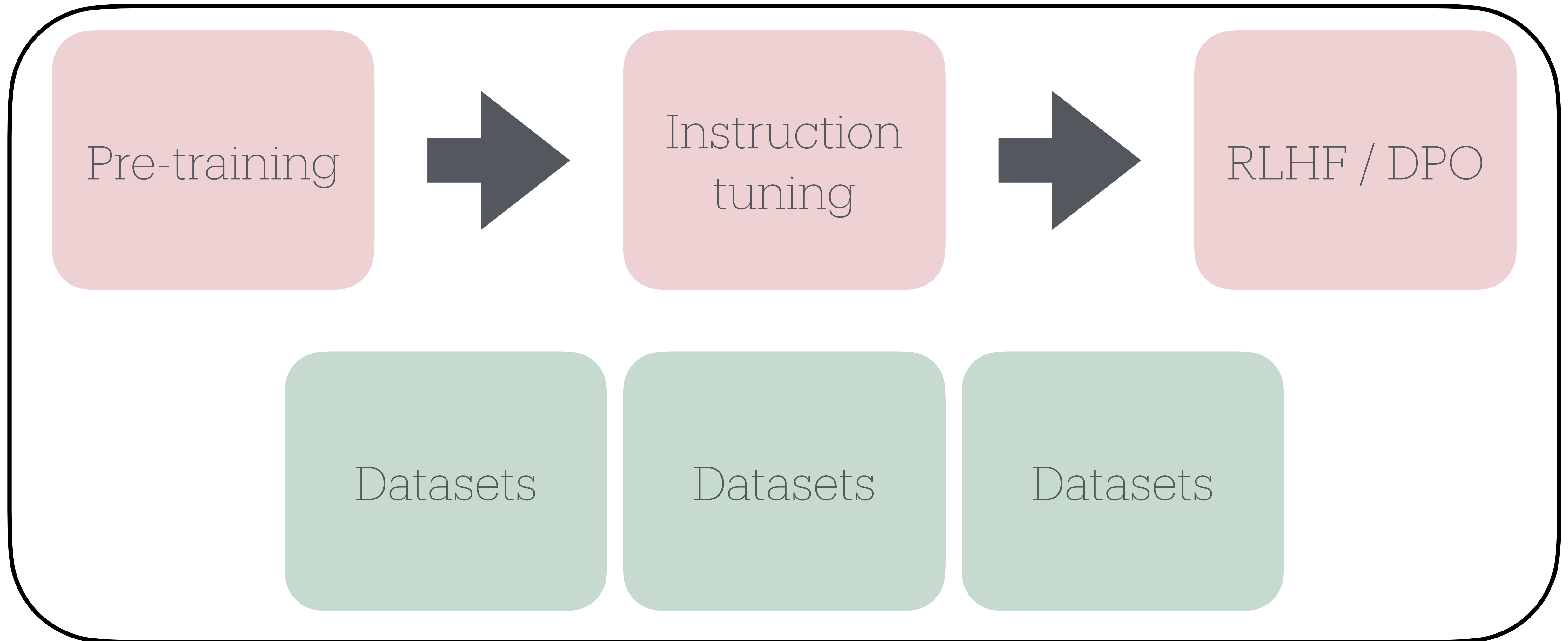


# Structured Dialogues

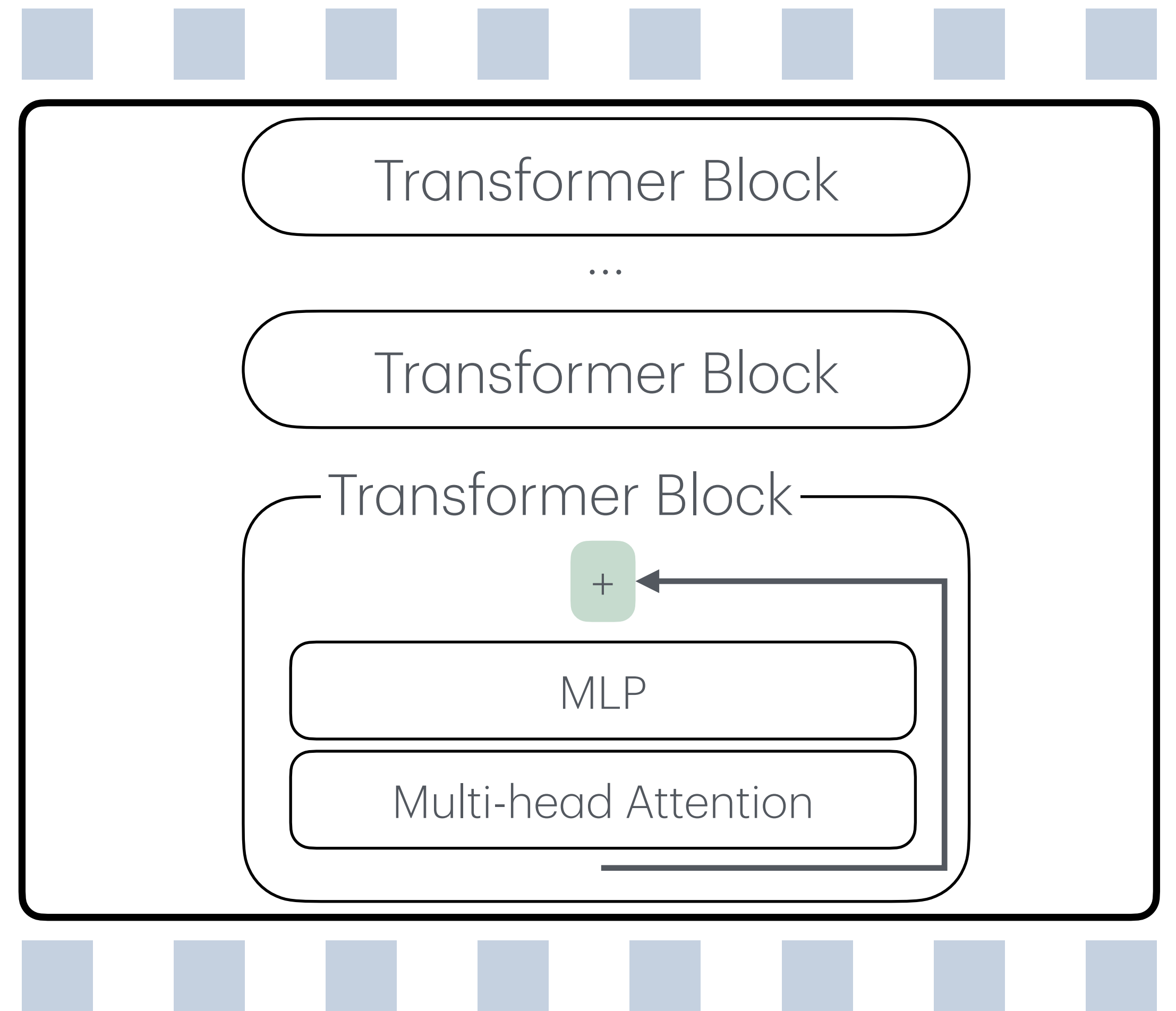
# Full Picture

Basic LLM



# Where does a LLM store information?

- Their weights
  - MLP and attention [1]
- Special tokens / activations [2,3]
  - Large activations or registers
- Their context



