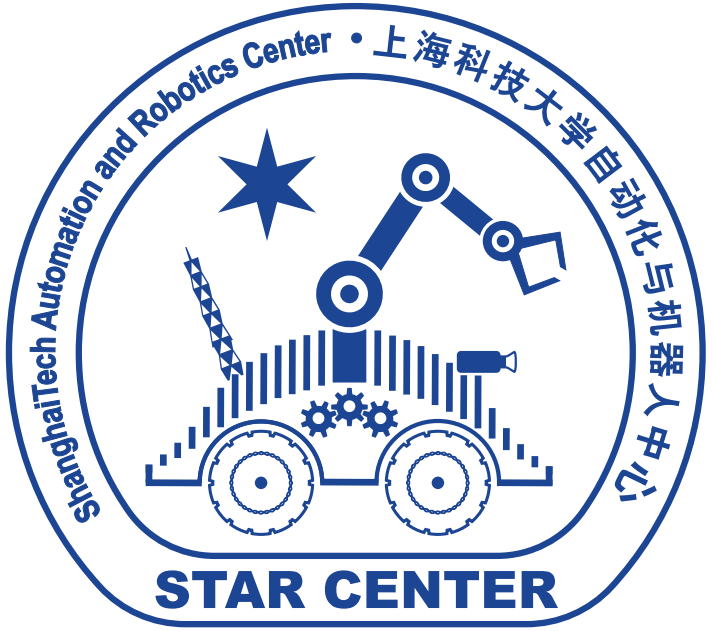
上海科技大学
ShanghaiTech University

CS289: **Mobile Manipulation Fall 2024**

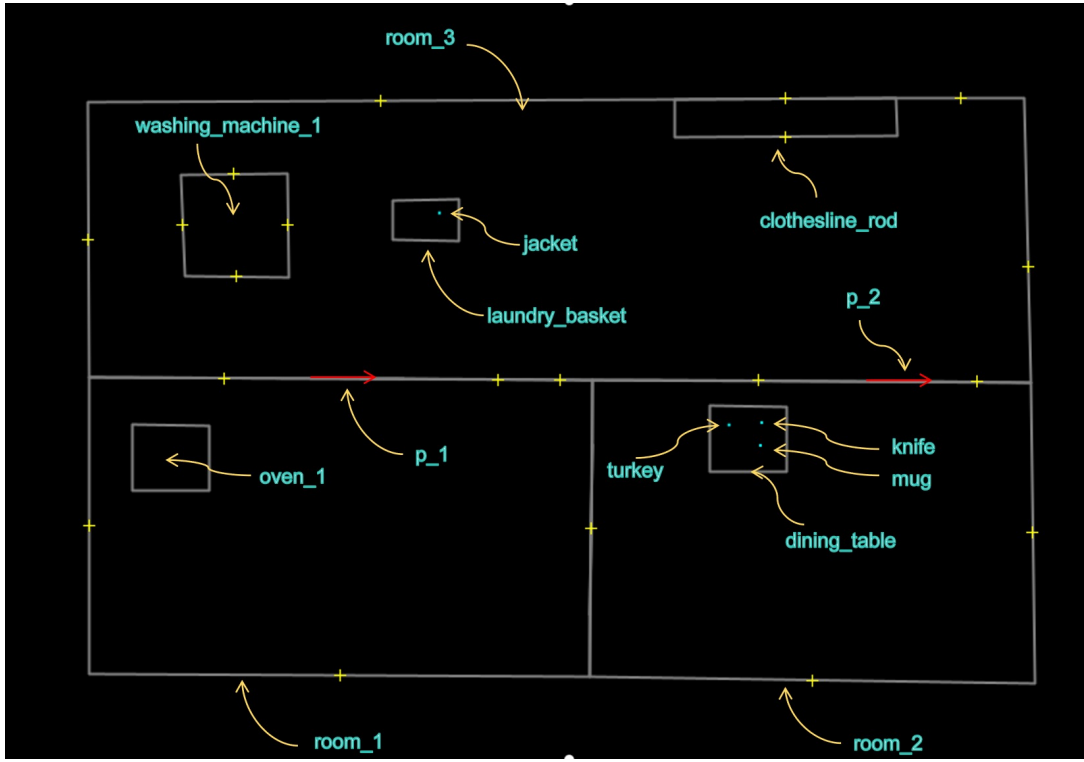
Sören Schwertfeger

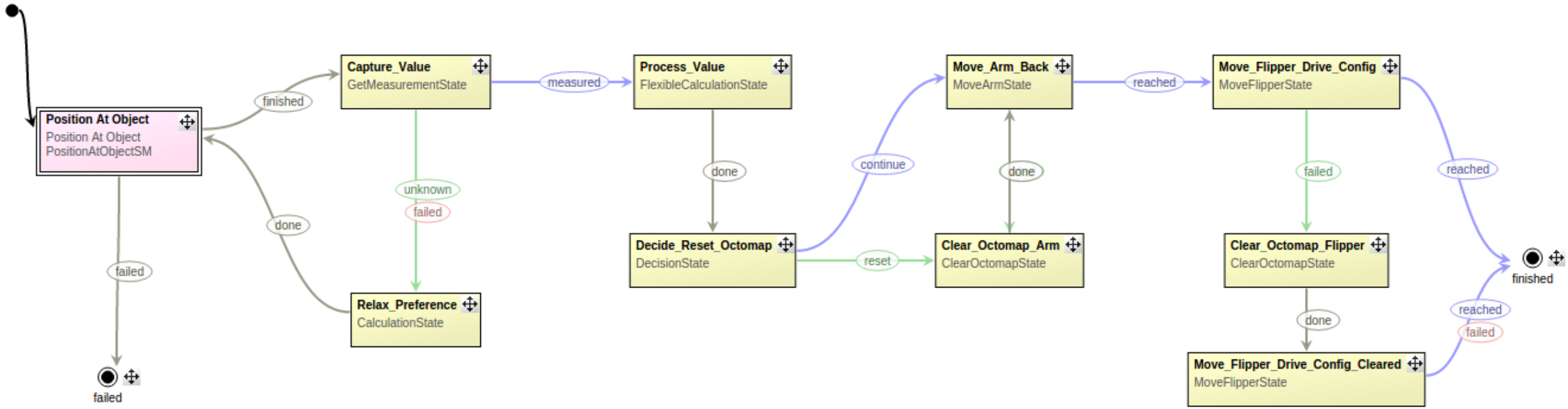
ShanghaiTech University



HW 4

- Execute Task: Cook Turkey & Wash Jacket
- We provide: simulator that executes 10 simple actions:
  - close_door, open_door; close_furniture; open_furniture; move_robot; pick; place; wash; cook; hang_clothes
- Map is fixed
- Robot start room and initial turkey location are randomized
- Robot is clumsy: door opening has failure rate of 50%!
- Goal: execute actions (via ROS Service) such that goal state is reached
- Use: BT; State Machine; Symbolic Planner; LLM
  - Work in groups of 4 – one approach per student!





Paper:
Flexible Navigation: Finite State Machine-Based Integrated Navigation and Control for ROS Enabled Robots
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7925266

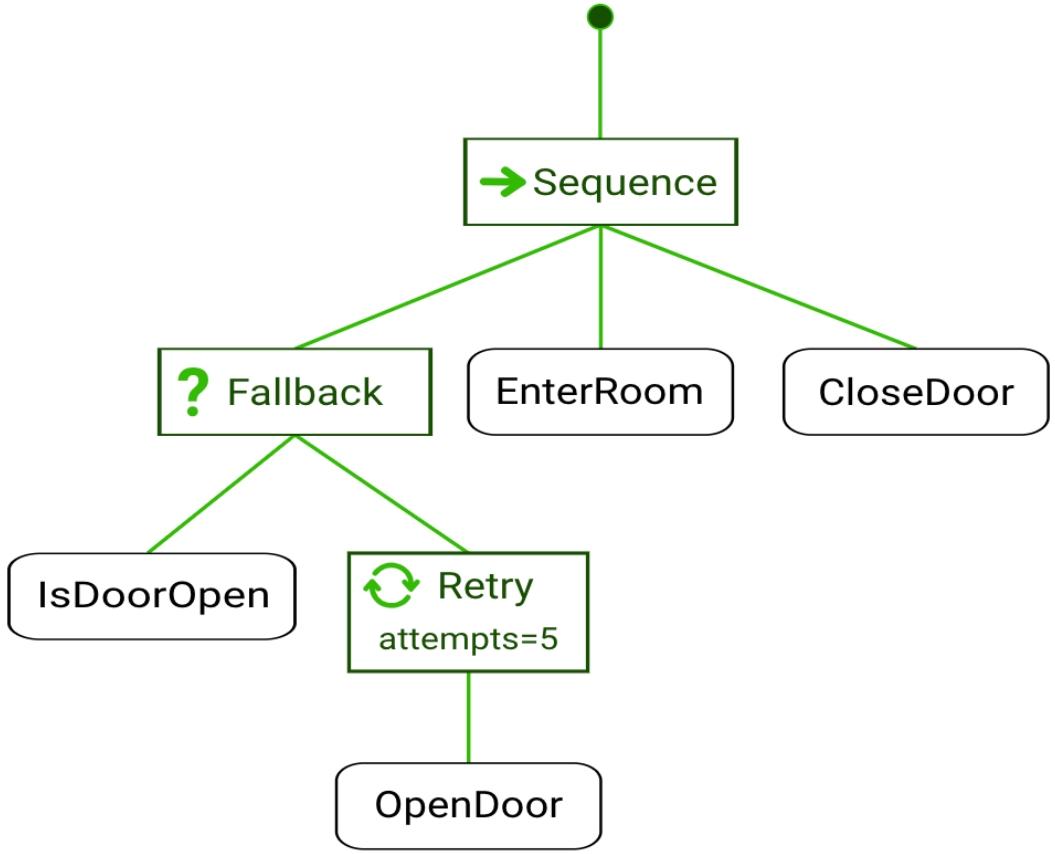
http://wiki.ros.org/flexbe

https://flexbe.readthedocs.io/en/latest/

Behavior Tree (BT)

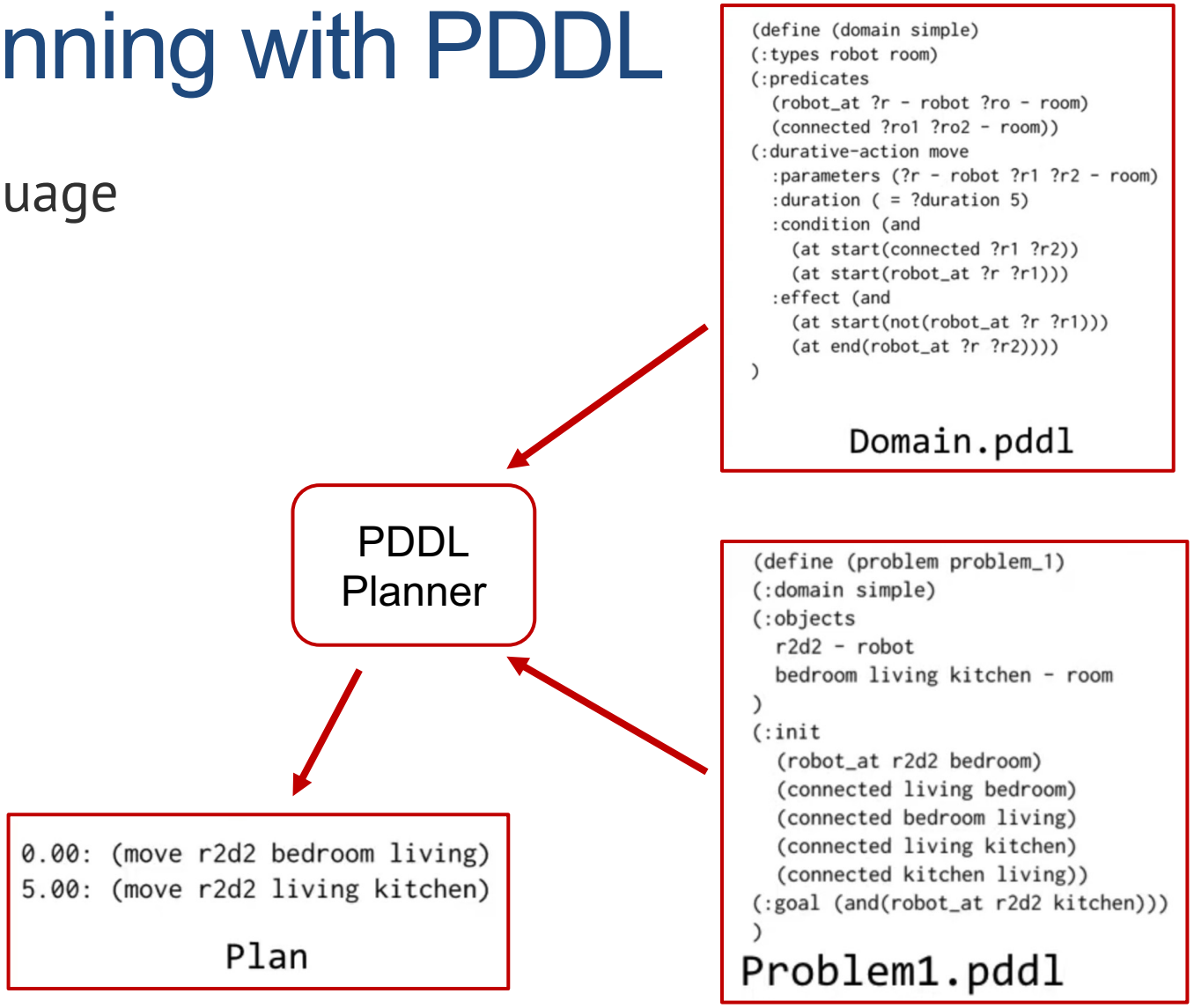
Behavior Trees in Robotics and AI: An Introduction
Michele Colledanchise, Petter Ögren
https://arxiv.org/abs/1709.00084

