

Lecture 12:

Learned World Models vs. Simulators Based on Human-Authored Models

**Visual Computing Systems
Stanford CS348K, Spring 2025**

The great CS348K debate

Team World Models



Team Traditional Simulator



The scenario

- Your job is to lead a team for the next 10 years to develop the ultimate virtual world simulator for training AI agents on a immensely wide variety of tasks
- Those AI agents might be trained with RL, be LLM-based agents, or use technology that has not been developed yet
- Consider technologies we have talked about in class so far... what technology bets will you make?

Things to consider

- Consider types of tasks you may want to support
- Consider the fidelity of simulation you want to achieve?
 - What does “high fidelity” even mean?
- Consider various costs
 - Costs to create (or acquire via capture) world content
 - Costs to execute simulations and training
 - Costs to develop the systems or debug agents that you train

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.

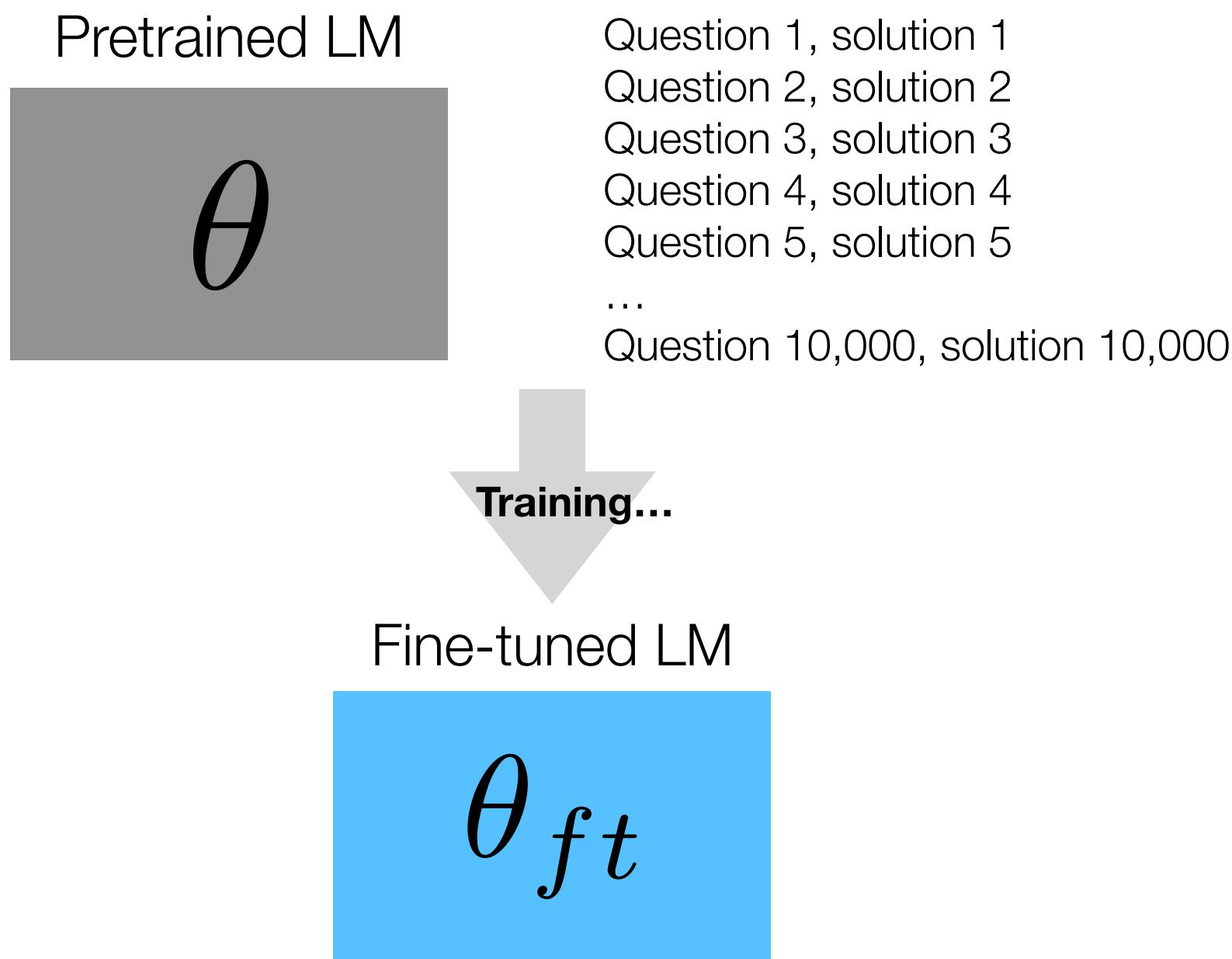
5. **Season the Avocado:** Add a pinch of salt and some black pepper to the mashed avocado. You can also add a dash of lemon juice for some extra flavor and to prevent browning.
6. **Toast the Bread:** While you're preparing the avocado, toast the bread slices to your desired level of crispiness.
7. **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!

