NEURAL INFORMATION PROCESSING SYSTEMS

Stanford

Massachusetts Institute of Technology

ReCAP: Recursive Context-Aware Reasoning and Planning for Large Language Model Agents

Zhenyu Zhang[1*], Tianyi Chen[1*], Weiran Xu[1*], Alex Pentland[2,3], Jiaxin Pei[2]

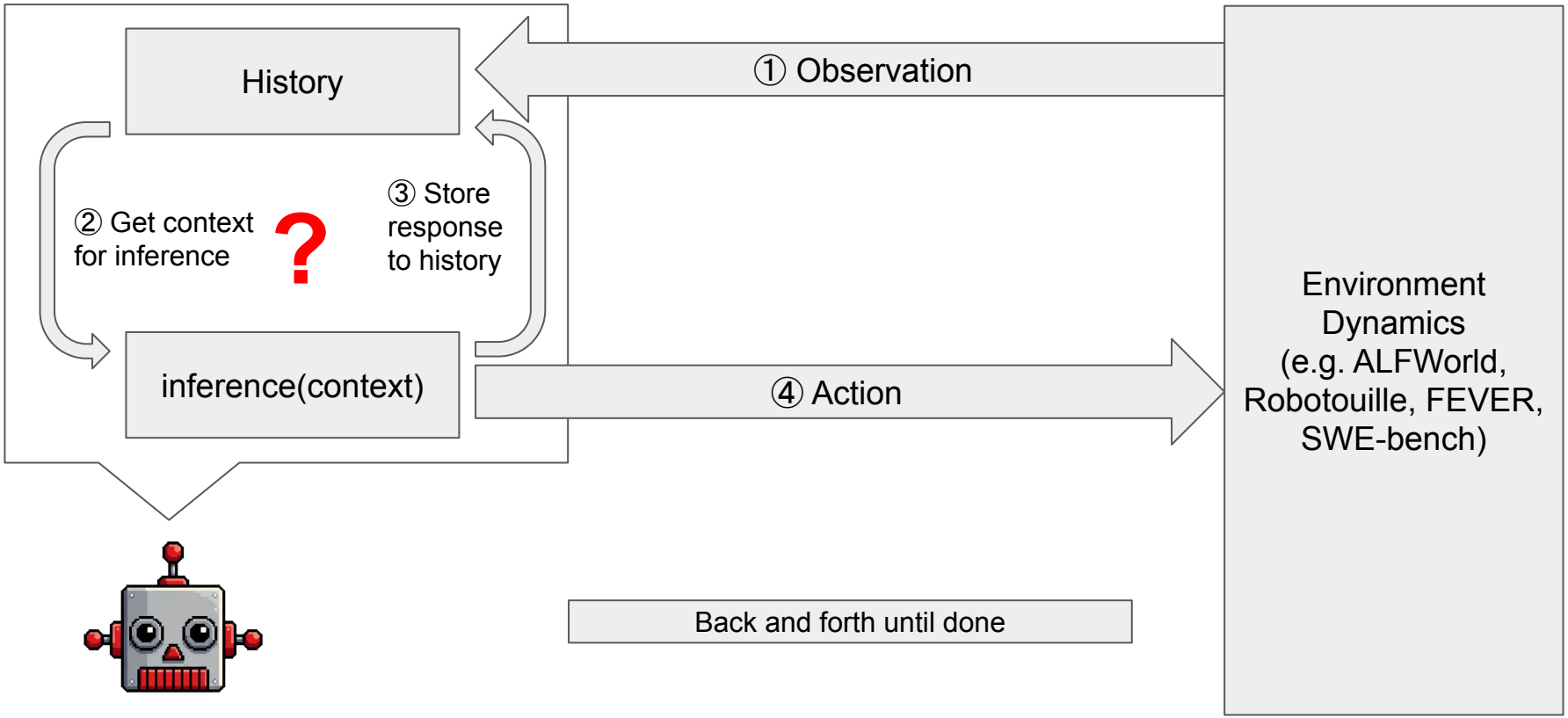
[1]Department of Computer Science, Stanford University [2]Stanford Institute for Human-Centered AI [3]MIT Media Lab

