering:** Use **code-augmented Chain-of-Thought synthesis** to remove low-quality trajectories and extensive rollouts to refine Q-values.
- **Outcome:** Curated training set of high-quality, Q-valued reasoning trajectories for supervised or RL fine-tuning.



Code-Augmented Chain-of-Thought (CoT) Generation

- **Input & Root Node**

Begin with the original problem statement x . Treat x as the root node of the Monte-Carlo Tree Search (MCTS).

- **Initialize Search Parameters**

- Choose the exploration constant c .
- Decide the total iteration / rollout budget R (how many MCTS cycles you will run).

- **Selection Phase (tree policy)**

- From the root, repeatedly pick the child node s that maximizes the UCT score

$$\text{UCT}(s) = Q(s) + c \sqrt{\frac{\ln N_{\text{parent}}(s)}{N(s)}}; \quad \text{where} \quad Q(s) = \frac{q(s)}{N(s)} \quad (1)$$

where $N(s)$ denotes the number of visits to node s , and $N_{\text{parent}}(s)$ is the visit count of s 's parent node. The predicted reward $q(s)$ is provided by the PPM and will be updated through back-propagation. c is a constant that balances exploitation and exploration.

Question: Bill walks $\frac{1}{2}$ mile south, then $\frac{3}{4}$ mile east, and finally $\frac{1}{2}$ mile south. How many miles is he, in a direct line, from his starting point? Express your answer as a decimal to the nearest hundredth.

```
# Step 1: Calculate the total distance walked south → NL CoT as Python Comment
total_south = 1/2 + 1/2
# Step 2: Calculate the total distance walked east
total_east = 3/4
# Step 3: Use the Pythagorean theorem to find the direct distance from the starting point
import math
direct_distance = math.sqrt(total_south**2 + total_east**2)
# Step 4: Round the direct distance to the nearest hundredth
direct_distance_rounded = round(direct_distance, 2)
From the result, we can see that the direct distance from the starting point is \boxed{1.25} miles
```

Python code execution for step 1:

```
# Step 1: Calculate the total distance walked south
total_south = 1/2 + 1/2
```

Python code execution for step 2:

```
# Step 1: Calculate the total distance walked south
total_south = 1/2 + 1/2
# Step 2: Calculate the total distance walked east
total_east = 3/4
...
```

Figure 2: An example of Code-augmented CoT.

- **Expansion Phase**

- Collect the current reasoning trajectory
 $x \oplus s_1 \oplus s_2 \oplus \dots \oplus s_{i-1}$
- **Prompt the policy model** with this trajectory to produce **n candidate next steps**
 $s_{i,0}, \dots, s_{i,n-1}$
- Each candidate consists of:
 - an NL sub-step (as a Python comment), and
 - a Python code snippet meant to carry out that sub-step.

- **Code-Execution Filtering**

- Concatenate the Python code from all prior accepted steps with each new candidate's code.
- Execute the resulting script.
- **Discard** any candidate whose code raises an error; keep only those that run to completion.

continued

MCTS prompt for next step generation:

<user>:

{input question}<assistant>: Let's think step by step and solve the problem with code. **{trajectory of previous steps}****SFT Prompt Example:**

<user>:

How many numbers can you get by multiplying two or more distinct members of the set 1, 2, 3, 5, 11 together?

<assistant>: Let's think step by step and solve the problem with code.

<code>

Step 1: Create the set of numbers

numbers = {1, 2, 3, 5, 11}

<end_of_step>

Step 2: Define a function to generate all possible products of two or more distinct members of the

from itertools import combinations

def generate_products(numbers):

products = set()

for r in range(2, len(numbers) + 1):

For combinations of 2, 3, 4, and 5 elements

for combo in combinations(numbers, r):

product = 1

for num in combo:

product *= num

products.add(product)

return products

<end_of_step>

Step 3: Calculate the products

unique_products = generate_products(numbers)

<end_of_step>

Step 4: Calculate the number of unique products

num_unique_products = len(unique_products)

<end_of_step>

Now print the final answer

print(num_unique_products)

<end_of_code>

<output>15<end_of_output>

<answer>From the result, we can see that there are 15 unique products.

<end_of_answer>

Prompt examples

continued

- **Scoring with the PPM**

- For every surviving candidate node, ask the PPM to predict its reward $q(s)$
- Update the node's running average

$$Q(s) = \frac{q(s)}{N(s)}$$

where $N(s)$ denotes the number of visits to node s ,
The predicted reward $q(s)$ is provided by the PPM

- **Back-Propagation**

- Walk back up the tree from the expanded node to the root.
- Increment visit counts $N(\cdot)$ and update cumulative value estimates with the obtained reward.

continued

- **Iterate**

Repeat **Selection** → **Expansion** → **Execution** → **Back-Propagation** until the rollout budget R is exhausted (or another stopping criterion is met).

- **Extract the Final CoT & Answer**

- Select the path with the highest value (e.g., greatest visit count or best Q).
- Return:
 - the full NL chain-of-thought (comments),
 - the executable Python code that produced the result, and
 - the final answer generated by that code.

Process Reward Models and Challenges

Process reward models are essential for solving challenging math problems by providing granular step-level reward signals.

Current Methods:

Human annotations and MCTS-generated scores are used to assign a score to each step.

Training uses methods like MSE loss or pointwise loss to minimize the gap between predicted and labeled scores.

Main Challenge:

Precise per-step scoring is difficult.

Ranking and scoring fine-grained steps (correct or incorrect) is particularly complex.