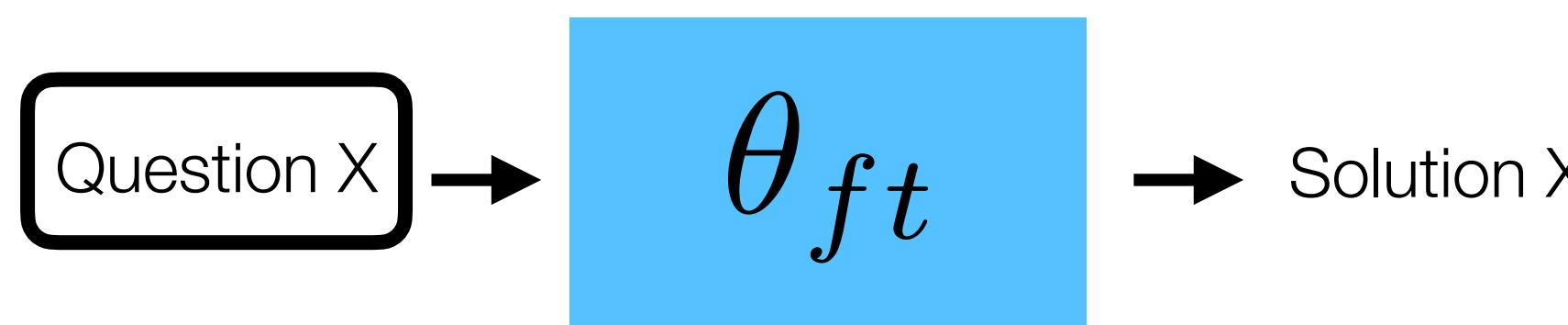
Fine-tuning vs. in-context learning

Fine-tuning:

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



Step 2: Test time execution



In context learning:

Test time:

Let's solve a problem:
Here are a few examples:
Question 1, solution 1
Question 2, solution 2
Question 3, solution 3
Now, given new question X, what is the solution?



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.

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. ✓

No fine-tuning step.

Step-by-step “training data” harder to come by, 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.

