Section 8.2: Agentic Planning and Reasoning

2026 Spring

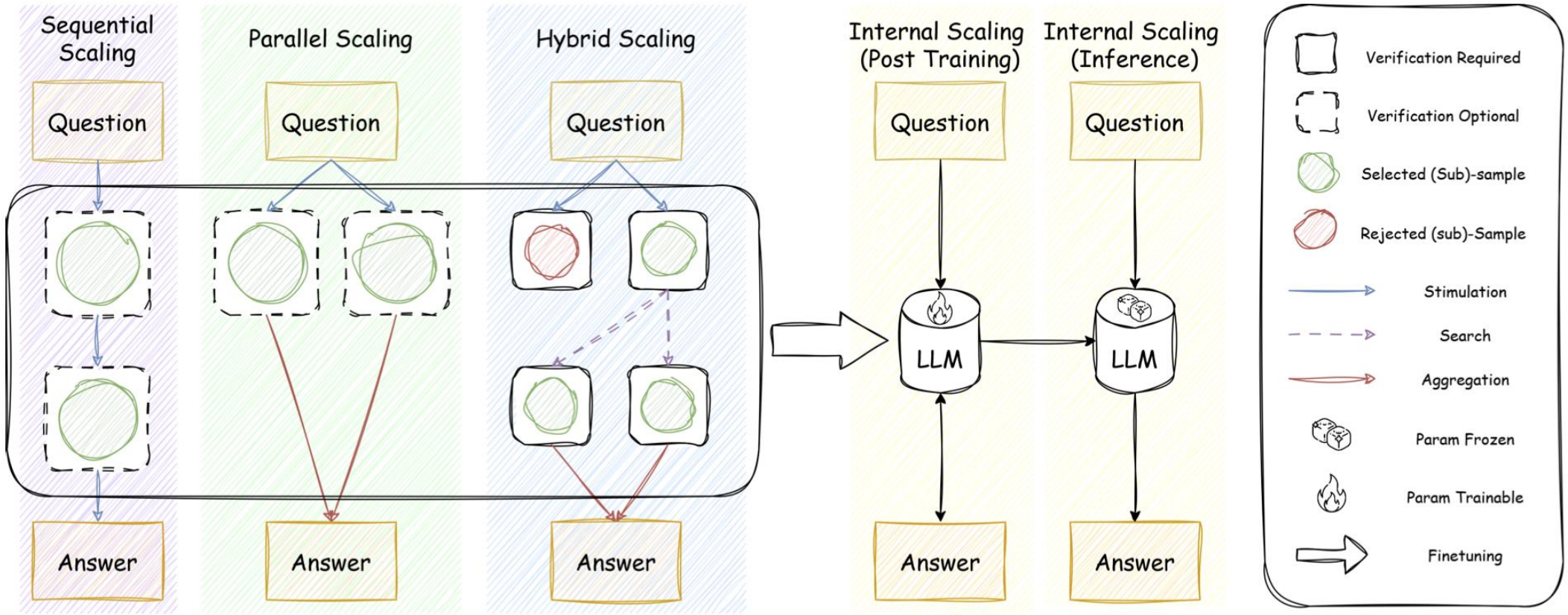
LLM Agents Foundation & Applications

Student Team / 20260407

Roadmap

- S1: LLM Basic Alignment
- S2: LLM Alignments for Reasoning
- S3: Agent applications
- S4: LLM Data synthesis
- S5: Agent Memory
- S6: LLM model serving
- S7: Agent Evaluation and Attack/Defense Landscape
- S8: Agent Planning / Testing Time Scaling ➔ This lecture!
- S9: World Modeling for GenAI Agents
- S10: Multi-Agents

What & How: Testing Time Scaling



Where to Scale: e.g., Reasoning Tasks

- Reasoning tasks:
  - Challenging tasks that require structured, explicit, and precise reasoning
  - Mathematical Reasoning: Complex computations, logical inference, and iterative verification
  - Programming and Code Generation: Syntactic accuracy, executable correctness, and iterative debugging

https://openreview.net/forum?id=VTF8yNQM66

Where to Scale: e.g., Planning?

- Planning is a sequence of actions (a_1, a_2, …, a_n) that lead us to the desired goal.

R. Epstein, C. E. Kirshnit, R. P. Lanza & L. C. Rubin, **Insight' in the pigeon: antecedents and determinants of an intelligent performance**, Nature, 1984

The Landscape of Agentic Reinforcement Learning for LLMs: A Survey

Why We Need Agentic RL?

What PBRFT (RLHF/DPO) handles

1. Prompt: “Fix this code” + source code + error message

3. Model outputs fixed code

3. Done

One prompt → one response → done

T = 1, episode ends

What real tasks need

“Fix the bug in this repo”

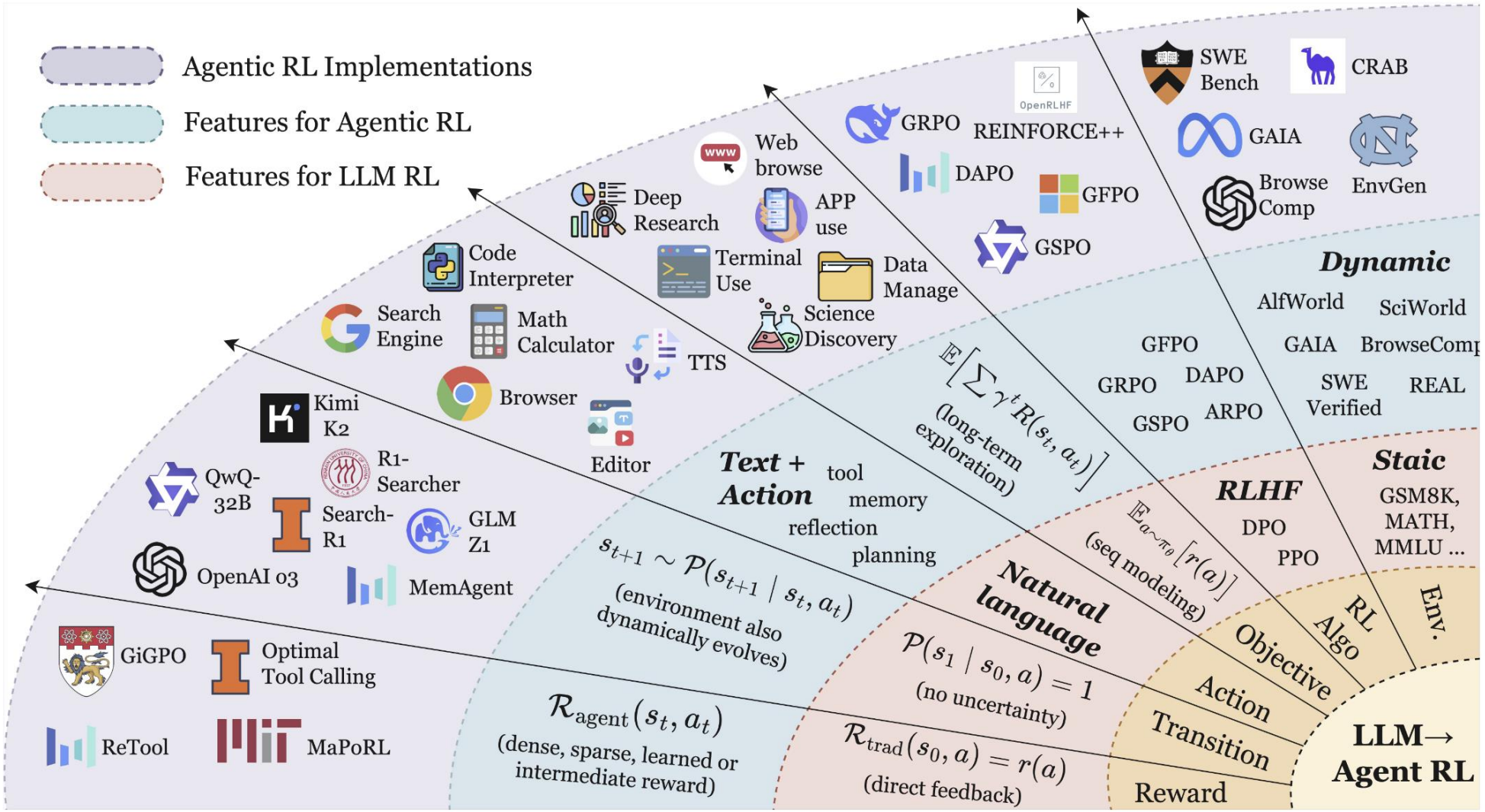
- → Read files, understand the error
- → Write a fix, run tests
- → New errors appear, revise
- → Repeat until all tests pass