Stanford University Human-Centered Artificial Intelligence

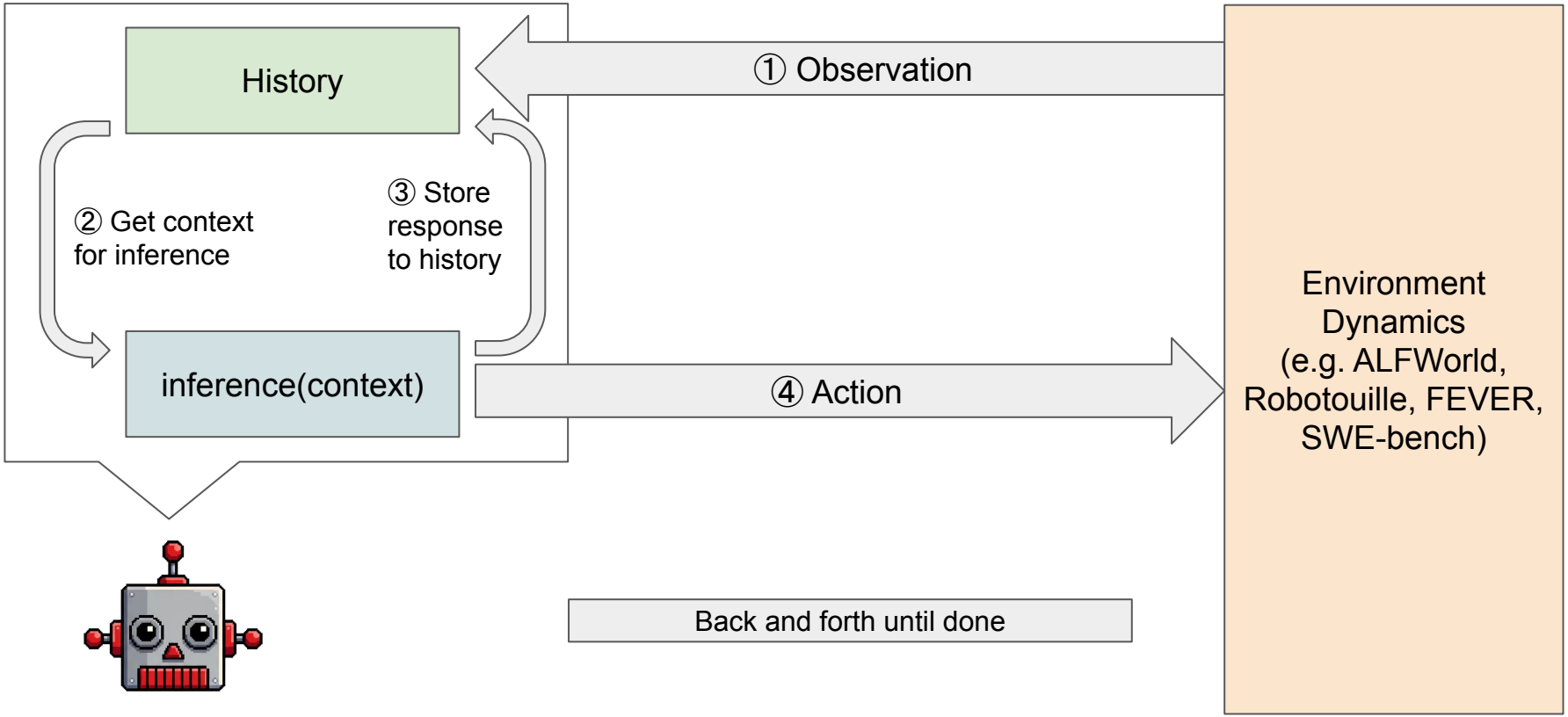
Stanford | ENGINEERING Computer Science

mit media lab

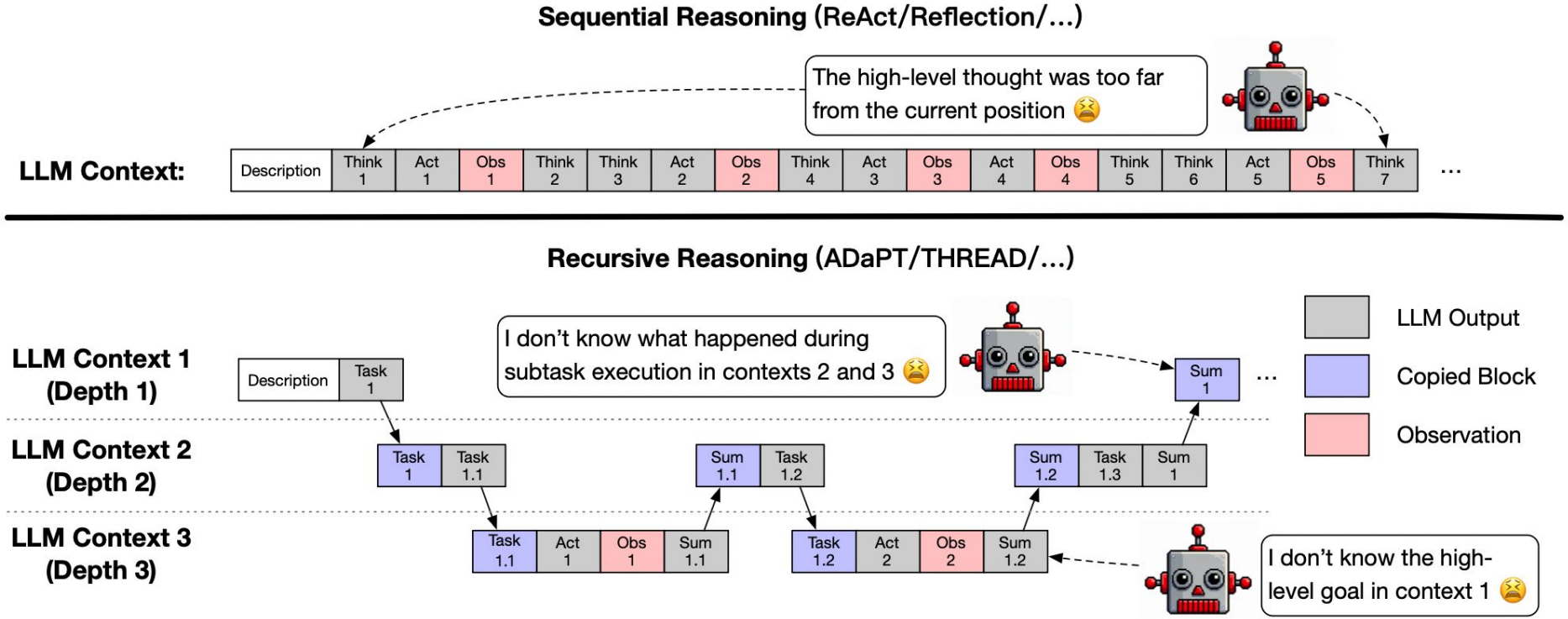
Background: Language Model Agent



Background: Language Model Agent



Limitations of pure sequential/hierarchical prompting



ReCAP: Recursive Context-Aware Reasoning and Planning

1. Recursive Task Decomposition with Plan-Ahead

2. Consistent Multi-level Context and Structured Injection

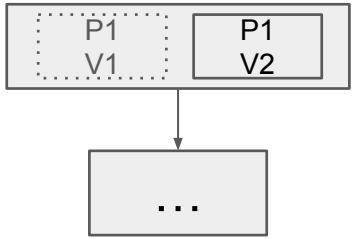
3. Scalable Memory Efficiency with a Context Tree

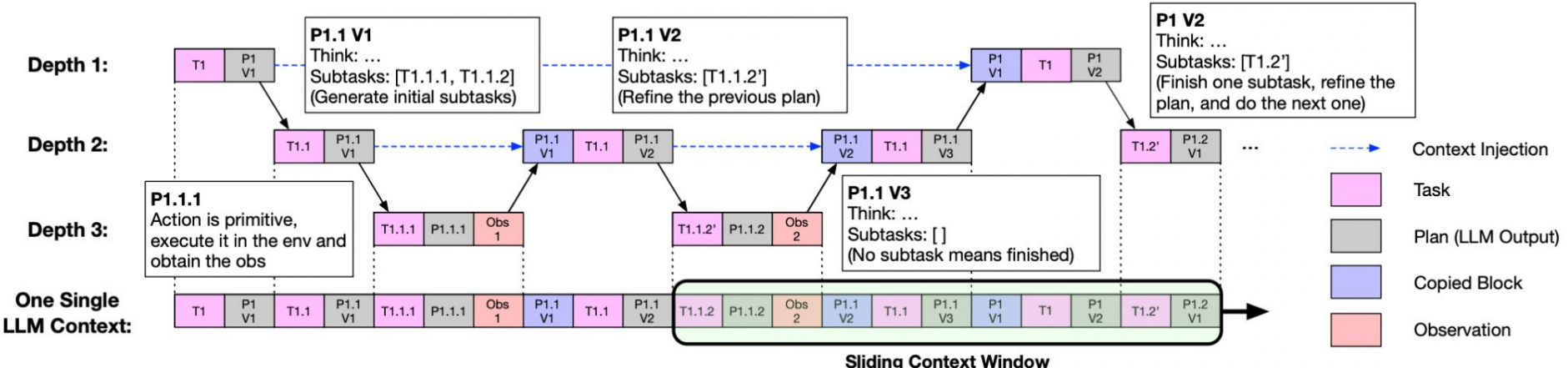
Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C

Context Tree





ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C

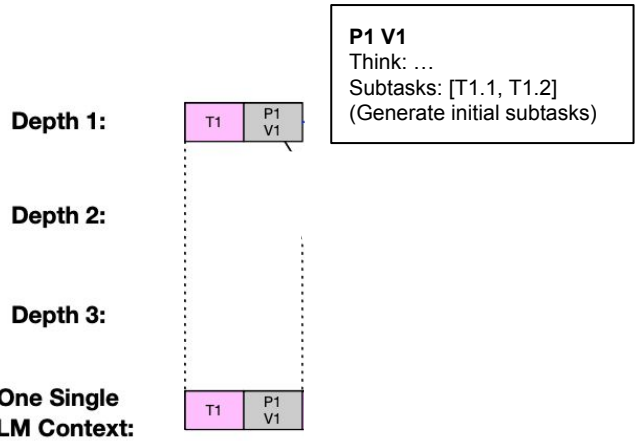
Empty

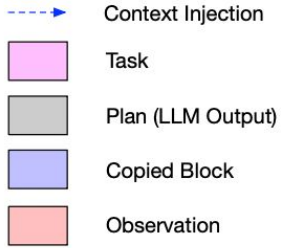
ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

