Lecture 13:

LLM-Based Problem Solving Agents

Visual Computing Systems
Stanford CS348K, Spring 2025

General design strategies for LLM-based agents

A simple problem solving activity

KA

Consider a multi-core processor that runs at 2 GHz and has 4 cores. Each core can perform up to one 8-wide SIMD vector instruction per clock What is the maximum throughput of the processor in units of floating point operations per second?

The answer is 64 GFLOPS (64 giga-floating point operations per second).

Another problem solving activity

KA

I want to eat avocado toast, what are steps for making it?

Avocado toast is a delicious and nutritious meal that's quick and easy to prepare. Here are the steps to make a basic avocado toast:

Ingredients:

- 1 ripe avocado
- 2 slices of your favorite bread (such as whole wheat, sourdough, or rye)
- Salt and pepper to taste
- Optional toppings (e.g., red pepper flakes, sliced tomatoes, poached or fried egg, feta cheese, herbs, etc.)

Instructions:

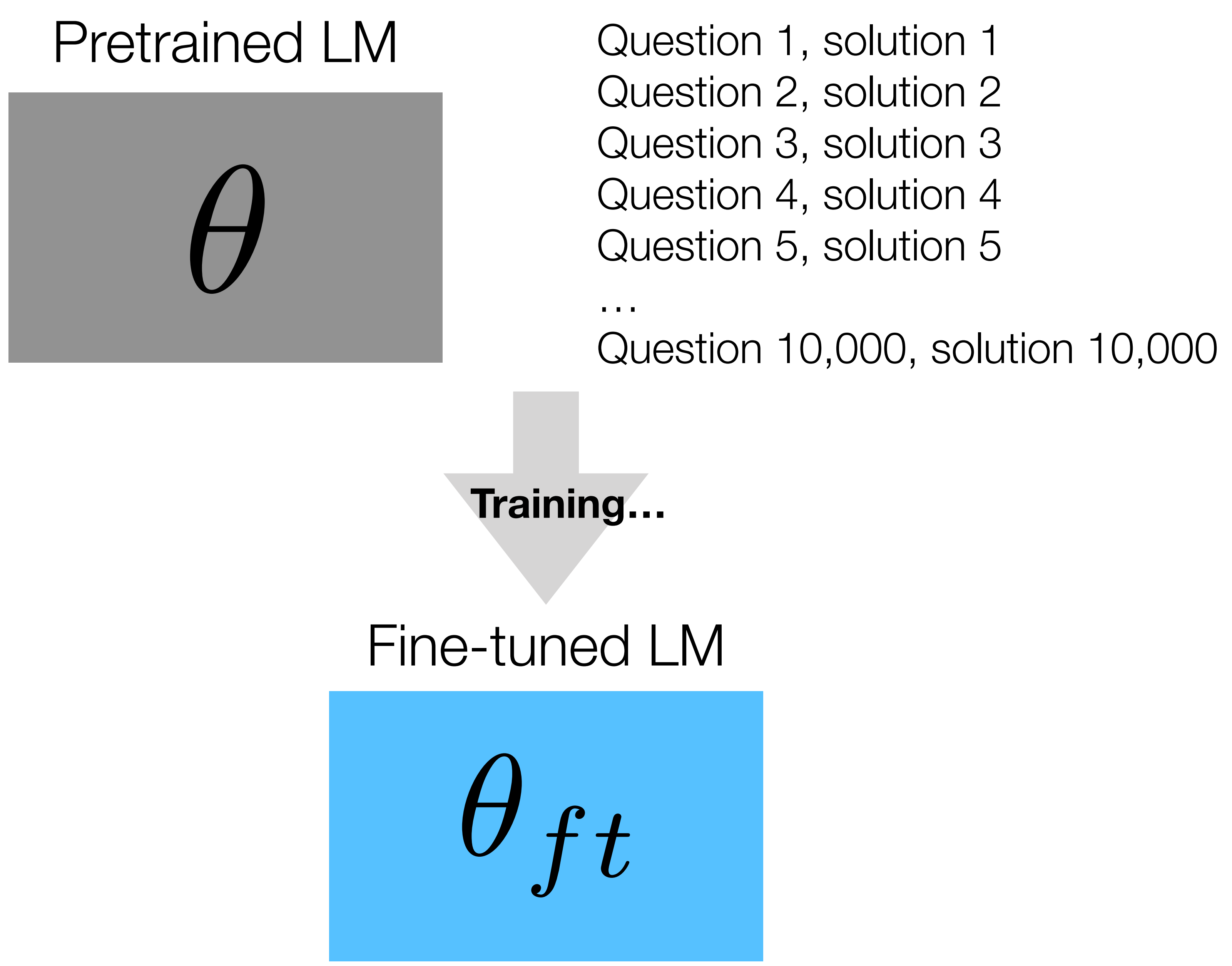
1. **Select a Ripe Avocado:** Choose a ripe avocado that yields slightly to gentle pressure when you squeeze it. If it's too firm, it won't spread easily on the toast.
2. **Slice the Avocado:** Cut the avocado in half by running a knife around the pit. Twist the two halves apart. Remove the pit by carefully striking it with a knife and twisting it out. Be very careful when doing this to avoid injury.
3. **Scoop the Flesh:** Use a spoon to scoop out the avocado flesh from each half. Place it in a bowl.
4. **Mash the Avocado:** Mash the avocado with a fork. You can leave it slightly chunky or make it smoother, depending on your preference.
5. **Season the Avocado:** Add a pinch of salt and some black pepper to the mashed avocado. You can also add a dash of lemon juice for some extra flavor and to prevent browning.
6. **Toast the Bread:** While you're preparing the avocado, toast the bread slices to your desired level of crispiness.
7. **Spread the Avocado:** Once the bread is toasted, spread the mashed avocado evenly over each slice.
8. **Add Toppings (Optional):** You can get creative with your toppings. Some popular options include sliced tomatoes, red pepper flakes, a poached or fried egg, feta cheese, herbs like cilantro or basil, or a drizzle of olive oil.
9. **Serve and Enjoy:** Your avocado toast is ready to be served! It makes a great breakfast, brunch, or even a quick and healthy snack.