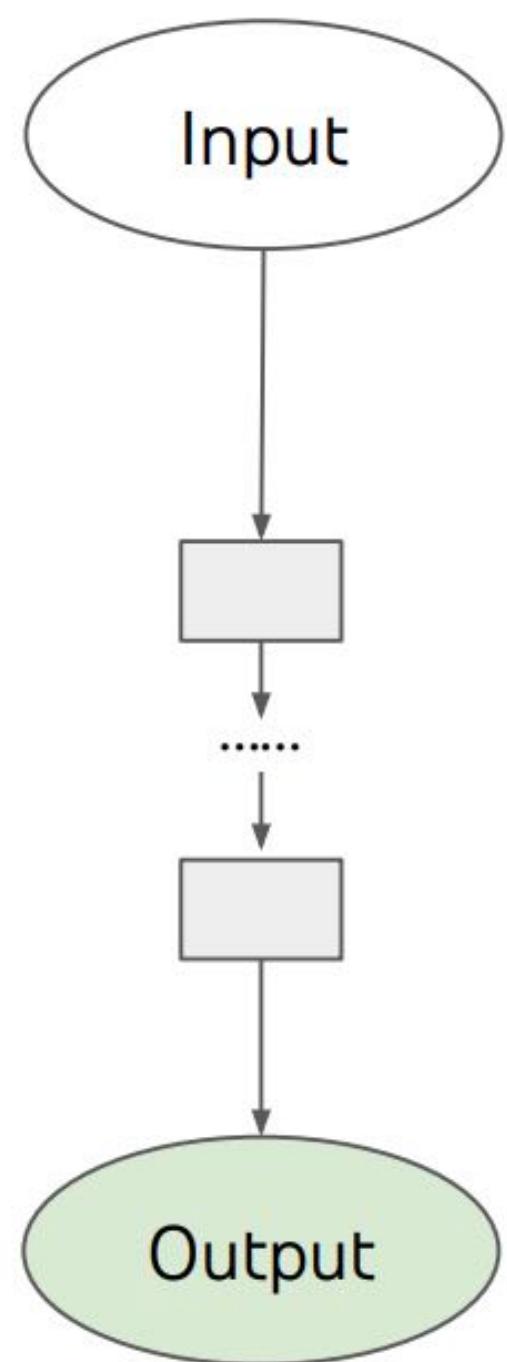
Empirically: chain of thought needs larger language models to work.

Tree of thoughts

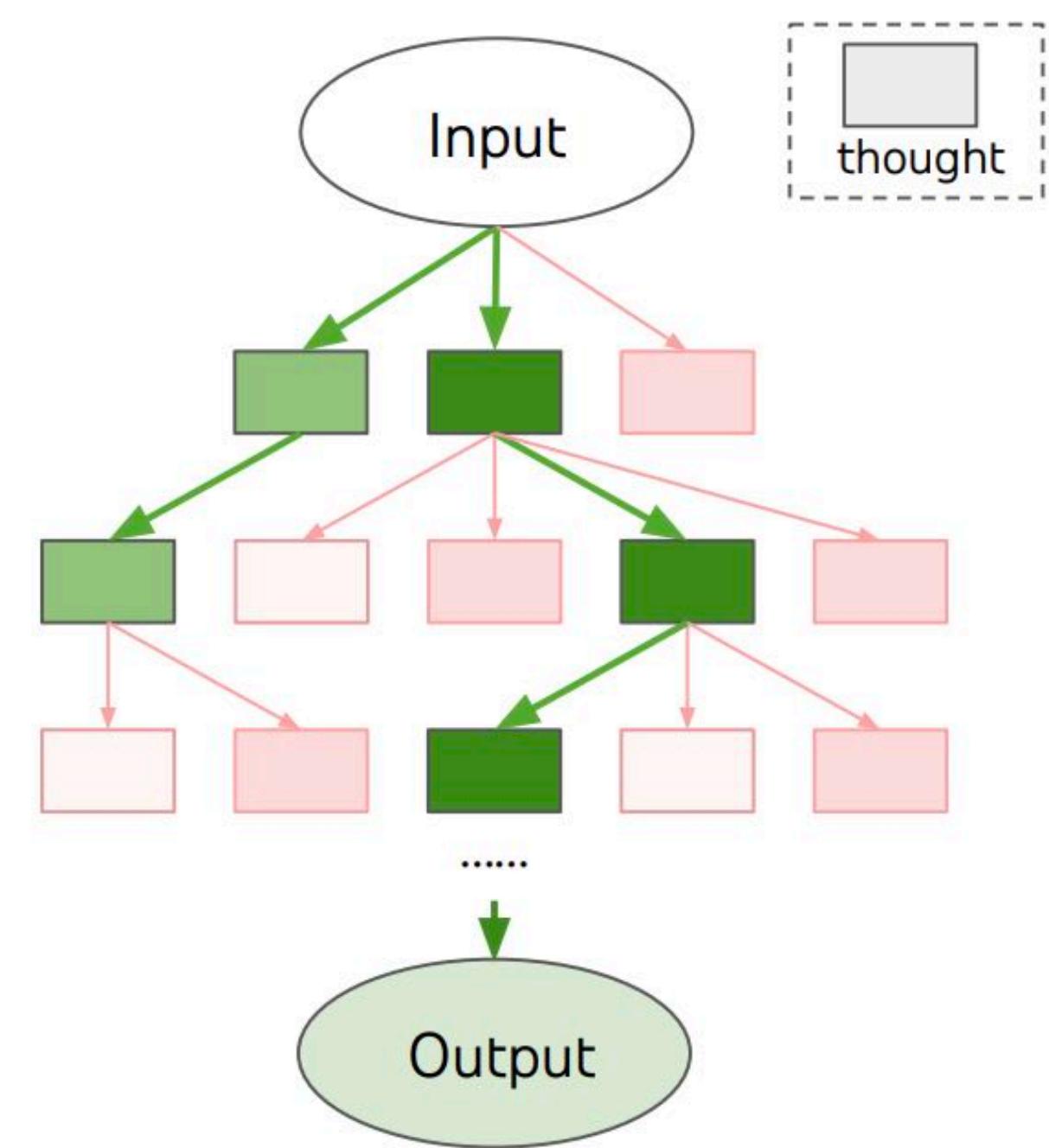
[Yao et al. 2023]

