Towards Manipulator Task-Oriented Programming: Automating Behavior-Tree Configuration

PhD Final Defense

Yue Cao

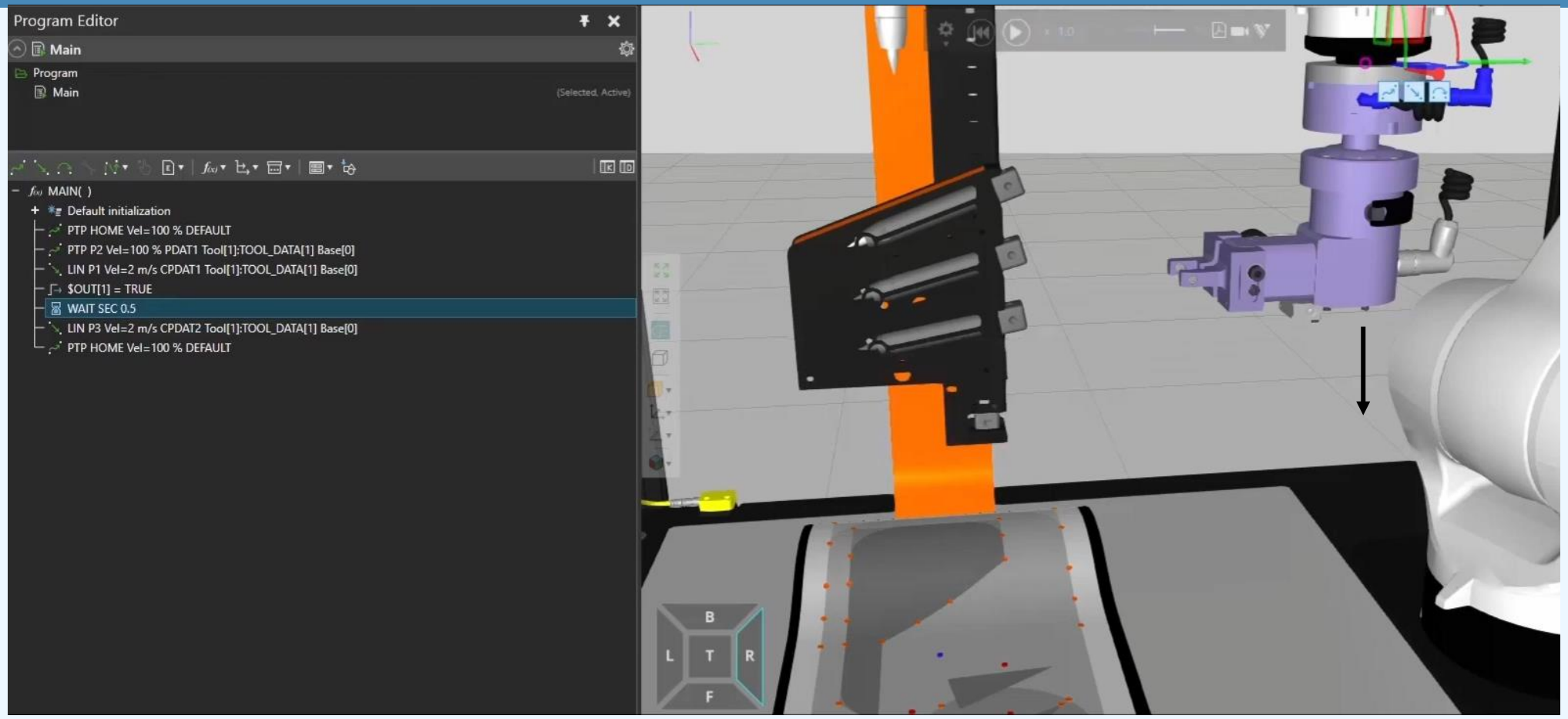
June 24, 2024

Thesis Committee

- C. S. George Lee (Chair)
- Jianghai Hu
- Shreyas Sundaram
- Shaoshuai Mou

ARTLAB
Assistive Robotics Technology

PURDUE UNIVERSITY
Elmore Family School of Electrical and Computer Engineering



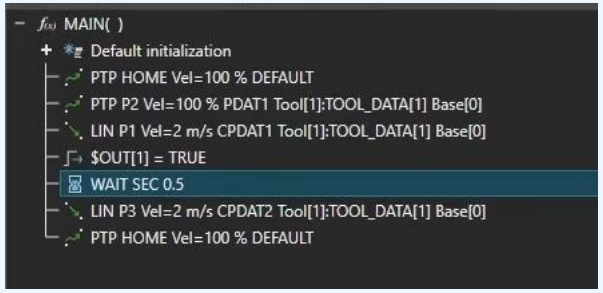
A manipulator programming example: KUKA.Sim offline programming

↓

How about simply using such instruction instead:

Retrieve the cube from the bottom of the rack

Manipulator Programming – Categories



Retrieve the cube from the bottom of the rack

Robot-oriented programming

Program manipulators
in terms of explicit motions

Task-oriented programming

Program manipulators
in terms of high-level tasks

Advantages:

- Towards non-expert end-users
- Transferability among manipulators

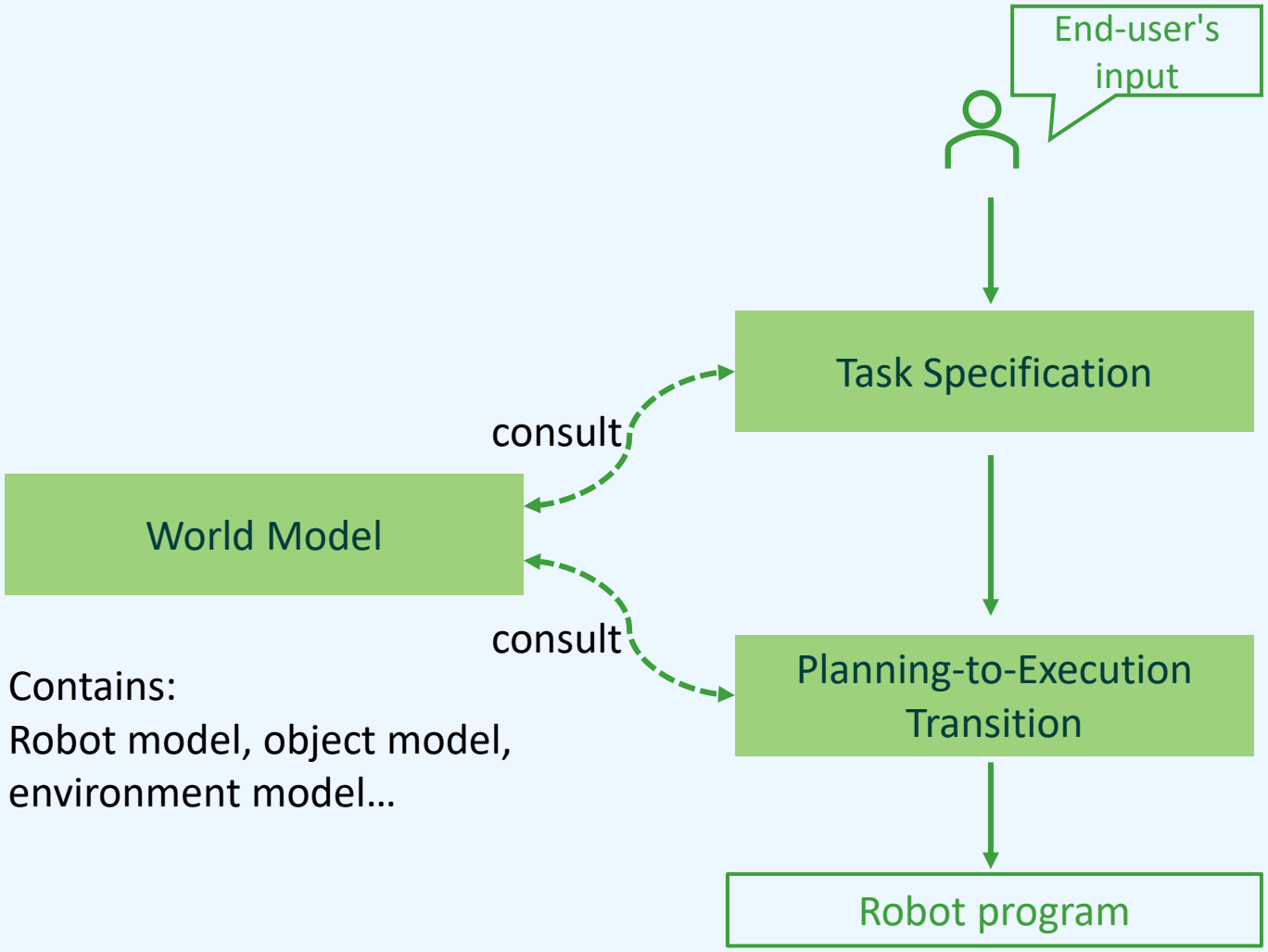
Manipulator Programming – Current Status

- Robot-oriented programming is still used by major robot manufacturers.
- Task-oriented programming is un-achieved in industry practice. [Hägele 2016]
  It is recognized as an important next-step tech advancement. [Sanneman 2021]