Remember that avocado toast is highly customizable, so feel free to experiment with different toppings and seasonings to suit your taste. Enjoy your delicious and nutritious meal!

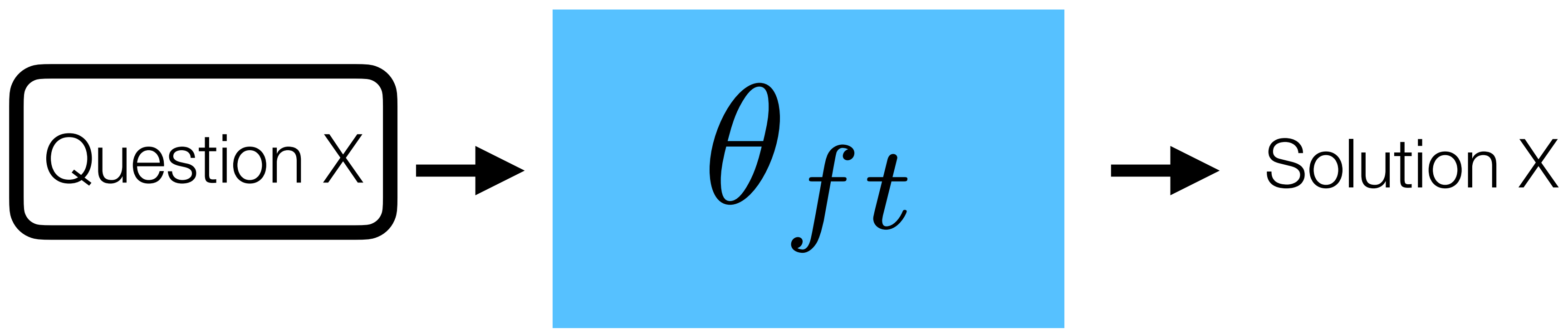
Fine-tuning vs. in-context learning

Fine-tuning:

Step 1: Fine tune existing model on in-domain training set:

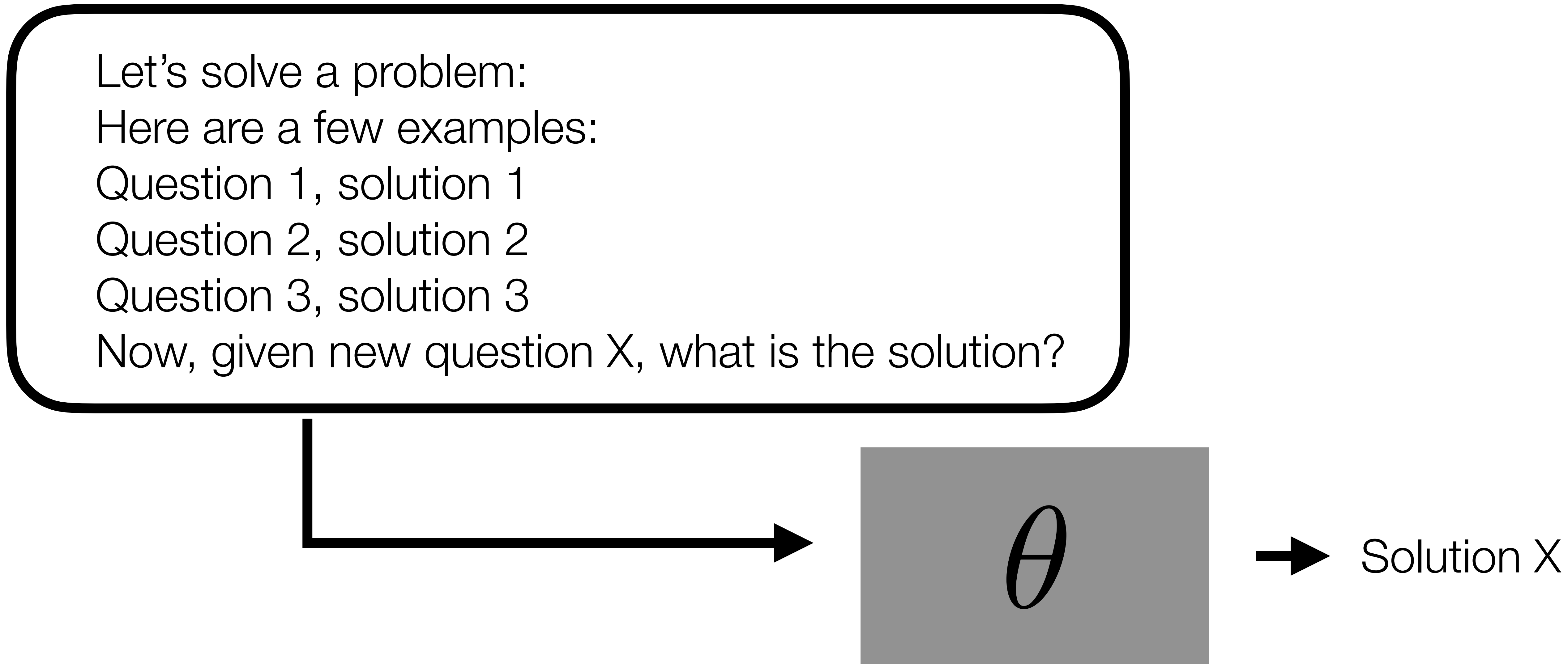


Step 2: Test time execution



In context learning:

Test time execution:



In-context learning example (failure)

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?