1: $(T, S) \leftarrow \pi(C)$
2: **while** S not empty **do**
3: **if** $S[0]$ is primitive **then**
4: $O \leftarrow \mathcal{E}(S[0])$
5: $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
6: **else**
7: $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
8: **end if**
9: $C \leftarrow C \parallel \langle T, S[1:] \rangle$
10: $(T, S) \leftarrow \rho(C)$
11: **end while**
12: **return** C



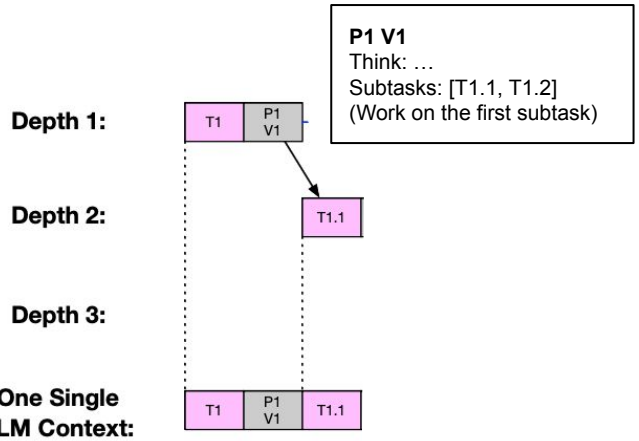


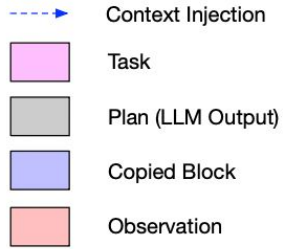
ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C



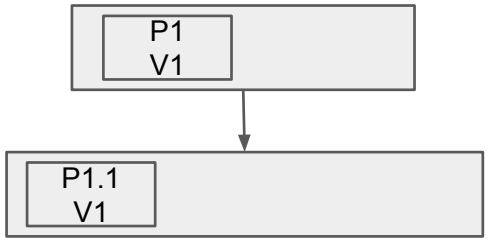


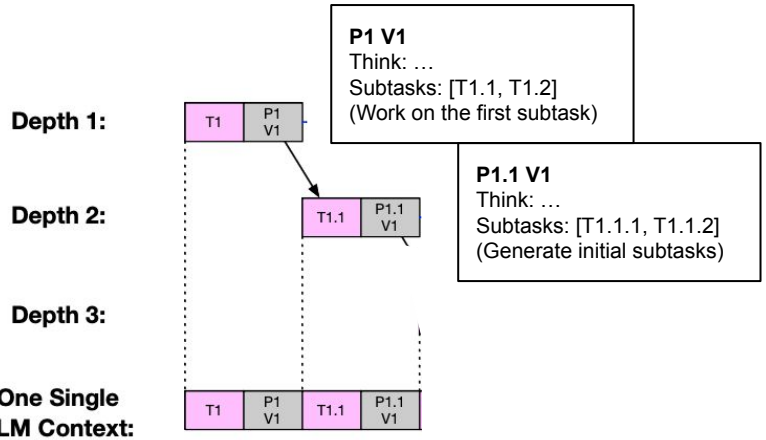
ReCAP

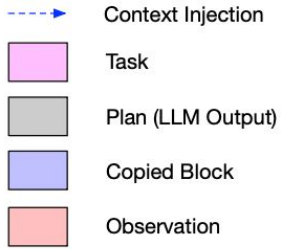
Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

1: $(T, S) \leftarrow \pi(C)$
2: **while** S not empty **do**
3: **if** $S[0]$ is primitive **then**
4: $O \leftarrow \mathcal{E}(S[0])$
5: $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
6: **else**
7: $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
8: **end if**
9: $C \leftarrow C \parallel \langle T, S[1:] \rangle$
10: $(T, S) \leftarrow \rho(C)$
11: **end while**
12: **return** C



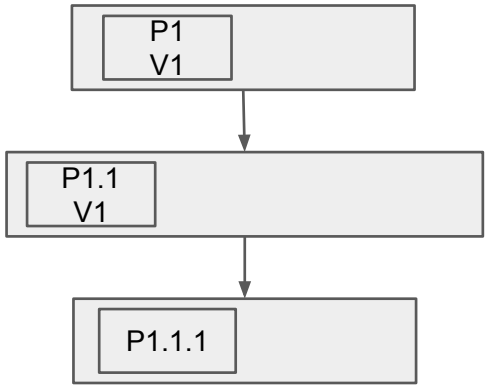


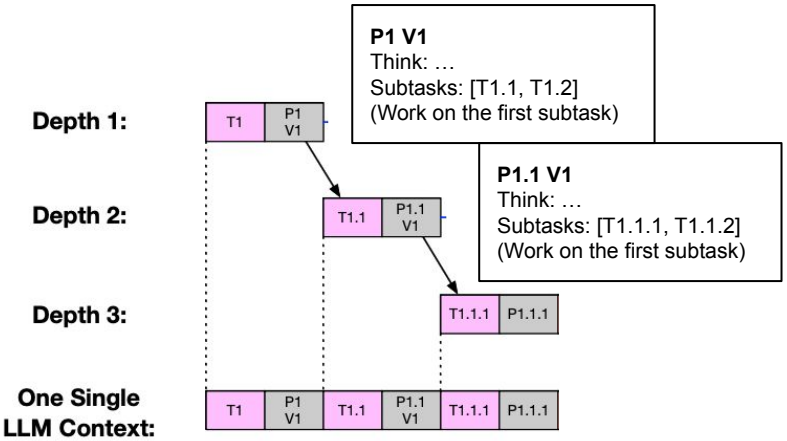


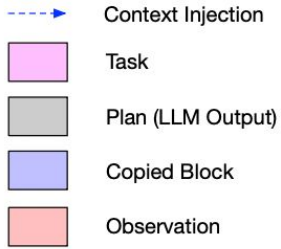
ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C
1: $(T, S) \leftarrow \pi(C)$
2: **while** S not empty **do**
3: **if** $S[0]$ is primitive **then**
4: $O \leftarrow \mathcal{E}(S[0])$
5: $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
6: **else**
7: $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
8: **end if**
9: $C \leftarrow C \parallel \langle T, S[1:] \rangle$
10: $(T, S) \leftarrow \rho(C)$
11: **end while**
12: **return** C





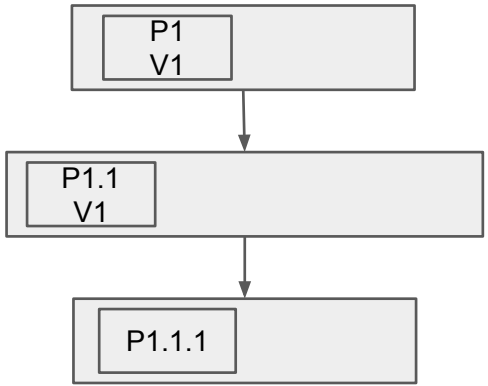


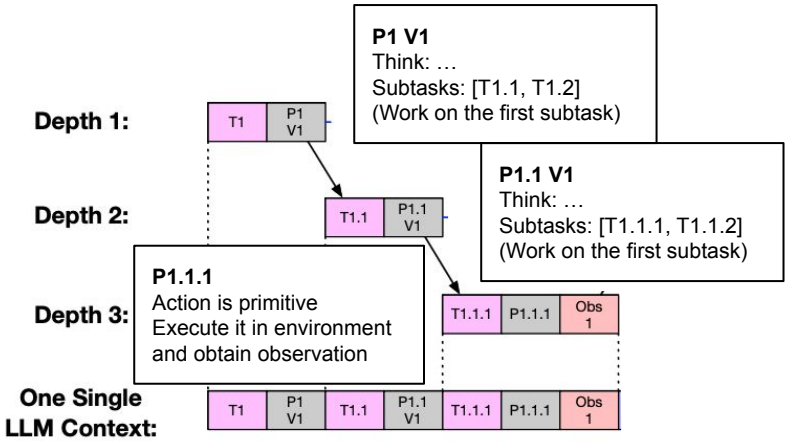
ReCAP

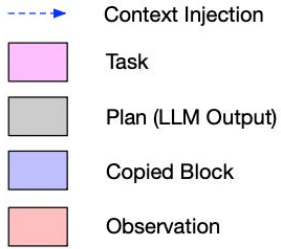
Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C