ABB Robotics, 2020

KUKA Robotics, 2021

Built-up of the Task-Oriented Programming System



Contains:
Robot model, object model, environment model…

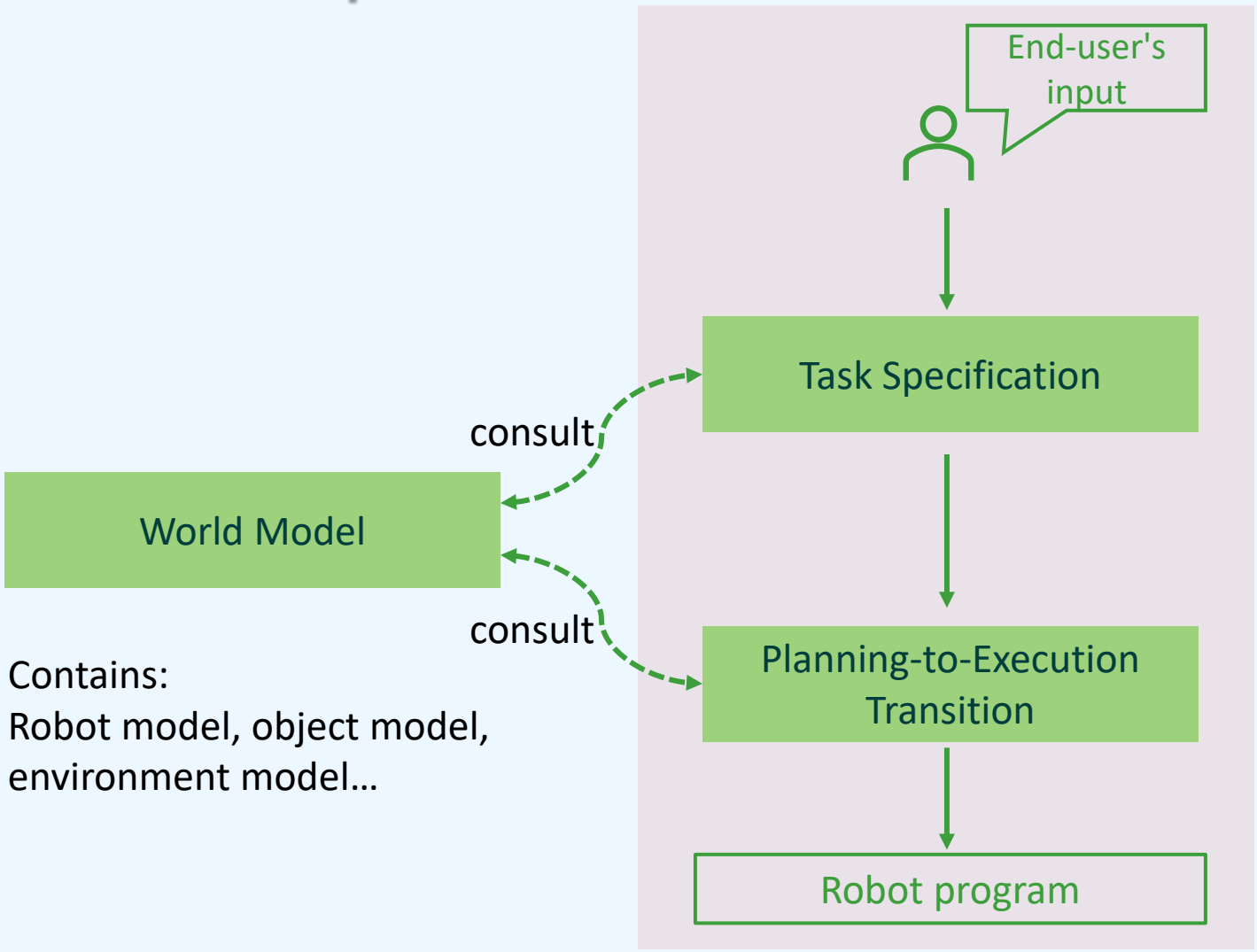
"Clean the classroom floor using vacuum"

-> pick vacuum
-> turn on vacuum
-> move vacuum over floor
-> …

Linear_move (1.30, 1.90, 0.70) m;
Rotate_gripper_z 90 deg;
Open_gripper;
Wait 3 sec;
…

[Lieberman 1977], [Lozano-Pérez 1983], [Fu 1987]

Built-up of the Task-Oriented Programming System



Contains:
Robot model, object model, environment model…

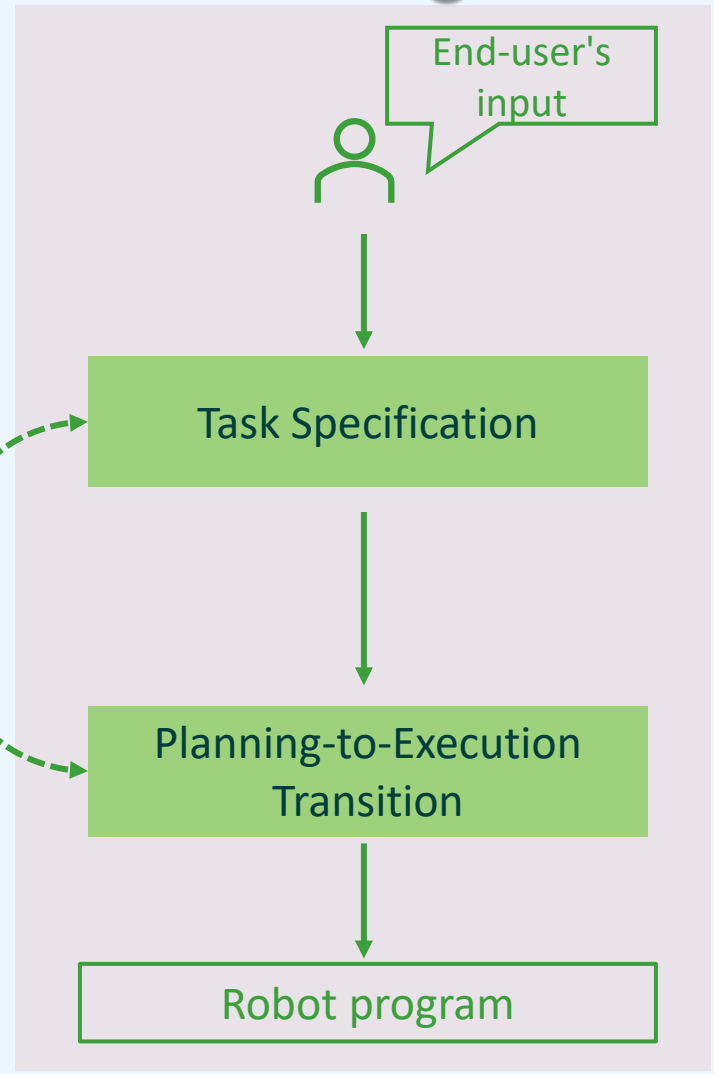
Thesis Focus

"Clean the classroom floor using vacuum"