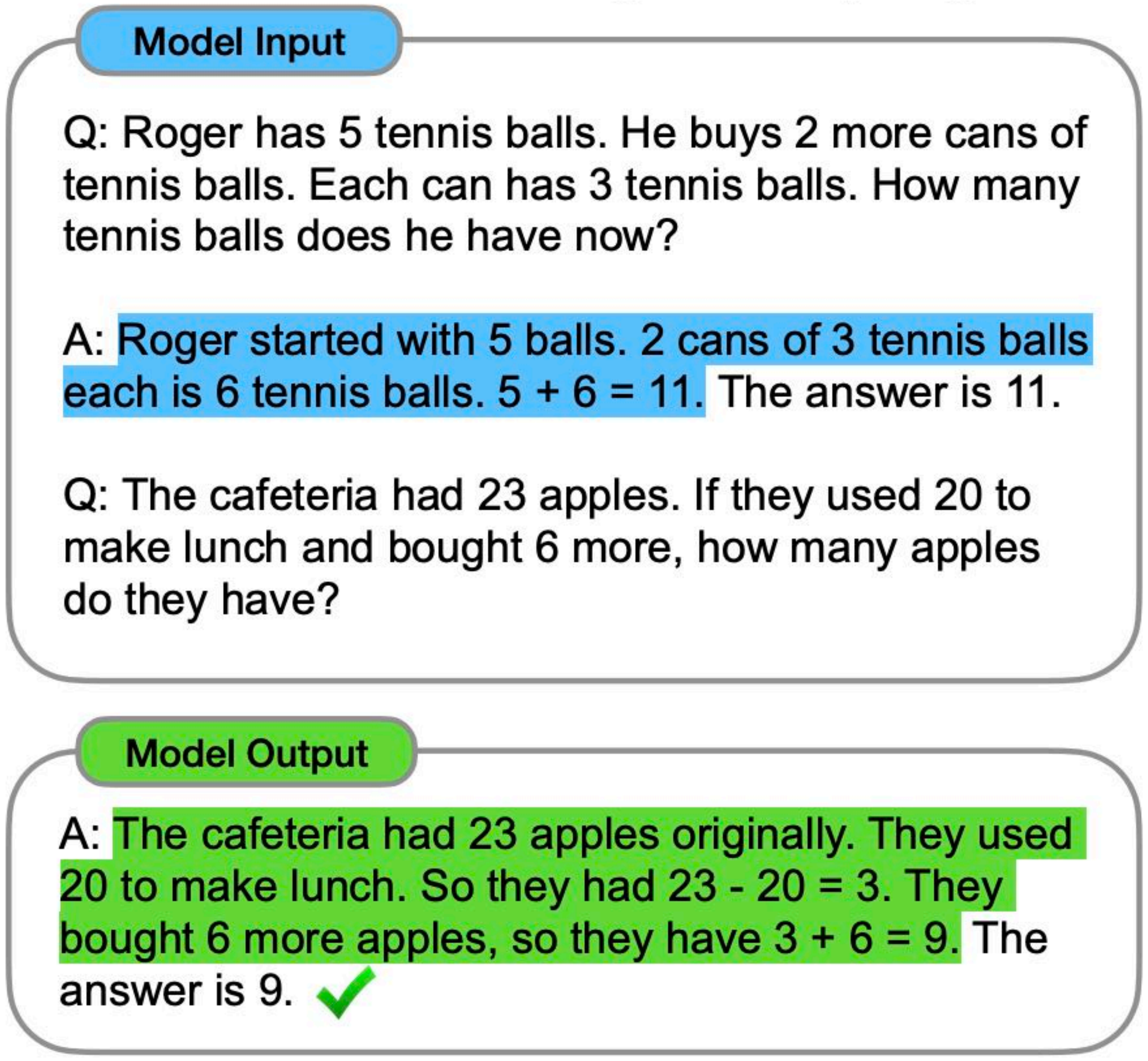
Model Output

A: The answer is 27. ✗

Chain of thought prompting

[Wei et al. 2022]

Main idea: modify the (in-context) prompted examples so they explicitly break down the solution into steps. By "coaching" the model to think step by step, the model is able to be more successful in its reasoning.



No fine-tuning step.

Step-by-step "training data" is harder to come by, and in-context reasoning needs less of it.