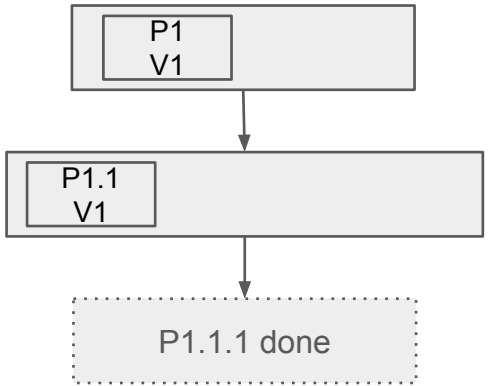


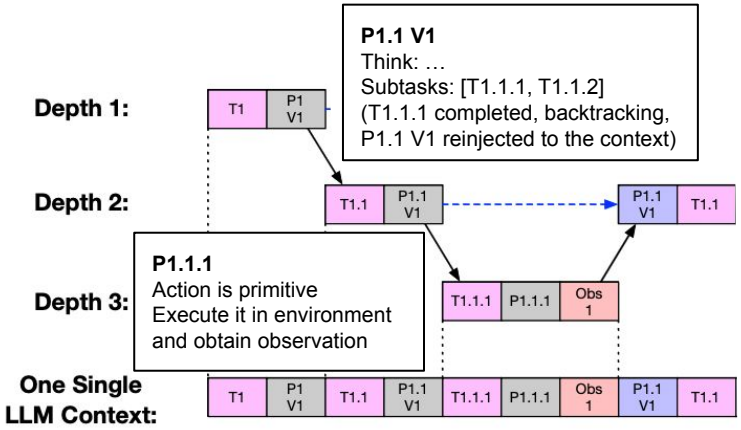
ReCAP

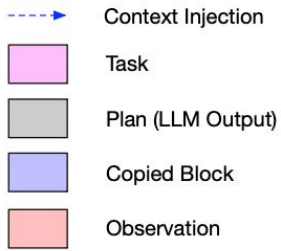
Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C





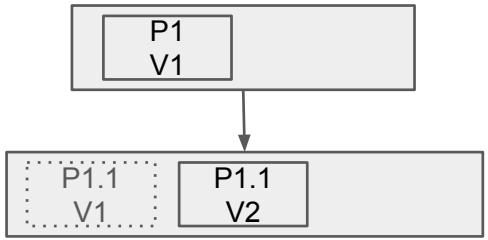


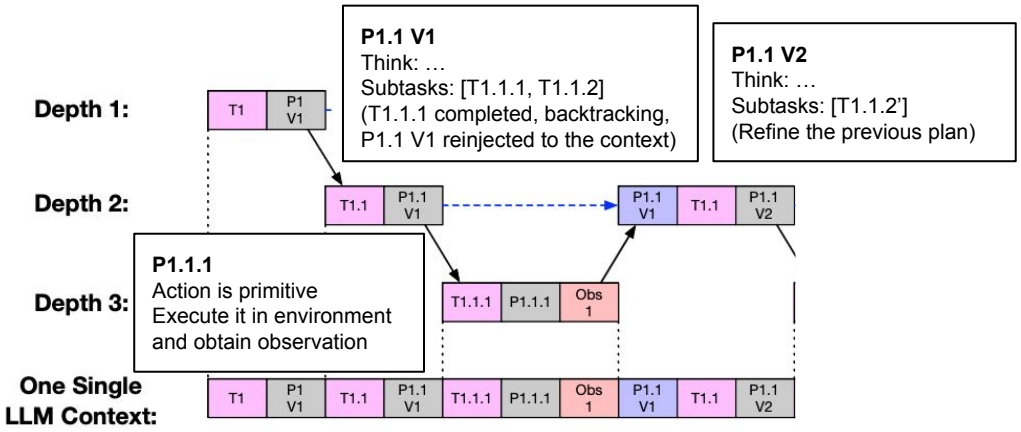
ReCAP

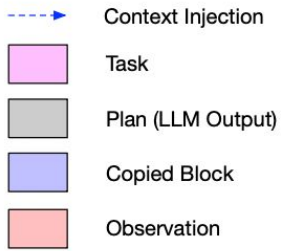
Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C





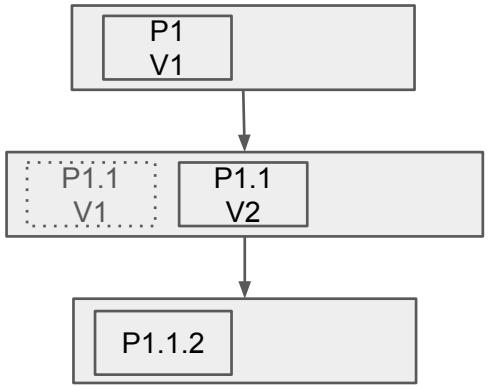


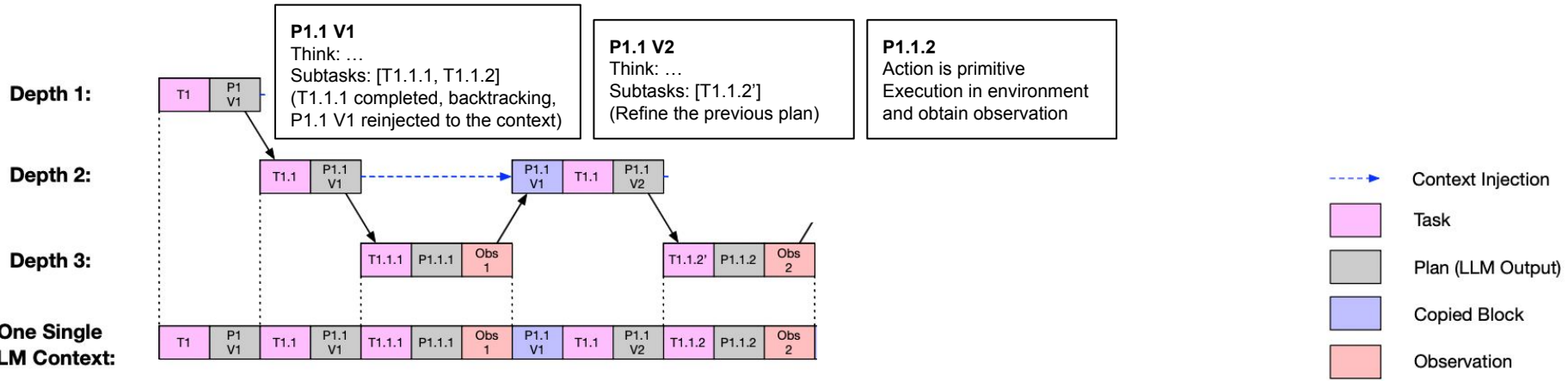
ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C