-> pick vacuum
-> turn on vacuum
-> move vacuum over floor
-> …

Linear_move (1.30, 1.90, 0.70) m;
Rotate_gripper_z 90 deg;
Open_gripper;
Wait 3 sec;
…

Challenge 1: The Missing of Natural Languages



Contains:
Robot model, object model, environment model…

"Clean the classroom floor using vacuum"

-> pick vacuum
-> turn on vacuum
-> move vacuum over floor
-> …

Linear_move (1.30, 1.90, 0.70) m;
Rotate_gripper_z 90 deg;
Open_gripper;
Wait 3 sec;
…

Challenge 1: The Missing of Natural Languages

- Prototypes use English-like statement, not natural languages.

  PLACE … ON … DRIVE IN … INTO … USING… CONTACTS…

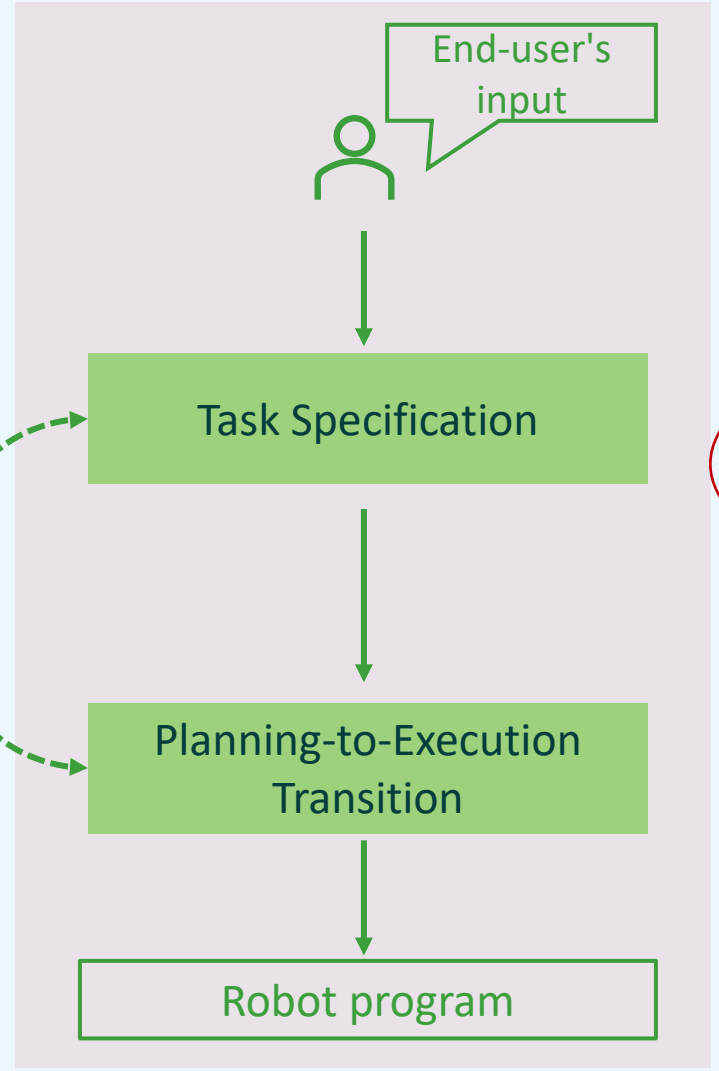
  fixed syntax, limited vocabulary.

- A compromise solution from the historical development.
  "The highest level (natural language input) is deemed infeasible to implement at present"
  [Lieberman 1977]

  "At the highest level one would like to have natural languages as the input...
  However, this level of input is still quite far away." [Fu 1987]

Thesis Goal 1: Natural languages as system input

Challenge 2: The Dilemma of Primitive-Task Libraries



Contains:
Robot model, object model, environment model…

"Clean the classroom floor using vacuum"

-> pick vacuum
-> turn on vacuum
-> move vacuum over floor
-> …

Linear_move (1.30, 1.90, 0.70) m;
Rotate_gripper_z 90 deg;
Open_gripper;
Wait 3 sec;
…

Challenge 2: The Dilemma of Primitive-Task Libraries

- Primitive tasks: fundamental building-blocks.

- Non-uniform: researchers define their own ad-hoc primitive-task libraries.
  Domain-specific: unable to work for all manipulation tasks.