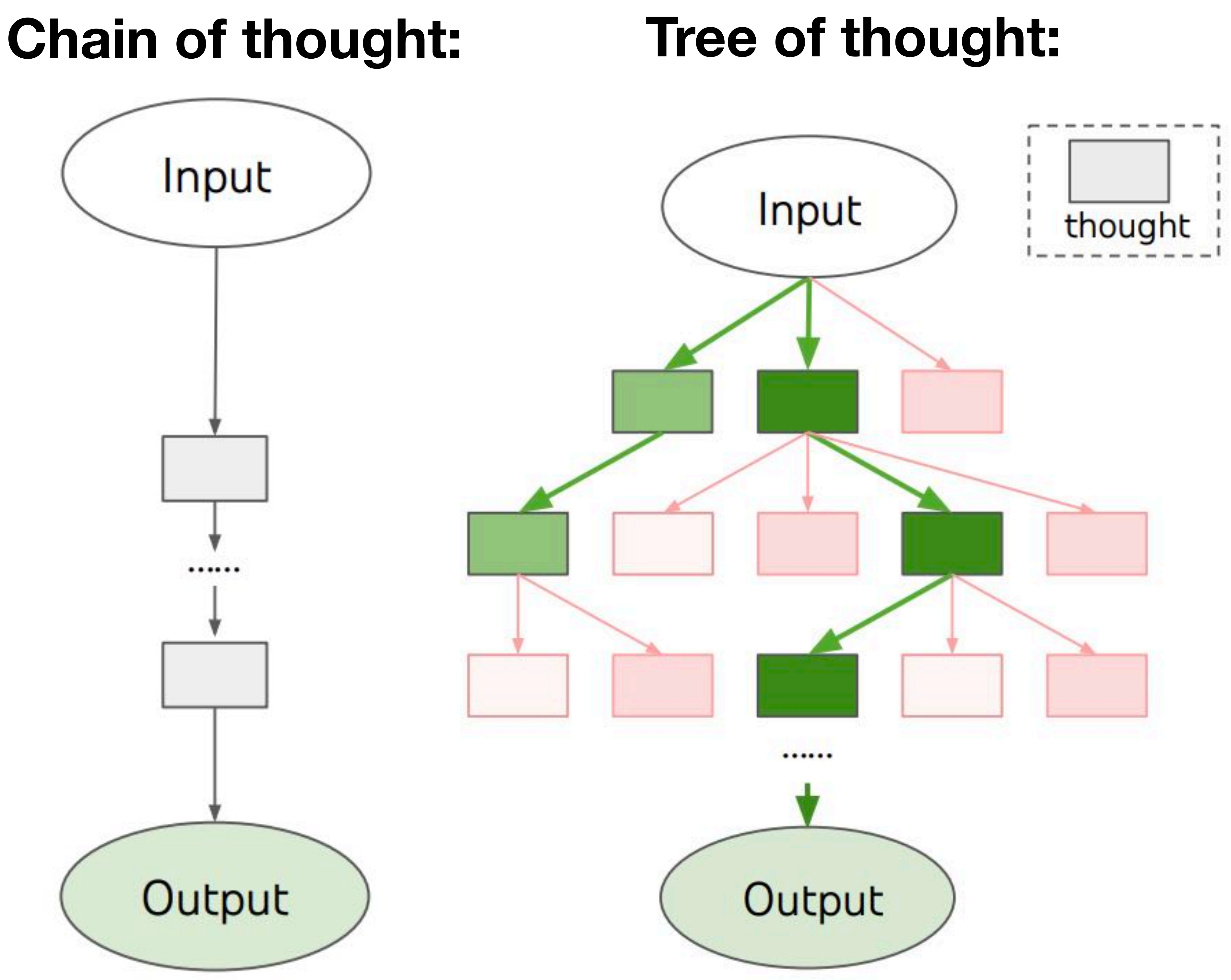
By running in steps, model can expend more computation to solve problem

More interpretable answer, since chain of reasoning is revealed.

Tree of thoughts

[Yao et al. 2023]

- Brings together traditional idea of backtracking search and chain-of-thought
- Generate answer step-by-step (like chain of thought)
- But generate multiple possibilities each step
- Choose most promising next step: use LM a judge ** to "score" the possibilities



Judge possibilities:

Given this question, which step do you think is best?

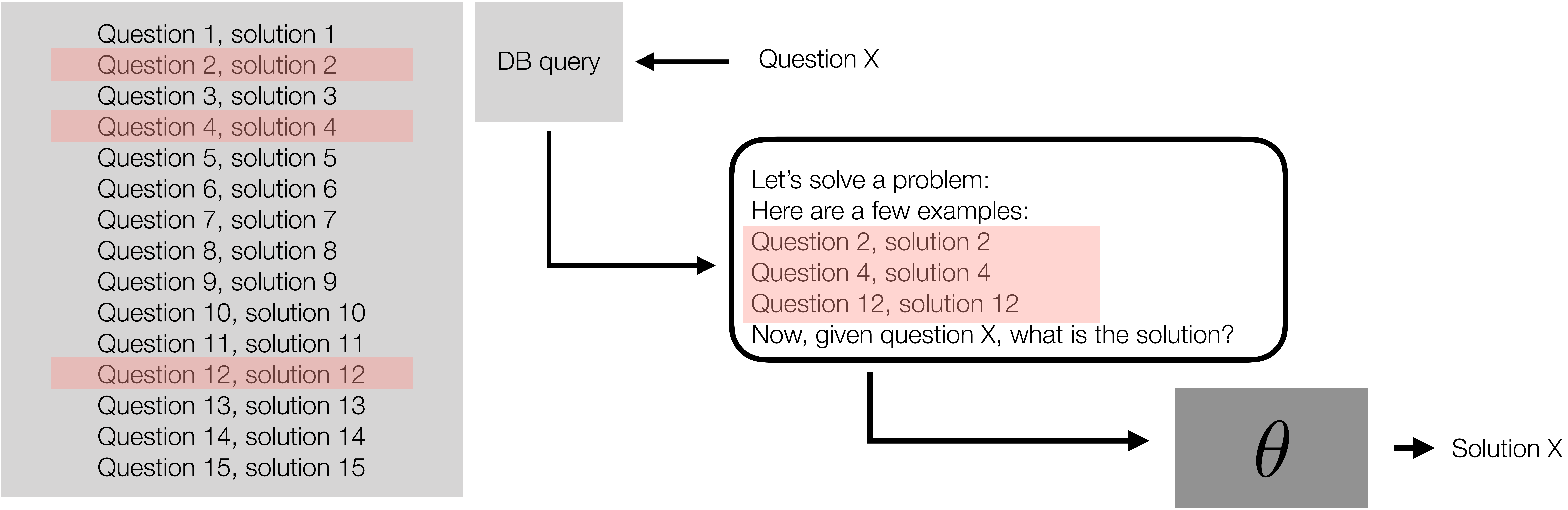
OR

Compute some independent score for each option, take best.