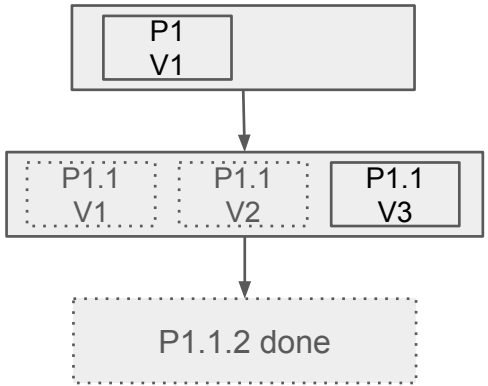


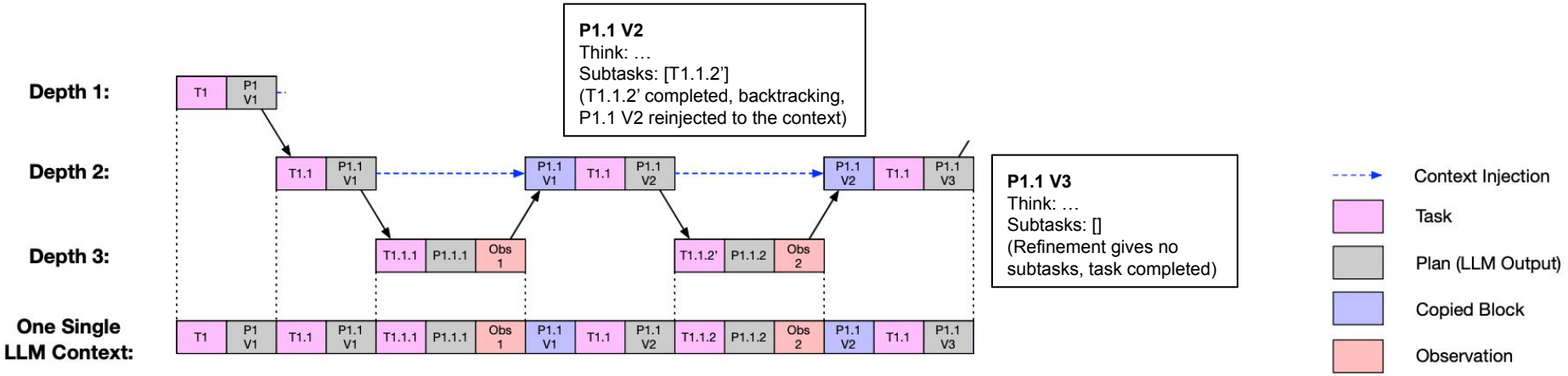
ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C



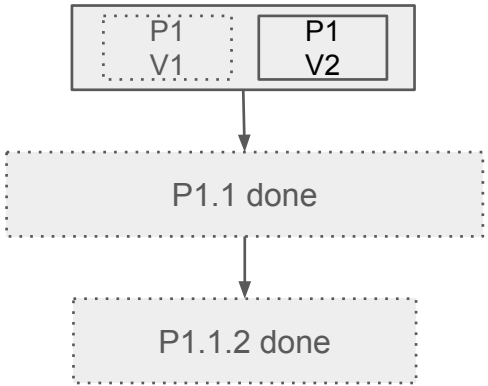


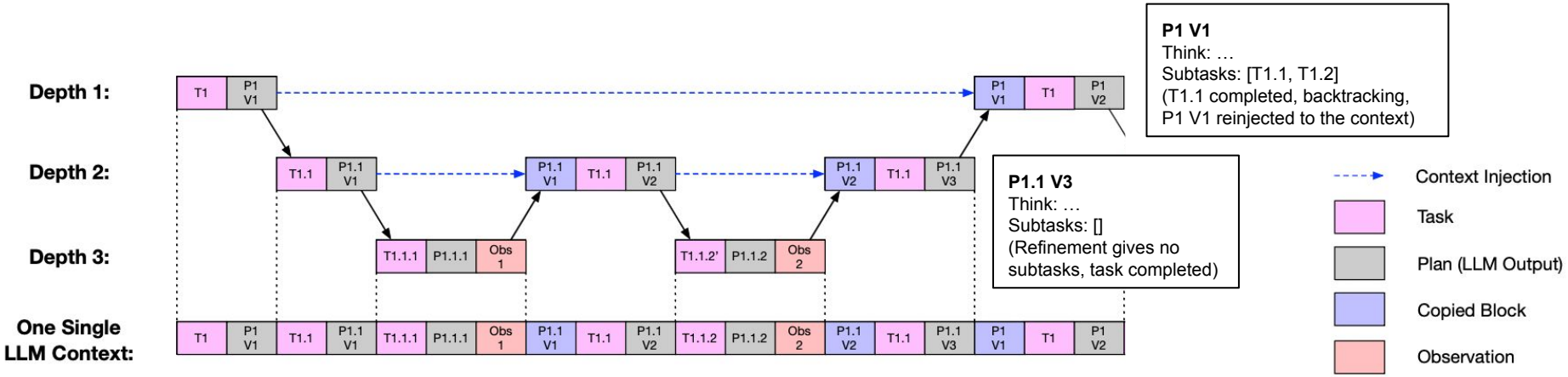
ReCAP

Algorithm 1 ReCAP

Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C



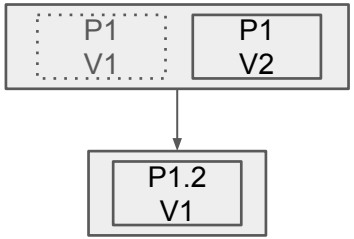


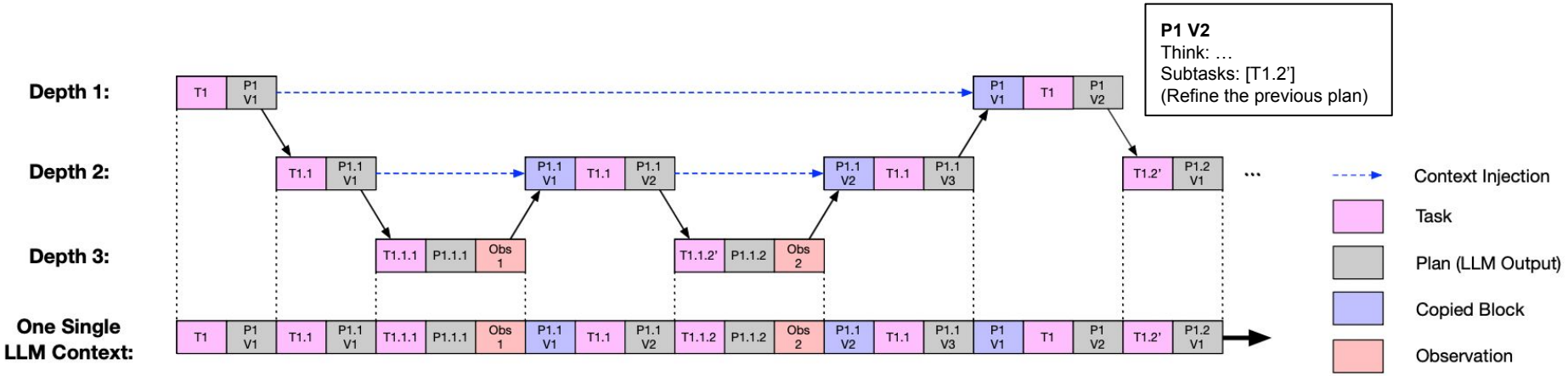
ReCAP

Algorithm 1 ReCAP

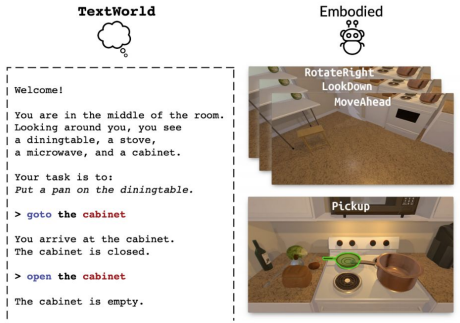
Require: LLM Context C
Ensure: Updated context C