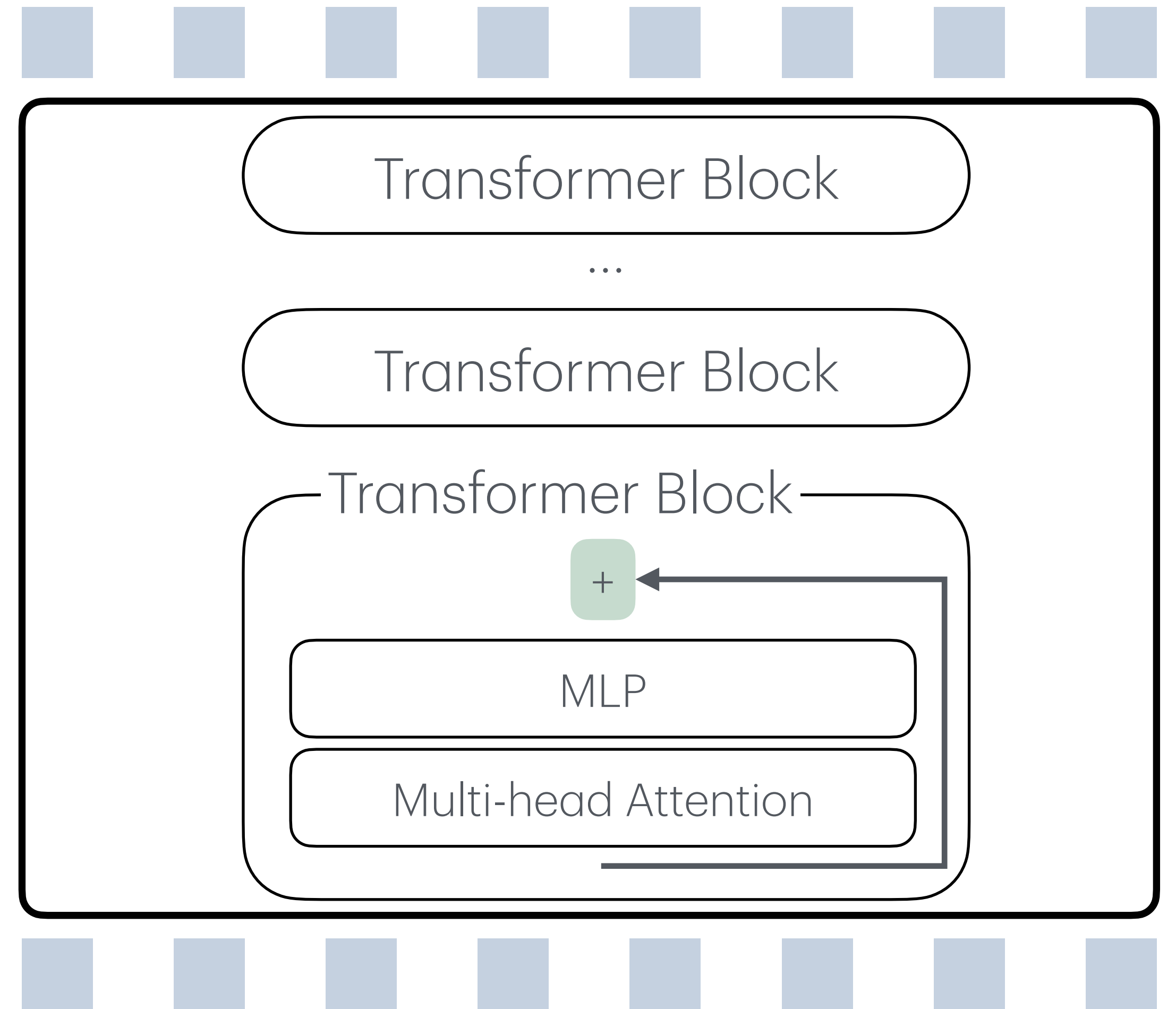
[1] Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws, Allen-Zhu 2024