Process Preference Model (PPM) - Novel Training Method

- **New Approach:**

- **Step-Level Positive-Negative Pairs:** We train a Process Preference Model (PPM) using preference pairs instead of direct Q-values.

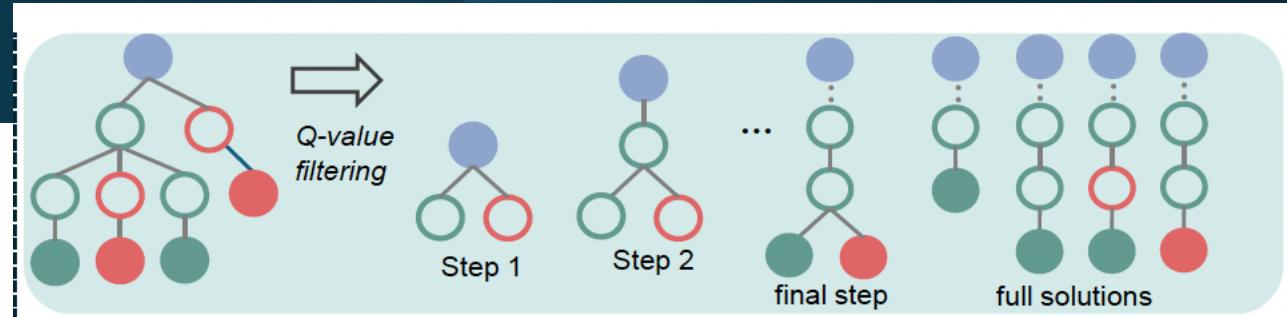
- **Generation Process:**

- **Positive Steps:** Two highest Q-value steps that lead to a correct answer.
- **Negative Steps:** Two lowest Q-value steps that lead to an incorrect answer.

- **Final Step Relaxation:**

- For the final answer step, positive pairs are selected based on the highest average Q-values, while negative pairs are from incorrect trajectories with the lowest Q-values.

- **Advantages:** This method helps overcome challenges related to ranking fine-grained steps and reduces noise in training data.



(b) Construction of per-step preference pairs based on Q-values

Extensive Rollouts for Q-value Annotation

Accurate Q-value $Q(s)$ annotation is crucial for guiding MCTS node selection towards correct problem-solving paths and identifying high-quality steps within trajectories.

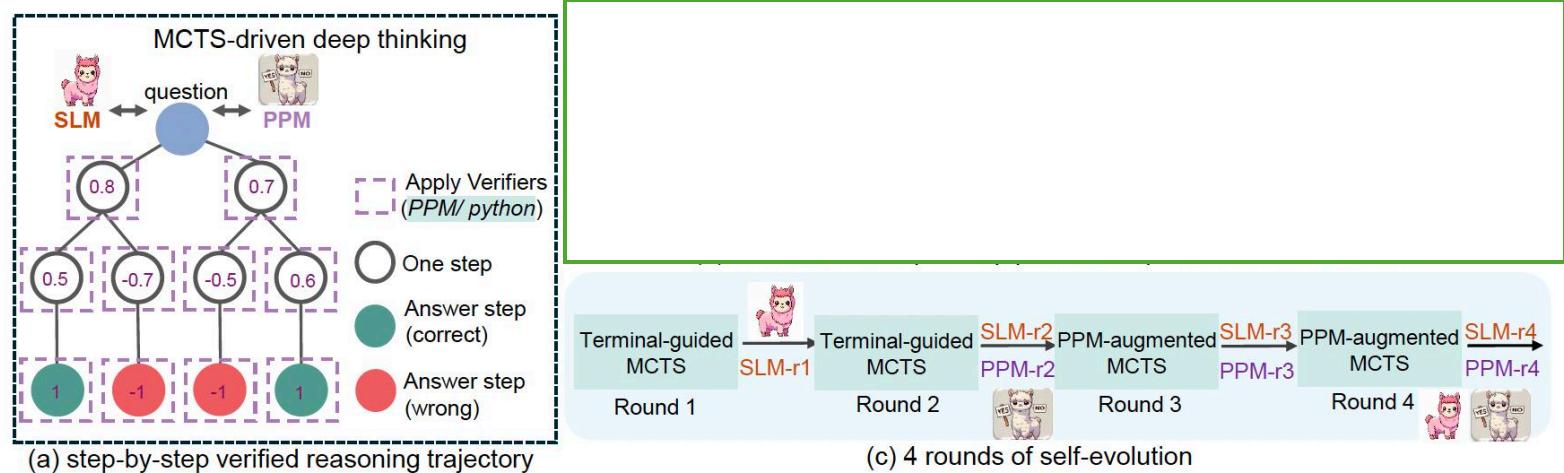


Figure 1: The overview of rStar-Math.

Terminal-guided annotation. During the first two rounds, when the PPM is unavailable or insufficiently accurate, we use terminal-guided annotation. Formally, let $q(s_i)^k$ denote the q value for step s_i after back-propagation in the k^{th} rollout. Following AlphaGo [Silver et al., 2017] and rStar [Qi et al., 2024], we score each intermediate node based on its contribution to the final correct answer:

$$q(s_i)^k = q(s_i)^{k-1} + q(s_d)^k; \quad (2)$$

where the initial q value $q(s_i)^0 = 0$ in the first rollout. If this step frequently leads to a correct answer, its q value will increase; otherwise, it decreases. Terminal nodes are scored as $q(s_d) = 1$ for correct answers and $q(s_d) = -1$ otherwise, as shown in Fig. 1.