$(T, S) \leftarrow \pi(C)$
while S not empty **do**
  if $S[0]$ is primitive **then**
    $O \leftarrow \mathcal{E}(S[0])$
    $C \leftarrow C \parallel \langle T, S, S[0], O \rangle$
  else
    $C \leftarrow \texttt{ReCAP}(C \parallel \langle T, S, S[0] \rangle)$
  end if
  $C \leftarrow C \parallel \langle T, S[1:] \rangle$
  $(T, S) \leftarrow \rho(C)$
end while
return C





Benchmarks



ALFWorld

Symbolic Household Reasoning
5-15 steps

Robotouille

Embodied Cooking (long horizon)
10-80 steps

FEVER

Knowledge Retrieval & Verification
<10 steps

SWE-bench Verified

Real-world Code Editing
5-257 steps

Experiment & Main Results

Baselines: Sequential (Direct Output / CoT / ReAct / Act), Hierarchical (ADaPT)

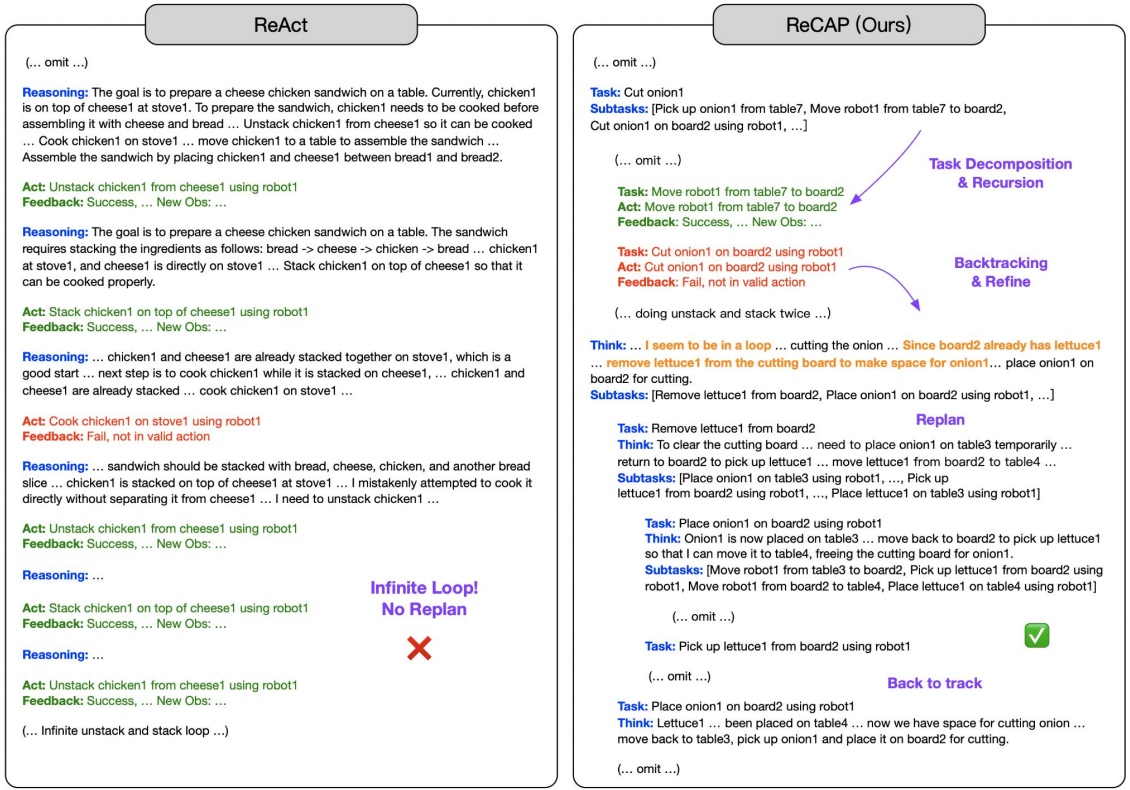
Protocol: Strict pass@1 (single run, no retry or ensembling)

LLM: GPT-4o (2024-08-06 build)

Metric: Success Rate (%)

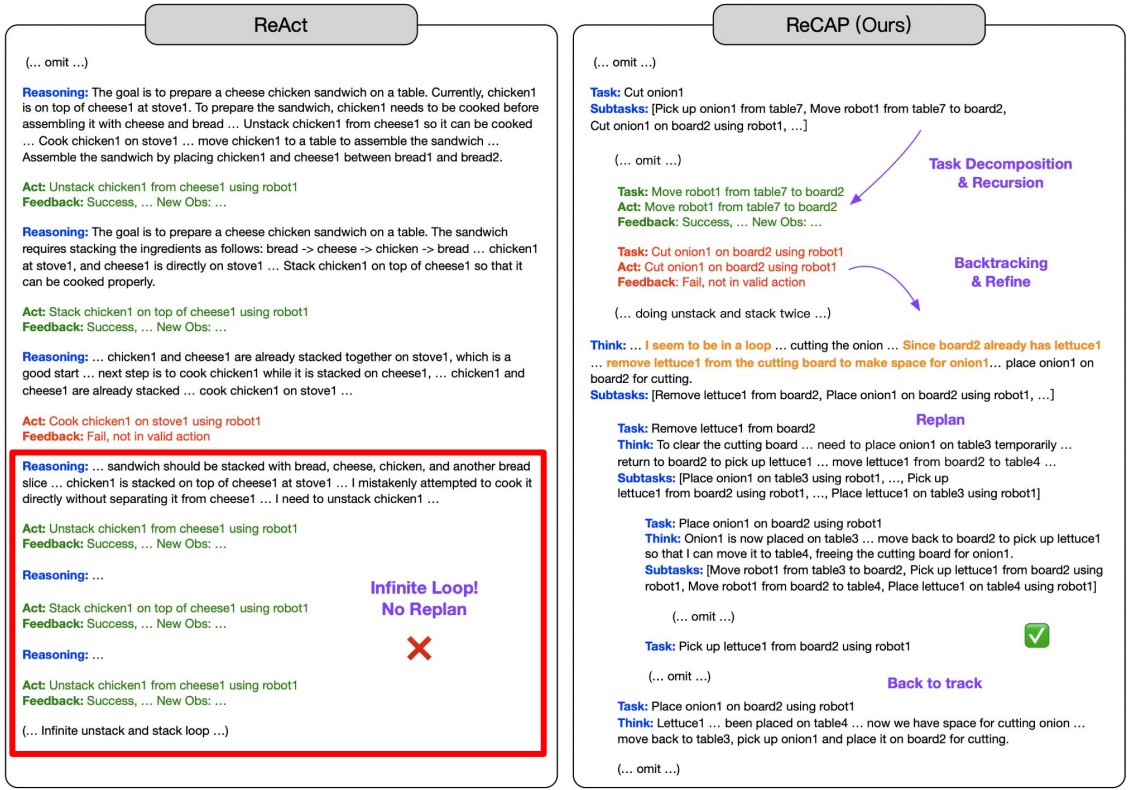
Benchmark	ReCAP (%)	Best Baseline (%)	Δ (%)
ALFWorld	**91.0**	ReAct 84.0	+7
Robotouille (Sync)	**70.0**	ADaPT 40.0	+32
Robotouille (Async)	**53.0**	ReAct 24.0	+29
FEVER	63.5	ReAct 63.5	-
SWE-bench Verified	**44.8**	ReAct 39.6	+5