[2] Vision Transformers Need Registers, Darcet et al 2023

[3] Massive Activations in Large Language Models, Sun et al 2024

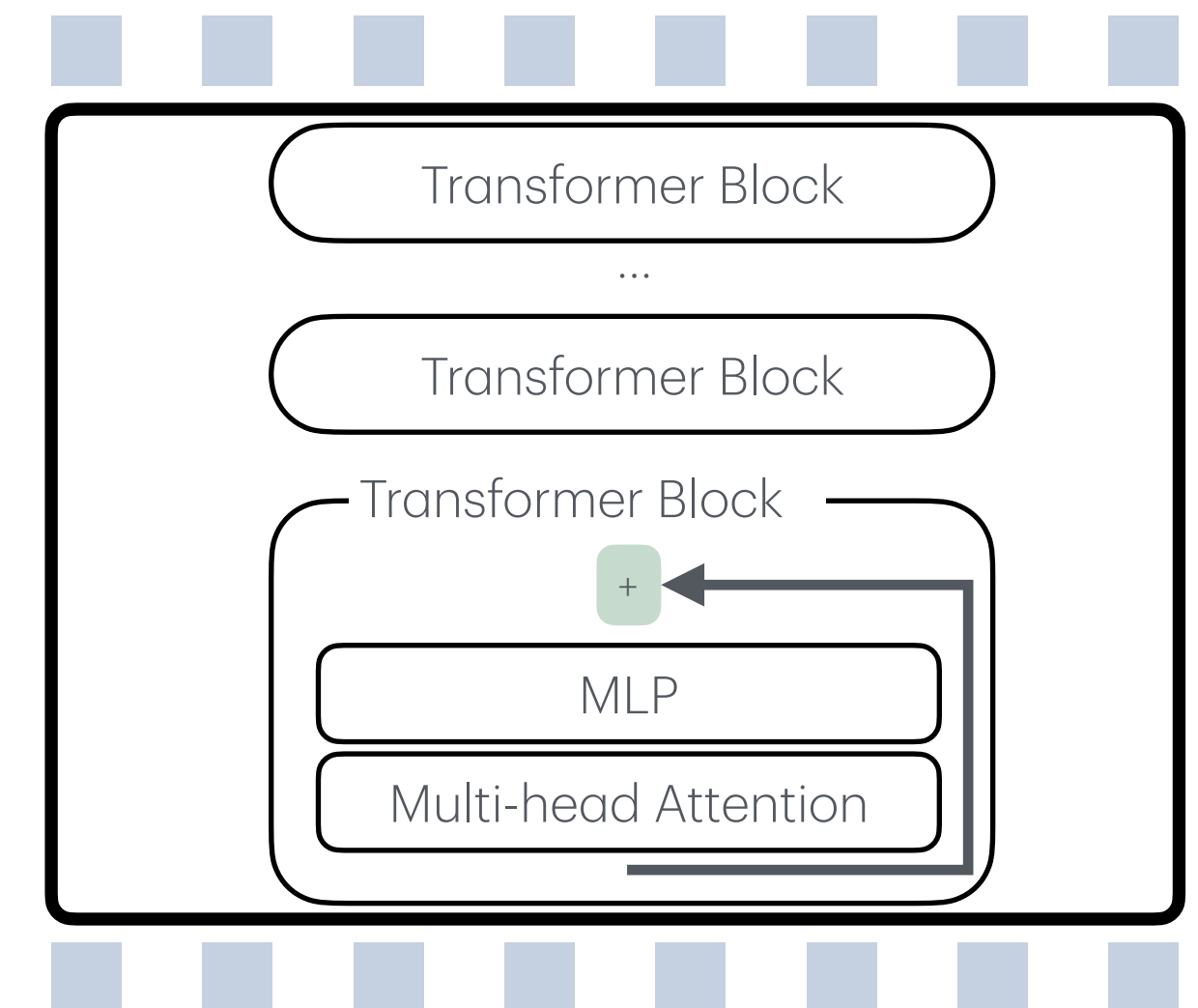
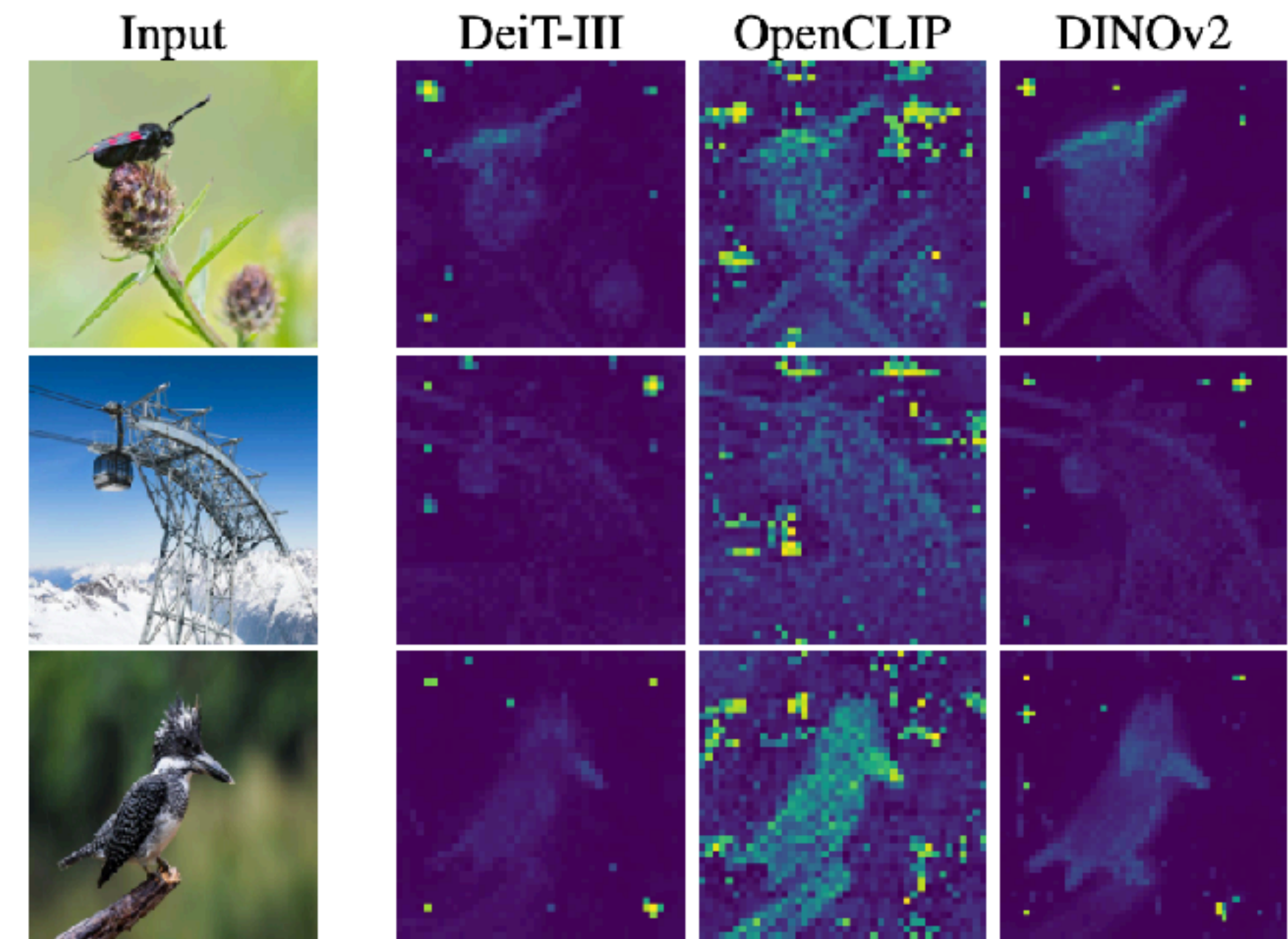
# Information in weights

- LLMs can store up to 2 bits of information per weight [1]
  - In MLP
  - In Attention
- 2 bits require very long training and multiple (up to 1000) augmentations of same information



# Special tokens / activations

- LLMs use special tokens to store information
  - LLMs attend to <BOS> token
  - VLMs attend to background