** The judge is a form of self reflection, see later in talk

Enhancement: dynamic lookup into "Memory"

- **Challenge: LMs have finite input context length**
- **Training dataset may be large, and may have many examples**
- **Idea: choose examples are the "most relevant" examples to provide the LM as context**



- **Typical query solution: embed question string, K-NN lookup in "vector database"**

Use of memory

- Lookup most relevant Q-A examples

- Examples:
  - Robot planning: If the task is making a plan to "cook an egg", then in-context examples should be as relevant as possible. i.e, if you have an example plan for "cook a potato" in the DB then use it.

  - Social decision making: If the task is deciding what to do next when you see Kayvon, DB lookup might be retrieve all items in history that record interactions with Kayvon

Self reflection

[Yao et al. 2022]

- If chain-of-thought was about forcing the model to “show your work”, then self reflection is about forcing the model to “tell me why” you took a particular step

- Model outputs two pieces of information:

1. A list of actions (steps in a plan)
    - Actions modify “the world” and solicit new observations
2. A list of thoughts
    - Thoughts do not modify the world, but update the state of the planner (by forcing it to emit more tokens)

Self-reflection example (explain your reasoning)

[Yao et al. 2022]

Example task:

(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.
```

LM agent emits actions only

(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 dishsponge 2, and a spoon 1.

Act 3: Go to sinkbasin 1
Obs 3: On the sinkbasin 1, you see a dishsponge 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.
```