No curly block

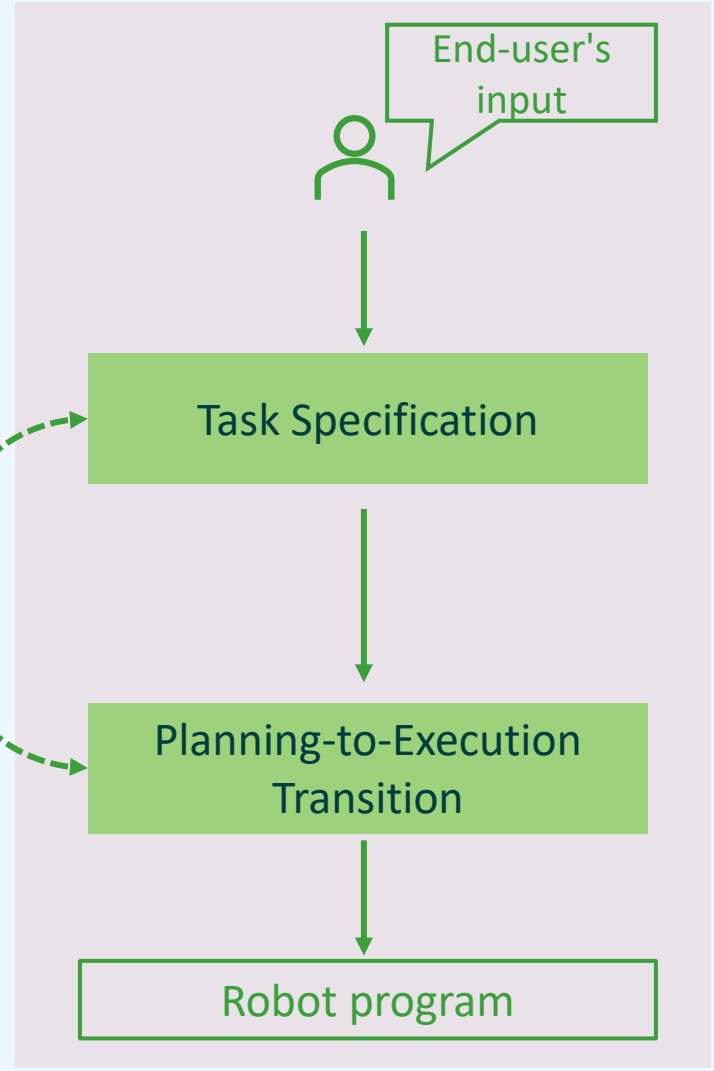
grasp object
release object
move object

No way to open milk bottle

Make latte

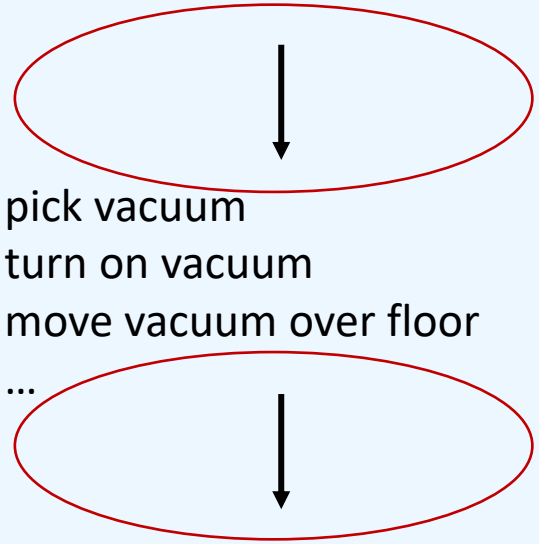
Thesis Goal 2: Go beyond fixed primitive-task libraries

Challenge 3: The “Mechanical Turk” Like System



Contains:
Robot model, object model, environment model…

“Clean the classroom floor using vacuum”



-> pick vacuum
-> turn on vacuum
-> move vacuum over floor
-> …

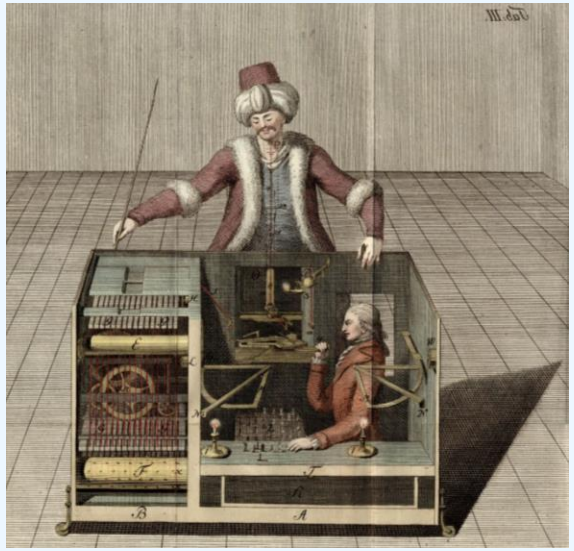
Linear_move (1.30, 1.90, 0.70) m;
Rotate_gripper_z 90 deg;
Open_gripper;
Wait 3 sec;
…

Challenge 3: The “Mechanical Turk” Like System

- Claim to be automatic, but still need human intervention.

- Task specification:
  In limited domains, this module can be automated.
  But in many other domains, it still requires end-users to select primitive tasks and chain them.

- Planning-to-execution transition:
  require experts to design such transition.