Qualitative Example & Insights



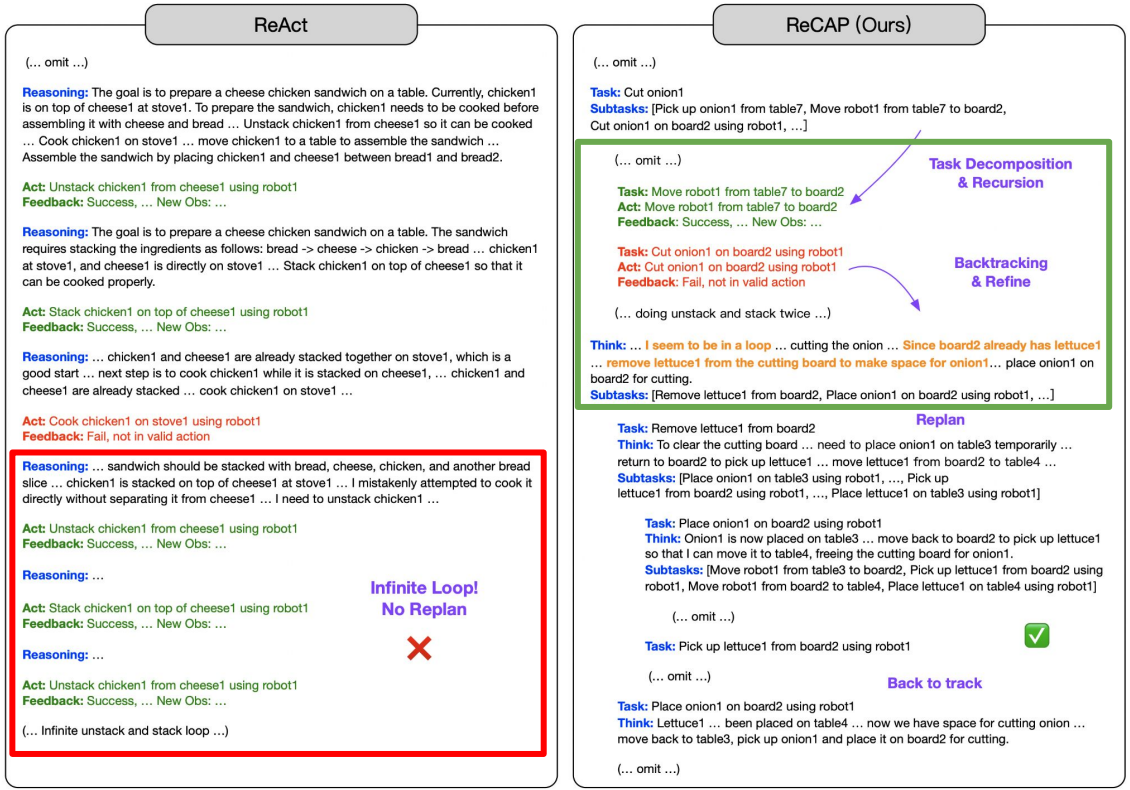
Detects failure loops → recursive context repair

Qualitative Example & Insights



Detects failure loops → recursive context repair

Qualitative Example & Insights



Detects failure loops → recursive context repair

Qualitative Example & Insights

ReAct

(… omit …)

Reasoning: The goal is to prepare a cheese chicken sandwich on a table. Currently, chicken1 is on top of cheese1 at stove1. To prepare the sandwich, chicken1 needs to be cooked before assembling it with cheese and bread … Unstack chicken1 from cheese1 so it can be cooked … Cook chicken1 on stove1 … move chicken1 to a table to assemble the sandwich … Assemble the sandwich by placing chicken1 and cheese1 between bread1 and bread2.

Act: Unstack chicken1 from cheese1 using robot1
Feedback: Success, … New Obs: …

Reasoning: The goal is to prepare a cheese chicken sandwich on a table. The sandwich requires stacking the ingredients as follows: bread -> cheese -> chicken -> bread … chicken1 at stove1, and cheese1 is directly on stove1 … Stack chicken1 on top of cheese1 so that it can be cooked properly.

Act: Stack chicken1 on top of cheese1 using robot1
Feedback: Success, … New Obs: …

Reasoning: … chicken1 and cheese1 are already stacked together on stove1, which is a good start … next step is to cook chicken1 while it is stacked on cheese1, … chicken1 and cheese1 are already stacked … cook chicken1 on stove1 …

Act: Cook chicken1 on stove1 using robot1
Feedback: Fail, not in valid action

Reasoning: … sandwich should be stacked with bread, cheese, chicken, and another bread slice … chicken1 is stacked on top of cheese1 at stove1 … I mistakenly attempted to cook it directly without separating it from cheese1 … I need to unstack chicken1 …

Act: Unstack chicken1 from cheese1 using robot1
Feedback: Success, … New Obs: …

Reasoning: …

Act: Stack chicken1 on top of cheese1 using robot1
Feedback: Success, … New Obs: …

Infinite Loop!
No Replan

❌

Reasoning: …

Act: Unstack chicken1 from cheese1 using robot1
Feedback: Success, … New Obs: …

(… Infinite unstack and stack loop …)

ReCAP (Ours)

(… omit …)

Task: Cut onion1
Subtasks: [Pick up onion1 from table7, Move robot1 from table7 to board2, Cut onion1 on board2 using robot1, …]

(… omit …)

Task Decomposition & Recursion

Task: Move robot1 from table7 to board2
Act: Move robot1 from table7 to board2
Feedback: Success, … New Obs: …

Task: Cut onion1 on board2 using robot1
Act: Cut onion1 on board2 using robot1
Feedback: Fail, not in valid action

Backtracking & Refine

(… doing unstack and stack twice …)

Think: … **I seem to be in a loop** … cutting the onion … **Since board2 already has lettuce1** … **remove lettuce1 from the cutting board to make space for onion1**… place onion1 on board2 for cutting.
Subtasks: [Remove lettuce1 from board2, Place onion1 on board2 using robot1, …]

Replan

Task: Remove lettuce1 from board2
Think: To clear the cutting board … need to place onion1 on table3 temporarily … return to board2 to pick up lettuce1 … move lettuce1 from board2 to table4 …
Subtasks: [Place onion1 on table3 using robot1, …, Pick up lettuce1 from board2 using robot1, …, Place lettuce1 on table3 using robot1]

Task: Place onion1 on board2 using robot1
Think: Onion1 is now placed on table3 … move back to board2 to pick up lettuce1 so that I can move it to table4, freeing the cutting board for onion1.
Subtasks: [Move robot1 from table3 to board2, Pick up lettuce1 from board2 using robot1, Move robot1 from board2 to table4, Place lettuce1 on table4 using robot1]