PBRFT cannot do this ✕

Agentic RL reframes LLMs: from passive text generators → autonomous decision-making agents

The Paradigm Shift at a Glance

Source: Figure 2, Zhang et al. (2026)



PBRFT vs. Agentic RL: Coding example

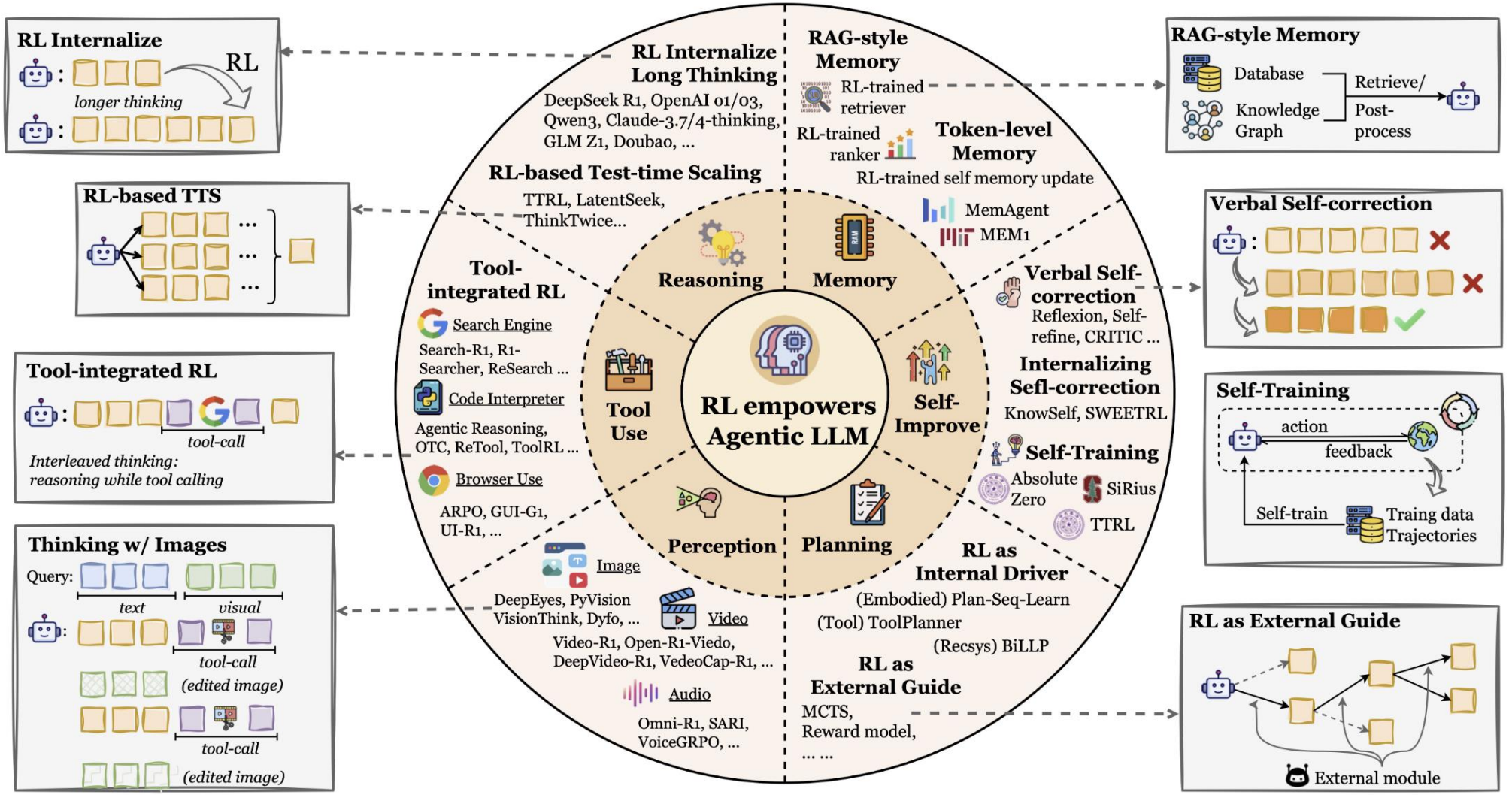
Concept	PBRFT (RLHF / DPO)	Agentic RL
S State	The prompt text *Episode ends after one response*	Current code in editor + latest error messages *State changes every step as new errors appear*
A Action	Output a fixed code string, nothing runs *Text only — no tool calls allowed*	Edit code, call run_tests(), search docs *Text + tool calls — agent interacts with environment*
P Transition	One output → terminal state, no environment *Deterministic, nothing can change after output*	Run tests → new errors appear unpredictably *Environment is stochastic — same fix, different errors*
R Reward	Preference model scores the output once *Single scalar at end of episode, no step feedback*	Test pass rate at each step *3/10 tests pass → fix → 8/10 tests pass → fix → 10/10*
J(θ) Objective	Make this one output score as high as possible *No planning, no looping, $T = 1$*	Maximize total tests passing across all steps *Rewards accumulated over the whole episode, $T > 1$*

PBRFT vs. Agentic RL: Formal Comparison

Concept	Traditional PBRFT	Agentic RL
$\mathcal{S}$: State space	$\{s_0\}$ (single prompt); episode ends immediately.	$s_t \in \mathcal{S}_{\text{agent}}$; $o_t = O(s_t)$; horizon $T > 1$.
$\mathcal{A}$: Action space	Pure text sequences.	$\mathcal{A}_{\text{text}} \cup \mathcal{A}_{\text{action}}$.
$\mathcal{P}$: Transition	Deterministic transition to the terminal state.	Dynamic transition function $P(s_{t+1} \mid s_t, a_t)$.
$\mathcal{R}$: Reward	Single scalar $r(a)$.	Step-wise $R(s_t, a_t)$; combines sparse task and dense sub-rewards.
$J(\theta)$: Objective	$\mathbb{E}_{a \sim \pi_\theta}[r(a)]$.	$\mathbb{E}_{\tau \sim \pi_\theta}[\sum_t \gamma^t R(s_t, a_t)]$.

Source: Zhang et al. (2026), Table 1

Six Core Capabilities of an Agentic LLM



Reasoning vs. Planning in LLM Agents

Reasoning - The ability to derive conclusions from given information.

"Given these facts, what is true?"