- 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

Chain of thought:



Tree of thought:



Judge possibilities:

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

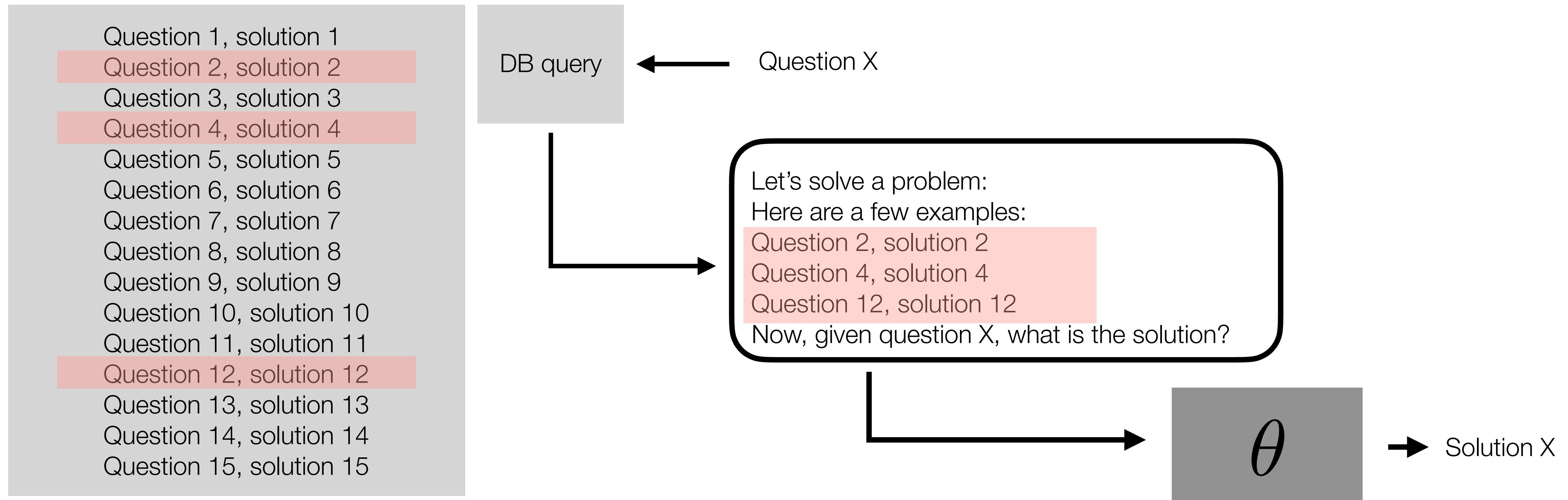
OR

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 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 Maneesh, DB lookup might be retrieve all items in history that record interactions with Maneesh