PPM-augmented annotation. Starting from the third round, we use PPM to score each step for more effective generation. Compared to terminal-guided annotation, which requires multiple rollouts for a meaningful q value, PPM directly predicts a non-zero initial q value. PPM-augmented MCTS also helps the policy model to generate higher-quality steps, guiding solutions towards correct paths. Formally, for step s_i , PPM predicts an initial $q(s_i)^0$ value based on the partial trajectory:

$$q(s_i)^0 = PPM(x \oplus s_1 \oplus s_2 \oplus \dots \oplus s_{i-1} \oplus s_i) \quad (3)$$

This q value will be updated based on terminal node's $q(s_d)$ value through MCTS *back-propagation* in Eq. 2. For terminal node s_d , we do not use PRM for scoring during training data generation. Instead, we assign a more accurate score based on ground truth labels as terminal-guided rewarding.

Radowan Mahmud Redoy (snf4za)

Self-Evolved Deep Thinking



rStar-Math trains small language models (SLMs) to become strong math solvers

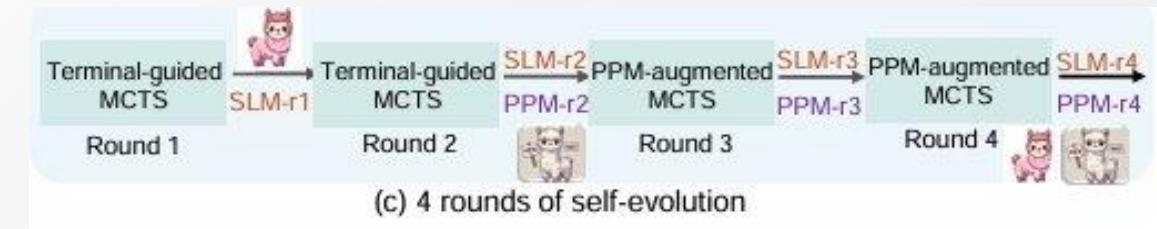


Uses a **4-round self-evolution process**



Progressively improves both the **policy model** (the model that generates math solutions) and the **Process Preference Model (PPM)** (which scores solution steps).

Stages Self-Evolution



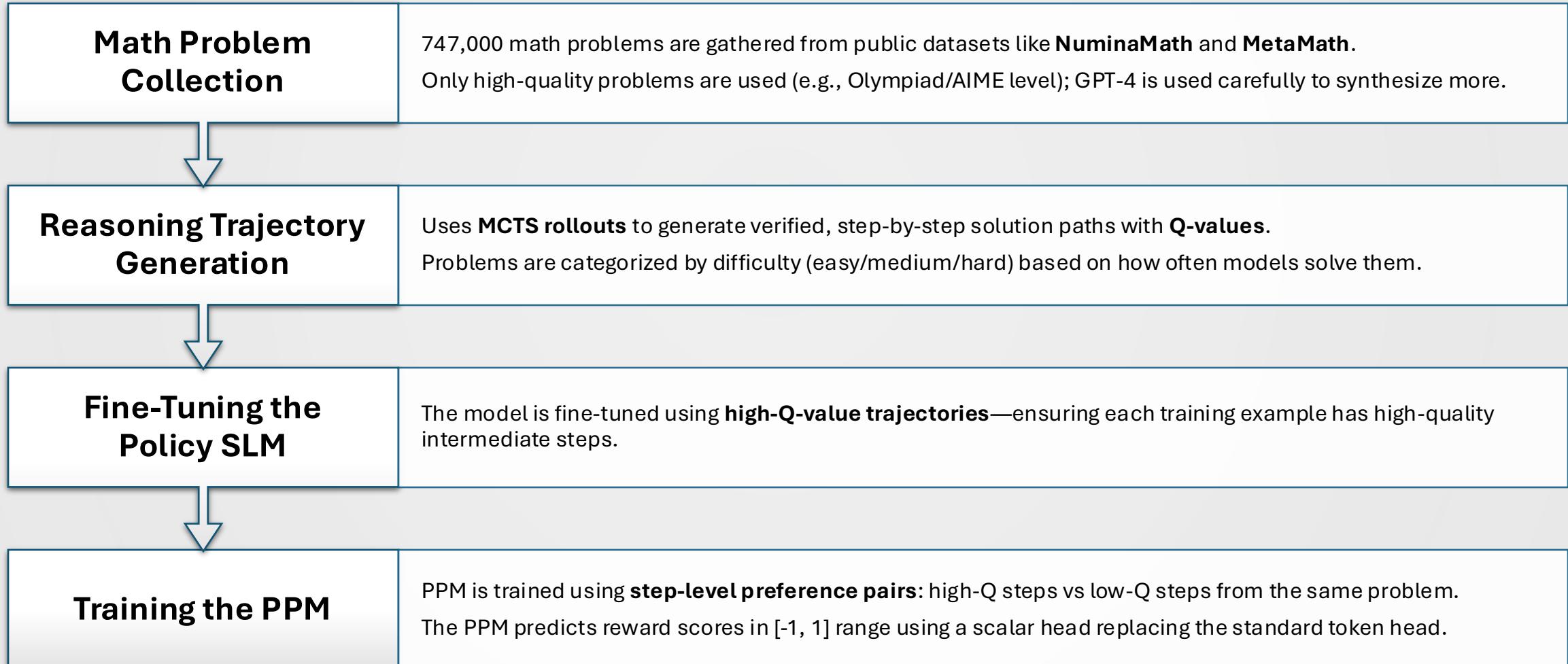
◆ Round 1 – Bootstrapping

◆ Round 2 – Training Reliable PPM

◆ Round 3 – PPM-Augmented MCTS

◆ Round 4 – Solving the Hardest Problems

Training with Step-by-Step Verified Reasoning Trajectory



Round 1 – Bootstrapping

What happens?