- Thinking inside a single step
- No environment, no tool calls
- Output: an answer or judgment
- Failure: wrong conclusion

Coding: "this bug is an index error because array length is 0"

Planning - The ability to decompose a goal into a sequence of actions to be executed over time.

"Given this goal, what do I do next?"

- Multi-step, across time
- External — tools, env, world state
- Output: an action sequence
- Failure: wrong or incomplete plan

Coding: read file → run tests → edit function → loop

In the agent loop:

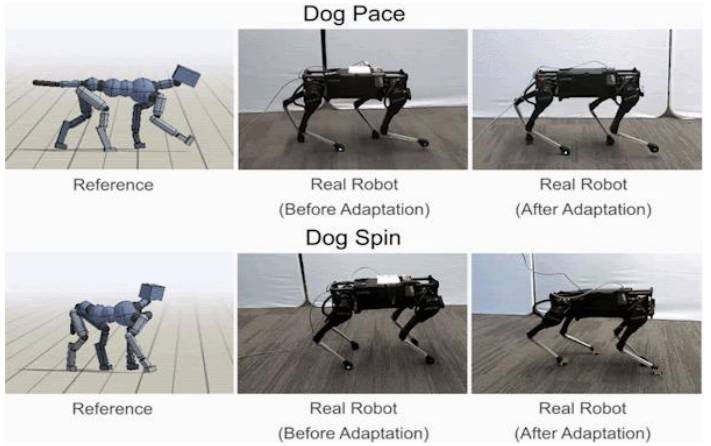
Planning decides the sequence → **Reasoning** thinks through each step → **Action** → Observation → **Replan if needed**

Why Planning is Important?

1) To handle complex task

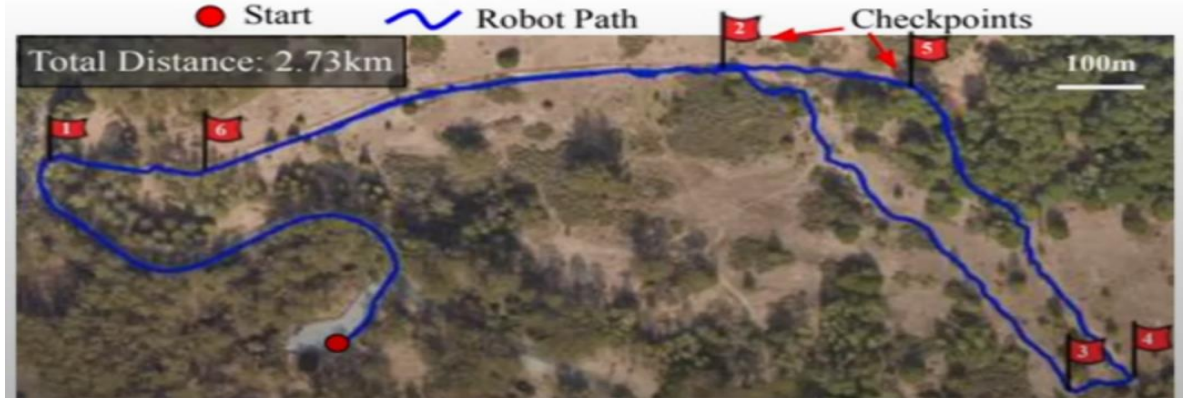
2) Can provide fine grained motor skills

Wu et al., **Discrete Policy: Learning Disentangled Action Space for Multi-Task Robotic Manipulation**, 2025



Peng et al., **Learning Agile Robotic Locomotion Skills by Imitating Animals,** 2020

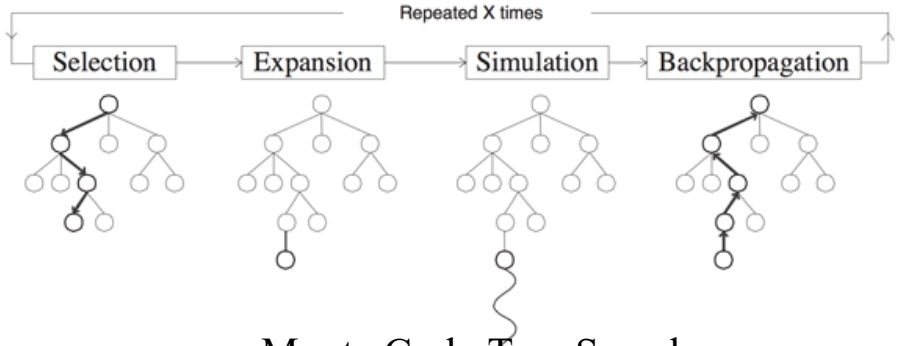
3) Planning should give the optimal path



Classical Planning vs Reinforcement Learning (RL)

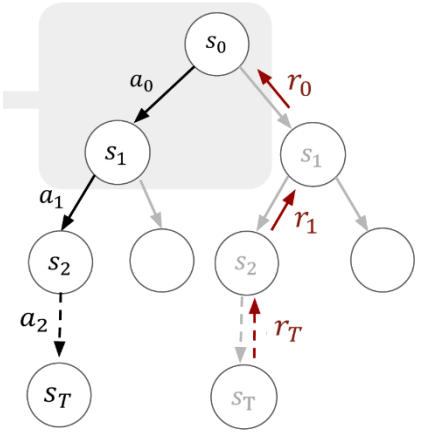
1) Classical Planning:
tree search-based planning

e.g., A* search, Greedy best first search, Monte Carlo search etc



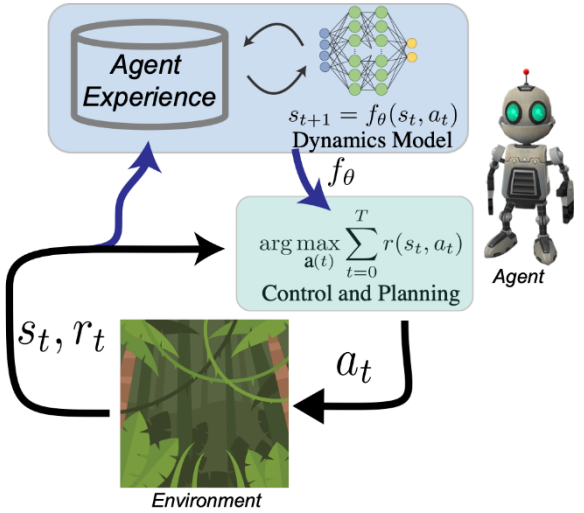
Monte Carlo Tree Search

Figure from Chaslot (2006)



Reasoning via Planning (RAP)

2) Reinforcement Learning based Planning



3) Traditional AI Planner (logic based)

Two Planning Paradigms

RL as External Guide

LLM role	Action proposer — generates candidate actions
RL optimizes	An external value or heuristic function
LLM params	Frozen ✗
Strength	Stable training, interpretable search process
Limitation	Ceiling bounded by LLM's existing knowledge

RL as Internal Driver

LLM role	Policy model — LLM is what gets trained
RL optimizes	The LLM's planning policy directly
LLM params	Updated ✓
Strength	Adaptive, generalizes from experience
Limitation	Less stable; joint optimization harder

External Guide: Reasoning with Language Model is Planning with World Model

Task: stack blocks A, B, C in order C-B-A

CoT generates:

Step 1 **Pick up A, place on C**

Step 2 **Pick up B, place on A**

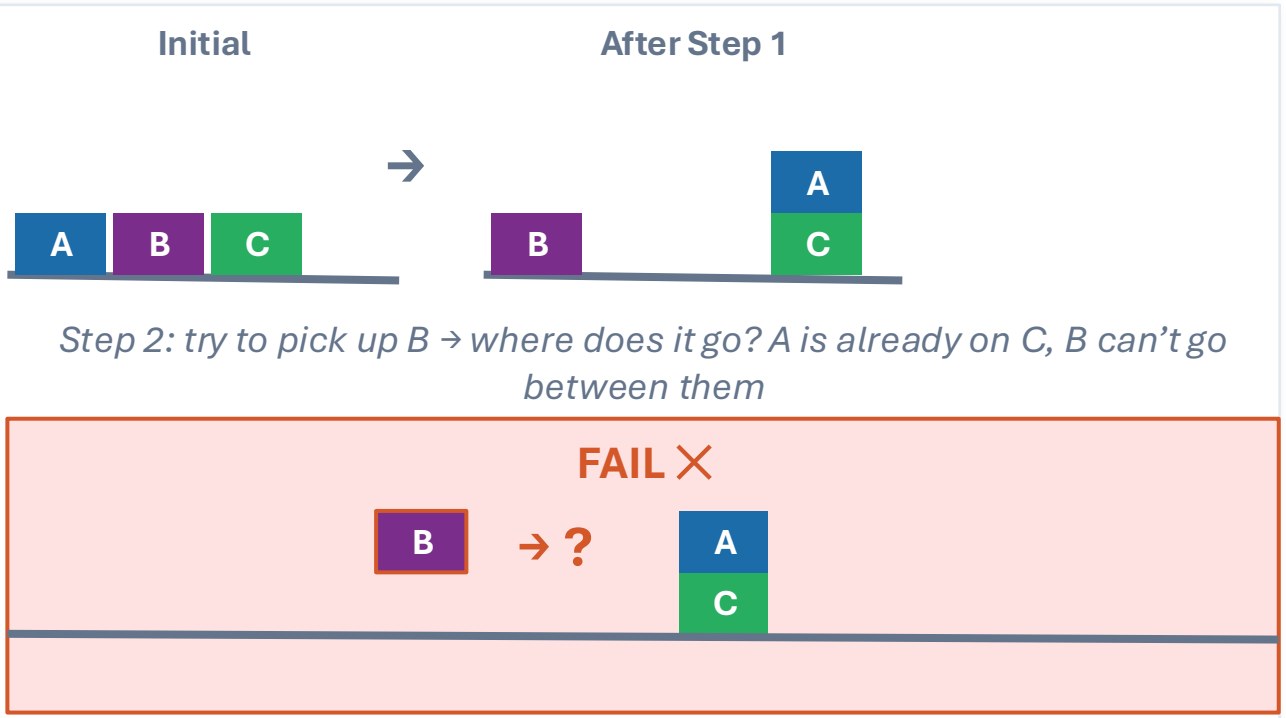
FAIL ✕

A is not on table — can't pick up B

Why?

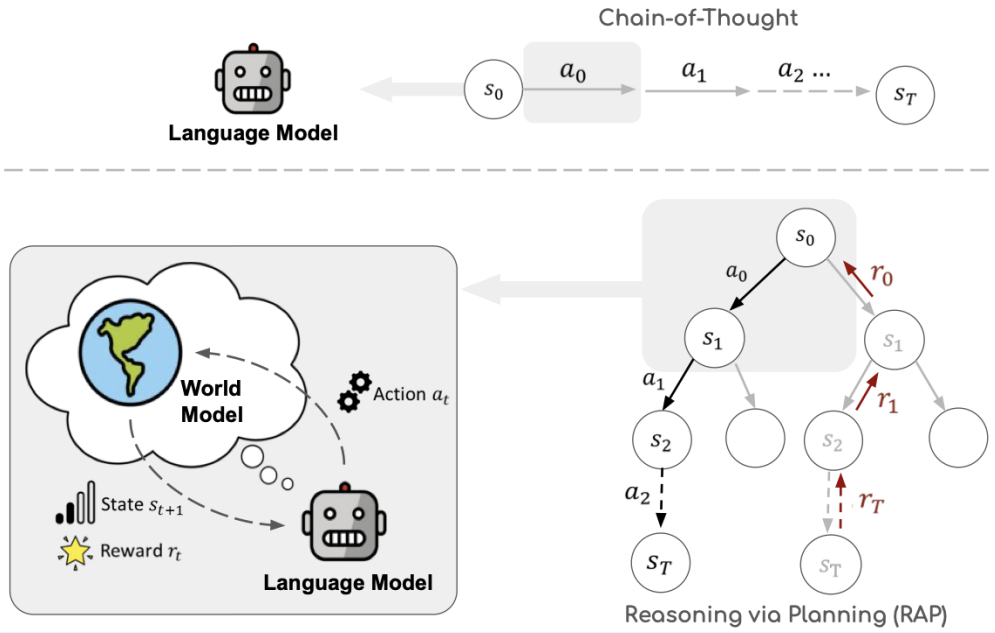
✕ No backtracking — commits blindly

✕ No world model — never checks if valid



External Guide: Reasoning with Language Model is Planning with World Model

Hao et al., EMNLP 2023



Problem with CoT

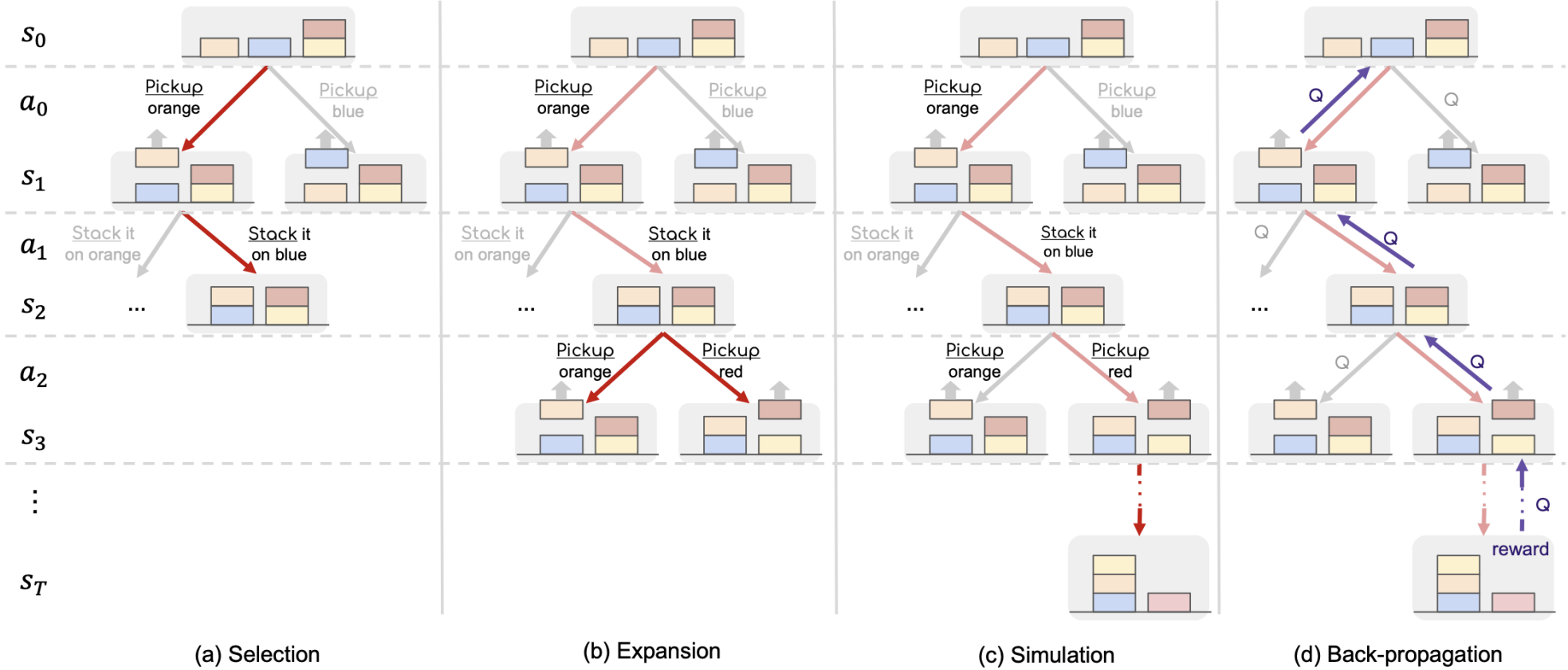
- Generates steps linearly — no backtracking
- No world model — can't simulate "what happens if I do X?"