(… omit …)

Task: Pick up lettuce1 from board2 using robot1

✅

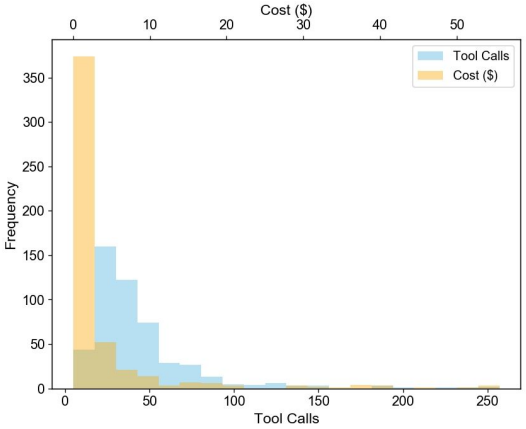
(… omit …)

Back to track

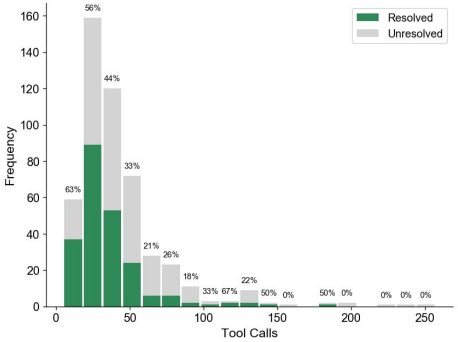
Task: Place onion1 on board2 using robot1
Think: Lettuce1 … been placed on table4 … now we have space for cutting onion … move back to table3, pick up onion1 and place it on board2 for cutting.

(… omit …)

Detects failure loops → recursive context repair



Tool Call and Cost Distributions
(SWE-bench Verified)



Resolve Rate vs Tool Calls
(SWE-bench Verified)

Ablations & Generalization

Method	GPT-4o	Qwen 2.5-32B	Qwen 2.5-72B	LLaMA-4 (400 B)	DeepSeek-V3 (671 B)
ReAct	63.0 %	10.0 %	23.0 %	37.0 %	57.0 %
ReCAP	**90.0 %**	**33.0 %**	**53.0 %**	**60.0 %**	**87.0 %**

Cross-Model Generalization on Robotouille (#2, #4, #6 tasks). Success rate (%).

Ablations & Generalization

Method	GPT-4o	Qwen 2.5-32B	Qwen 2.5-72B	LLaMA-4 (400 B)	DeepSeek-V3 (671 B)
ReAct	63.0 %	10.0 %	23.0 %	37.0 %	57.0 %
ReCAP	**90.0 %**	**33.0 %**	**53.0 %**	**60.0 %**	**87.0 %**

Cross-Model Generalization on Robotouille (#2, #4, #6 tasks). Success rate (%).

Variant	Success Rate (%)	Observation
Original ReCAP	**80**	**Full recursion + reasoning traces** → best performance
Think Many	70	**Keeps too much reasoning history** but still strong
No Think	60	**Removing reasoning text** hurts alignment
Name Only	55	**Without reasoning traces** → poor context recall
Level 5	70	**Deep recursion** works well
Level 4	60	**Slightly shallower depth** → small drop
Level 3	10	**Too shallow** → fails to decompose tasks
Level 2	0	**No recursive decomposition** → task collapse

Ablation Study – Recursive Depth and Reasoning Traces

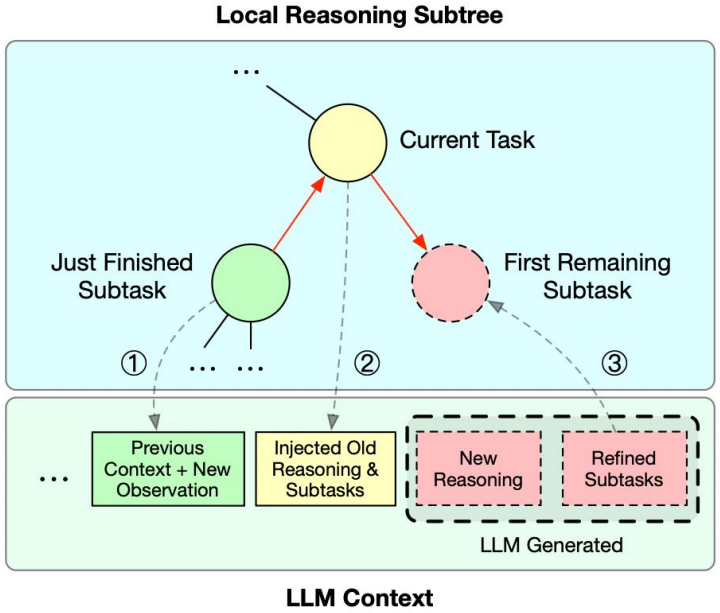
Conclusion

Main Contributions

- 3 mechanisms:
  1. **Plan-ahead decomposition**: generate full subtask list, execute first, refine the rest
  2. **Structured context re-injection**: preserve cross-level coherence during recursion
  3. **Sliding-window efficiency** – bound active prompt while reintroducing essentials

- Unifies sequential and hierarchical prompting under one shared context

- Enables long-horizon reasoning without training or fine-tuning



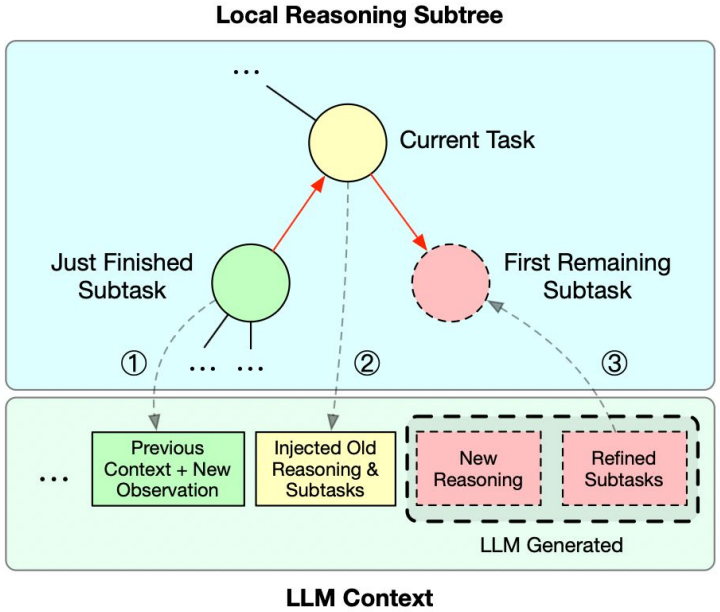
Conclusion

Main Contributions

- 3 mechanisms:
  1. **Plan-ahead decomposition**: generate full subtask list, execute first, refine the rest
  2. **Structured context re-injection**: preserve cross-level coherence during recursion
  3. **Sliding-window efficiency** – bound active prompt while reintroducing essentials

- Unifies sequential and hierarchical prompting under one shared context

- Enables long-horizon reasoning without training or fine-tuning



Future Directions

- Modular planning–execution architecture
- Reasoning compression
- Context graphs / memory routing
- **Goal:** make reasoning efficient, recursive, and memory-aware

Stanford

Massachusetts
Institute of
Technology

Thank You!

Stanford University
Human-Centered
Artificial Intelligence

Stanford | ENGINEERING
Computer Science