- Uses a **very large model (DeepSeek-Coder-236B)** to generate initial solutionpaths via MCTS.
- Trains the **first policy SLM (SLM-r1)** with these trajectories.
- No reliable PPM yet, so Q-values come from **terminal-guided annotation** (just based on whether the final answer is correct).

Why?

- Kickstarts the self-training process since small models aren't yet capable of generating useful data on their own.

Round 2 – Training Reliable PPM

What happens?

- Now using **SLM-r1** for MCTS, they run **16 rollouts per problem** for more accurate Q-values.
- These help train the **first reliable PPM (PPM-r2)**.

Why?

- A good reward model is key for guiding better solution search in future rounds.

Round 3 – PPM-Augmented MCTS

What happens?

- MCTS now uses **PPM-r2** to score steps *during* search.
- This results in **much better reasoning trajectories**, enabling training of even better models (**SLM-r3 and PPM-r3**).

Why?

- PPM-guided MCTS generates higher-quality data, expanding coverage to harder math problems.

Round 4 – Solving the Hardest Problems

What happens?

- Focuses on **unsolved hard problems**, especially Olympiad-level ones.
- Increases MCTS rollouts (to 64 or even 128) and varies random seeds to maximize solution discovery.
- Leads to **SLM-r4 and PPM-r4**.

Why?

- Pushes the model to solve extremely hard problems that even GPT-4 fails at.

Results of Self Evolution

Table 2: Percentage of the 747k math problems correctly solved in each round. Only problems have correct solutions are included in the training set. The first round uses DeepSeek-Coder-Instruct as the policy LLM, while later rounds use our fine-tuned 7B policy SLM.

#	models in MCTS	GSM-level	MATH-level	Olympiad-level	All
Round 1	DeepSeek-Coder-V2-Instruct	96.61%	67.36%	20.99%	60.17%
Round 2	policy SLM-r1	97.88%	67.40%	56.04%	66.60%
Round 3	policy SLM-r2, PPM-r2	98.15%	88.69%	62.16%	77.86%
Round 4	policy SLM-r3, PPM-r3	98.15%	94.53%	80.58%	90.25%

Table 3: Pass@1 accuracy of the resulting policy SLM in each round, showing continuous improvement until surpassing the bootstrap model.

Round#	MATH	AIME 2024	AMC 2023	Olympiad Bench	College Math	GSM8K	GaokaoEn 2023
DeepSeek-Coder-V2-Instruct (bootstrap model)	75.3	13.3	57.5	37.6	46.2	94.9	64.7
Base (Qwen2.5-Math-7B)	58.8	0.0	22.5	21.8	41.6	91.6	51.7
policy SLM-r1	69.6	3.3	30.0	34.7	44.5	88.4	57.4
policy SLM-r2	73.6	10.0	35.0	39.0	45.7	89.1	59.7
policy SLM-r3	75.8	16.7	45.0	44.1	49.6	89.3	62.8
policy SLM-r4	78.4	26.7	47.5	47.1	52.5	89.7	65.7

Evaluation



Evaluation Setup

- **Datasets:**
Evaluates on a broad set of math benchmarks — covering grade-school, competition-level (AIME, AMC), Olympiad, college math, and out-of-domain tasks like GaoKao.
- **Models Used:**
Tests rStar-Math on four small LLMs (1.5B–7B), including both general-purpose and math-specialized models.
- **Training Setup:**
Full 4-round self-evolution done only on Qwen2.5-Math-7B; other models fine-tuned using its final outputs.
- **Baselines Compared:**
Includes top closed-source models (GPT-4o, Claude), open-source systems (e.g., LLaMA3), and Best-of-N setups.
- **Metric:**
Uses Pass@1 accuracy; System 2 methods use MCTS-based test-time search with 8–64 trajectories.

Main Results



Model Performance:

rStar-Math significantly boosts math reasoning ability of SLMs across all benchmarks.



System 2 Advantage:

MCTS-based deep reasoning enables small models to match or outperform larger models that rely on one-shot or Best-of-N methods.



Generality:

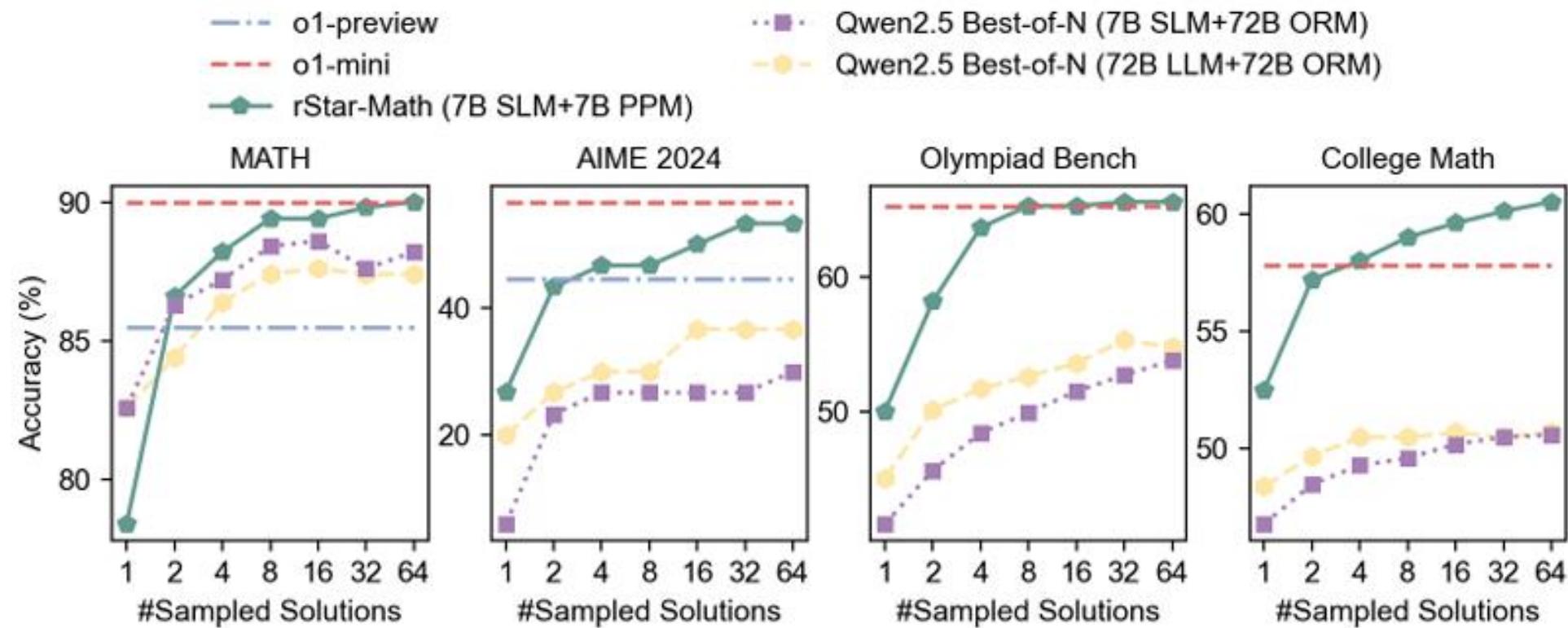
rStar-Math performs well not only on familiar benchmarks (like MATH or GSM8K) but also **generalizes** to new, harder benchmarks (e.g., Olympiad, College Math, GaoKao).

Main comparison of rStar-Math vs. baseline models across benchmarks

Table 5: The results of rStar-Math and other frontier LLMs on the most challenging math benchmarks. rStar-Math⁶⁴ shows the Pass@1 accuracy achieved when sampling 64 trajectories.

Model	Method	Competition and College Level						OOD	
		MATH	AIME 2024	AMC 2023	Olympiad Bench	College Math	GSM8K	Gaokao En 2023	
<i>Frontier LLMs</i>									
GPT-4o	System 1	76.6	9.3	47.5	43.3	48.5	92.9	67.5	
Claude3.5-Sonnet	System 1	78.3	16.0	-	-	-	96.4	-	
GPT-o1-preview	-	85.5	44.6	90.0	-	-	-	-	
GPT-o1-mini	-	90.0	56.7	95.0	65.3	57.8	94.8	78.4	
<i>Open-Sourced Reasoning LLMs</i>									
DeepSeek-Coder-V2-Instruct	System 1	75.3	13.3	57.5	37.6	46.2	94.9	64.7	
Mathstral-7B-v0.1	System 1	57.8	0.0	37.5	21.5	33.7	84.9	46.0	
NuminaMath-72B-CoT	System 1	64.0	3.3	70.0	32.6	39.7	90.8	58.4	
LLaMA3.1-8B-Instruct	System 1	51.4	6.7	25.0	15.4	33.8	76.6	38.4	
LLaMA3.1-70B-Instruct	System 1	65.4	23.3	50.0	27.7	42.5	94.1	54.0	
Qwen2.5-Math-72B-Instruct	System 1	85.6	30.0	70.0	49.0	49.5	95.9	71.9	
Qwen2.5-Math-72B-Instruct+72B ORM	System 2	85.8	36.7	72.5	54.5	50.6	96.4	76.9	
<i>General Base Model: Phi3-mini-Instruct (3.8B)</i>									
Phi3-mini-Instruct (base model)	System 1	41.4	3.33	7.5	12.3	33.1	85.7	37.1	
rStar-Math (3.8B SLM+7B PPM)	System 2	85.4	40.0	77.5	59.3	58.0	94.5	77.1	
rStar-Math⁶⁴ (3.8B SLM+7B PPM)	System 2	86.4	43.3	80.0	60.3	59.1	94.7	77.7	
<i>Math-Specialized Base Model: Qwen2.5-Math-1.5B</i>									
Qwen2.5-Math-1.5B (base model)	System 1	51.2	0.0	22.5	16.7	38.4	74.6	46.5	
Qwen2.5-Math-1.5B-Instruct	System 1	60.0	10.0	60.0	38.1	47.7	84.8	65.5	
Qwen2.5-Math-1.5B-Instruct+72B ORM	System 2	83.4	20.0	72.5	47.3	50.2	94.1	73.0	
rStar-Math (1.5B SLM+7B PPM)	System 2	87.8	46.7	80.0	63.5	59.0	94.3	77.7	
rStar-Math⁶⁴ (1.5B SLM+7B PPM)	System 2	88.6	46.7	85.0	64.6	59.3	94.8	79.5	
<i>Math-Specialized Base Model: Qwen2-Math-7B</i>									
Qwen2-Math-7B (base model)	System 1	53.4	3.3	25.0	17.3	39.4	80.4	47.3	
Qwen2-Math-7B-Instruct	System 1	73.2	13.3	62.5	38.2	45.9	89.9	62.1	
Qwen2-Math-7B-Instruct+72B ORM	System 2	83.4	23.3	62.5	47.6	47.9	95.1	71.9	
rStar-Math (7B SLM+7B PPM)	System 2	88.2	43.3	80.0	63.1	58.4	94.6	78.2	
rStar-Math⁶⁴ (7B SLM+7B PPM)	System 2	88.6	46.7	85.0	63.4	59.3	94.8	79.2	
<i>Math-Specialized Base Model: Qwen2.5-Math-7B</i>									
Qwen2.5-Math-7B (base model)	System 1	58.8	0.0	22.5	21.8	41.6	91.6	51.7	
Qwen2.5-Math-7B-Instruct	System 1	82.6	6.0	62.5	41.6	46.8	95.2	66.8	
Qwen2.5-Math-7B-Instruct+72B ORM	System 2	88.4	26.7	75.0	49.9	49.6	97.9	75.1	
rStar-Math (7B SLM+7B PPM)	System 2	89.4	50.0	87.5	65.3	59.0	95.0	80.5	
rStar-Math⁶⁴ (7B SLM+7B PPM)	System 2	90.0	53.3	87.5	65.6	60.5	95.2	81.3	