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 examples

[Yao et al. 2022]

Example 1: Robot acting in virtual environment

(Observations are symbolic representation of world state)

(b) ReAct (Human Thought edit)

.....

Obs 16: You open the drawer 4. The drawer 4 is open. In it, you see a keychain 3, and a watch 2.

Act 17: ~~Think [Now I find the first keychain (3). Next, I need to take it. I can find the second keychain (2) later in drawer 4.]~~

Act 18: take keychain 3 from drawer 4

Obs 18: You pick up the keychain 3 from the drawer 4.

.....

Act 22: put keychain 3 in/on safe 1

Obs 22: You put the keychain 3 in/on the safe 1.

Act 23: ~~Think [Now I put the first keychain in safe. Next, I need to find the second keychain, which is more likely to appear in dresser (1), garbagecan (1), safe (1), sidetable (1), sofa (1), shelf (1-12) ...]~~

Act 24: go to dresser 1

Obs 24: On the dresser 1, you see a box 1, a creditcard 1, a keychain 2, a keychain 1, a laptop 1...

Act 25: ~~Think [Now I find the second keychain (2). Next, I need to take it.]~~

Act 26: take keychain 2 from dresser 1

Obs 26: You pick up the keychain 2 from the dresser 1.

Act 27: ~~Think [Now I take the second keychain (2). Next, I need to put it in/on safe 1.]~~

Act 28: go to safe 1

Obs 28: The safe 1 is open. In it, you see a keychain 3, a statue 2, and a watch 3.

Act 29: put keychain 2 in/on safe 1

Obs 29: You put the keychain 2 in/on the safe 1.



Example 2: digital assistant browsing the web

(Observations are web page contents)

(1d) ReAct (Reason + Act)

Thought 1: I need to search Cirque du Soleil show Mystere, find the hotel it is in, then find the number of rooms in the hotel.

Act 1: **Search**[Cirque du Soleil show Mystere]

Obs 1: Could not find Cirque du Soleil show Mystere. Similar: ['Cirque du Soleil' ...]

Thought 2: To find the hotel, I can search Mystere (Cirque du Soleil).

Act 2: **Search**[Mystere (Cirque du Soleil)]

Obs 2: Mystère (my-steer) is ... Held in a custom theatre at the Treasure Island Hotel and Casino ...

Thought 3: Mystere is held in a custom theatre at the Treasure Island Hotel and Casino. So I need to search Treasure Island Hotel and Casino next and find the number of rooms in the hotel.

Act 3: **Search**[Treasure Island Hotel and Casino]

Obs 3: Treasure Island Hotel and Casino ... with 2,884 rooms and 220 suites ...

Thought 4: Treasure Island Hotel and Casino has 2,884 rooms and 220 suites. So the answer is 3,104.

Act 4: **Finish**[3,104]