Key Idea: LLM plays two roles

- **Reasoning agent — proposes candidate actions at each step**
- **World model — predicts next state, self-evaluates each node**

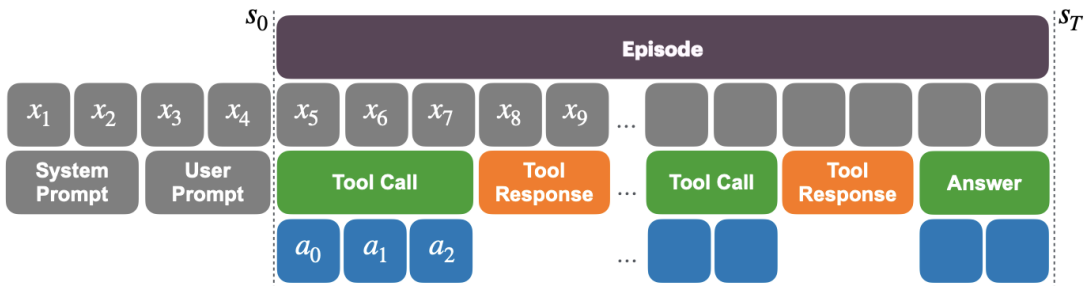
External Guide: Reasoning with Language Model is Planning with World Model

Hao et al., EMNLP 2023



Internal Driver: Planner-R1

Zhu et al., 2025



Planner-R1

Reward shaping for efficient agentic RL · Zhu et al., 2025

Outcome Reward	Process Reward
Signal only at task end	Step-wise credit per decision
Sparse, delayed feedback	Dense, immediate feedback
Slow convergence	Faster convergence

Result: process rewards markedly improve learning efficiency — smaller models reach competitive planning capability

Other Internal Driver works

ETO · VOYAGER · Dynamic Speculative Planning · AdaPlan · RLTR

Tradeoff & Prospective

External Guide

- ✓ Stable training
- ✓ Interpretable search
- ✓ No LLM update needed
- ✗ LLM quality ceiling
- ✗ Value fn requires training

Internal Driver

- ✓ Adaptive planning
- ✓ Generalizes from experience
- ✓ End-to-end optimization
- ✗ Less stable training
- ✗ Joint optimization harder

Prospective: The Synthesis of Deliberation and Intuition

The future paradigm is not External vs Internal — it is their **synthesis.**

An agent that internalizes structured search, learns a meta-policy governing when to explore vs commit, and seamlessly blends fast plan generation with deliberate reasoning.

"From a component that proposes actions → to an integrated reasoning engine"

Research Agent: OpenAI Deep Research

OpenAI Deep Research

Plan — Decompose user request into sub-goals

↓

Search — RL-shaped queries, page selection, trajectory refinement

↓

Memory — Persistent scratchpad notes across turns

↓

Verify — Cross-source verification before synthesis

RL shapes control decisions over search depth, branch selection, and evidence integration — not just what to retrieve, but how to plan the research

Open Challenges

① Temporal Credit Assignment

In long-horizon tasks, sparse rewards make it hard to identify which action in a multi-step sequence contributed to success or failure. GiGPO and SpaRL explore turn-level advantage estimation, but this remains largely unsolved.

② Scaling Agentic Environments

Prevalent environments like ALFWorld are insufficient for training general-purpose agents. The field needs adaptive environments that co-evolve with the agent — automating both reward design and curriculum generation.

③ Reward Hacking & Trustworthiness

RL agents optimize whatever reward is given. In agentic settings this risks unsafe tool use, hallucination amplification, and sycophancy — the agent learns to exploit loopholes rather than solve the task.

④ Does RL Create New Capabilities?

Open debate: does RL genuinely install new reasoning abilities, or mainly reshapes the sampling distribution over pre-existing ones? Evidence suggests both — depends on task structure, reward quality, and model regime.

Model-First Reasoning LLM Agents: Reducing Hallucinations through Explicit Problem Modeling