- Alternative to Finite State Machine (FSM)
  - BT (supposedly) more scalable, more human-understandable and easier to reuse than FSM
  - Intrinsically hierarchical
  - Graphical representation has meaning
  - Expressive
- BehaviorTree CPP V3
  https://www.behaviortree.dev/
- Defined in XML
- Execute top down, left first (similar to DFS)



Mission and Task Planning with PDDL

- Planning Domain Definition Language
- Establish the rules (Domain)
- Present a situation and goal (Problem)
- Use a plan solver
- Get a plan



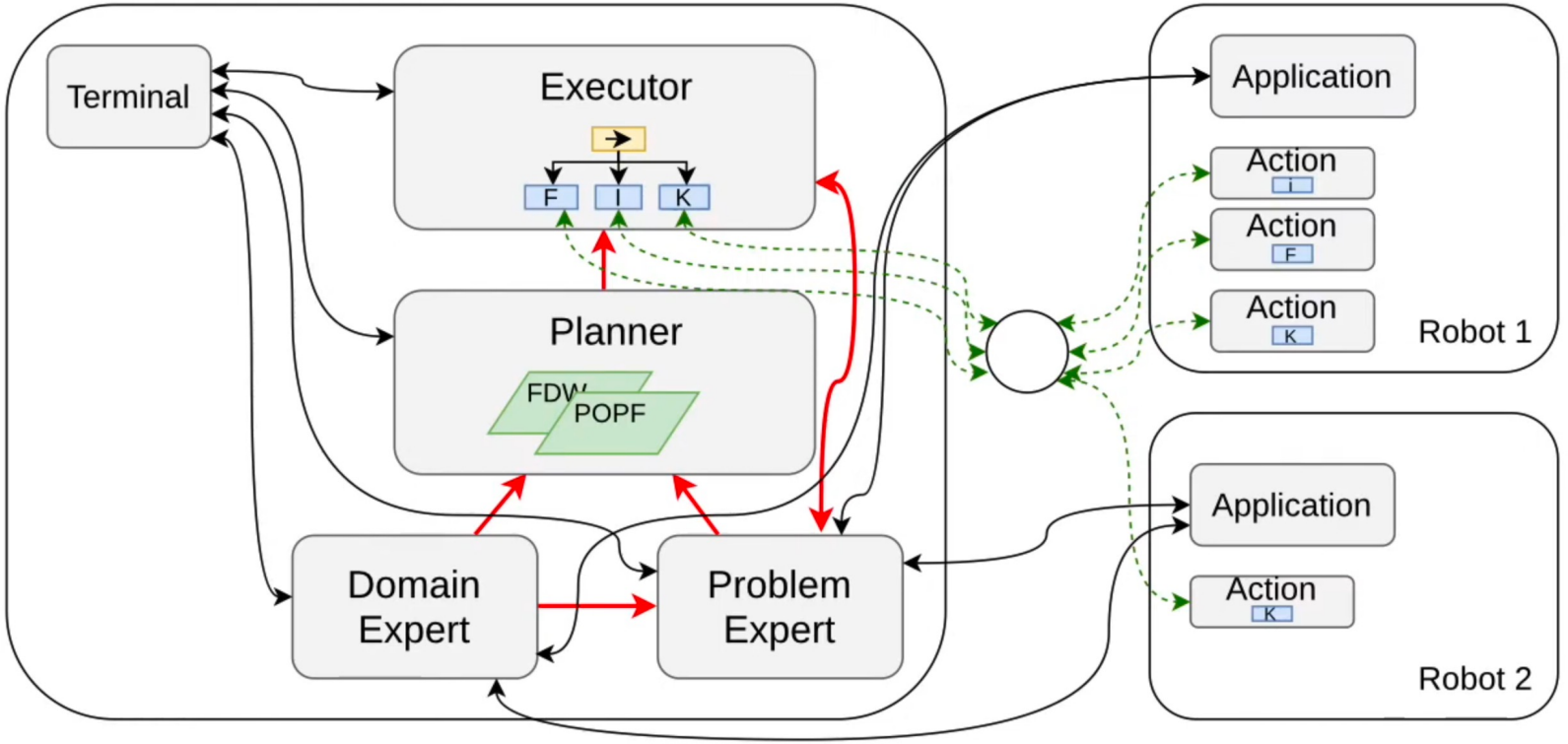
Solver/ Planners

- PDDL just a description language
- Standardized - to be used in different solvers
- Many different algorithms:
  - Classical planning:
    - forward chaining of state space, maybe with heuristics
    - Backward chaining (e.g. STRIPS from 1970s)
  - Reduction to other problems
    - Model checking (planning is a subclass of model checking)
  - Temporal planning
    - Actions may be executed in parallel/ overlapping in time
  - Probabilistic planning
    - MDP
    - POMDP

ROS 2 Planning System

PlanSys2

- Framework for AI Planning
- Modular Design
- PlanSyst2 Terminal
- Multirobot
- Parallel execution
- Load balancing
- Execution of plans as BT!



- Planners as plugins
- Default:
  - Partial Order Planning Forwards (POPF)
  - forwards-chaining temporal planner

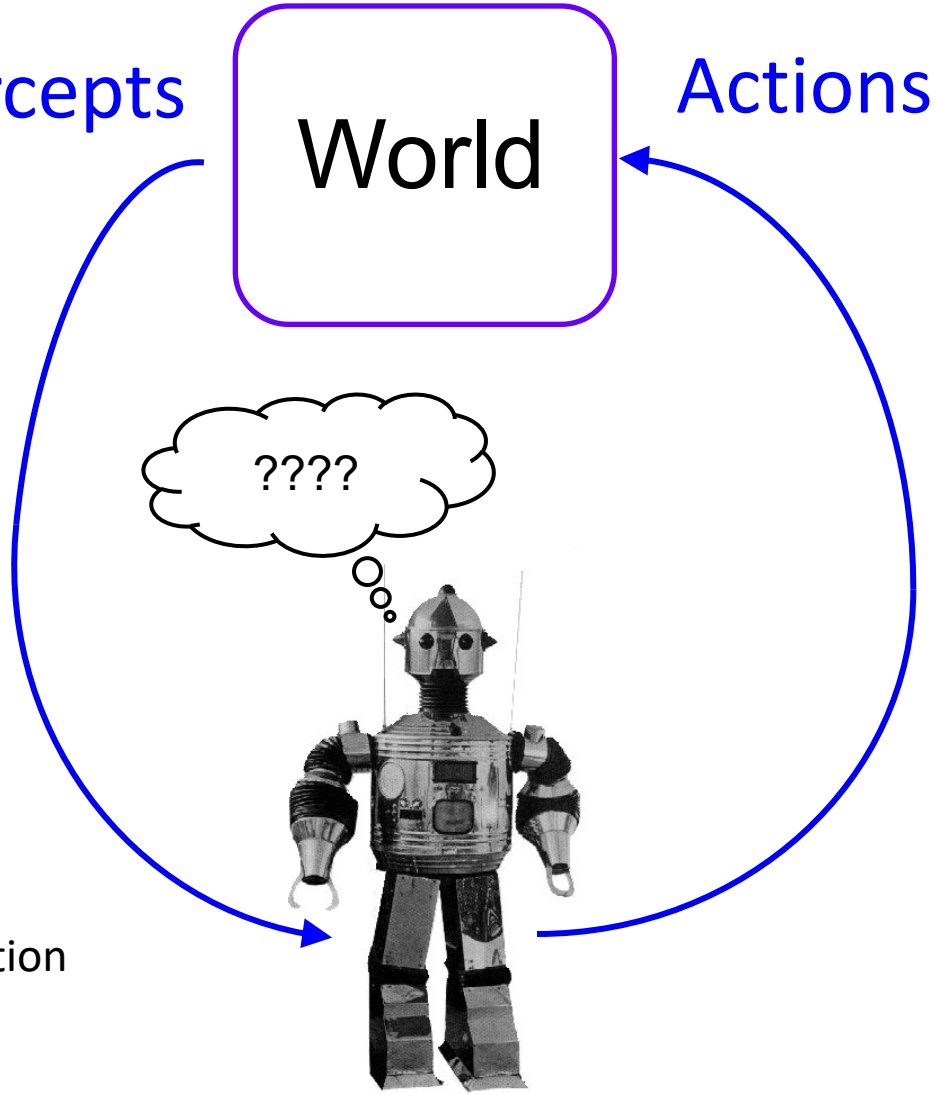
https://plansys2.github.io/

AI PLANNING

Languages for Planning Problems

- STRIPS
  - Stanford Research Institute Problem Solver
  - Historically important
- ADL
  - Action Description Languages
  - See Table 11.1 for STRIPS versus ADL
- PDDL
  - Planning Domain Definition Language
  - Revised & enhanced for the needs of the International Planning Competition
  - Currently version 3.1

Classical Planning Assumptions



perfect

fully observable

sole source of change

deterministic

instantaneous

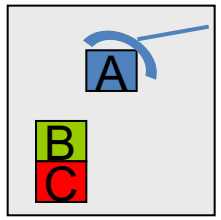
Goal

achieve goal condition

Material: Alan Fern; Daniel Weld
Oregon State University
https://web.engr.oregonstate.edu/~afern/classes/cs533/notes/strips-intro.pptx

Representing States

World states are represented as sets of facts.
We will also refer to facts as propositions.

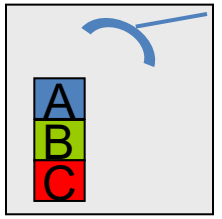


holding(A)
clear(B)
on(B,C)
onTable(C)

State 1

handEmpty
clear(A)
on(A,B)
on(B,C)
onTable(C)

State 2

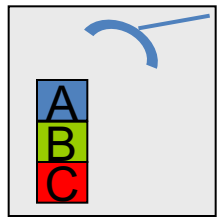


Closed World Assumption (CWA):
Fact not listed in a state are assumed to be false. Under CWA we are assuming the agent has full observability.

Representing Goals

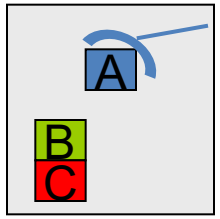
Goals are also represented as sets of facts.
For example { on(A,B) } is a goal in the blocks world.

A **goal state** is any state that contains all the goal facts.



handEmpty
clear(A)
on(A,B)
on(B,C)
onTable(C)

State 1

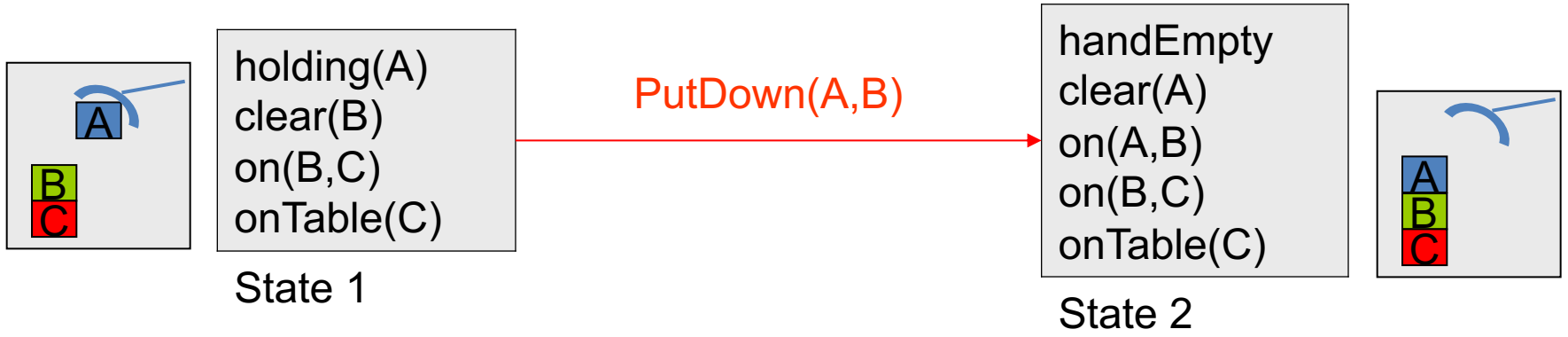


holding(A)
clear(B)
on(B,C)
onTable(C)

State 2

State 1 is a goal state for the goal { on(A,B) }.
State 2 is not a goal state for the goal { on(A,B) }.