Up-to-date

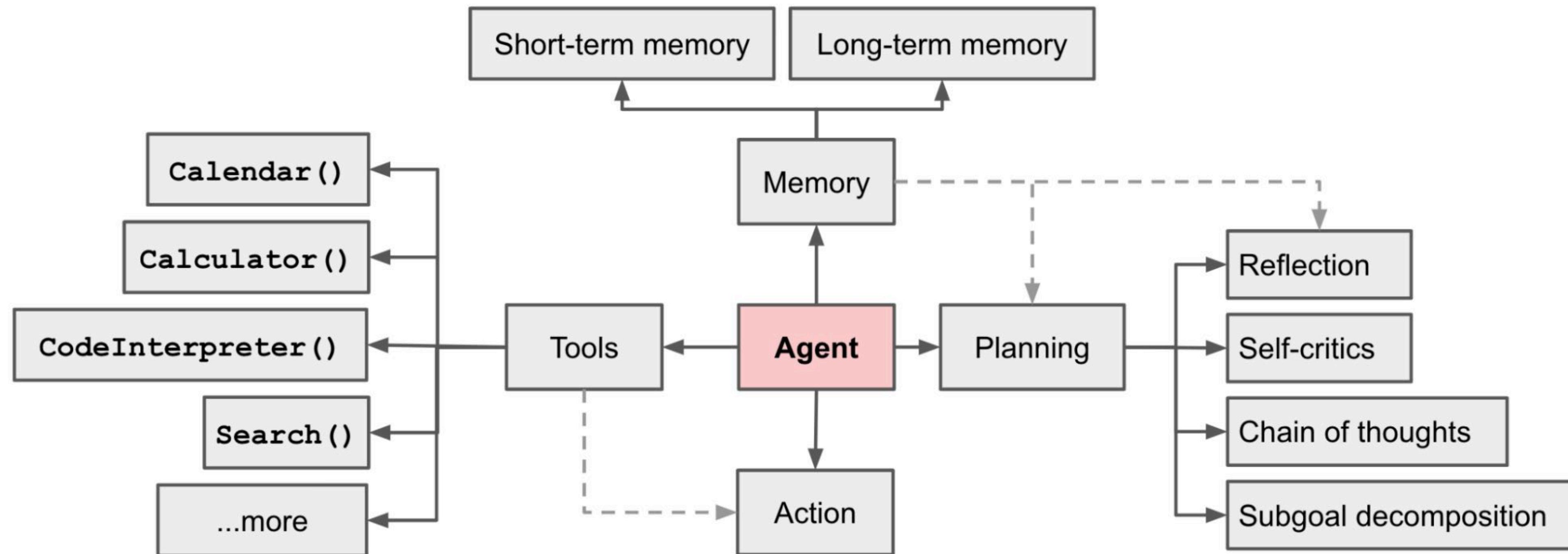


Another form of reflection

[Shin et al. 2023]

- Reinforcement learning formulation (assumes a world simulator exists)
- 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”
- Model model is asked to produce an English-language critique of what went wrong (e.g, what and where in the log)
- Now retry plan generation with: ($W, T, \text{log, reward, critique}$)
- We'll see the same idea in the Voyager paper you'll read tonight.