Authors: Gaurav Kumar and Annu Rana

Presented by: Yi Deng

Reasoning as a Core Agent Capability

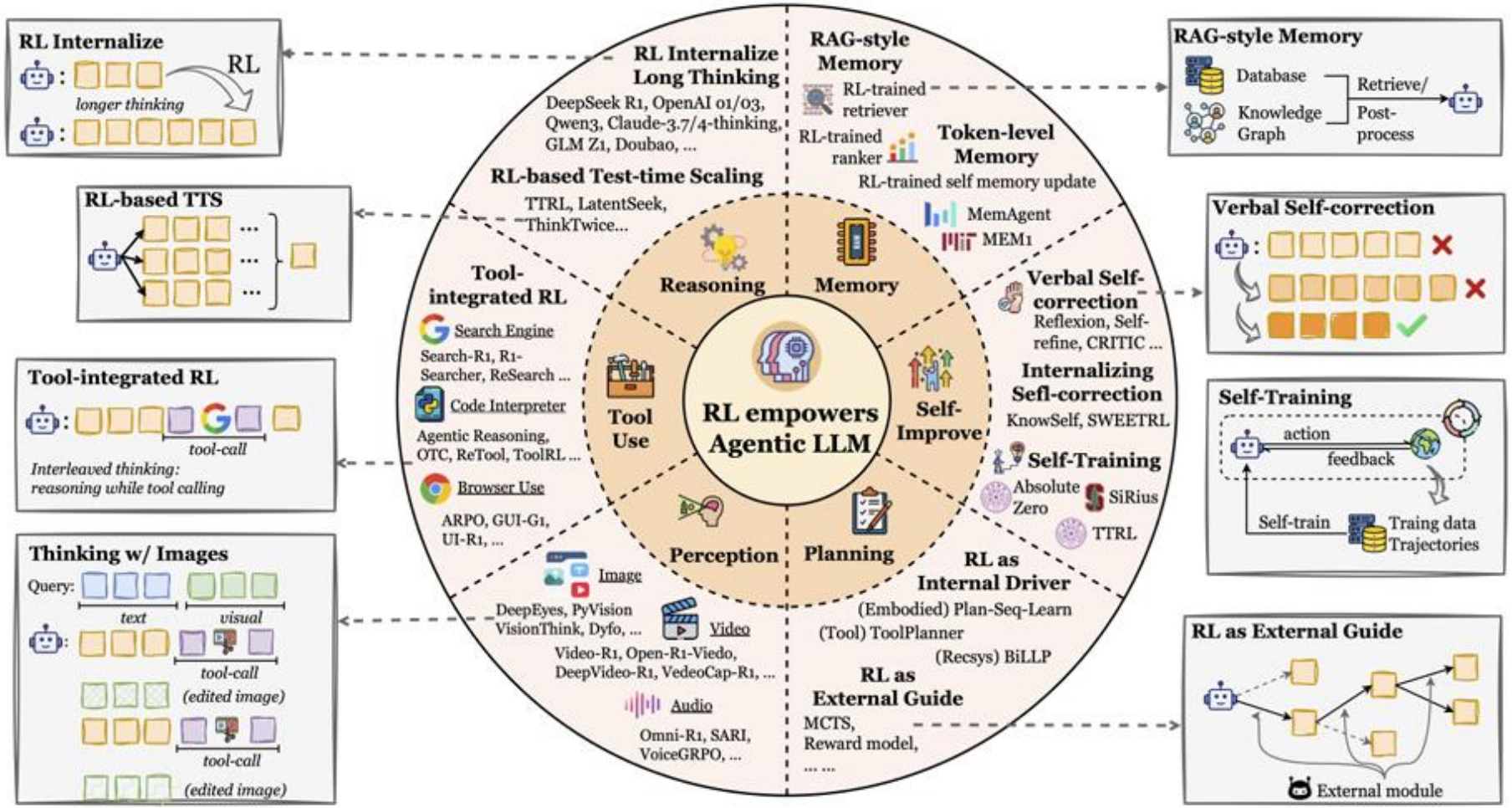


Figure 5: A conceptual overview of how RL empowers agentic LLMs across six core capabilities. The central panel summarizes the capability taxonomy, while the side panels illustrate representative RL mechanisms and interaction patterns. Listed methods are illustrative rather than exhaustive; see the main text for details.

- Reasoning is one of the six core capabilities of an agentic LLM.
- Section 3.5 frames reasoning through fast reasoning and slow reasoning.

Fast vs Slow Reasoning

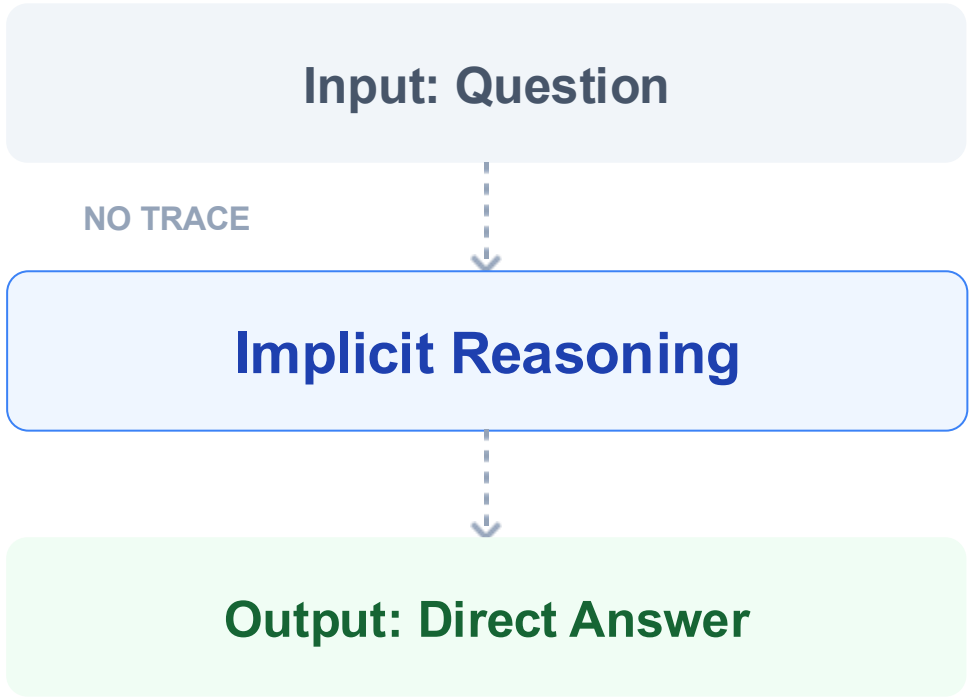
Fast Reasoning

- rapid, heuristic-driven
- minimal intermediate steps
- efficient and low-latency
- reasoning remains largely implicit

Slow Reasoning

- deliberate and structured
- explicit intermediate traces
- deeper reflection and greater logical consistency
- higher accuracy but longer inference

Fast Reasoning: Efficient but Implicit



Most conventional LLMs reason implicitly through **next-token prediction**.

✓ This works well when fluency, efficiency, and low latency matter

⚠ But the reasoning process is **not explicitly exposed**.

✗ As a result, models are more vulnerable to factual errors, biases, and shallow generalization

Slow Reasoning: CoT as a Foundational Example

Problem → **Step 1** → **Step 2** → **Step 3** → **Answer**

vis ibil ity

Explicit Reasoning

Chain-of-Thought (CoT) makes intermediate reasoning steps visible and structured.

de scr ipt io n

Key Benefits

Slow reasoning supports deeper reflection and greater logical consistency.
It often achieves higher accuracy and robustness on knowledge-intensive tasks.

Representative Modern Slow Reasoning in Section 3.5

DeepSeek-R1

OpenAI o1 / o3

Dynamic test-time scaling / RL-based reasoning

Common output characteristics

- longer and more explicit reasoning trajectories
- clear exploration and planning structure
- frequent verification and checking behaviors
- stronger on math, scientific reasoning, and multi-hop QA

Open Challenge: Efficient and Reliable Agentic Reasoning

Fast Reasoning — **Trade-off** — **Slow Reasoning**

Fast Reasoning Risk

Purely fast models may overlook critical reasoning steps, leading to **shallow errors**.

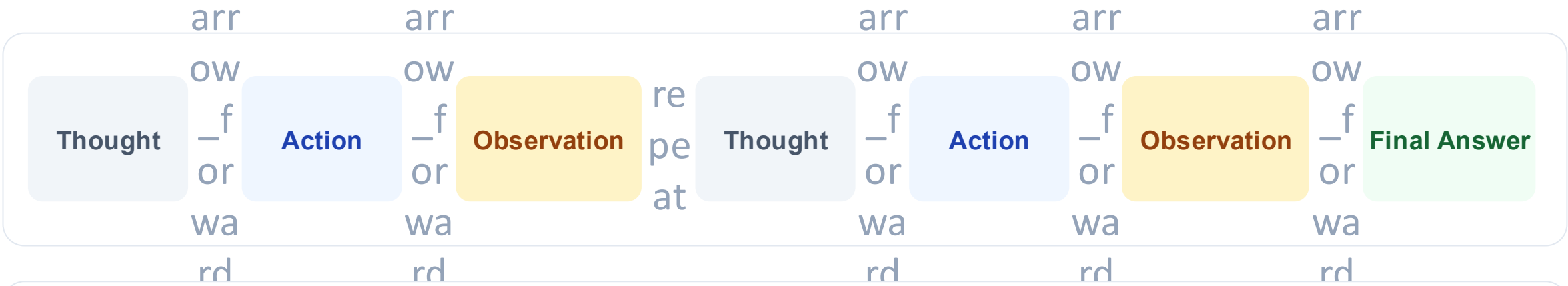
Slow Reasoning Risk

Purely slow models may suffer from **excessive latency** or overthinking.

Prospective Direction: Hybrid & Adaptive Reasoning

- The survey highlights the need for models that adapt reasoning depth dynamically.
- **Adaptive test-time scaling** is one concrete example of this efficiency.