Representing Action in STRIPS



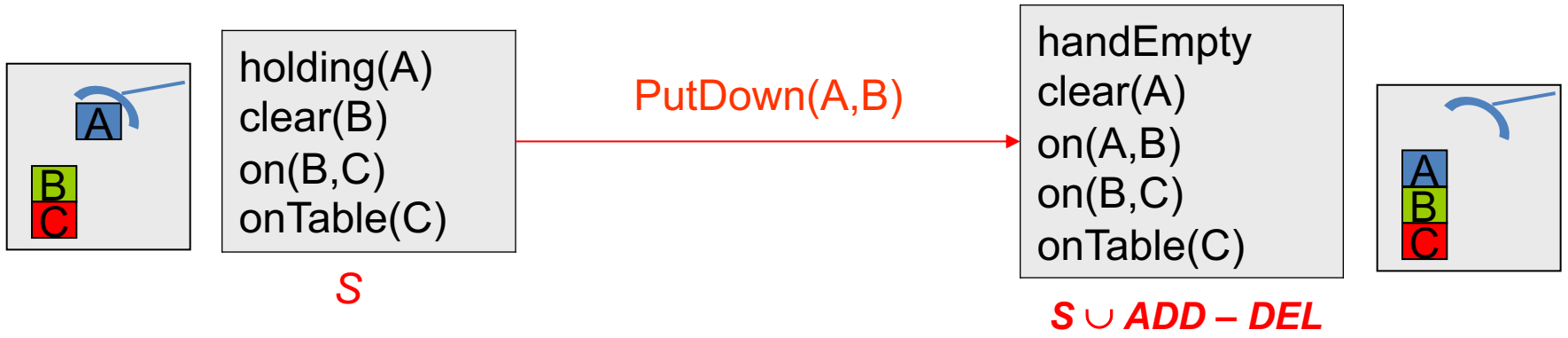
A STRIPS action definition specifies:

1) a set PRE of preconditions facts
2) a set ADD of add effect facts
3) a set DEL of delete effect facts

PutDown(A,B):

PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A) }
DEL: { holding(A), clear(B) }

Semantics of STRIPS Actions



- A STRIPS action is **applicable** (or allowed) in a state when its preconditions are contained in the state.
- Taking an action in a state ***S*** results in a new state ***S ∪ ADD – DEL*** (i.e. add the add effects and remove the delete effects)

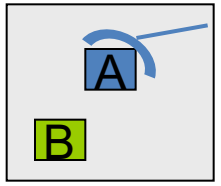
PutDown(A,B):
PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A)}
DEL: { holding(A), clear(B) }

STRIPS Planning Problem

A **STRIPS planning problem** specifies:
1) an initial state *S*
2) a goal *G*
3) a set of STRIPS actions

Objective: find a "short" action sequence reaching a goal state, or report that the goal is unachievable

Example Problem:



holding(A)
clear(B)
onTable(B)

Initial State

Solution: (PutDown(A,B))

on(A,B)

Goal

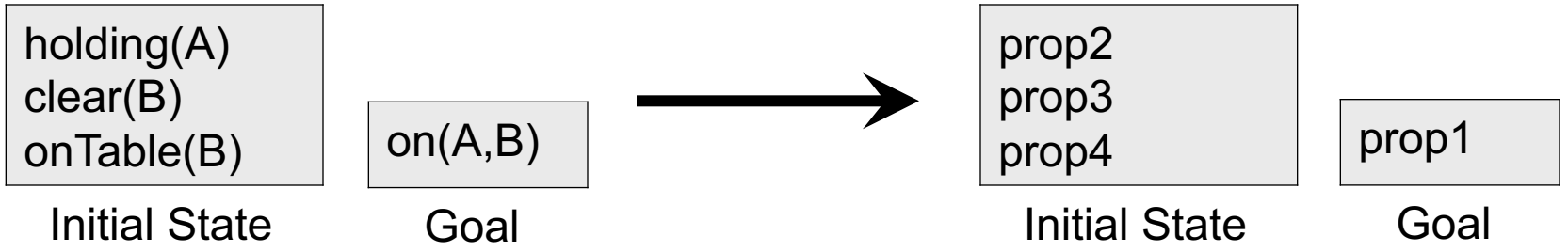
PutDown(A,B):
PRE: { holding(A), clear(B) }
ADD: { on(A,B), handEmpty, clear(A)}
DEL: { holding(A), clear(B) }

PutDown(B,A):
PRE: { holding(B), clear(A) }
ADD: { on(B,A), handEmpty, clear(B) }
DEL: { holding(B), clear(A) }

STRIPS Actions

Propositional Planners

- So far: written propositions (e.g. on(A,B)) in terms of objects (e.g. A and B) and predicates (e.g. on).
- But: planners ignore the internal structure of propositions such as on(A,B).
- Such planners are called **propositional planners** as opposed to first-order or relational planners
- => no difference to the planner if we replace “on(A,B)” in a problem with “prop1” (and so on)
- It feels wrong to ignore the existence of objects. But currently propositional planners are the state-of-the-art.



STRIPS Versus PDDL

- The Planning Domain Description Language (PDDL): standard language for defining planning problems
  - Includes STRIPS as special case along with more advanced features
  - simple additional features: type specification for objects, negated preconditions, conditional add/del effects
  - advanced features: allowing numeric variables and durative actions

- Most planners you can download take PDDL as input
  - Majority only support the simple PDDL features (essentially STRIPS)
  - PDDL syntax is easy to learn from examples packaged with planners, but a definition of the STRIPS fragment can be found at:

    http://eecs.oregonstate.edu/ipc-learn/documents/strips-pddl-subset.pdf

Properties of Planners

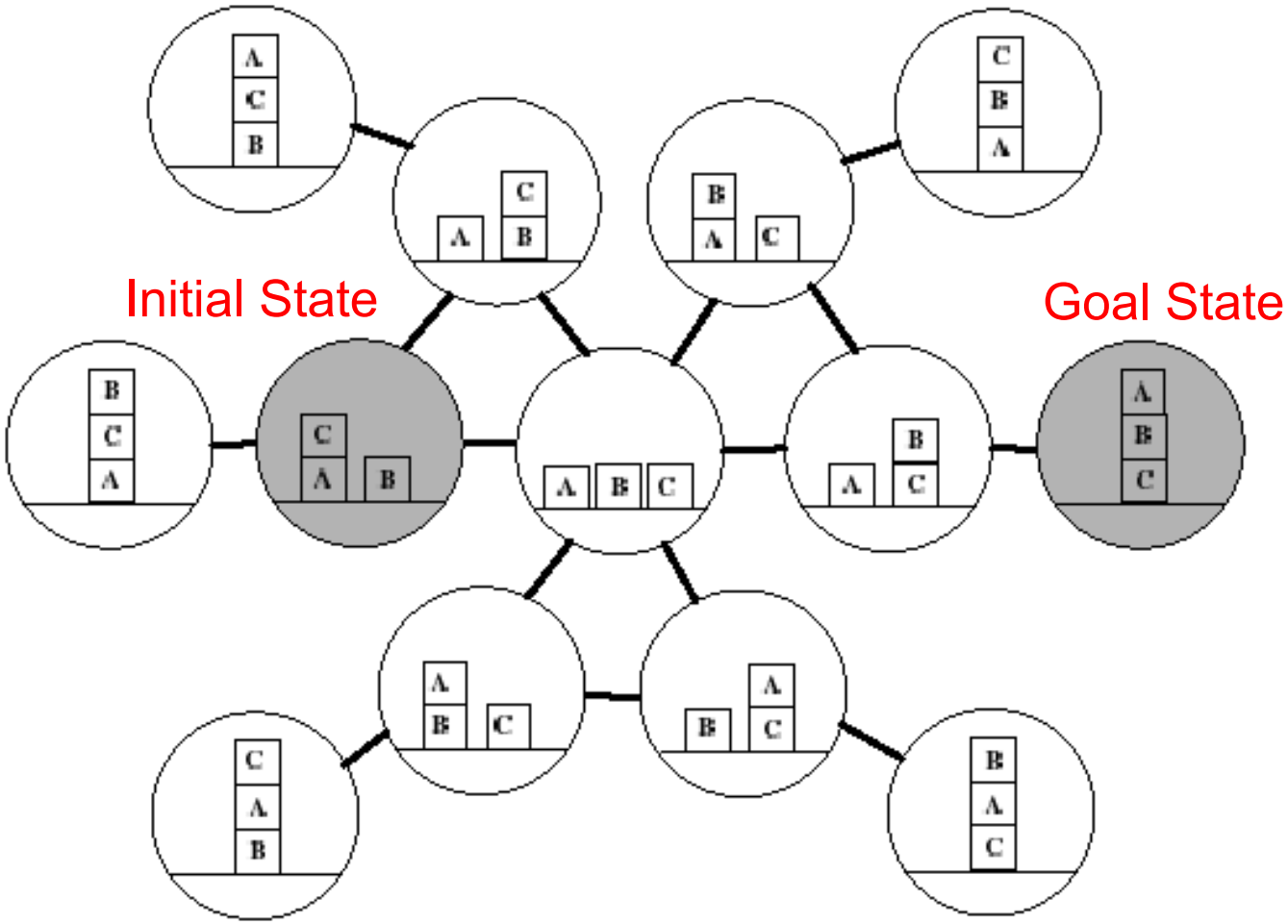
- A planner is **sound** if any action sequence it returns is a true solution
- A planner is **complete** if it outputs an action sequence or “no solution” for any input problem
- A planner is **optimal** if it always returns the shortest possible solution

Planning as Graph Search

- It is easy to view planning as a graph search problem
- Nodes/vertices = possible states
- Directed Arcs = STRIPS actions
- Solution: path from the initial state (i.e. vertex) to one state/vertices that satisfies the goal

Search Space: Blocks World

Graph is finite



Planning as Graph Search

- Planning is just finding a path in a graph
  - Why not just use standard graph algorithms for finding paths?
- **Answer:** graphs are exponentially large in the problem encoding size (i.e. size of STRIPS problems).
  - But, standard algorithms are poly-time in graph size
  - So standard algorithms would require exponential time

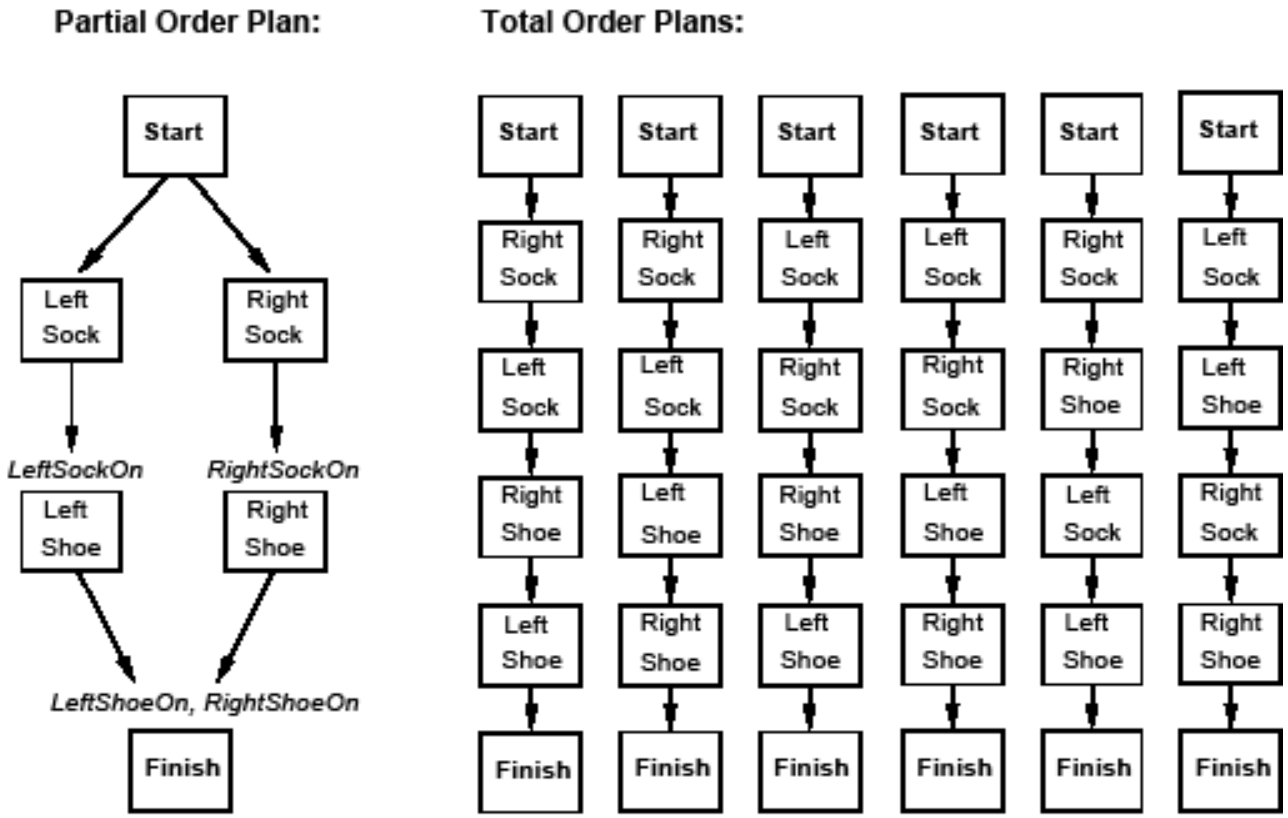
- Do better:
  - We can use A*, but we need an admissible heuristic
    1. Divide-and-conquer: sub-goal independence assumption
    - Problem relaxation by removing
    2. … all preconditions
    3. … all preconditions and negative effects
    4. … negative effects only: Empty-Delete-List

Partial Order Planning (POP)

- State-space search
  - Yields totally ordered plans (linear plans)
- POP
  - Works on subproblems independently, then combines subplans
  - Example
    - Goal(RightShoeOn $\wedge$ LeftShoeOn)
    - Init()
    - *Action*(RightShoe, PRECOND: RightSockOn, EFFECT: RightShoeOn)
    - *Action*(RightSock, EFFECT: RightSockOn)
    - *Action*(LeftShoe, PRECOND: LeftSockOn, EFFECT: LeftShoeOn)
    - *Action*(LeftSock, EFFECT: LeftSockOn)