✗

LM agent emits reasoning and actions

(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.
```

✓

A general template for a modern decision making agent

- **Here the agent uses an LLM to make high-level plans, and perform reasoning (self-reflection) and action selection each step.**
- **Note that prior to each LLM evaluation, relevant examples from a database of examples are chosen.**

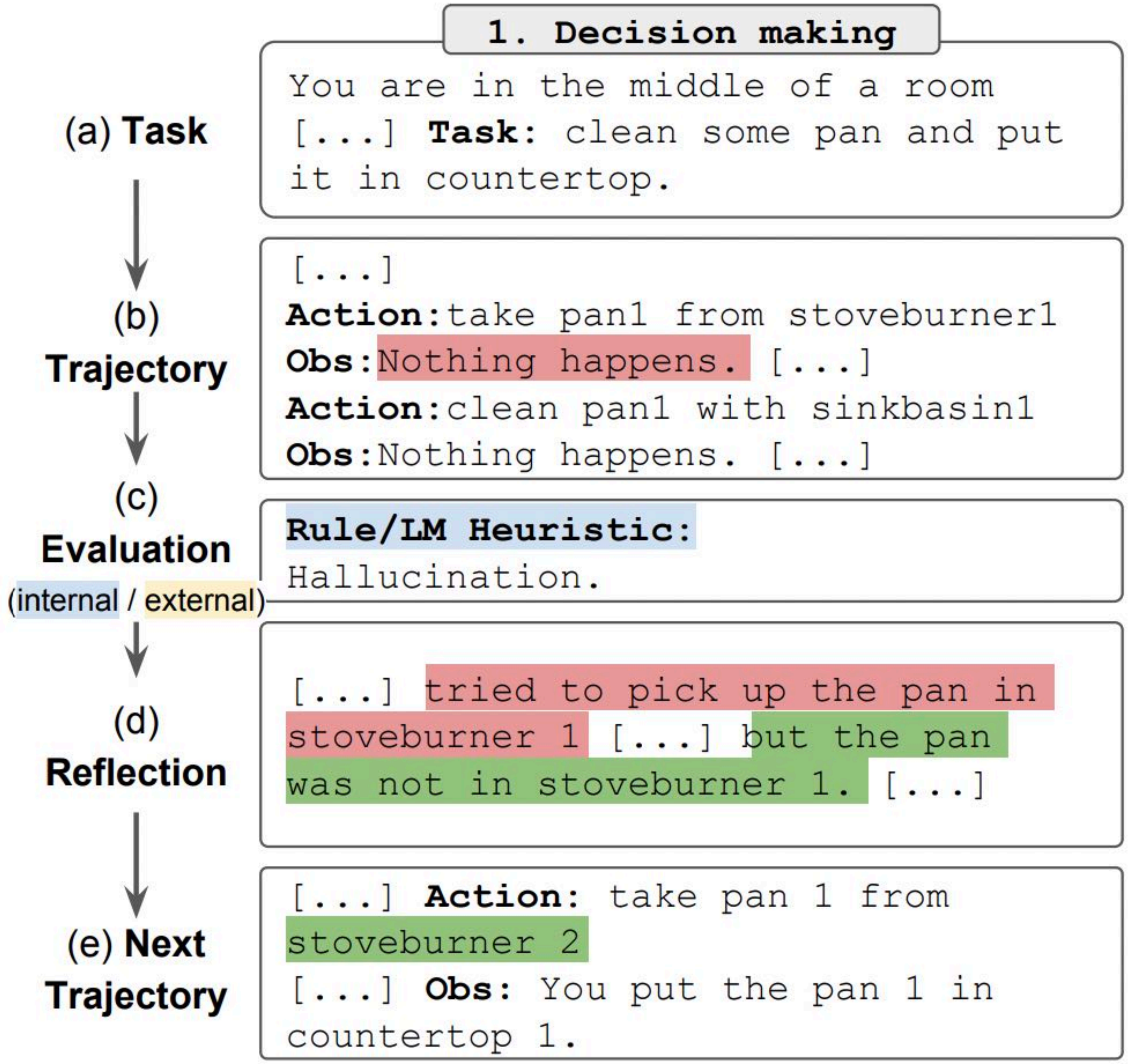
```
function agent(task_goal, db, env, actions):
    reasoning_examples = retrieve(db, task_goal)
    reasoning = LLM_do_planning_reasoning(task_goal, reasoning_examples);
    planning_examples = retrieve(db, task_goal, reasoning)
    plan = LLM_make_plan(task_goal, planning_examples)
    while (!env.done)
        obs = env.getobs()
        reasoning_examples = retrieve(db, [task_goal, plan, obs])
        reasoning = LLM_do_action_reasoning(task_goal, plan, obs, reasoning_examples);
        action_examples = retrieve(db, [task_goal, plan, obs, reasoning])
        action = LLM_choose_action(task_goal, plan, obs, action_examples)