Impact of scaling test-time compute



Ablation Study and Analysis

Self-Evolution Effectiveness

- Performance improves consistently across rounds.
- Round 2 introduces a strong PPM, which unlocks deeper reasoning capability.

Verified Reasoning Trajectories

- Fine-tuning with verified, code-checked trajectories outperforms distillation or random/rejection-based sampling.

PPM Effectiveness

- PPM (process-level reward model) outperforms outcome-based reward models.
- Enables more accurate guidance during step-by-step reasoning.

Self-Evolution Effectiveness

- Shows performance improvements from Round 1 → Round 4.
- Demonstrates how accuracy improves as policy SLM and PPM evolve.

Table 6: The continuously improved math reasoning capabilities through rStar-Math self-evolved deep thinking. Starting from round 2, the 7B base model powered by rStar-Math surpasses GPT-4o.

Round#	MATH	AIME 2024	AMC 2023	Olympiad Bench	College Math	GSM8K	GaokaoEn 2023
GPT-4o	76.6	9.3	47.5	43.3	48.5	92.9	67.5
Base 7B model	58.8	0.0	22.5	21.8	41.6	91.6	51.7
rStar-Math Round 1	75.2	10.0	57.5	35.7	45.4	90.9	60.3
rStar-Math Round 2	86.6	43.3	75.0	59.4	55.6	94.0	76.4
rStar-Math Round 3	87.0	46.7	80.0	61.6	56.5	94.2	77.1
rStar-Math Round 4	89.4	50.0	87.5	65.3	59.0	95.0	80.5

Verified Reasoning Trajectories

- Compares fine-tuning on different training datasets.
- Shows verified CoT trajectories outperform GPT-distilled, random, and rejection-sampled data.

Table 7: Ablation study on the effectiveness of our step-by-step verified reasoning trajectories as the SFT dataset. We report the SFT accuracy of Qwen2.5-Math-7B fine-tuned with different datasets.

	Dataset	MATH	AIME	AMC	Olympiad	Bench	College Math	GSM8K	GaokaoEn 2023
GPT-4o	-	76.6	9.3	47.5	43.3		48.5	92.9	67.5
GPT4-distillation (Open-sourced)	MetaMath	55.2	3.33	32.5	19.1		39.2	85.1	43.6
	NuminaMath-CoT	69.6	10.0	50.0	37.2		43.4	89.8	59.5
Self-generation by policy SLM-r3	Random sample	72.4	10.0	45.0	41.0		48.0	87.5	57.1
	Rejection sampling	73.4	13.3	47.5	44.7		50.8	89.3	61.7
	Step-by-step verified (ours)	78.4	26.7	47.5	47.1		52.5	89.7	65.7

PPM Effectiveness

- Compares three reward models: ORM, PQM, and PPM.
- PPM leads to the best math reasoning performance.