Berthe Y. Choueiry (Shu-we-ri)
University of Nebraska Lincoln

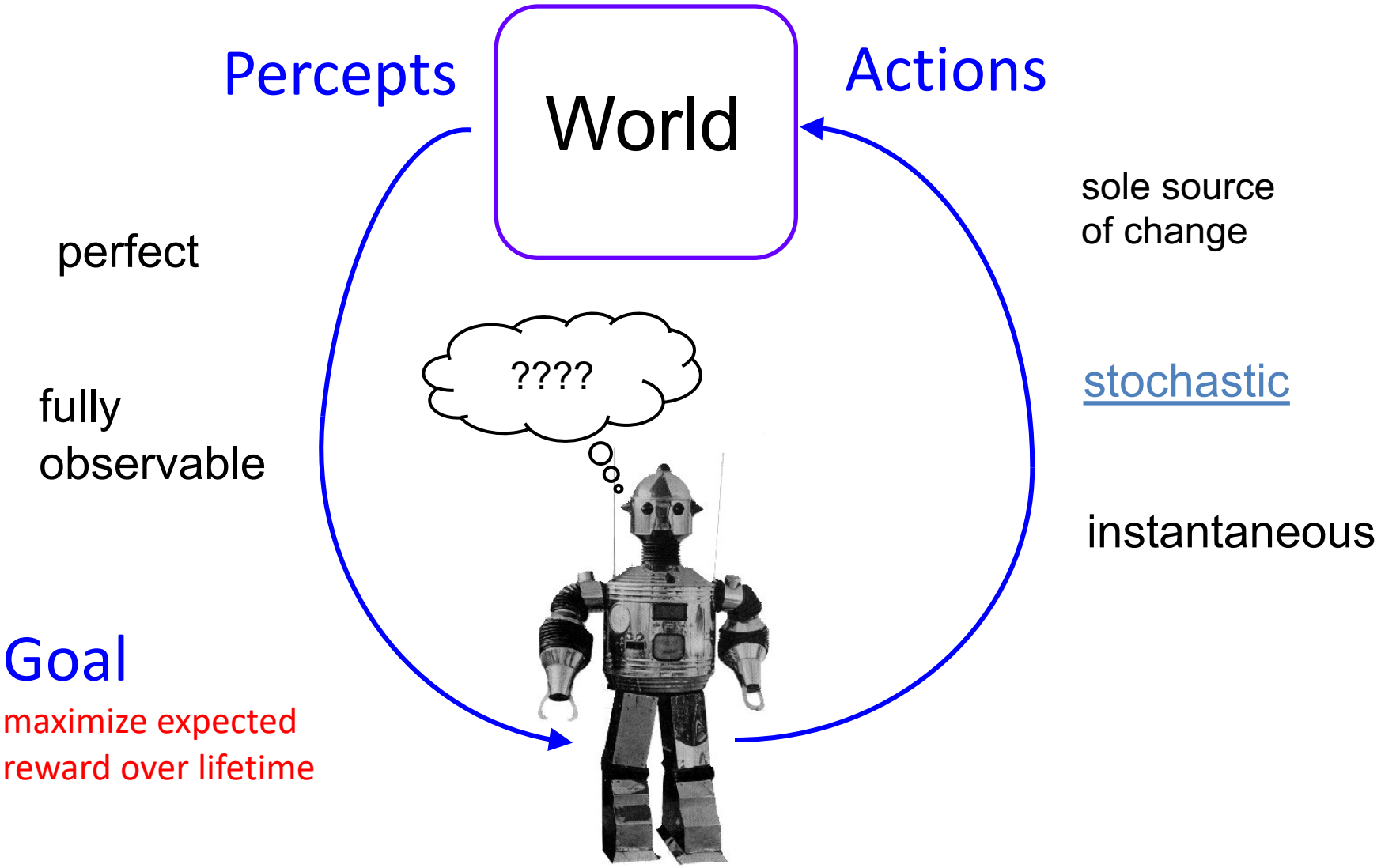
POP Example & its linearization



Berthe Y. Choueiry (Shu-we-ri)
University of Nebraska Lincoln

POMDP

Stochastic/Probabilistic Planning: Markov Decision Process (MDP) Model

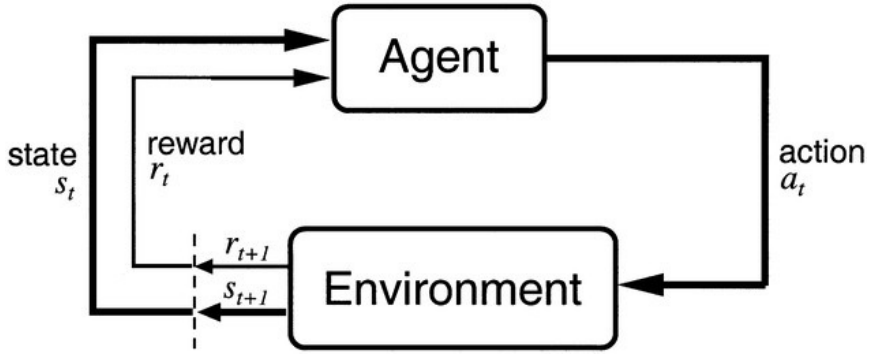


POMDPs

- A special case of the Markov Decision Process (MDP)
- MDP:
  - Sequential decision making
  - Outcome uncertain
  - the environment is fully observable
  - Markov assumption for the transition model
    - Distribution of next state only depends on current state and action =>
  - the optimal policy depends only on the current state.

- For POMDPs, the environment is only partially observable

Markov Decision Process (S, A, H, T, R)



Given

- S: set of states
- A: set of actions
- H: horizon over which the agent will act
- T Transition dynamics: S x A x S x {0,1,…,H} ➜ [0,1] , $T_t(s,a,s') = P(s_{t+1} = s' \mid s_t = s, a_t = a)$
- R Reward: S x A x S x {0, 1, …, H} ➜ $\mathbb{R}$, $R_t(s,a,s')$ = reward for $(s_{t+1} = s', s_t = s, a_t = a)$

Goal: Policy π

- Find : S x {0, 1, …, H} ➜ A that maximizes expected sum of rewards, i.e.,

$$\pi^* = \arg\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} R_t(S_t, A_t, S_{t+1})|\pi]$$

https://ai.stanford.edu/~gwthomas/notes/mdps.pdf

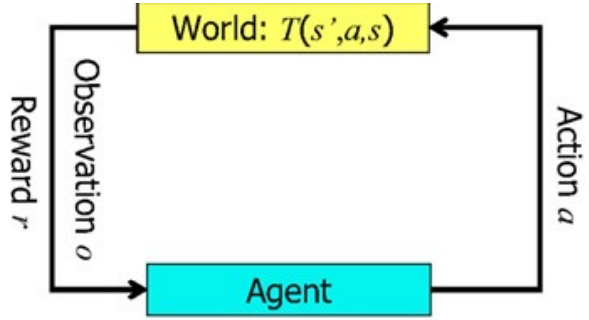
Material: Sachin Patil
Pieter Abbeel, Alex Lee
UC Berkeley

POMDP – Partially Observable MDP

= MDP

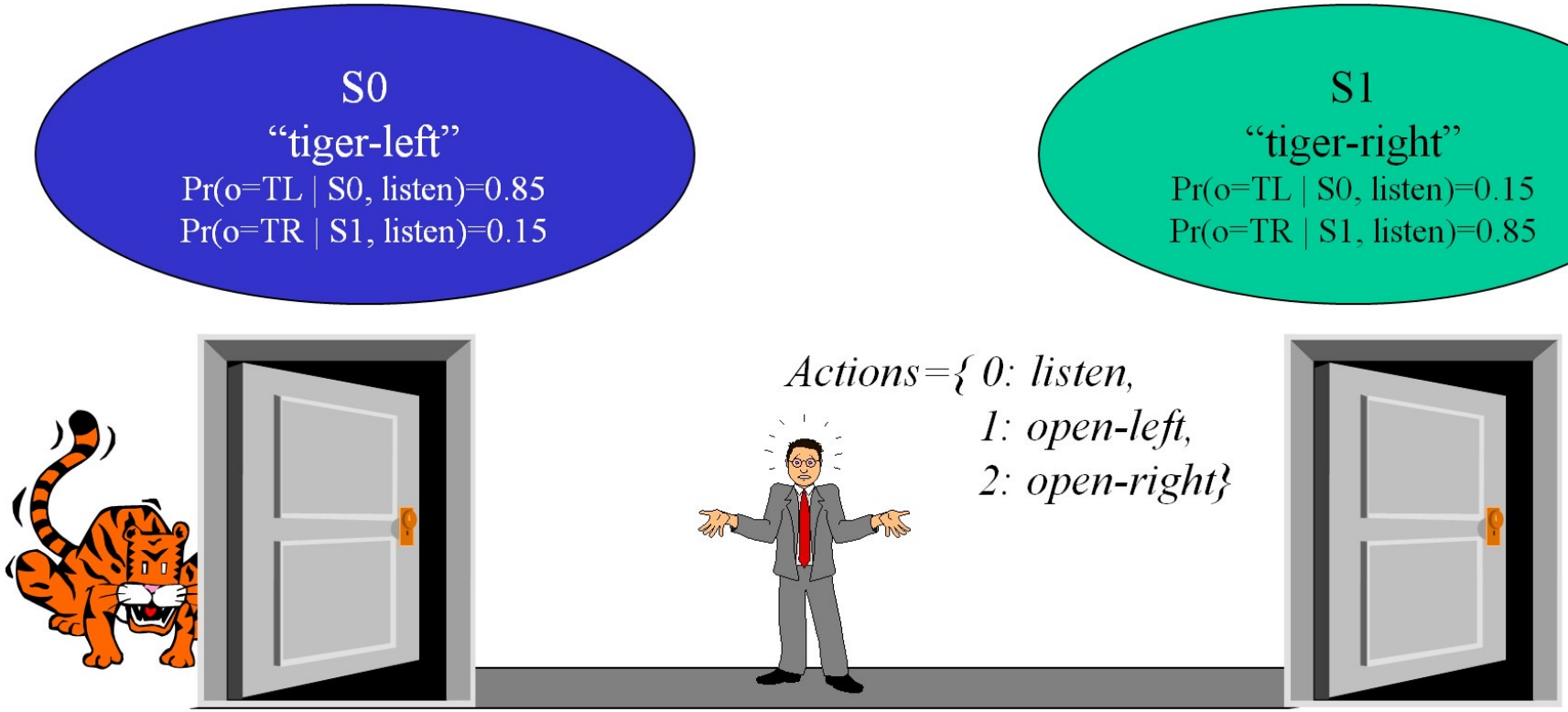
BUT

don’t get to observe the state itself, instead get sensory measurements



Now: what action to take given current probability distribution rather than given current state.

POMDPs: Tiger Example



Reward Function

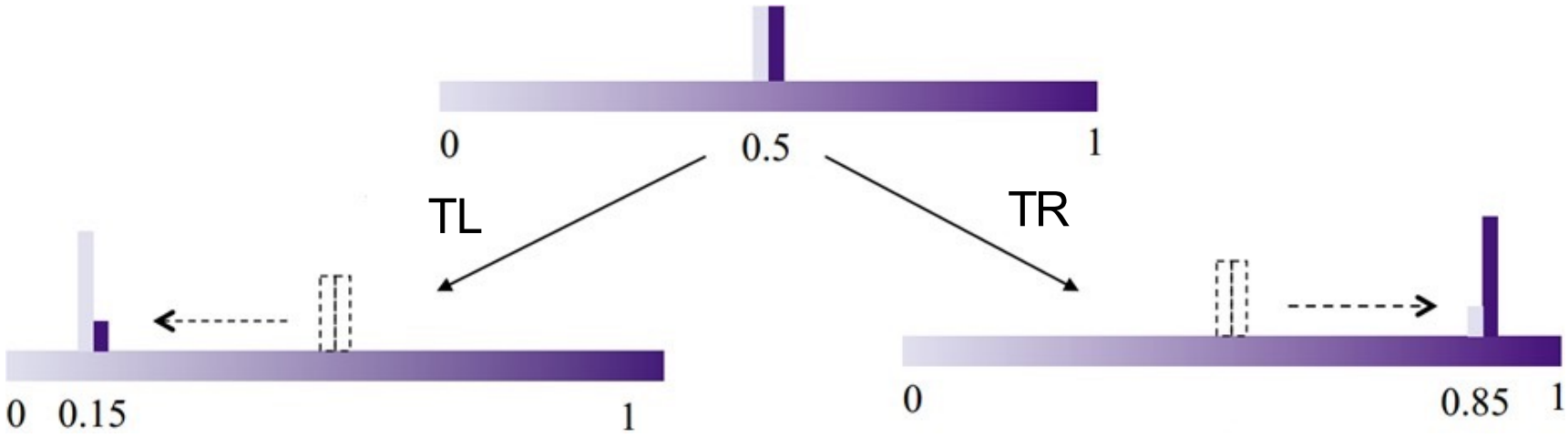
- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

Belief State

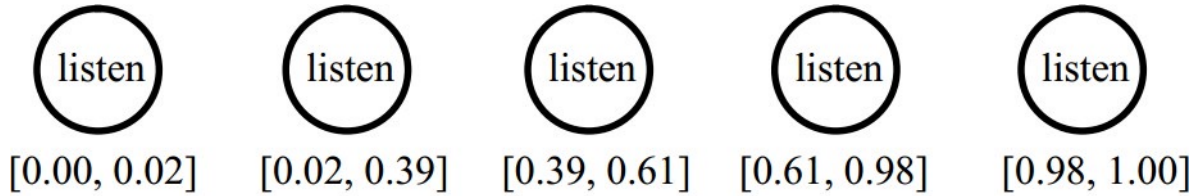
- Probability of S0 vs S1 being true underlying state
- Initial belief state: p(S0)=p(S1)=0.5
- Upon listening, the belief state should change according to the Bayesian update (filtering)



Policy – Tiger Example

- Policy π is a map from [0,1] → {listen, open-left, open-right}
- What should the policy be?
  - Roughly: listen until sure, then open
- But where are the cutoffs?