        env.step(action)
```

[Credit: Sarukkai et al.]

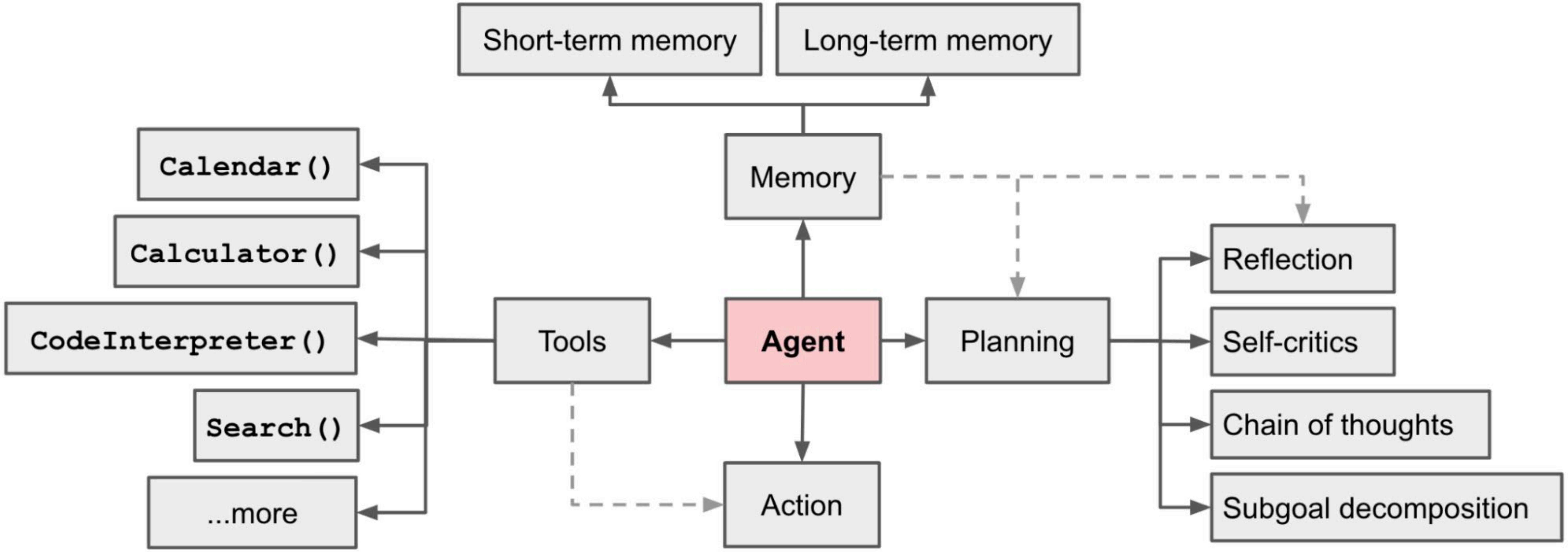
Another form of reflection (reflecting on how you did)

[Shin et al. 2023]

- Given world state W and task description T (e.g., "get a glass of milk"), the agent executes a plan in the simulated world ("rolls out plan")
- This generates a log of: (action_i, observation_i) pairs, and a "reward"
- Next, the model is asked to produce a critique of what went wrong (e.g, what went wrong and potentially even where in the log the error happened)
- Now retry plan generation with: (W, T, log, reward, critique)
- Note: this formulation requires environment where the agent has the ability to try a task multiple times



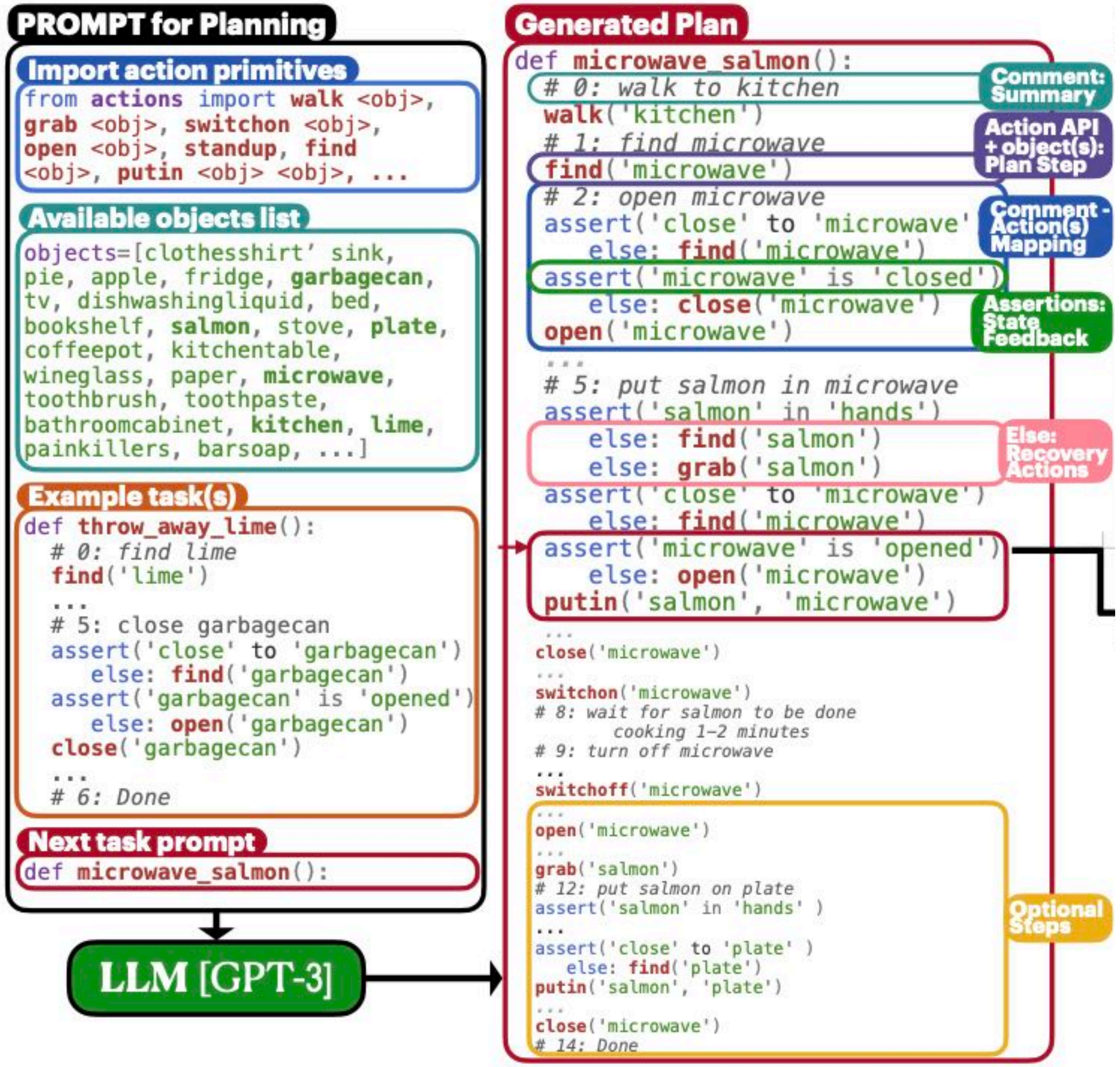
General LLM-agent architecture



[Image credit: Lilian Weng]

Recall idea from earlier lectures: representing plans as programs

"Microwave salmon":



Example: ProgPrompt [Sing et al. ICRA 23]

Key ideas:

- LM as a code generator: instead of invoking an LLM each step, LLM emits a valid python program with access to subroutines (action space defined by subroutines) that determines what action to take each step.

- Can perform reflection on the results of the programs execution.

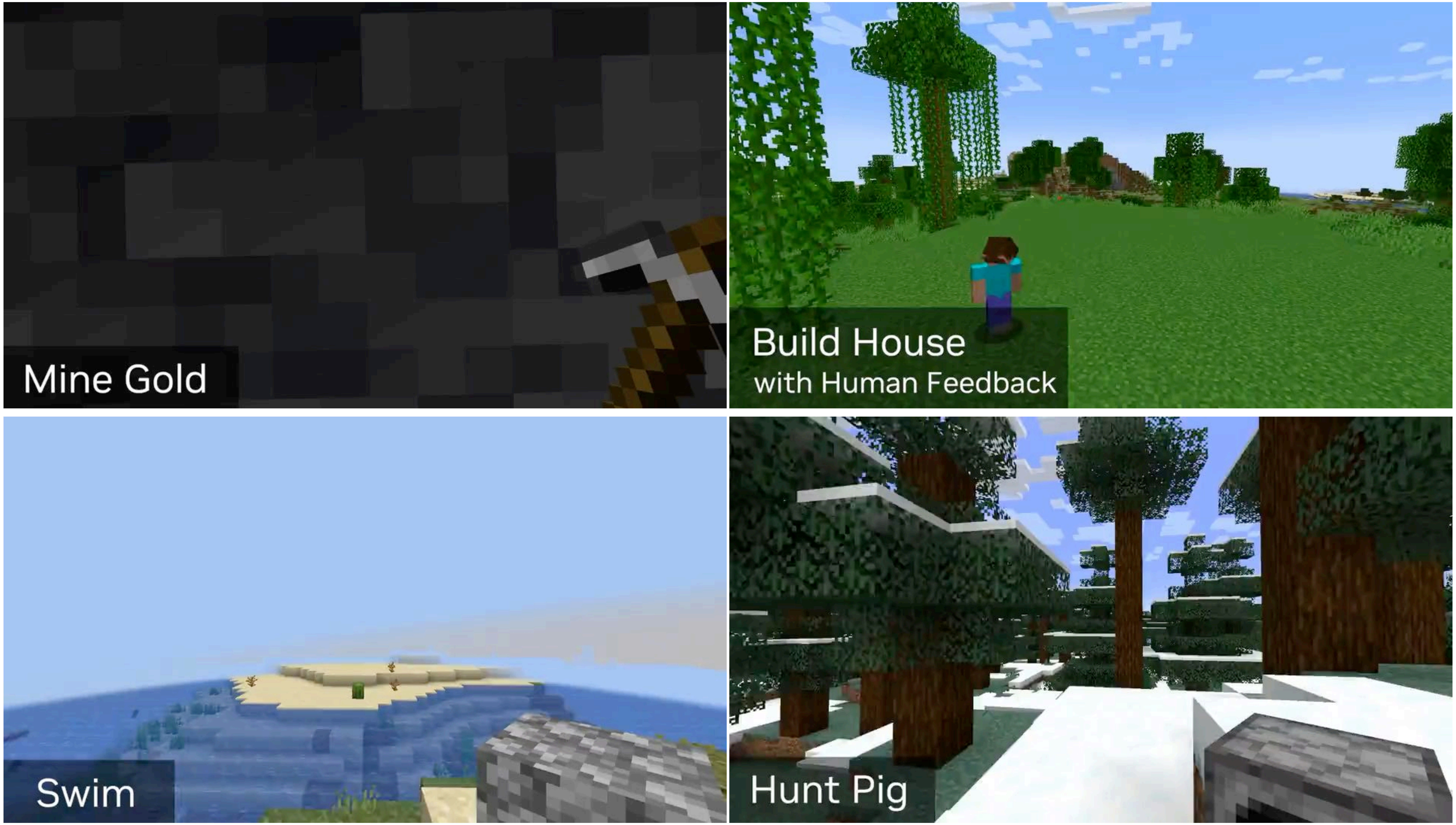
Voyager

- Task: do things in Minecraft: make new things, fight Zombies, etc.
- Major concepts:
  - Use LM to generate programs that execute the agent
  - Use LM to repair bad programs: (reflection)
    - Run programs to see if they work
    - Get compiler feedback and game state feedback
    - Use LM to repair program given feedback

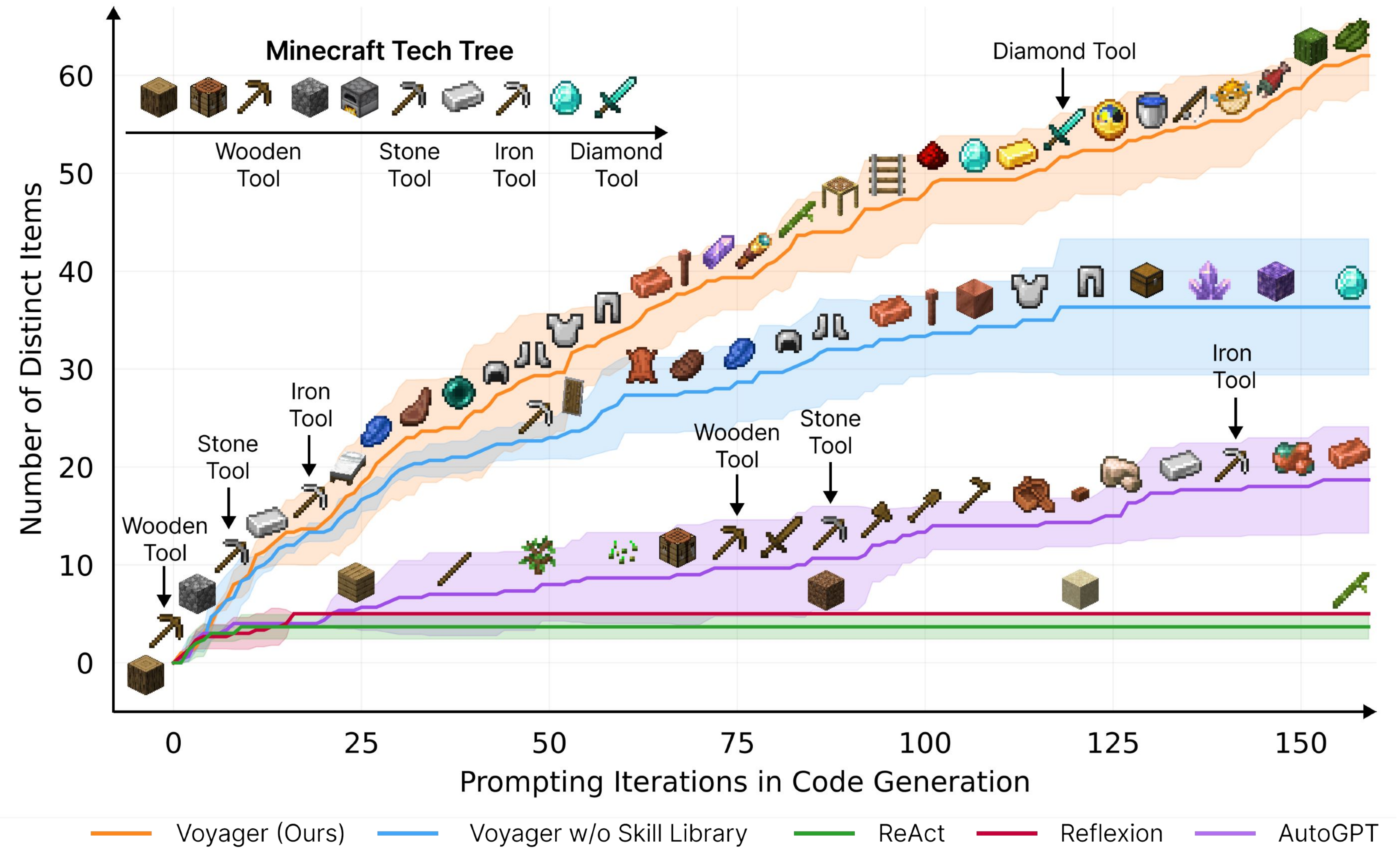
  - Use LM to propose new tasks (AI makes the curriculum)

  - Key idea: hierarchical skill library: new tasks get asked to skill library for use as future subroutines
    - As agent develops: increasing granularity and sophistication of the actions in the agent's action space

Example tasks



Voyager achieves SOTA performance



Compared both to Deep RL and other LLM Agents

Voyager solves simple problems first