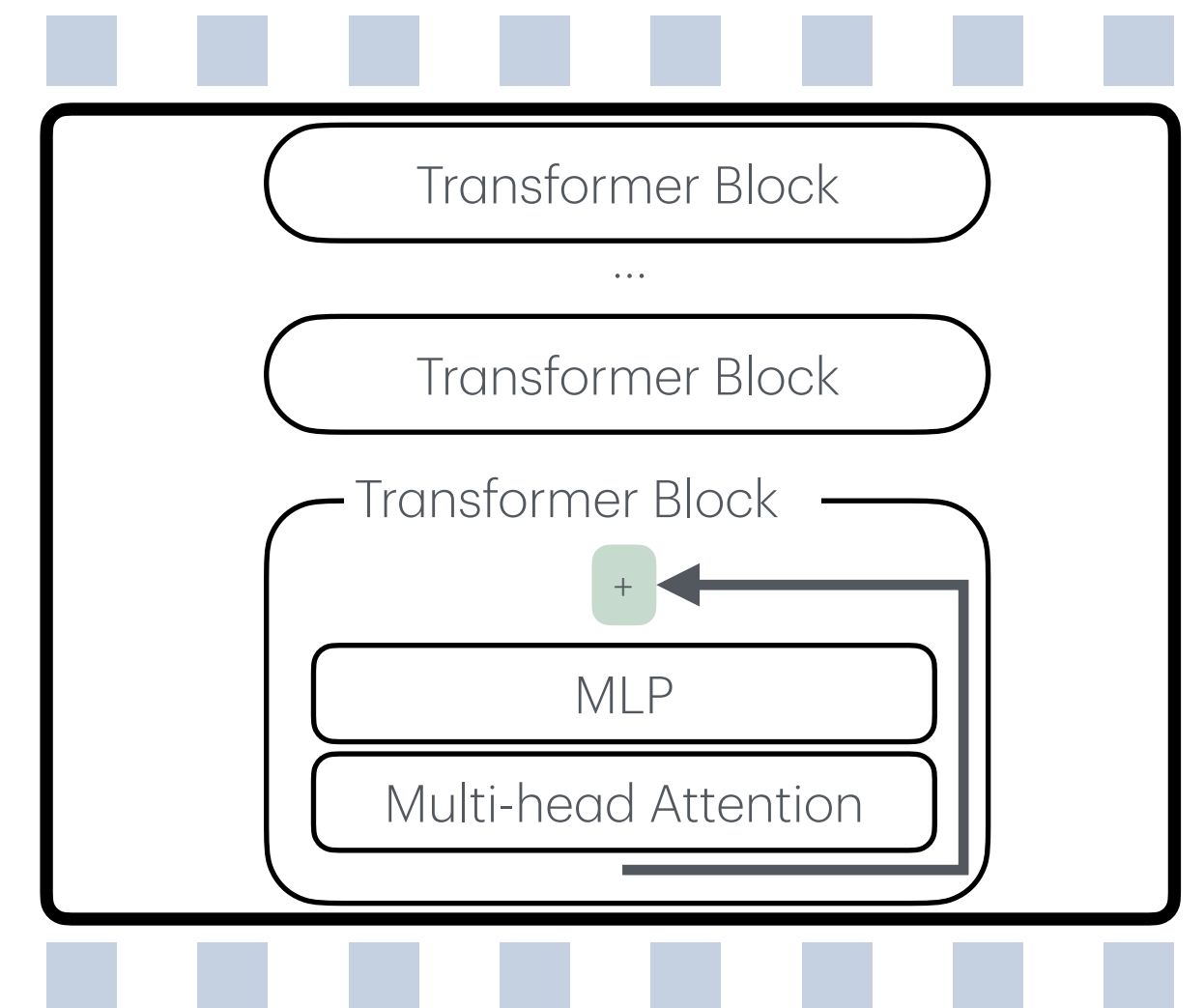


[1] Vision Transformers Need Registers, Darcet et al 2023

[2] Massive Activations in Large Language Models, Sun et al 2024

# Context

- LLMs store information in their context
- Examples
  - System prompt
  - Retrieval Augmented Generation
  - ...



[1] Vision Transformers Need Registers, Darcet et al 2023

[2] Massive Activations in Large Language Models, Sun et al 2024

# In context learning

- Describe the task
  - Give examples input - output pairs
  - Then ask for your specific

Translate words from English to German using JSON as an output. Here are some examples

Car

```
{"English": "Car", "German": "Auto"}
```

Sun

```
{"English": "Sun", "German": "Sonne"}
```

Moon

# In context learning

Why does it work?

- LLMs like repeating patterns
  - Likely exist in pre-training data
- Examples of in-context prompts and answers during training (instruction tuning, alignment)

Translate words from English to German using JSON as an output. Here are some examples

Car

```
{"English": "Car", "German": "Auto"}
```

Sun

```
{"English": "Sun", "German": "Sonne"}
```

Moon



# In context learning

What does it work for?

- Formatting outputs
- Simple requests

Translate words from English to German using JSON as an output. Here are some examples

Car  
{“English”: “Car”, “German”: “Auto”}

Sun  
{“English”: “Sun”, “German”: “Sonne”}

Moon

# Chain of thought

- Ask model to derive answer
  - Pre-instruction tuning: In-context example of reasoning
  - Post-instruction tuning
    - Ask model to think step-by-step before giving the answer
    - Guide model through thinking process

## Standard Prompting

### Model Input

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: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

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.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

# Chain of thought

## Why does it work?

- More output tokens = better performance
  - Delays making a decision
- Can work around tokenization issues
  - Break up numbers

### Standard Prompting

#### Model Input

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: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

#### Model Output

A: The answer is 27. ❌

### Chain-of-Thought Prompting

#### Model Input

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.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

#### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

# Chain of thought

- Order matters
  - Think first, then answer
- Chain-of-BS: Ask model to give answer and justify it

## Standard Prompting

### Model Input

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: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The answer is 27. ❌

## Chain-of-Thought Prompting

### Model Input

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.

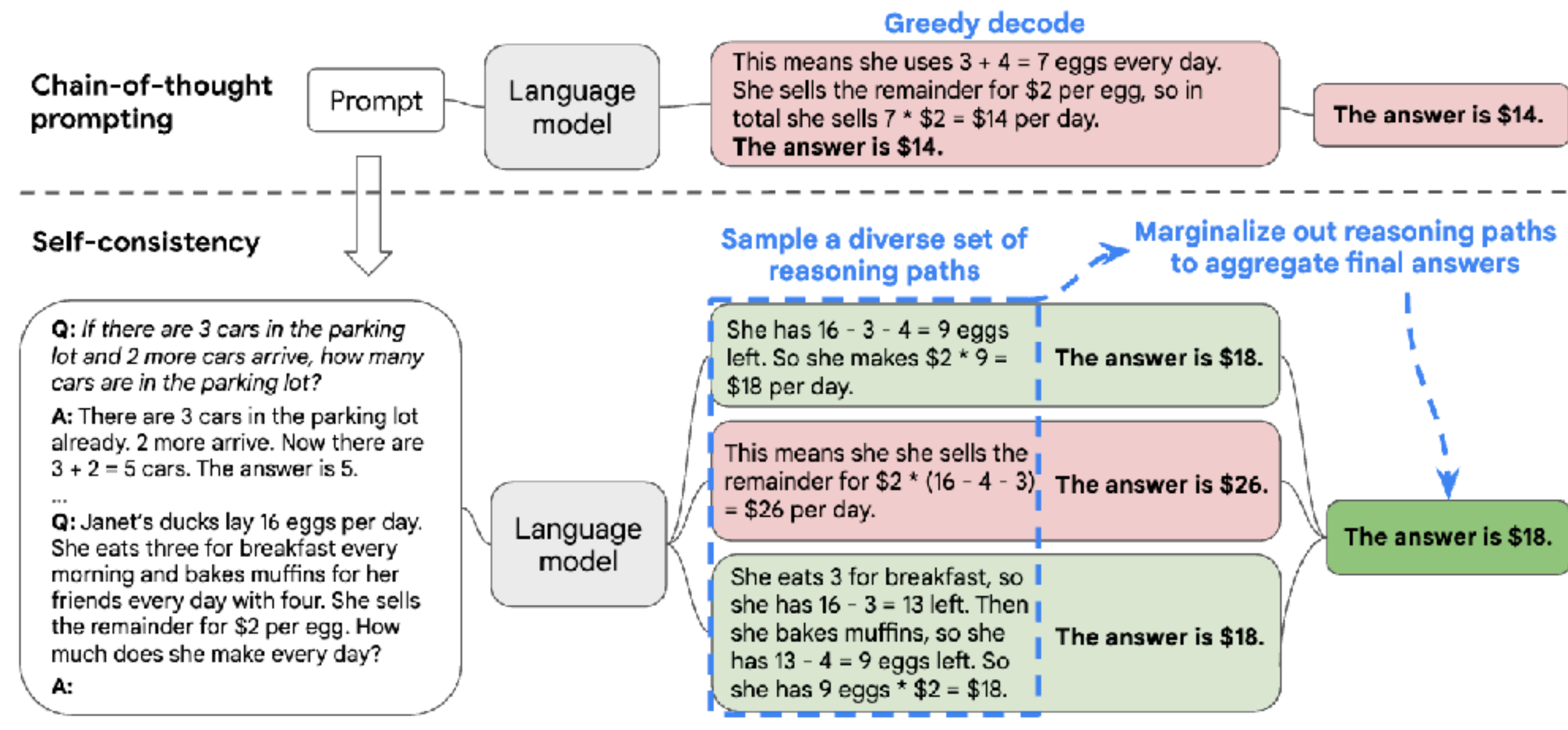
Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

### Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had  $23 - 20 = 3$ . They bought 6 more apples, so they have  $3 + 6 = 9$ . The answer is 9. ✅

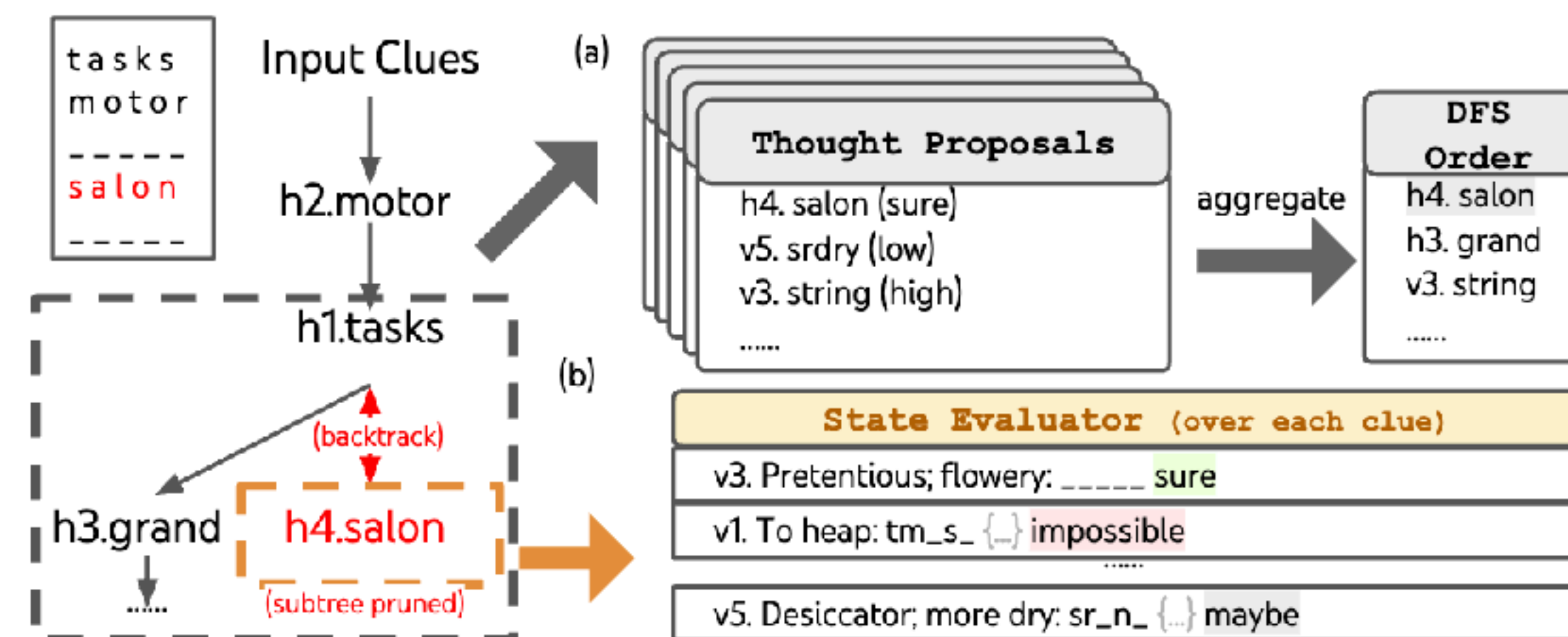
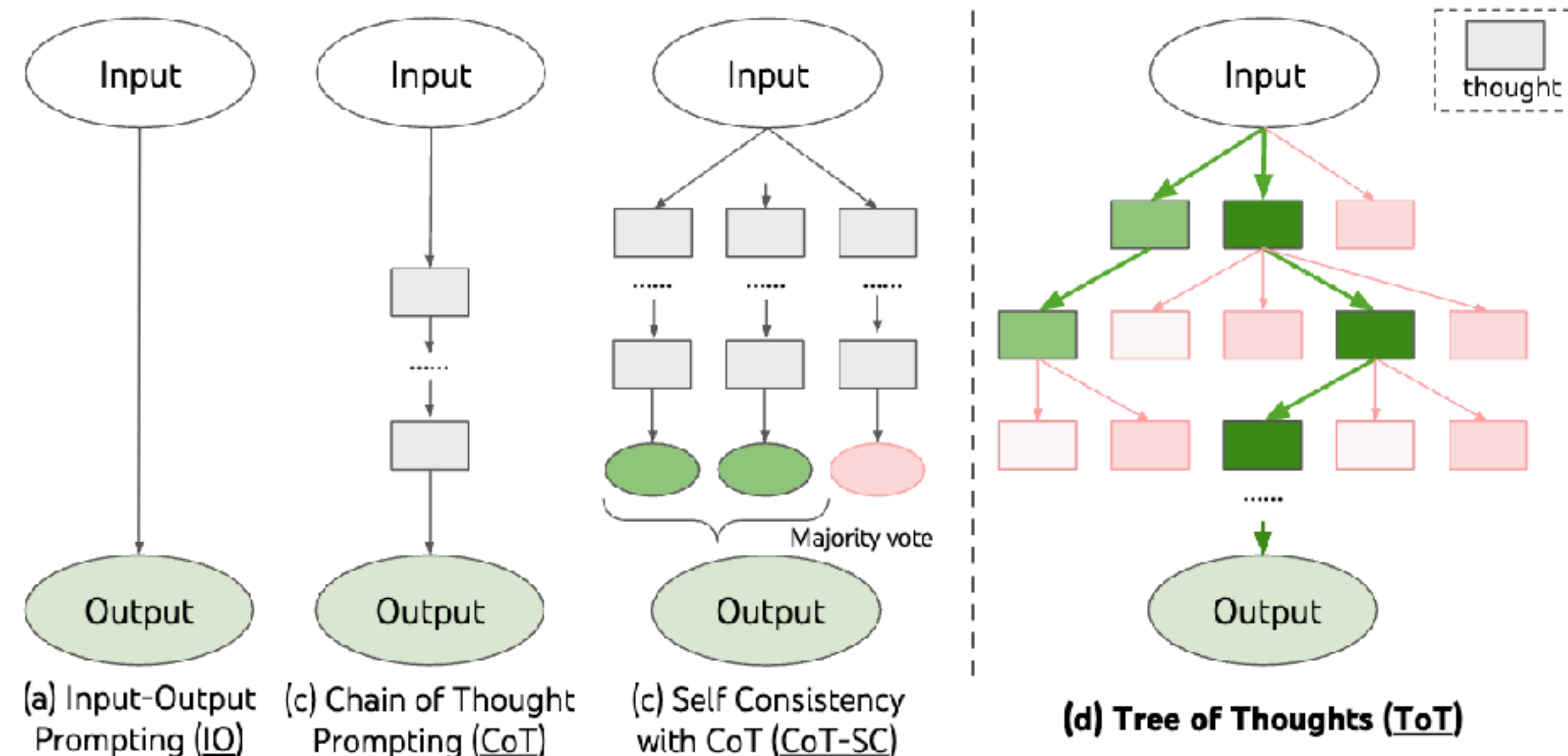
# Self-Consistency