Tiger example optimal policy for $t = 1$



Tiger example optimal policy for $t = 2$

Some Solution Techniques

- Most exact solution algorithms (value iteration, policy iteration) use dynamic programming techniques
  - transform from one value function (the transition model in physical space, which is piecewise linear and convex - PWLC) to another that can be used in an MDP solution technique
  - Dynamic programming algorithms: one-pass (1971), exhaustive (1982), linear support (1988), witness (1996)
  - Better method – incremental pruning (1996)

COMBINED TASK & MOTION PLANNING

Differentiable Task Assignment and Motion Planning

Jimmy Envall, Roi Poranne, Stelian Coros

ETH zürich

CRL Computational Robotics Lab

אוניברסיטת חיפה
University of Haifa
جامعة حيفا

Envall, J., Poranne, R., & Coros, S. (2023, October). Differentiable task assignment and motion planning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2049-2056). IEEE. https://arxiv.org/pdf/2304.09734

https://www.youtube.com/watch?v=zq3WyQFE5mA

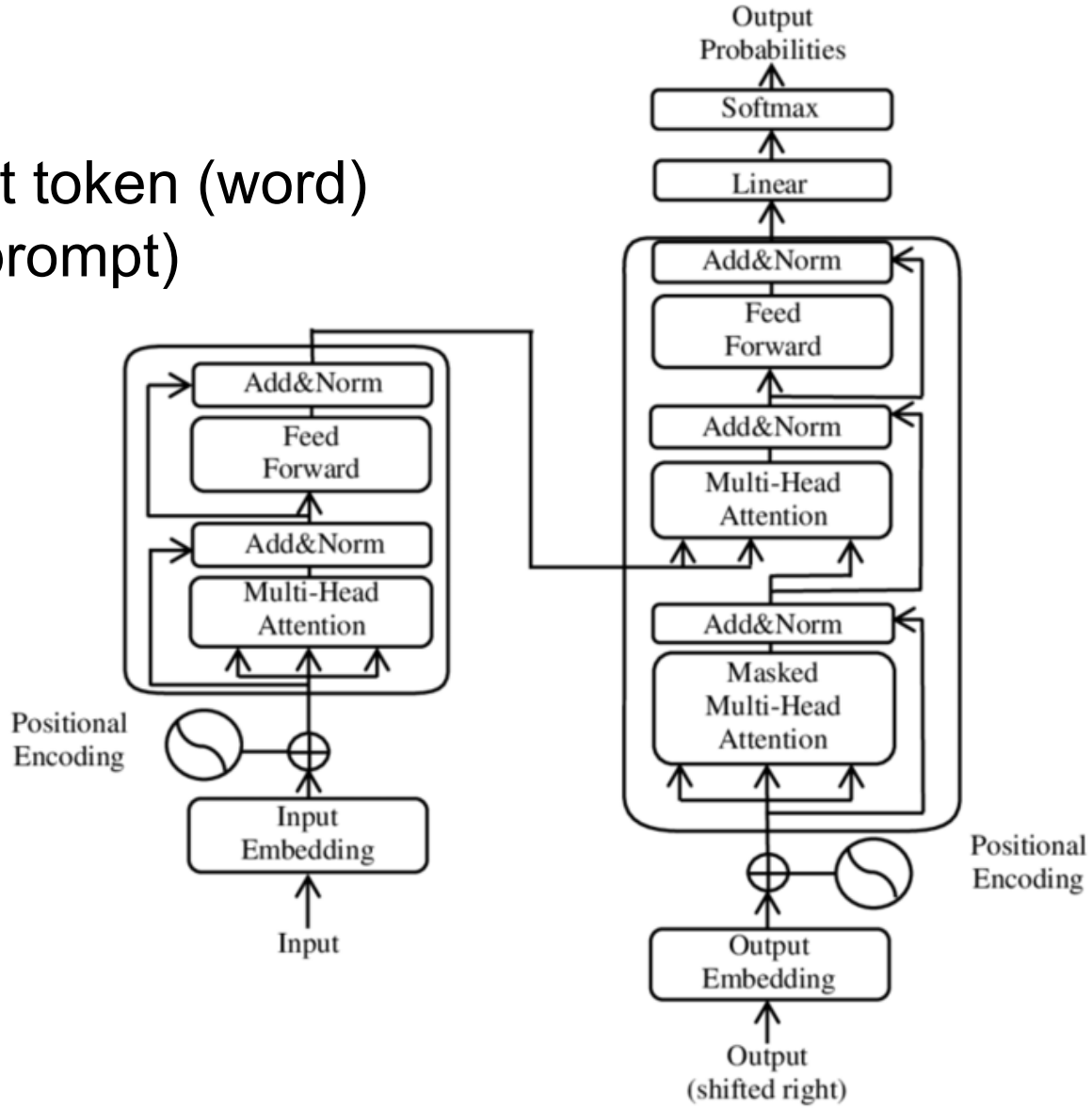
LLM

Large Language Model

- Transformer-based model outputting next token (word) with highest probability based on input (prompt)

- Powerful AI tool because:
  - “Understands” natural language (and other structured text) input
  - Through training has good world knowledge
  - Some form of (limited) reasoning

  - => useful in robotics, especially
    - Human Robot Interaction
    - All kinds of planning tasks
    - Various other purposes



LLM for Robotics

- Do As I Can, Not As I Say: Grounding Language in Robotic Affordances
  —4 Apr 2022: Robotics at Google, Everyday Robots
- ChatGPT for Robotics: Design Principles and Model Abilities
  —20 Feb 2023: Microsoft Autonomous Systems and Robotics Research
- Palm-E: An embodied multimodal language model
  —6 Mar 2023: Robotics at Google, TU Berlin, Google Research
- RT-2: New model translates vision and language into action
  —28 Jul. 2023: Google DeepMind
- RobotGPT: Robot Manipulation Learning from ChatGPT
  —3 Dec 2023: Samsung Research China, Tsinghua University
- Large Language Models for Robotics:Opportunities, Challenges, and Perspectives
  —9 Jan 2024: Northwestern Polytechnical University, Univ. Georgia, etc.
- Large Language Models for Robotics: A Survey
  —13 Nov 2023: Jinan University, University of Illinois Chicago
- Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis
  —14 Dec 2023: CMU, FAIR at Meta, Google DeepMind
- A Survey of Language-Based Communication in Robotics
  —6 Jun 2024: University of Southampton

LLM for Robotics

https://github.com/GT-RIPL/Awesome-LLM-Robotics

Awesome-LLM-Robotics awesome

This repo contains a curative list of **papers using Large Language/Multi-Modal Models for Robotics/RL**. Template from awesome-Implicit-NeRF-Robotics

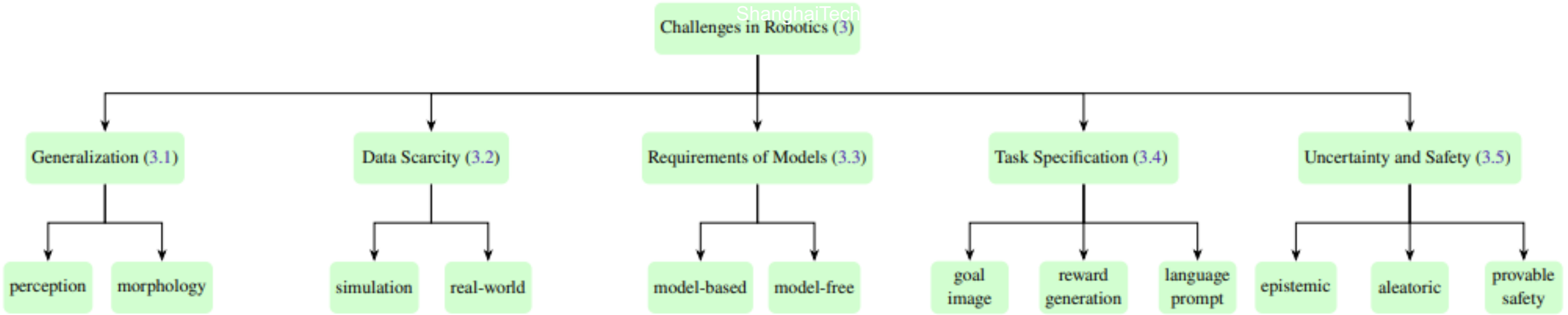
Please feel free to send me pull requests or email to add papers!

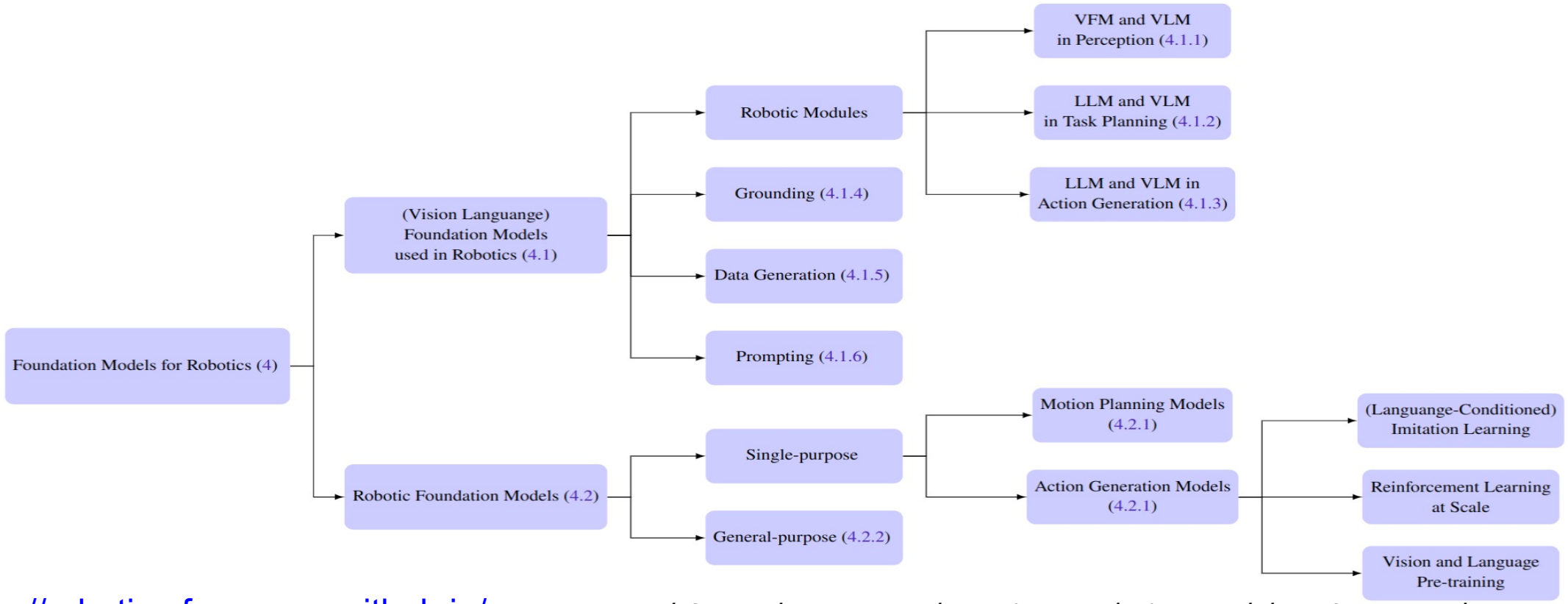
If you find this repository useful, please consider citing and STARing this list. Feel free to share this list with others!

Overview

Surveys

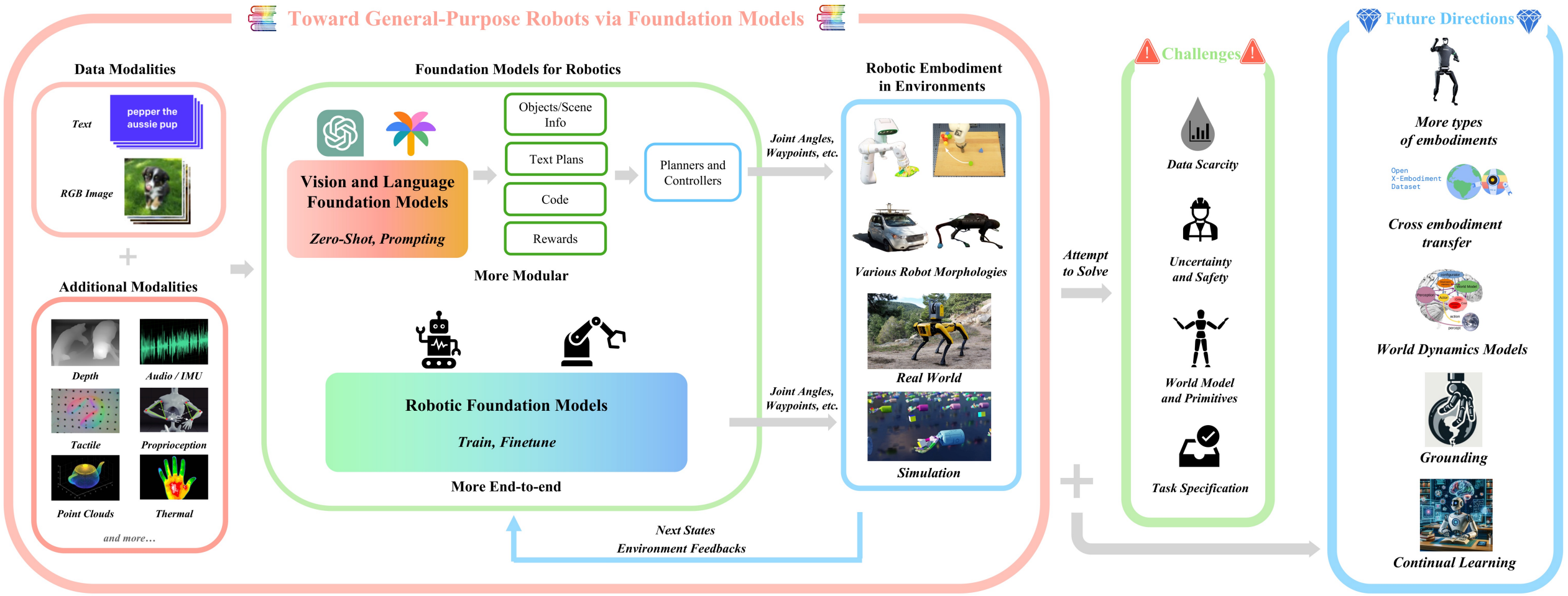
- "Neural Scaling Laws for Embodied AI", *arXiv, May 2024*. [Paper]
- "Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis", *arXiv, Dec 2023*. [Paper] [Paper List] [Website]
- "Language-conditioned Learning for Robotic Manipulation: A Survey", *arXiv, Dec 2023*, [Paper]
- "Foundation Models in Robotics: Applications, Challenges, and the Future", *arXiv, Dec 2023*, [Paper] [Paper List]
- "Robot Learning in the Era of Foundation Models: A Survey", *arXiv, Nov 2023*, [Paper]
- "The Development of LLMs for Embodied Navigation", *arXiv, Nov 2023*, [Paper]





https://robotics-fm-survey.github.io/ - *Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis*

Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis



ChatGPT for Robotics: Design Principles and Model Abilities

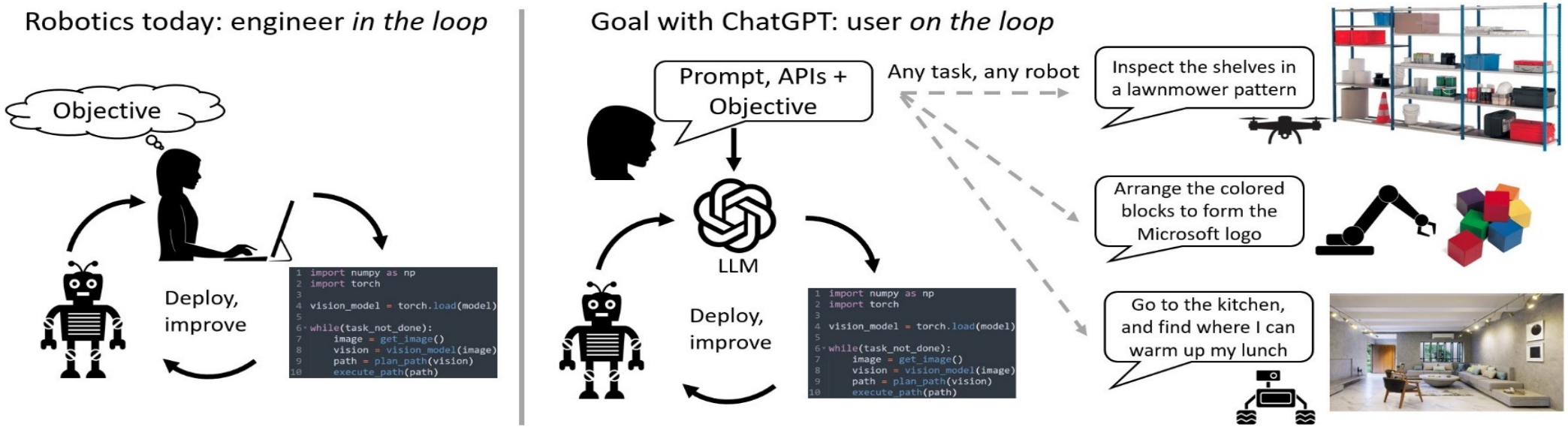


Figure 1: Current robotics pipelines require a specialized engineer in the loop to write code to improve the process. Our goal with ChatGPT is to have a (potentially non-technical) user on the loop, interacting with the language model through high-level language commands, and able to seamlessly deploy various platforms and tasks.

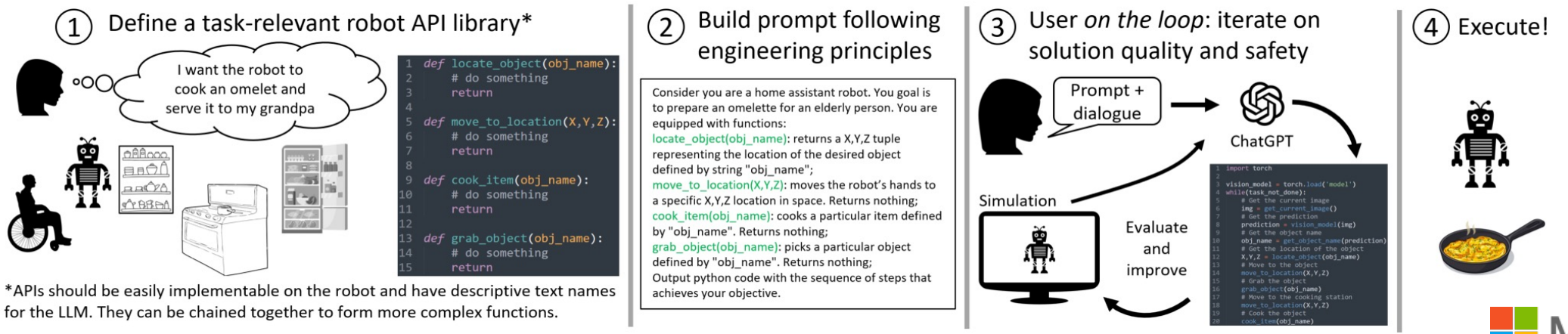
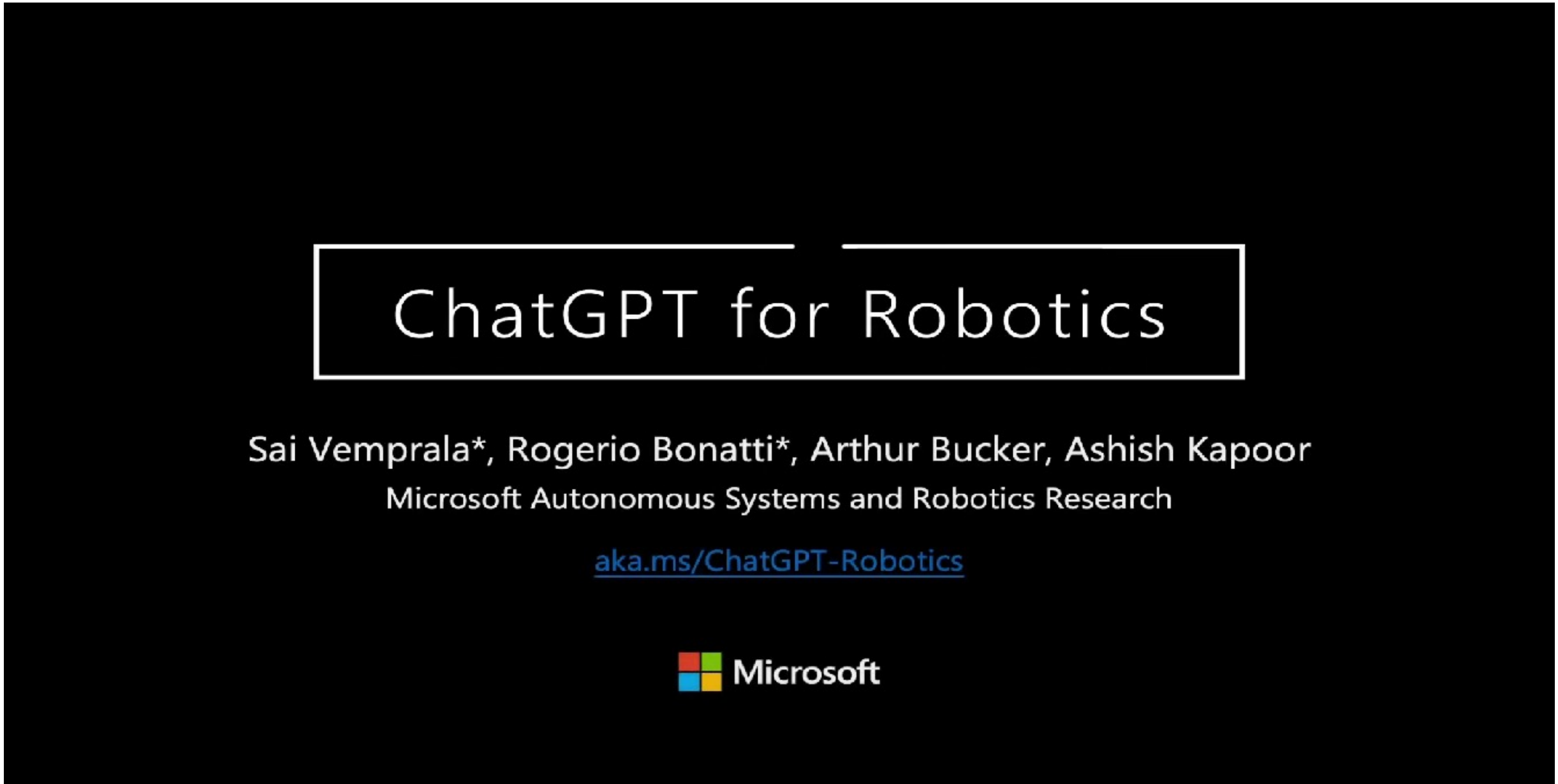


Figure 2: Robotics pipeline employing ChatGPT with the user on the loop to evaluate the output's quality and safety.

Microsoft

ChatGPT for Robotics: Design Principles and Model Abilities



PaLM-E: An Embodied Multimodal Language Model

Robotics at Google

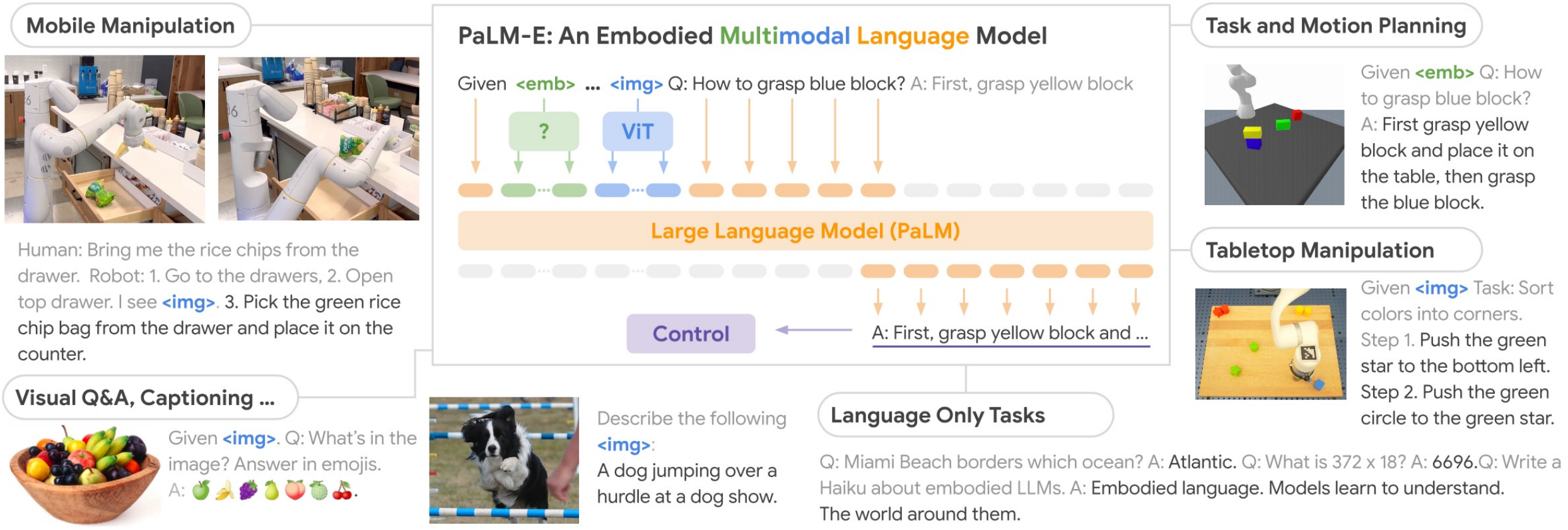
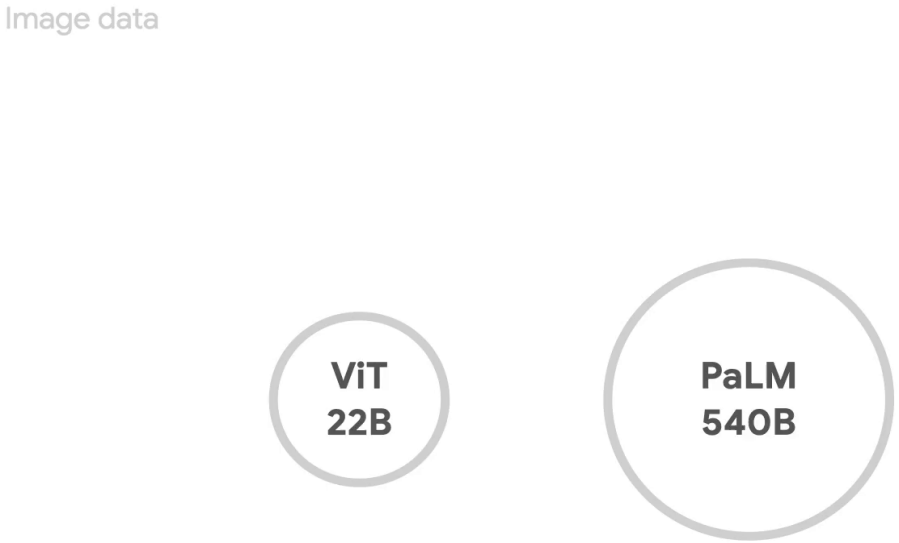


Figure 1: PaLM-E is a single general-purpose multimodal language model for embodied reasoning tasks, visual-language tasks, and language tasks. PaLM-E transfers knowledge from visual-language domains into embodied reasoning – from robot planning in environments with complex dynamics and physical constraints, to answering questions about the observable world. PaLM-E operates on multimodal sentences, i.e. sequences of tokens where inputs from arbitrary modalities (e.g. images, neural 3D representations, or states, in green and blue) are inserted alongside text tokens (in orange) as input to an LLM, trained end-to-end.

PaLM-E: An Embodied Multimodal Language Model



Abstract

Large language models have been demonstrated to perform complex tasks. However, enabling general inference in the real world, e.g. for robotics problems, raises the challenge of grounding. We propose embodied language models to directly incorporate real-world continuous sensor modalities into language models and thereby establish the link between words and percepts. Input to our embodied language model are multi-modal sentences that interleave visual, continuous state estimation, and textual input encodings.