Thesis Goal 3: Minimize human intervention

Motivation Summary

Challenge 1: The Missing of Natural Languages

Natural languages as system input

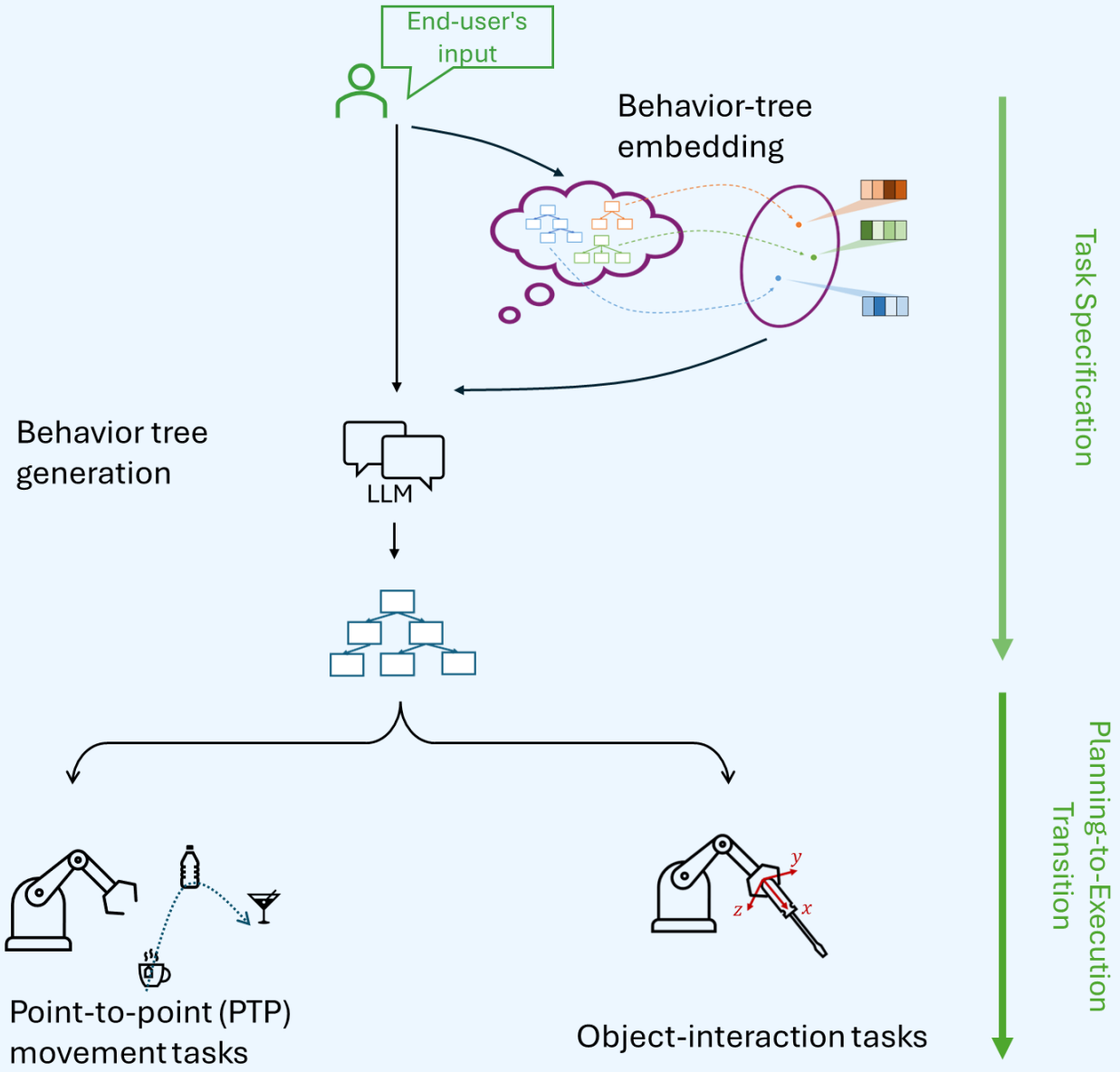
Challenge 2: The Dilemma of Primitive-Task Libraries

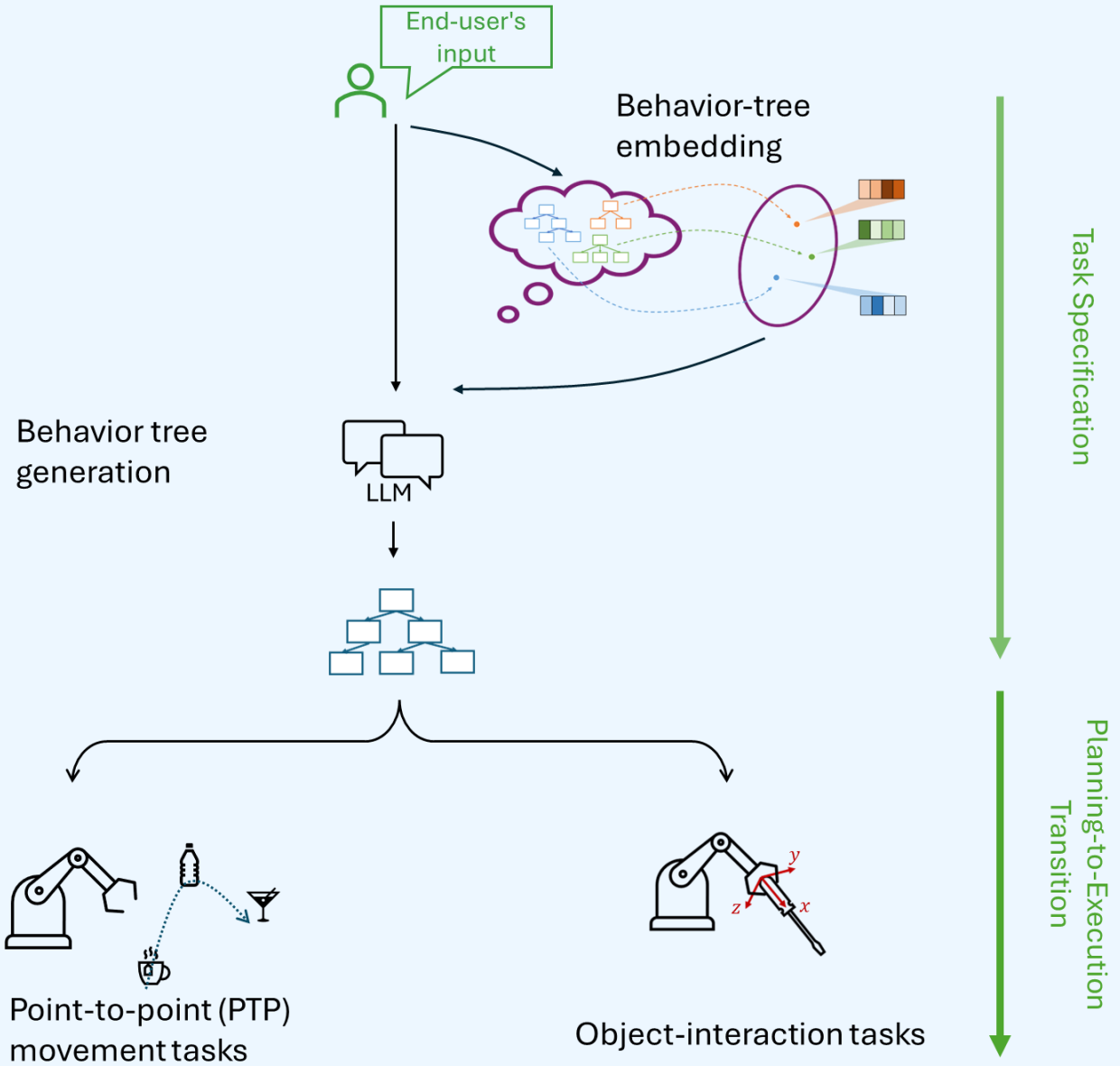
Go beyond fixed primitive-task libraries