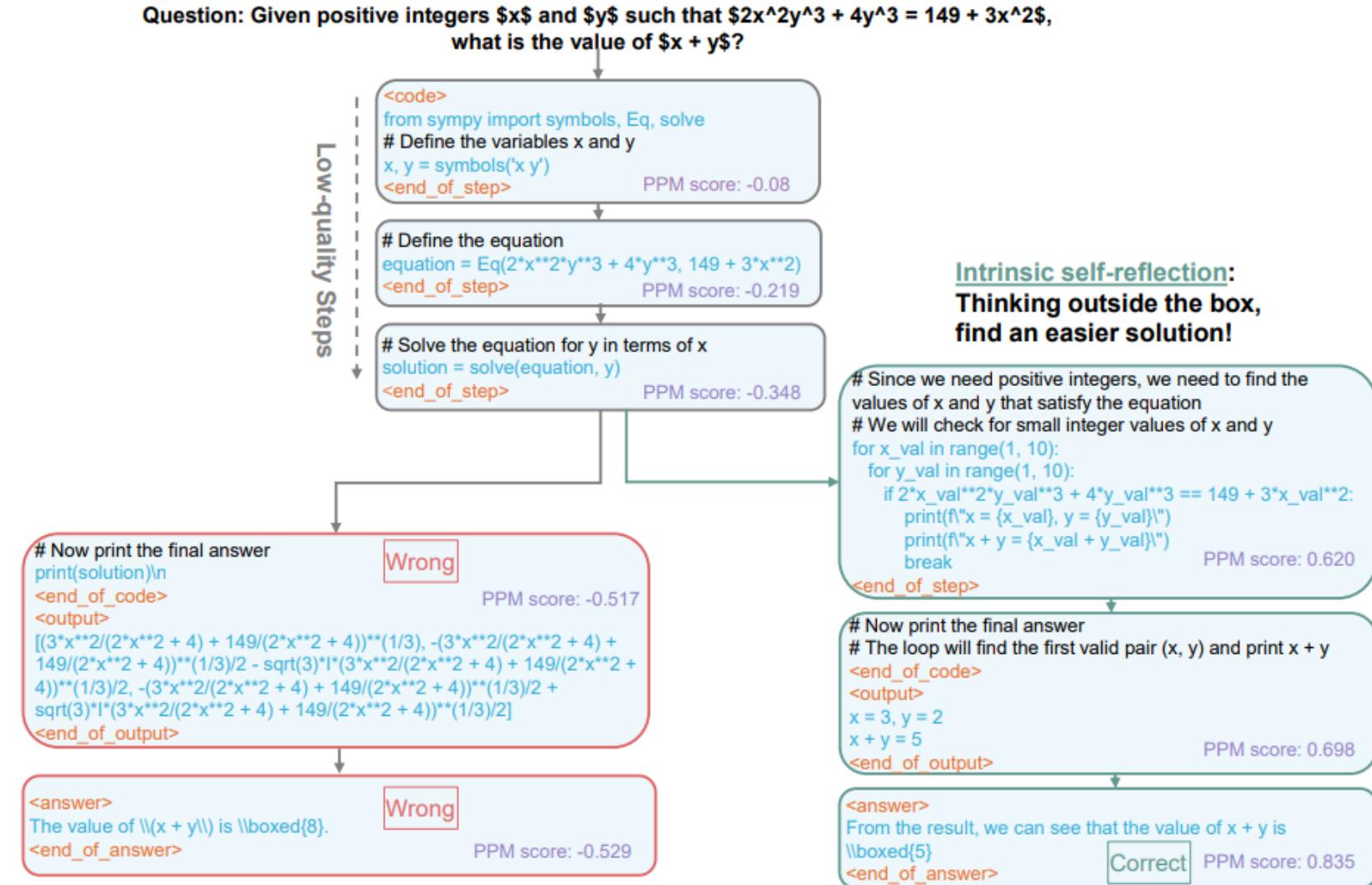
Table 8: Ablation study on the reward model. Process reward models (PQM and PPM) outperform ORM, with PPM pushing the frontier of math reasoning capabilities.

RM	Inference	MATH	AIME	AMC	Olympiad Bench	College Math	GSM8K	GaokaoEn
ol-mini	-	<u>90.0</u>	<u>56.7</u>	<u>95.0</u>	<u>65.3</u>	55.6	94.8	78.6
ORM	Best-of-N	82.6	26.7	65.0	55.1	55.5	92.3	72.5
PQM	MCTS	88.2	46.7	85.0	62.9	<u>57.6</u>	94.6	<u>79.5</u>
PPM	MCTS	<u>89.4</u>	<u>50.0</u>	<u>87.5</u>	<u>65.3</u>	<u>59.0</u>	<u>95.0</u>	<u>80.5</u>

Findings and Discussions

-  **Intrinsic Self-Reflection**
 - rStar-Math **exhibits self-correction** during MCTS rollouts.
 - The model **detects low-quality reasoning paths** and **backtracks** to try better approaches.
-  **PPM Recognizes Key Theorem Applications**
 - PPM assigns **higher scores to steps that apply mathematical theorems**.
 - Helps the model focus on **conceptually meaningful moves**
-  **Generalization Potential**
 - rStar-Math's methodology is **domain-agnostic**.
 - Can generalize to:
 - Theorem proving
 - Code reasoning (via test cases)
 - Commonsense reasoning (via LLM mutual verification)