Robotics at Google

This video has audio

Language Models as Zero-Shot Planners:

Extracting Actionable Knowledge for Embodied Agents

Wenlong Huang
UC Berkeley

Pieter Abbeel
UC Berkeley

Deepak Pathak*
CMU

Igor Mordatch*
Google

- Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents, ICML 2022

RT-2: Vision-Language-Action Models, Transfer Web Knowledge to Robotic Control

Google DeepMind

RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control

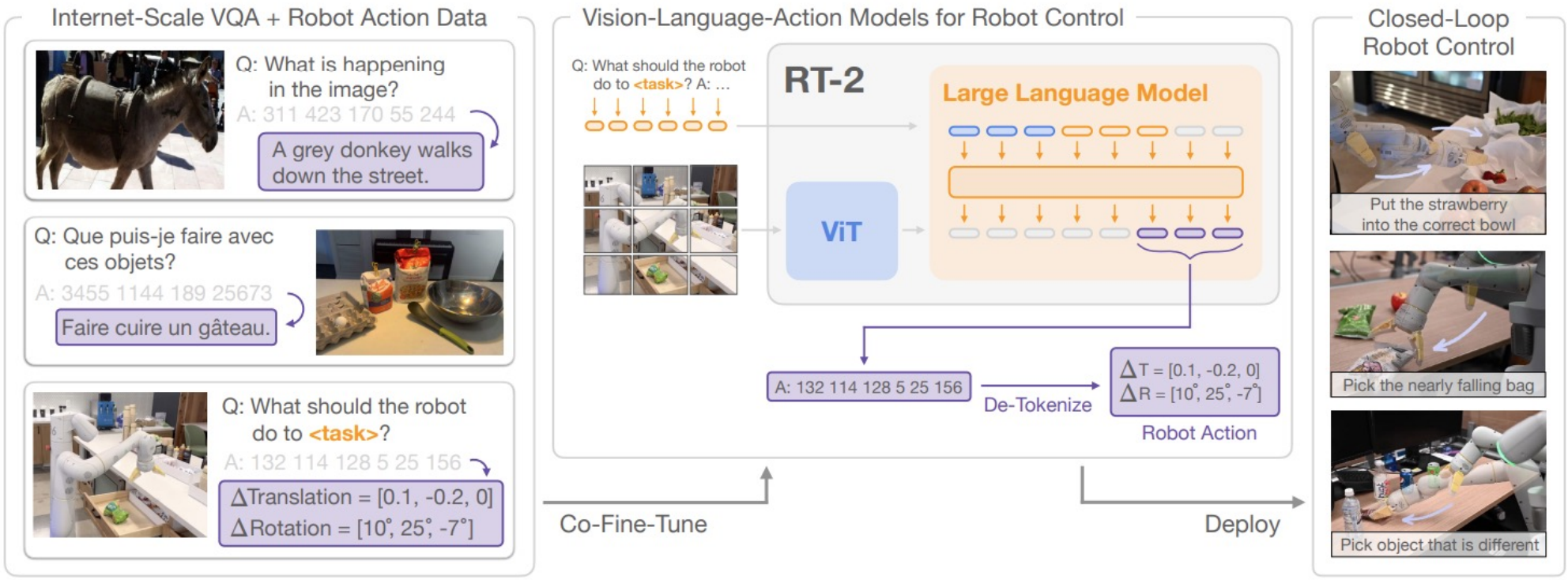
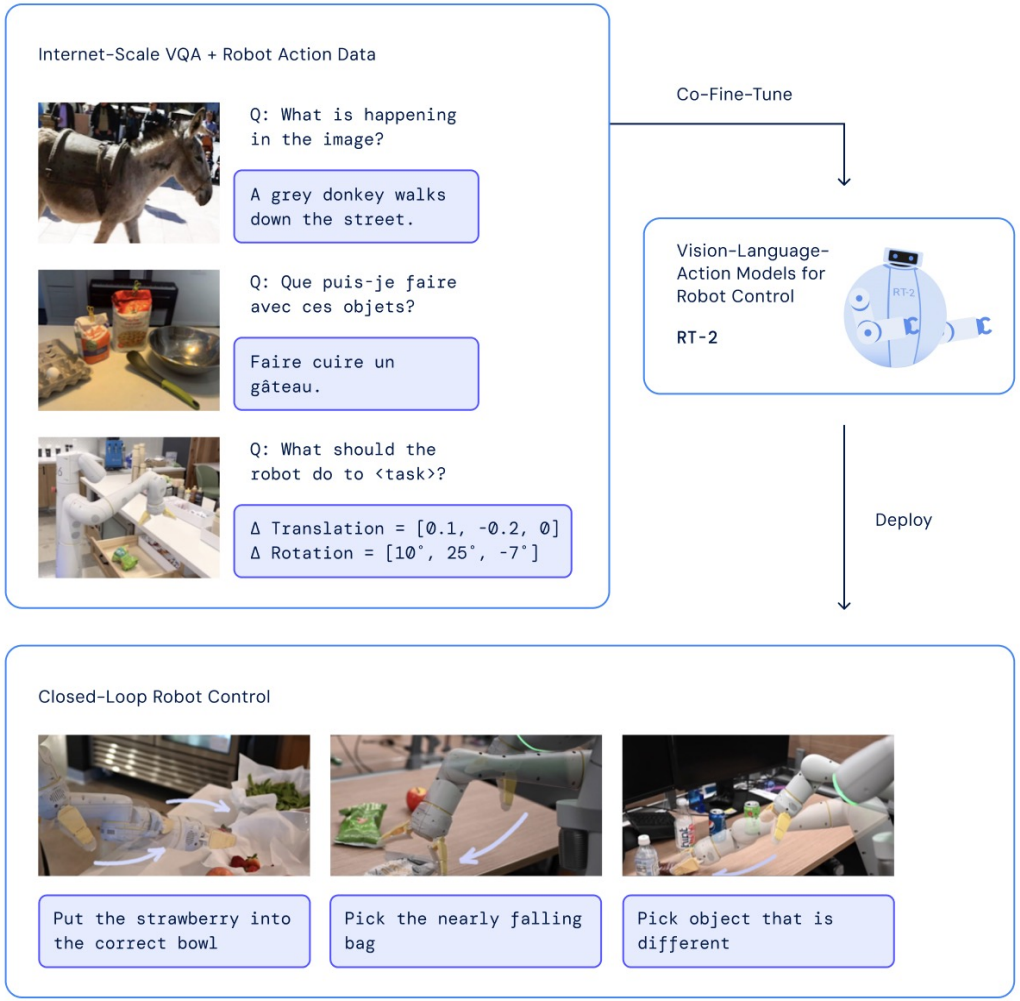


Figure 1 | RT-2 overview: we represent robot actions as another language, which can be cast into text tokens and trained together with Internet-scale vision-language datasets. During inference, the text tokens are de-tokenized into robot actions, enabling closed loop control. This allows us to leverage the backbone and pretraining of vision-language models in learning robotic policies, transferring some of their generalization, semantic understanding, and reasoning to robotic control. We demonstrate examples of RT-2 execution on the project website: robotics-transformer2.github.io.

RT-2: Vision-Language-Action Models, Transfer Web Knowledge to Robotic Control

Google DeepMind

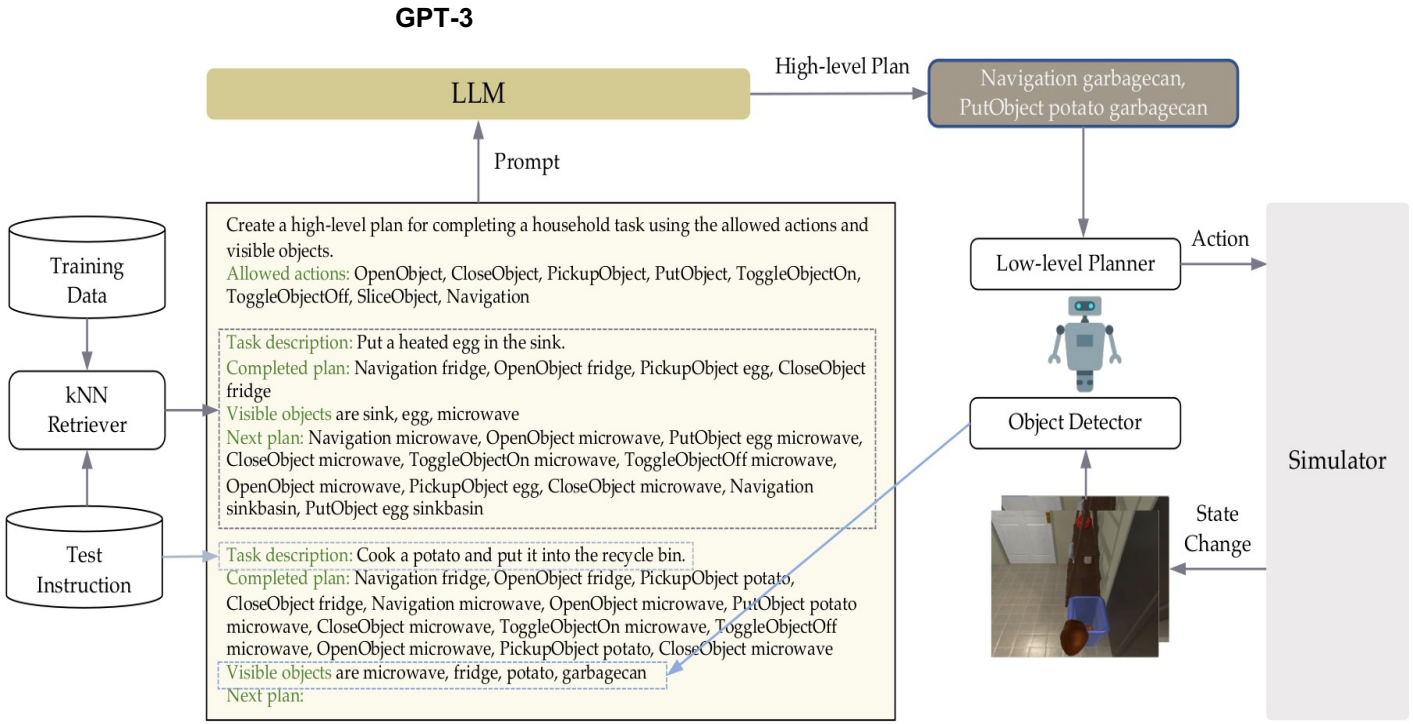


RT-2 architecture and training

- Song C H, Wu J, Washington C, et al.

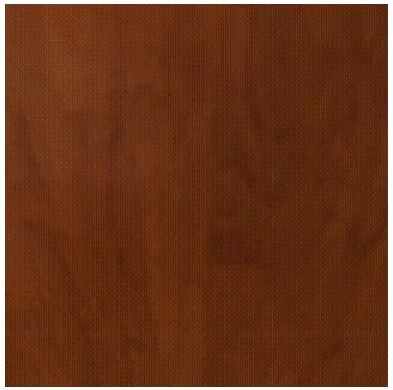
LLM-planner: Few-shot grounded planning for embodied agents with large language models

Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023: 2998-3009.



- Dataset/ Simulator:

Alfred: vision-and-language navigation dataset: https://askforalfred.com/



- Another Embodied AI Simulator: Habitat: https://research.facebook.com/publications/habitat-a-platform-for-embodied-ai-research/

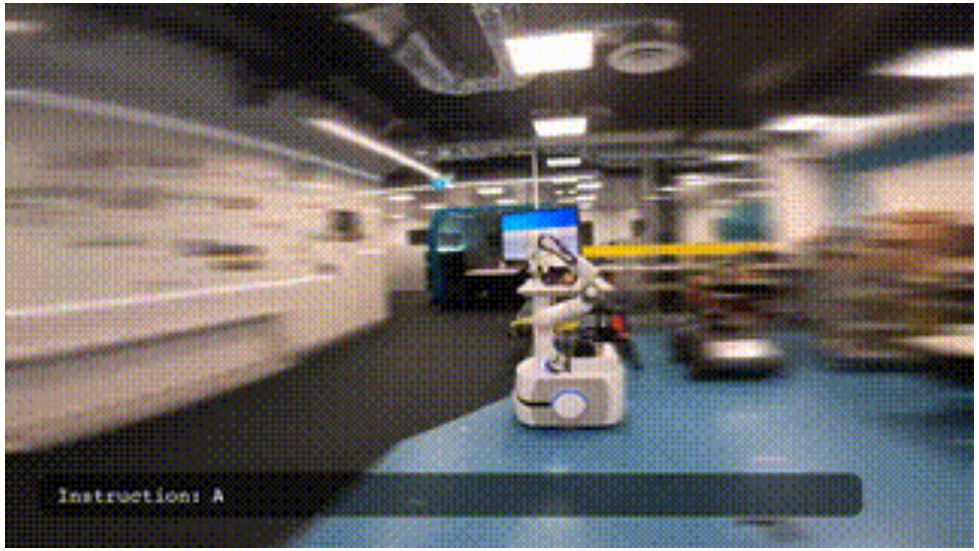
Table V: Robot datasets collected by recent VLAs. VIMA skills refer to "meta-tasks" in their original paper. We use the newer BridgeData V2. PC: point cloud.