- Let the model reason multiple times
  - Pick the most frequent answer
- Mathematically: Marginalize out reasoning to obtain most likely answer



# Tree of Thoughts

- Combine tree search with CoT
  - Requires a state-evaluator (i.e. reward/cost/scoring function or second LLM)



# ReACT

- Chain of thought for iterative actions / tool use

- Thought

- Action

- Observation (from external tool)

**(1) Hotspot QA**

**Question:** Aside from the Apple Remote, what other device can control the program Apple Remote was originally designed to interact with?

**(1a) Standard**

**Answer:** iPod ✗

**(1b) CoT (Reason Only)**

**Thought:** Let's think step by step. Apple Remote was originally designed to interact with Apple TV. Apple TV can be controlled by iPhone, iPad, and iPod Touch. So the answer is iPhone, iPad, and iPod Touch. ✗

**Answer:** iPhone, iPad, iPod Touch ✗

**(1c) Act-Only**

**Act 1:** Search[Apple Remote]  
**Obs 1:** The Apple Remote is a remote control ...

**Act 2:** Search[Front Row]  
**Obs 2:** Could not find [Front Row]. Similar: ...

**Act 3:** Search[Front Row (software)]  
**Obs 3:** Front Row is a discontinued media center software ...

**Act 4:** Finish[yes] ✗

**(1d) ReAct (Reason + Act)**

**Thought 1:** I need to search Apple Remote and find the program it was originally designed to interact with.  
**Act 1:** Search[Apple Remote]  
**Obs 1:** The Apple Remote is a remote control introduced in October 2005 by Apple ... originally designed to control the Front Row media center program ...

**Thought 2:** Apple Remote was originally designed to control the Front Row media center program. I need to search Front Row next and find what other device can control it.  
**Act 2:** Search[Front Row]  
**Obs 2:** Could not find [Front Row]. Similar: ['Front Row Seat to Earth', 'Front Row Motorsports', 'Front Row (software)', ...]

**Thought 3:** Front Row is not found. I need to search Front Row (software) .  
**Act 3:** Search[Front Row (software)]  
**Obs 3:** Front Row is a discontinued media center software ...

**Thought 4:** Front Row (software) is controlled by an Apple Remote or the keyboard function keys. So the answer is keyboard function keys.  
**Act 4:** Finish[keyboard function keys] ✓

**(2) AlfWorld**

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.  
**Your task is to:** Put some pepper shaker on a drawer.

**(2a) Act-Only**

**Act 1:** Go to drawer 1  
**Obs 1:** The drawer 1 is closed.

**Act 2:** Open drawer 1  
**Obs 2:** You open the drawer 1. The drawer 1 is open. In it, you see a dish sponge 2, and a spoon 1.

**Act 3:** Go to sinkbasin 1  
**Obs 3:** On the sinkbasin 1, you see a dish sponge 3, a spatula 1, and a spoon 2.

**Act 4:** Take peppershaker 1 from sinkbasin 1  
**Obs 4:** Nothing happens.

**Act 5:** Take peppershaker 1 from sinkbasin 1  
**Obs 5:** Nothing happens. ✗

**(2b) ReAct (Reason + Act)**

**Act 1:** Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ...]  
**Act 2:** Go to cabinet 1  
**Obs 2:** On the cabinet 1, you see a vase 2.  
(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

**Act 6:** Go to countertop 3  
**Obs 6:** On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.

**Act 7:** Take pepper shaker 1 from countertop 3  
**Obs 7:** You pick up the pepper shaker 1 from the countertop 3.

**Act 8:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1. ]  
**Act 9:** Go to drawer 1  
**Obs 9:** Drawer 1 is closed.

**Act 10:** Open drawer 1  
**Obs 10:** You open Drawer 1 ...

**Act 11:** Put pepper shaker 1 in/on drawer 1  
**Obs 11:** You put pepper shaker 1 in/on the drawer 1. ✓

# Structured Dialogues

- Break down problem / tasks for LLM
  - Higher performance
  - Lots of human engineering / prompting

User:

Request

LLM:

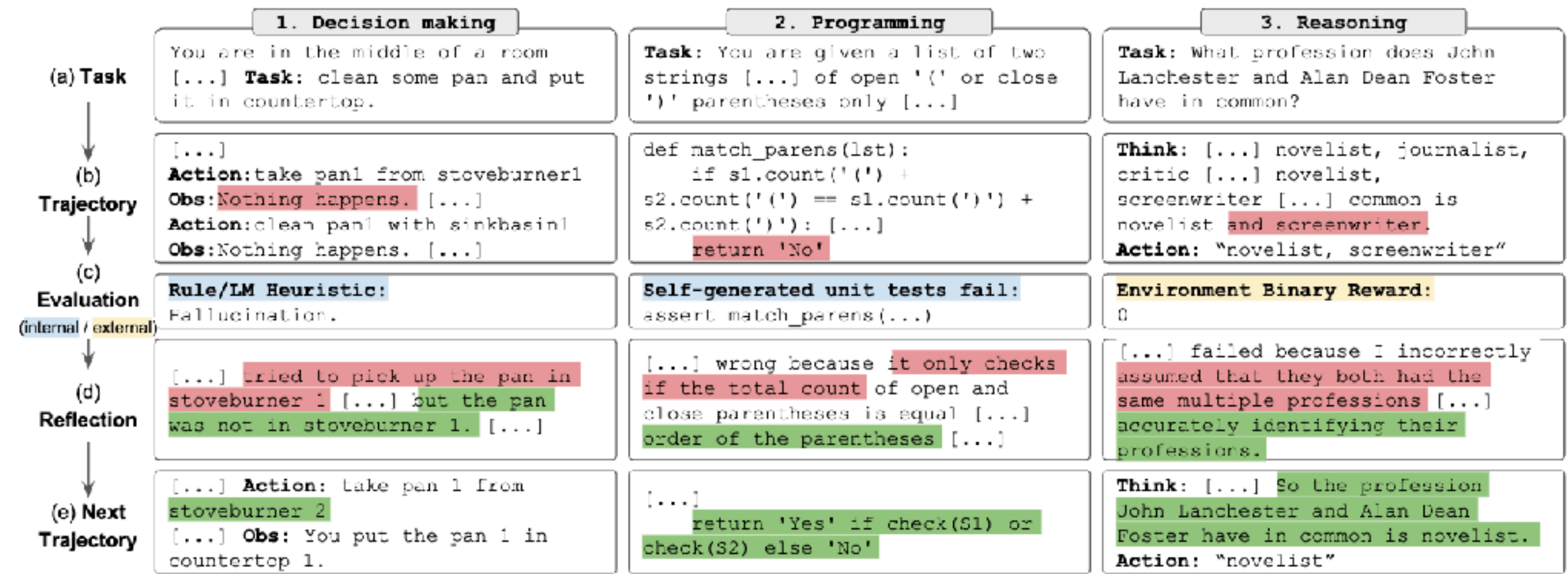
Thought

Answer



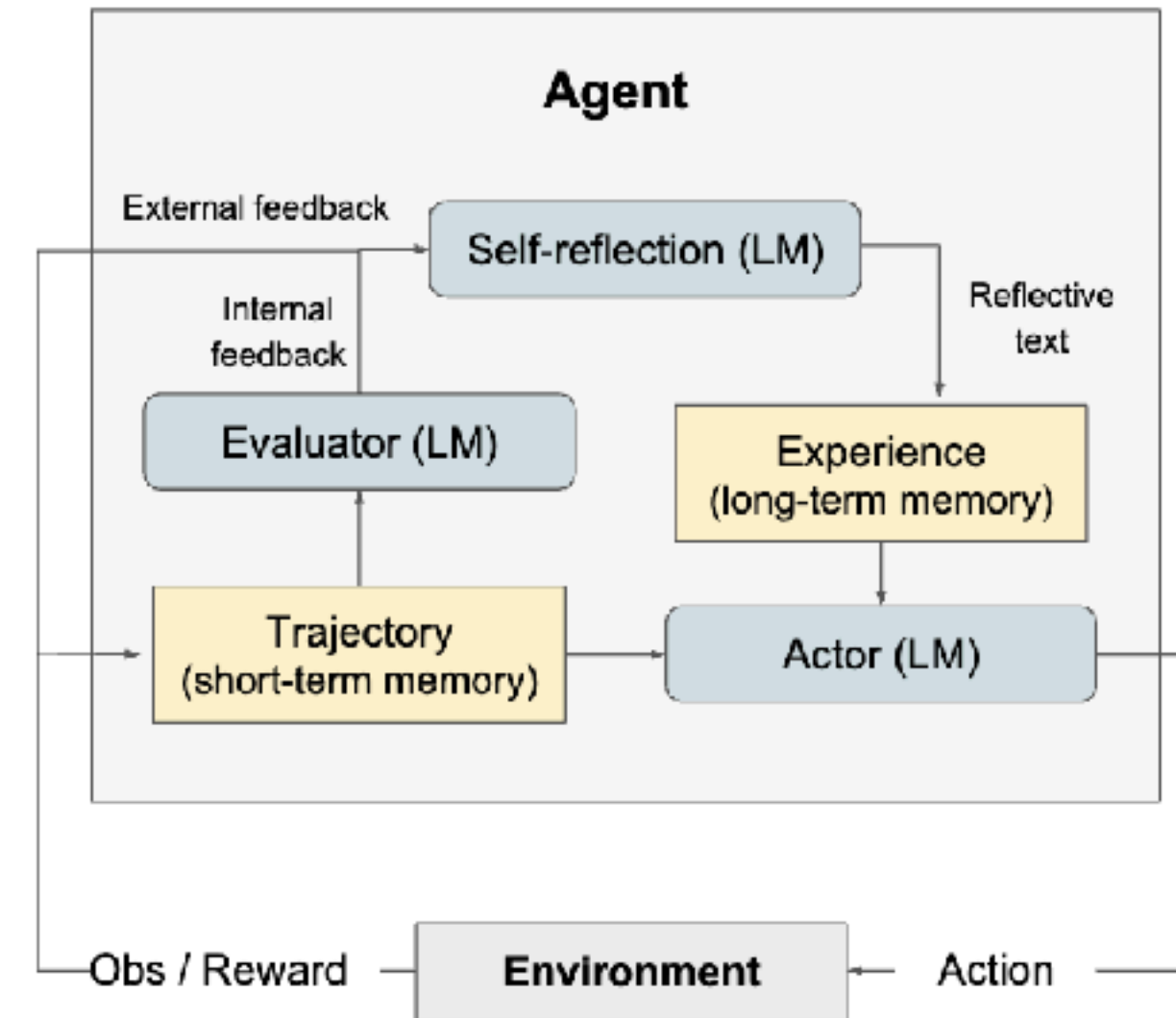
# Reflexion

- Chain of Thought / ReACT
- Obtain observation / result
- Reflect on outcome
- Repeat



# Reflexion

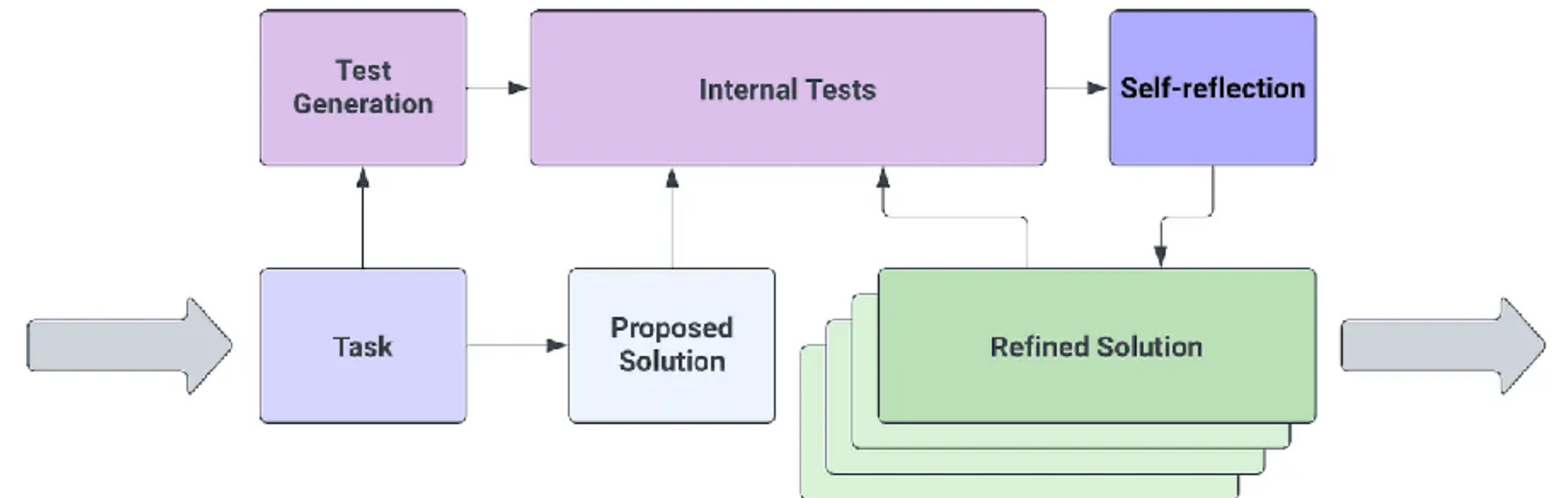
- Connections to reinforcement learning
  - More strictly planning
- Requires an evaluator (cost function)
  - External environment (i.e. simulator, code interpreter)
  - LLM generated tests
  - Trained LLM verifier [1]