ReAct in Agent Settings



- ReAct interleaves reasoning with actions and observations.
- This brings explicit reasoning closer to interactive agent settings.
- It can improve grounding through external observations.
- However, the task state is still maintained implicitly across the trajectory.

LLM Agents Fail in Complex, Long-Horizon Tasks

report_problem

Constraint Violations

Agents break explicit task rules or requirements during execution.

sync_problem

Inconsistent State Tracking

Lose track of intermediate states over long trajectories.

extension_of

Brittle Solutions

Fail under small changes or unexpected environmental conditions.

Examples: Medical scheduling, resource allocation, and procedural execution.

The Limitation of Implicit Reasoning

psychology

Chain-of-Thought (CoT)

IMPLICIT REASONING

- Step-by-step reasoning
- Entities, states, and constraints remain implicit
- May stay locally coherent while failing globally

settings_suggest

ReAct

DISTRIBUTED TRACKING

- Interleaves reasoning with actions and observations
- State tracking is distributed across text traces
- Global constraints are rarely enforced explicitly

Many failures are representational, not inferential

Reasoning often fails because the problem model is implicit, unstable, and unverifiable.

layers_clear

Undefined Parameters

Entities, states, and constraints are never explicitly defined.

trending_down

State Drift

State tracking can drift significantly over long trajectories.

grid_off

Global Inconsistency

Locally coherent reasoning can still become globally inconsistent.

Why Explicit Problem Models Matter

Why reasoning needs a model

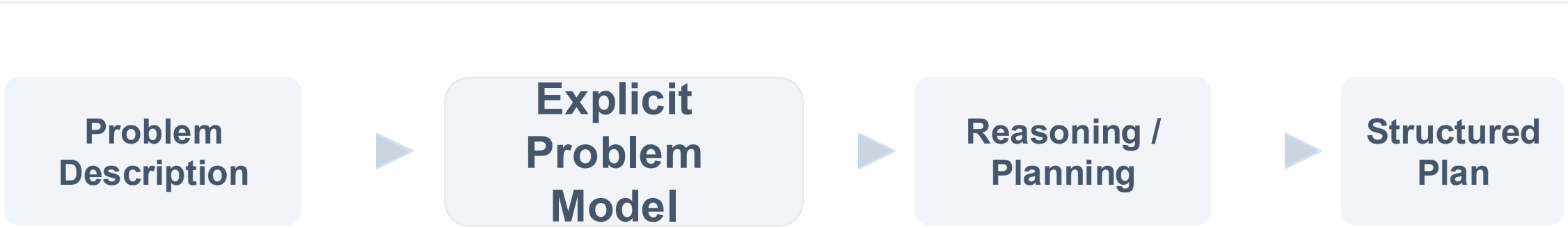
- Human reasoning operates over internal models
- Missing variables or constraints can lead to wrong conclusions
- Reliable reasoning needs an explicit representation of the problem

Classic planning makes structure explicit

- Define entities, state variables, actions, and constraints first
- Planning is then performed over a fixed, verifiable structure
- Inconsistencies become visible when the plan violates the model

Reasoning does not create structure — it operates on structure.

Model-First Reasoning: Core Idea



- **Pre-solution Construction:** The model is built *before* any solution generation begins.
- **Comprehensive Definition:** It captures entities, state variables, actions, and constraints.
- **Reasoning Constraint:** All downstream reasoning is strictly bound by this explicit model.

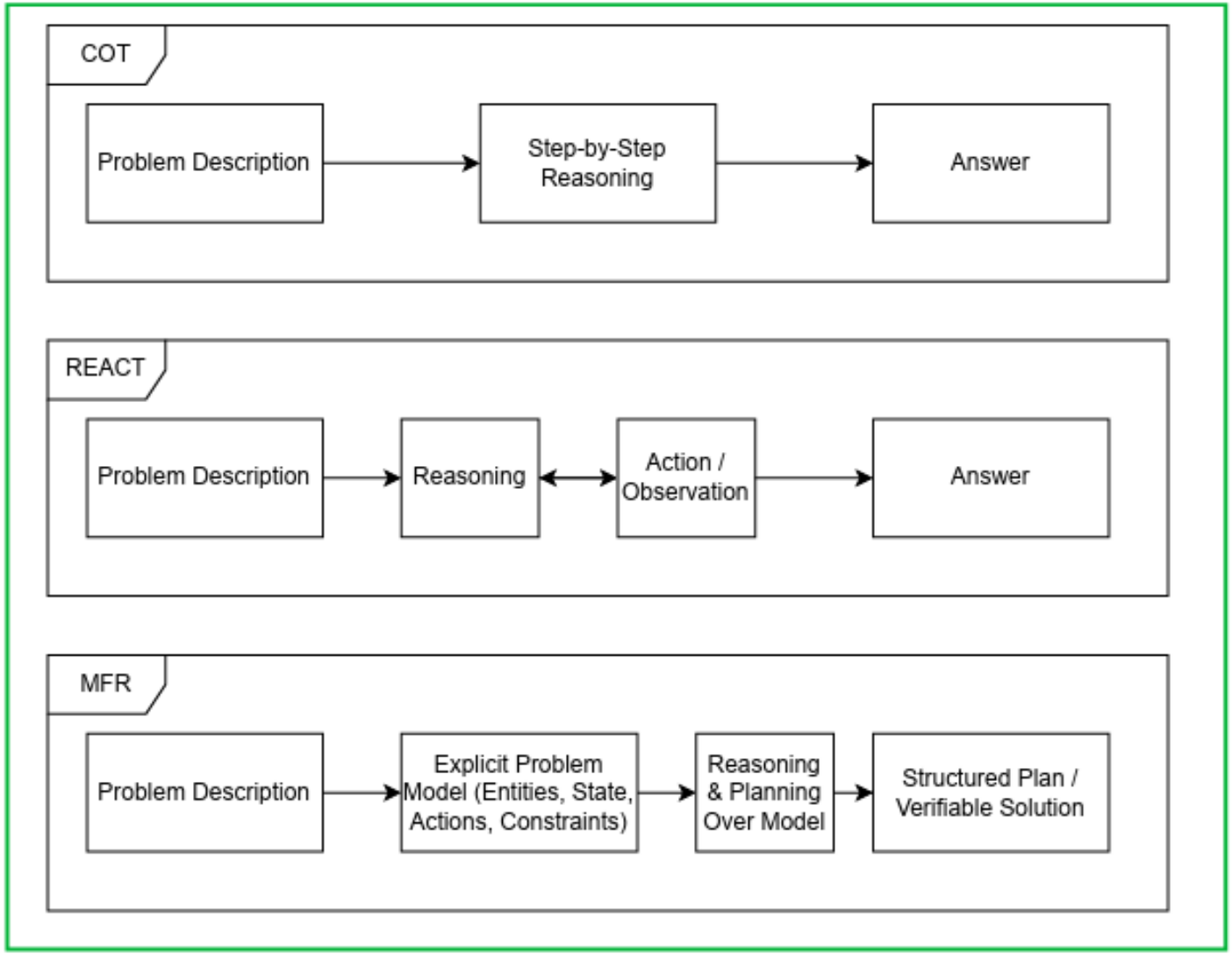


Figure 1: Comparison of reasoning paradigms: CoT, ReAct, and Model-First Reasoning (MFR).

Phase 1: Model Construction

Entities

Objects or agents in the task

(e.g., people, resources, locations)

State Variables

Properties that change over time

(e.g., availability, location, status)

Actions

Allowed operations with optional preconditions and effects

Constraints

Rules or invariants that must always be respected

No solution is generated in this phase. This enforces a clean separation between representation and reasoning.