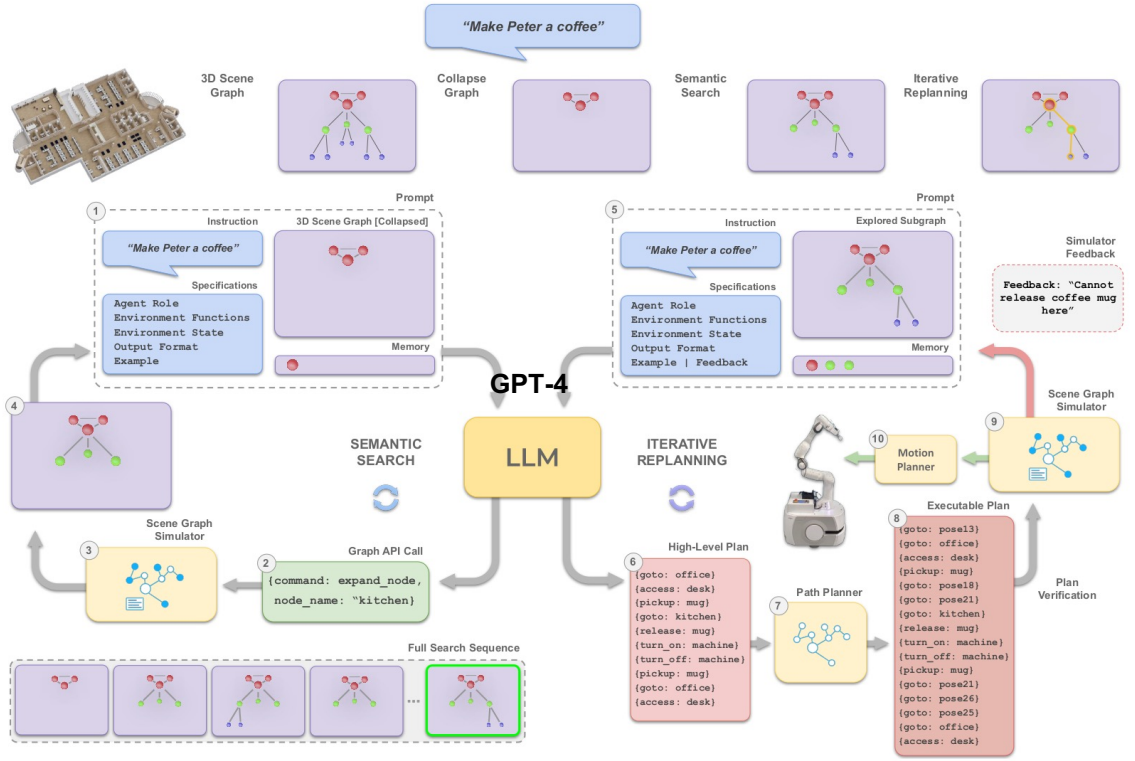
Name	Type	Instruction	Observation	Robot	Skills	Tasks	Objects	Episodes
RoboNet [165]	Real-world	Goal image	RGB	7 embodiments	-	-	-	162K
MT-Opt [71]	Real-world	Lang	RGB	7 embodiments	2	12	-	800K
BC-Z [24]	Real-world	Lang, Demo	RGB	Everyday	9	100	-	25.9K
RT-1 [33]	Real-world	Lang	RGB	Everyday	12	700+	16	130K
MOO [42]	Real-world	Multi-modal	RGB	Everyday	5	-	106	59.1K
VIMA [41]	Simulator	Multi-modal	RGB	UR5	17	-	29	650K
RoboSet [166]	Real-world	Lang	RGB, D	Franka	12	38	-	98.5K
BridgeData [167]	Real-world	Lang	RGB, D	WidowX 250	13	-	100+	60.1K
OXE [37]	Real-world	Lang	RGB, D, PC	22 embodiments	527	160,266	-	1M+

Ma Y, Song Z, Zhuang Y, et al. A Survey on Vision-Language-Action Models for Embodied AI[J]. arXiv preprint arXiv:2405.14093, 2024.

Planning: Grounding large language models using 3d scene graphs for scalable task planning

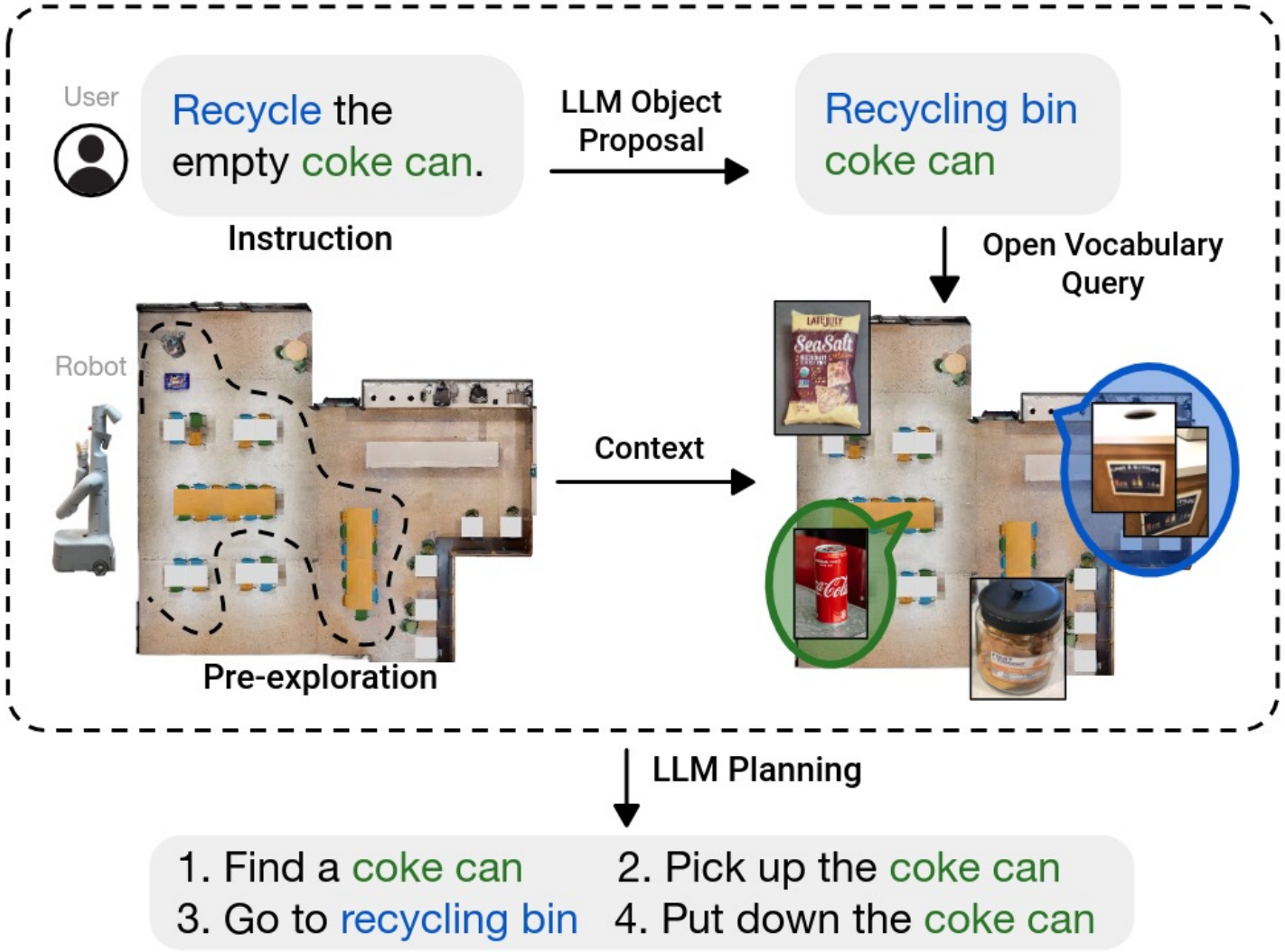
- Rana K, Haviland J, Garg S, et al. Sayplan: Grounding large language models using 3d scene graphs for scalable task planning[J]. arXiv preprint arXiv:2307.06135, 2023.





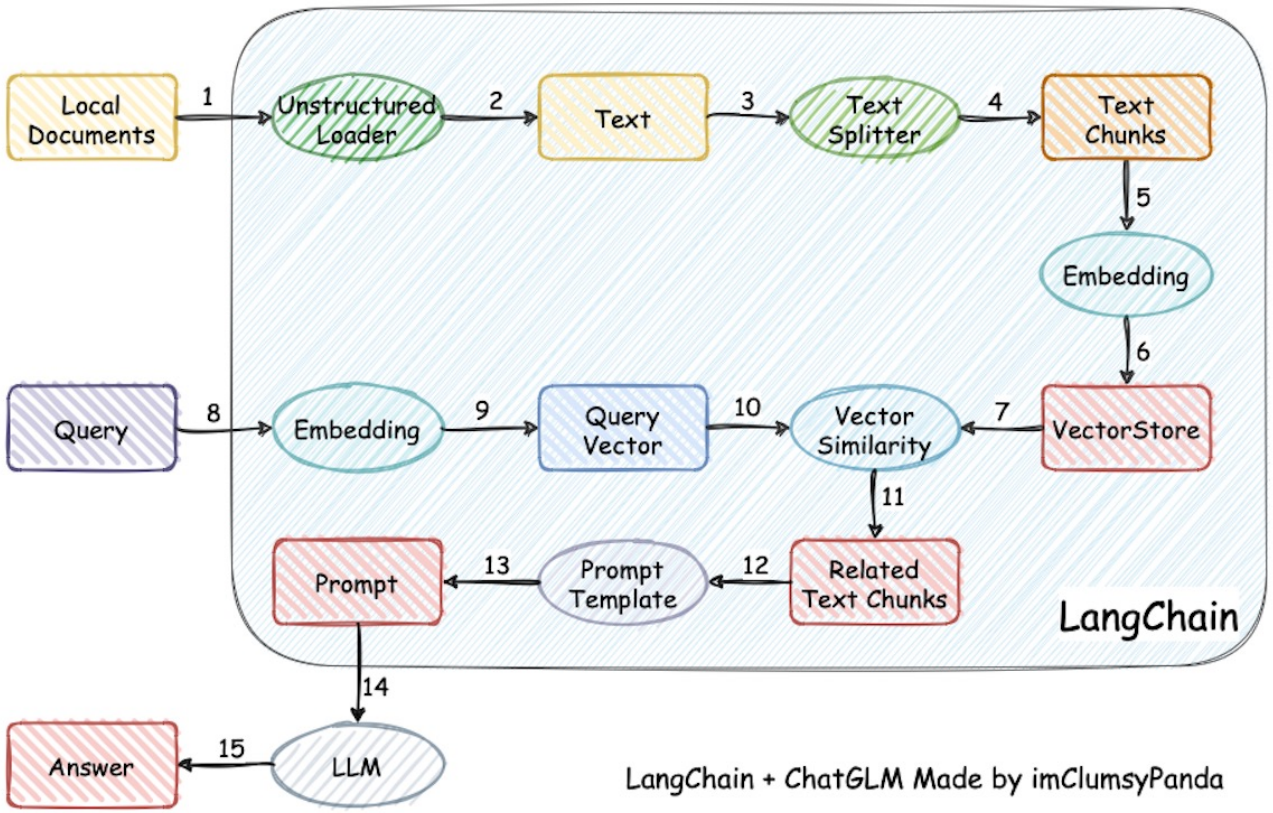
Mission Planning :

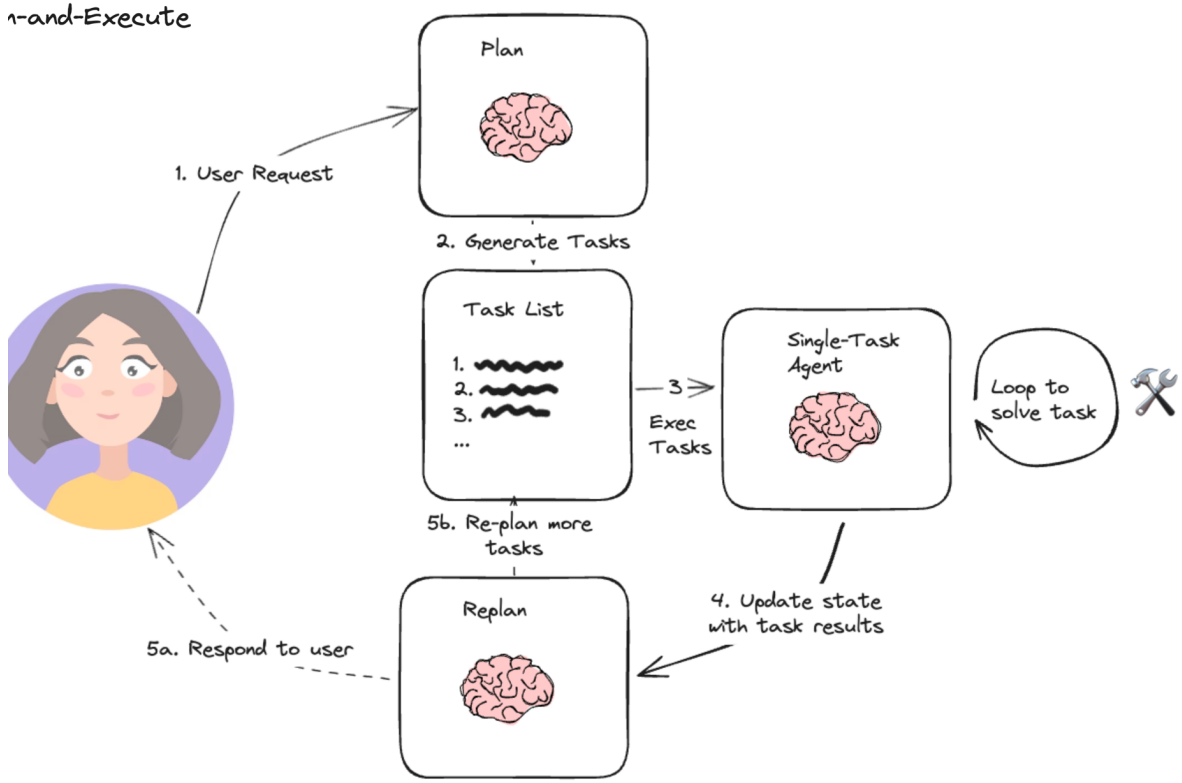
Open-vocabulary queryable scene representations for real world planning



Chen, B., Xia, F., Ichter, B., Rao, K., Gopalakrishnan, K., Ryoo, M. S., ... & Kappler, D. (2023, May). Open-vocabulary queryable scene representations for real world planning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 11509-11522). IEEE.

Retrieval Augmented Generation: LangGraph





https://langchain-ai.github.io/langgraph/

Prompt engineering

- Ask the LLM to provide uncertainty of the prompts:
  - Ren A Z, Dixit A, Bodrova A, et al. Robots that ask for help: Uncertainty alignment for large language model planners[J]. arXiv preprint arXiv:2307.01928, 2023.
  - SaySelf: Teaching LLMs to Express Confidence with Self-Reflective Rationales