Example of intrinsic self-reflection



Performance comparison across models with different RMs

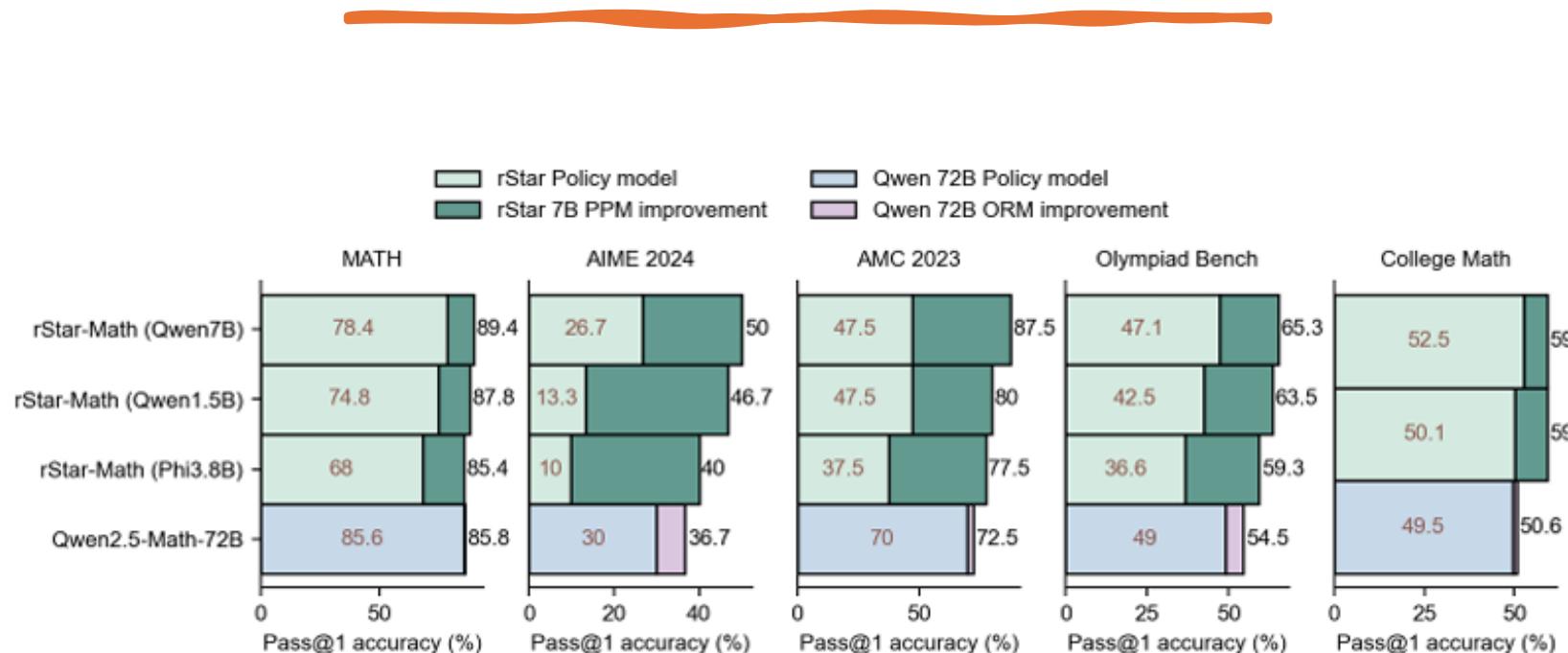


Figure 5: Pass@1 accuracy of policy models and their accuracy after applying System 2 reasoning with various reward models, shows that reward models primarily determine the final performance.

Recognizes Key Theorem Applications

Math statement proof using Fermat's little theorem

Question: If $(pt+a)$ is a perfect square for some value of (t) , then $(a^{\frac{p-1}{2}} - 1)$ is divisible by (p) (where (p) is a prime number).

Our trajectory for proving:

<code>

```
# Step 1: Define the variables
from sympy import symbols, isprime, sqrt, mod_inverse
# Define the variables
p, t, a = symbols('p t a')\\
<end_of_step>
```

```
# Step 2: Assume  $p*t + a$  is a perfect square
# This means there exists an integer  $k$  such that  $p*t + a = k^2$ 
# Rearrange to find  $a$  in terms of  $p$ ,  $t$ , and  $k$ 
#  $a = k^2 - p*t$ 
<end_of_step>
```

```
# Step 3: Use Fermat's Little Theorem
# Fermat's Little Theorem states that if  $p$  is a prime and  $a$  is an integer not divisible by  $p$ ,
then  $a^{(p-1)} \equiv 1 \pmod{p}$ 
# We need to show that  $a^{((p-1)/2)} - 1$  is divisible by  $p$ 
<end_of_step>
```

```
# Step 4: Substitute  $a = k^2 - p*t$  into Fermat's Little Theorem
#  $a^{((p-1)/2)} \equiv (k^2 - p*t)^{((p-1)/2)} \pmod{p}$ 
# Since  $p*t$  is divisible by  $p$ ,  $(k^2 - p*t)^{((p-1)/2)} \equiv k^{(p-1)} \pmod{p}$ 
# By Fermat's Little Theorem,  $k^{(p-1)} \equiv 1 \pmod{p}$ 
<end_of_step>
```

```
# Step 5: Therefore,  $a^{((p-1)/2)} \equiv 1 \pmod{p}$ 
# This implies  $a^{((p-1)/2)} - 1 \equiv 0 \pmod{p}$ 
# Hence,  $a^{((p-1)/2)} - 1$  is divisible by  $p$ 
<end_of_step>
```

```
# Now print the final answer
print("The expression  $a^{((p-1)/2)} - 1$  is divisible by  $p$ .")
```

<end_of_code>

<output>

The expression $a^{((p-1)/2)} - 1$ is divisible by p .
The statement is true.

<end_of_output>

Conclusion

rStar-Math enables **small language models (1.5B–7B)** to achieve **state-of-the-art math reasoning**.

Uses a **self-evolving, System 2-style approach**

Achieves performance **comparable to or better than GPT-4-level models**, without distillation.

Introduces key capabilities like **self-reflection** and **theorem-aware reasoning**.

Generalizable to other domains like **code**, **logic**, and **commonsense reasoning**.

Questions ?



Limitations and Future Directions

Limitation 3: Agentic Formulation of Language Tasks

- Framing NLP tasks (e.g., QA) as multi-step agentic processes (like MDPs) may overcomplicate simple queries.
- Recommendation
 - Reconsidering when decomposition is necessary
 - Especially for tasks that don't benefit from subgoal planning.

Future Direction

- **Workflow Composition:**
 - Combining feedback sources with tool-use,
 - Or embedding MCTS into validation loops.
- **Prompt Optimization:**
 - Dynamically generating prompt templates via meta-learning or context-aware sampling.
- **Grounded Evaluation:**
 - Designing benchmarks with realistic feedback sources (e.g., user reviews, external APIs) rather than gold labels.