## Algorithm 1 Reinforcement via self-reflection

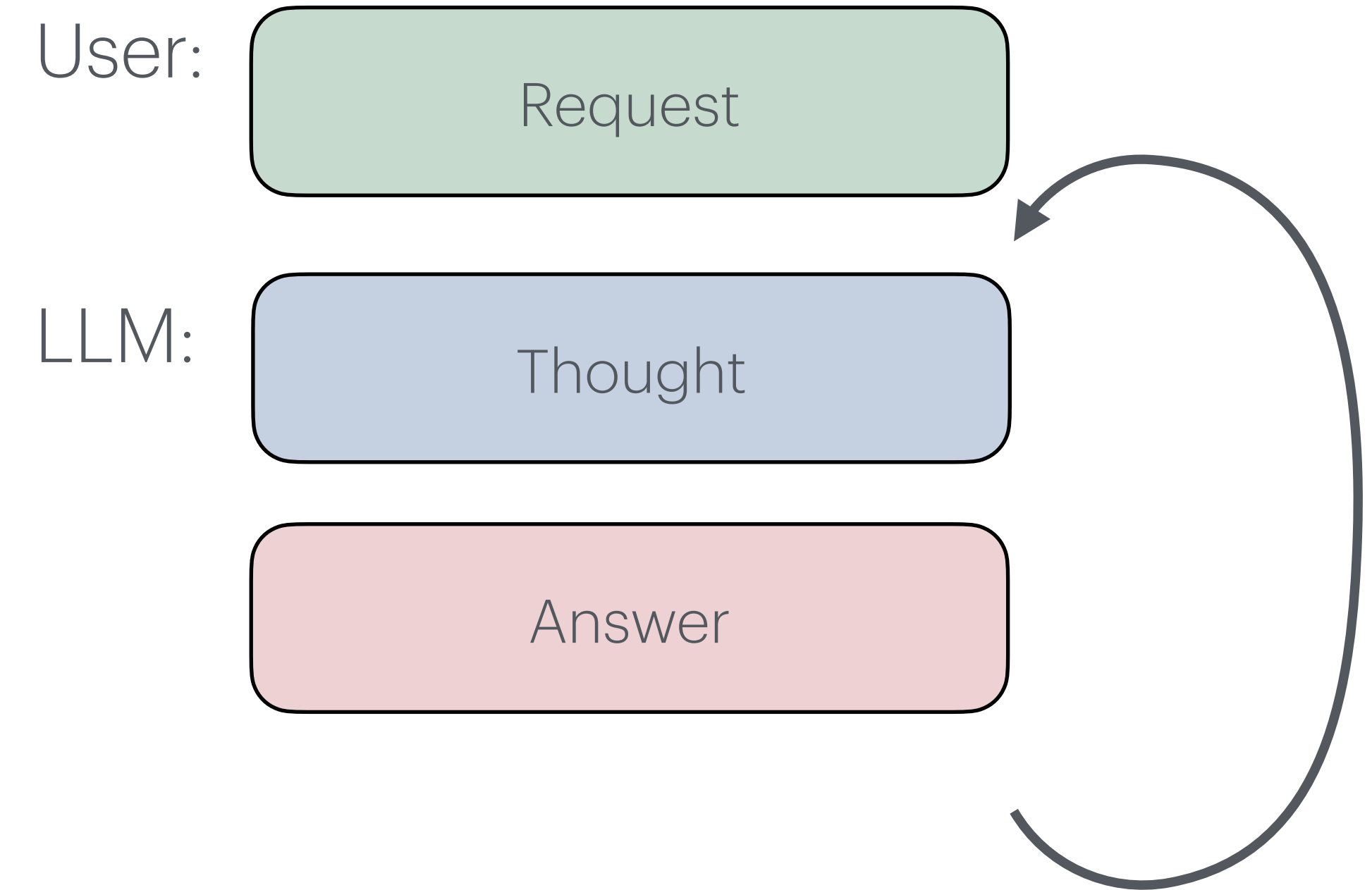
```

Initialize Actor, Evaluator, Self-Reflection:
 $M_a, M_e, M_{sr}$ 
Initialize policy  $\pi_\theta(a_i|s_i), \theta = \{M_a, mem\}$ 
Generate initial trajectory using  $\pi_\theta$ 
Evaluate  $\tau_0$  using  $M_e$ 
Generate initial self-reflection  $sr_0$  using  $M_{sr}$ 
Set  $mem \leftarrow [sr_0]$ 
Set  $t = 0$ 
while  $M_e$  not pass or  $t < \text{max trials}$  do
    Generate  $\tau_t = [a_0, o_0, \dots, a_i, o_i]$  using  $\pi_\theta$ 
    Evaluate  $\tau_t$  using  $M_e$ 
    Generate self-reflection  $sr_t$  using  $M_{sr}$ 
    Append  $sr_t$  to  $mem$ 
    Increment  $t$ 
end while
return
    
```



# Reflexion

- Break down problem / tasks for LLM
  - Higher performance
  - Lots of human engineering / prompting



# References

- [1] Physics of Language Models: Part 3.3, Knowledge Capacity Scaling Laws, Allen-Zhu 2024
- [2] Vision Transformers Need Registers, Darcet etal 2023
- [3] Massive Activations in Large Language Models, Sun etal 2024
- [4] Language Models are Few-Shot Learners, Brown etal 2020
- [5] Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, Wei etal 2022
- [6] Self-Consistency Improves Chain of Thought Reasoning in Language Models, Wang etal 2022
- [7] Tree of Thoughts: Deliberate Problem Solving with Large Language Models, Yao etal 2023
- [8] ReAct: Synergizing Reasoning and Acting in Language Models, Yao etal 2022
- [9] Reflexion: Language Agents with Verbal Reinforcement Learning, Shin etal 2023
- [10] Generative Verifiers: Reward Modeling as Next-Token Prediction, Zhang etal 2024