Phase 2: Reasoning Over the Model

Explicit Problem Model ▶ **Reasoning / Planning** ▶ **Structured Plan**

RULES

- Actions must respect **preconditions**
- State transitions must match **defined effects**
- **Constraints** must remain satisfied

Errors become visible as **inconsistencies** between the plan and the model.

Prompt Structure

Phase 1: Model Construction

- Define entities
- Define state variables
- Define actions with preconditions/effects
- Define constraints
- Do not solve yet

Phase 2: Reasoning & Planning

- Use only the model above
- Generate a step-by-step plan
- Respect constraints and state transitions

Can be implemented in a single prompt or two sequential prompts.

Why MFR Works

psychology

Latent State Tracking

Reduces reliance on implicit memory by making the model state explicit.

Long-Horizon Consistency

Ensures multi-step plans remain valid from start to finish relative to the model.

fact_check

Unstated Assumptions

Prevents hidden logical leaps by requiring formal definitions of the problem space.

rule

Verification Capabilities

Enables both human auditors and automated systems to verify step-by-step logic.

Compatible with existing paradigms

- Can be combined with CoT in Phase 2
- Can be integrated into ReAct as persistent state

MFR acts as a form of **soft symbolic grounding**.

Experimental Objective

The goal is to evaluate whether MFR improves reliability, constraint adherence, and structural clarity of LLM-generated plans compared with CoT and ReAct.

verified_user

Reliability

More consistent and accurate planning behavior

gavel

Constraint Adherence

Better compliance with task rules and limits

account_tree

Structural Clarity

More interpretable and verifiable plans

These are the high-level goals; the concrete evaluation criteria are introduced next.

Tasks and Setup

Experimental Tasks

- Medication scheduling
- Route planning
- Resource allocation
- Logic puzzle solving
- Procedural synthesis

Experimental Setup

- **Task descriptions:** Uniform across all tested methods.
- **Variable:** Only the reasoning instructions differ.
- **Models:** Evaluated across multiple Large Language Models (LLMs).

Comparing performance metrics across diverse reasoning domains.

Evaluation Criteria

1. Constraint Satisfaction

Does the generated plan respect the explicit or implicit task constraints?

2. Implicit Assumptions

Are there unstated or inferred actions/states that could impact correctness?

3. Structural Clarity

Is the plan interpretable and verifiable, with clear logical structure?

Main Qualitative Results

Reasoning Strategy	Constraint Violations	Implicit Assumptions	Structural Clarity
Chain-of-Thought (CoT)	Medium	Frequent	Low
ReAct	Medium–Low	Occasional	Medium
Model-First	Low	Rare	High

Table 1: Comparison of reasoning strategies across tasks (qualitative assessment).

MFR shows lower constraint violations and fewer implicit assumptions, with higher structural clarity.

Main Qualitative Results

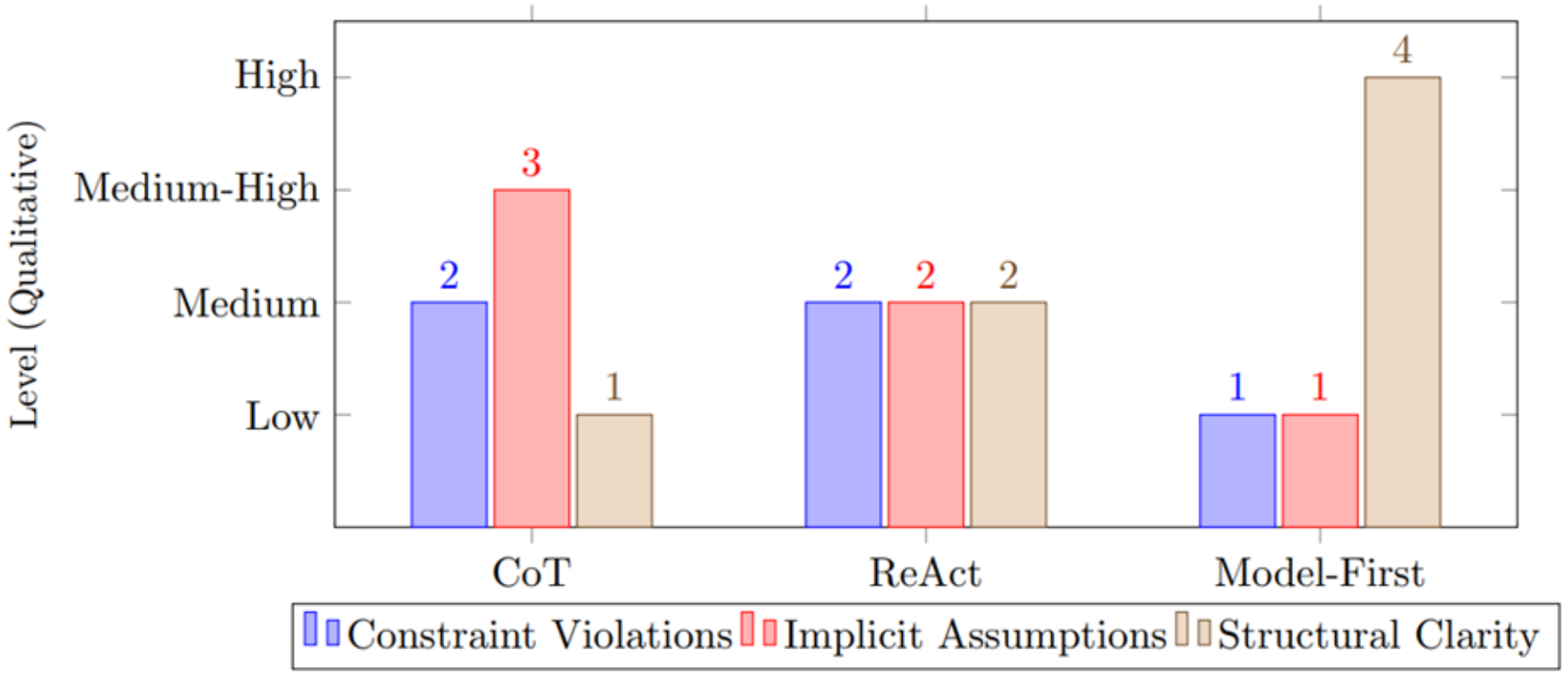


Figure 2: Qualitative comparison of reasoning strategies across tasks. Levels: Low=1, Medium=2, Medium-High=3, High=4. Rare/Frequent/Occasional mapped as 1/3/2 respectively.

- Lower is better for constraint violations and implicit assumptions
- Higher is better for structural clarity
- Ratings are qualitative and manually verified

Why CoT and ReAct Still Fail

Chain-of-Thought (CoT)

- Skips critical intermediate states
- Introduces unstated assumptions
- Fails to maintain global consistency

ReAct

- Improves local reasoning
- Observations may be assumed
- Global constraints remain unstable over long horizons

Why MFR Performs Better

fact_check

Explicit Constraint Grounding

MFR ensures that every generated step is strictly tied to the defined task constraints, minimizing errors.

psychology

Reduced Implicit Assumptions

By formalizing transitions, the model avoids relying on unstated or inferred information that could lead to failure.

account_tree

Improved Structural Clarity

The clear, hierarchical reasoning process makes the resulting plans easier to interpret, verify, and execute.

Limitations

- selected examples rather than exhaustive benchmarking
- qualitative ratings only
- depends on accurate model construction
- increased token overhead
- not a formal verifier