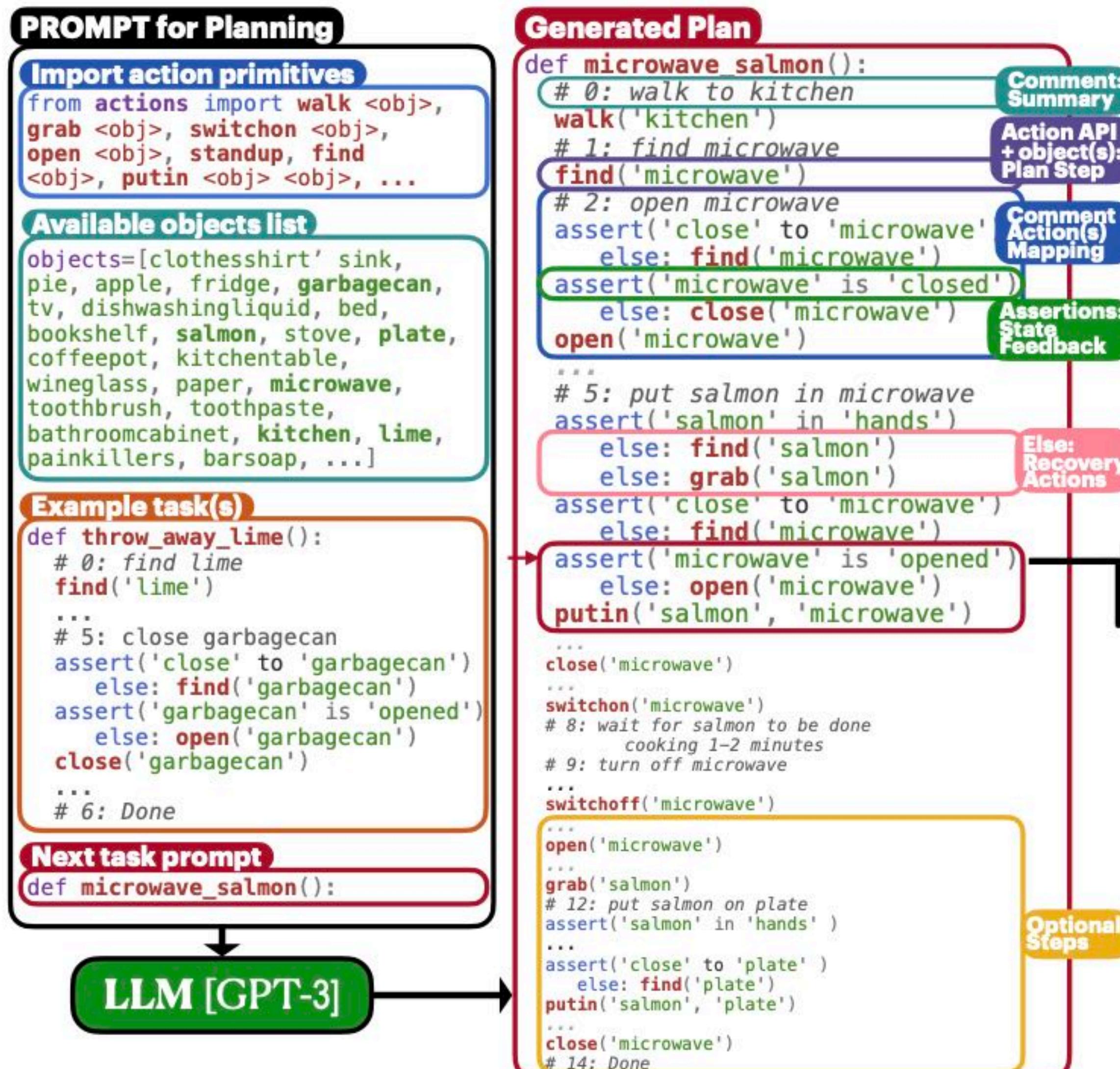
General LLM-agent architecture



[Image credit: Lilian Weng]

Recall from prior lecture: plans as valid programs

“Microwave salmon”:



Example: ProgPrompt [Sing et al. ICRA 23]

Key ideas:

- LM as a code generator: a plan is a valid python program with access to subroutines (action space defined by subroutines)
- World state is given by a list of available objects
- Through conditional logic, plans can have grounded recovery policies:
 - If condition is not true, do X

Voyager

- Task: do things in Minecraft: make new things, fight Zombies, etc.

- Big ideas:

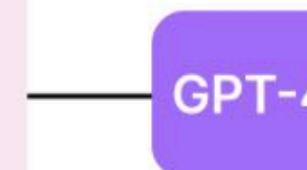
- Use LM to generate plans as programs
- Use LM to repair bad programs: (reflection)
 - Run programs to see if they work.
 - Get compiler feedback and game state feedback
 - Use LM to repair program given feedback



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



Inventory (5/36): {'oak_planks': 3, 'stick': 4, 'crafting_table': 1, 'stone': 3, 'wooden_pickaxe': 1}



Reasoning: Since you have a wooden pickaxe and some stones, it would be beneficial to upgrade your pickaxe to a stone pickaxe for better efficiency.
Task: Craft 1 stone pickaxe.

- 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