Challenge 3: The “Mechanical Turk” Like System

Minimize human intervention

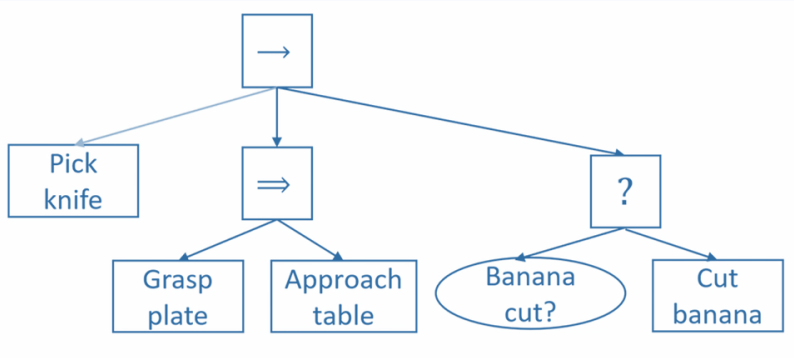
Research Overview



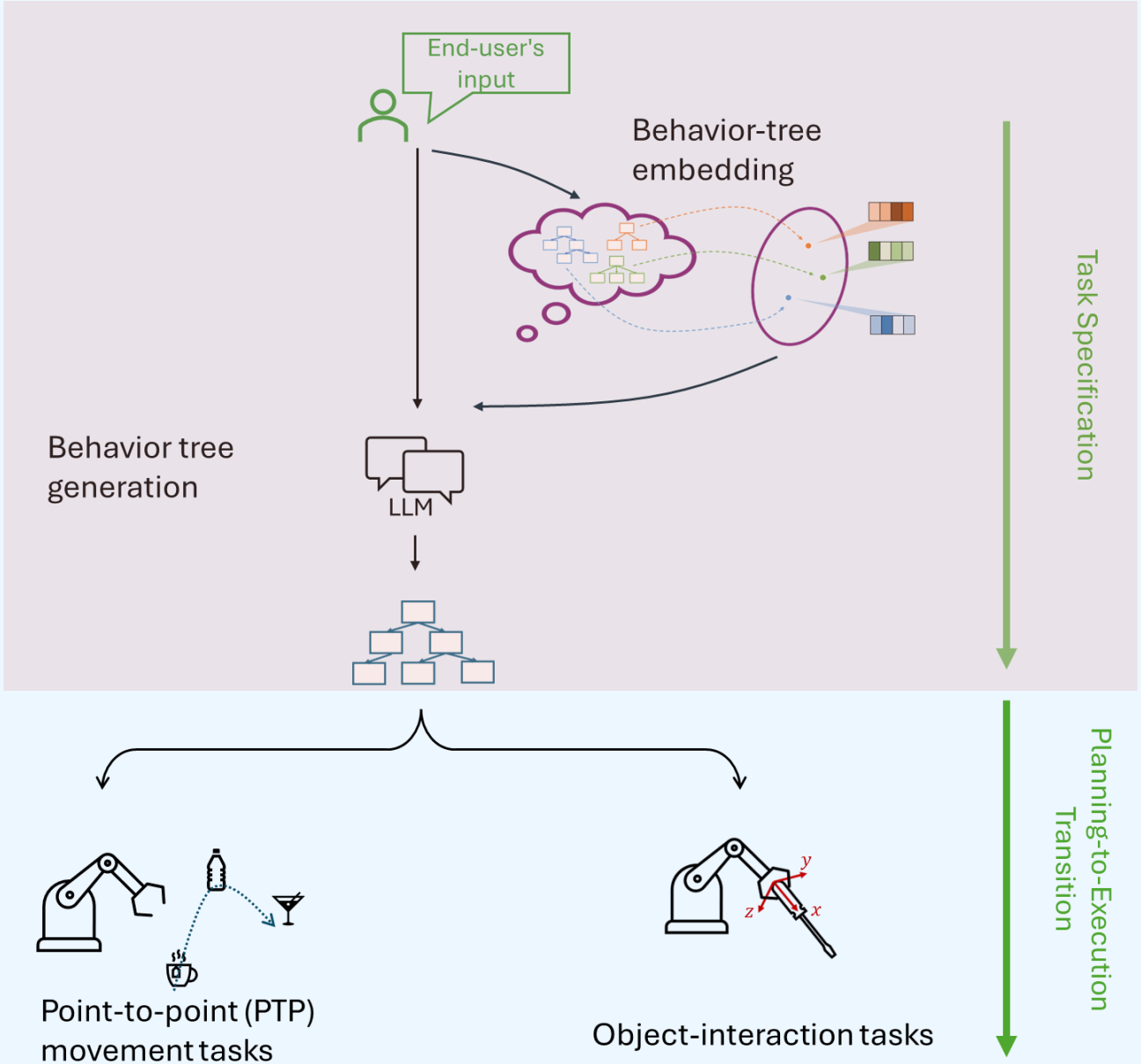


Research Overview

- Behavior tree:
  an advanced representation for task specification.
- Quick intro on behavior tree:

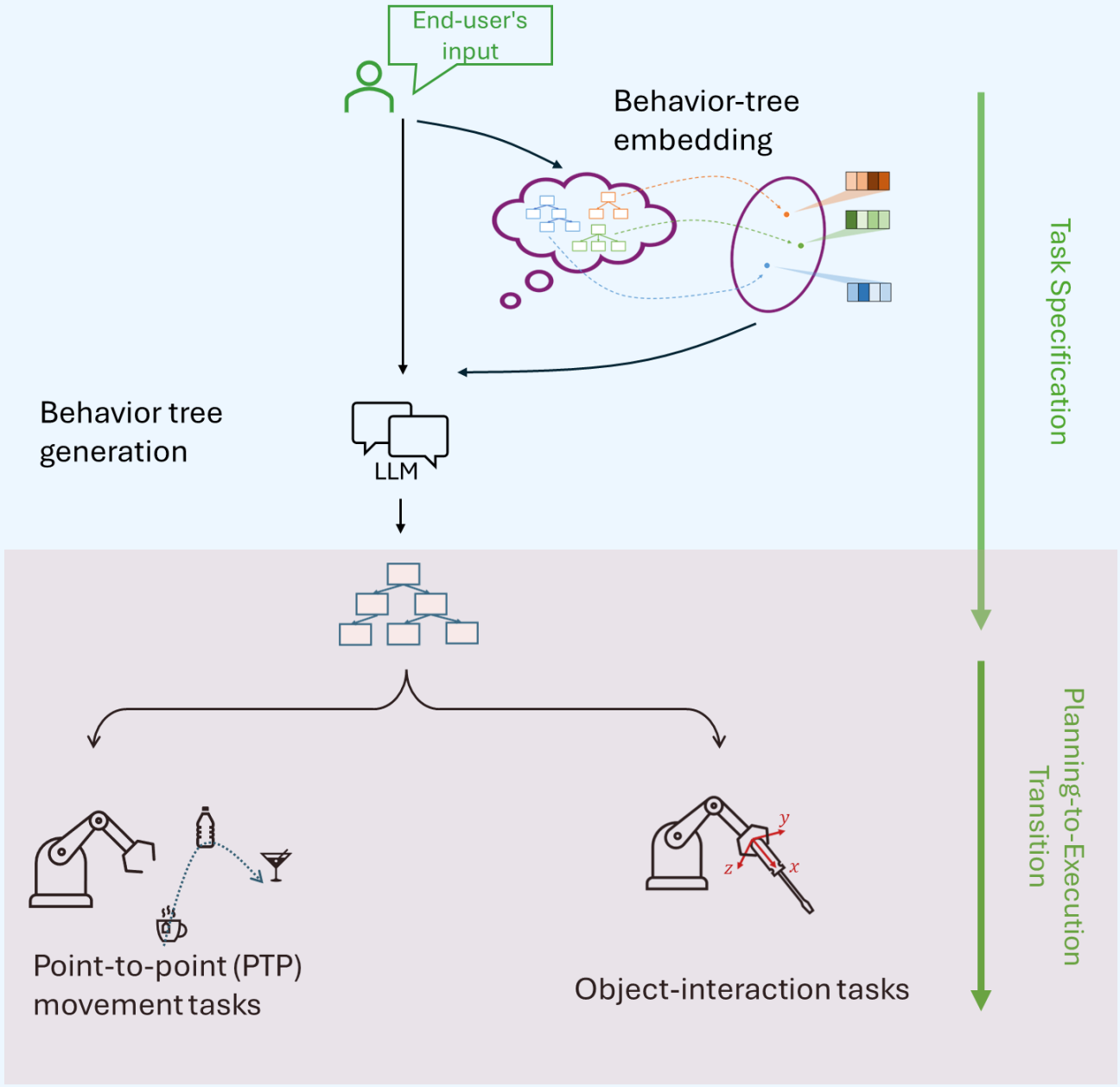


Control-flow nodes		Execution nodes	
Sequence	→	Condition	Condition
Fallback	?	Action	Primitive task
Parallel	⇒		
Decorator	δ		



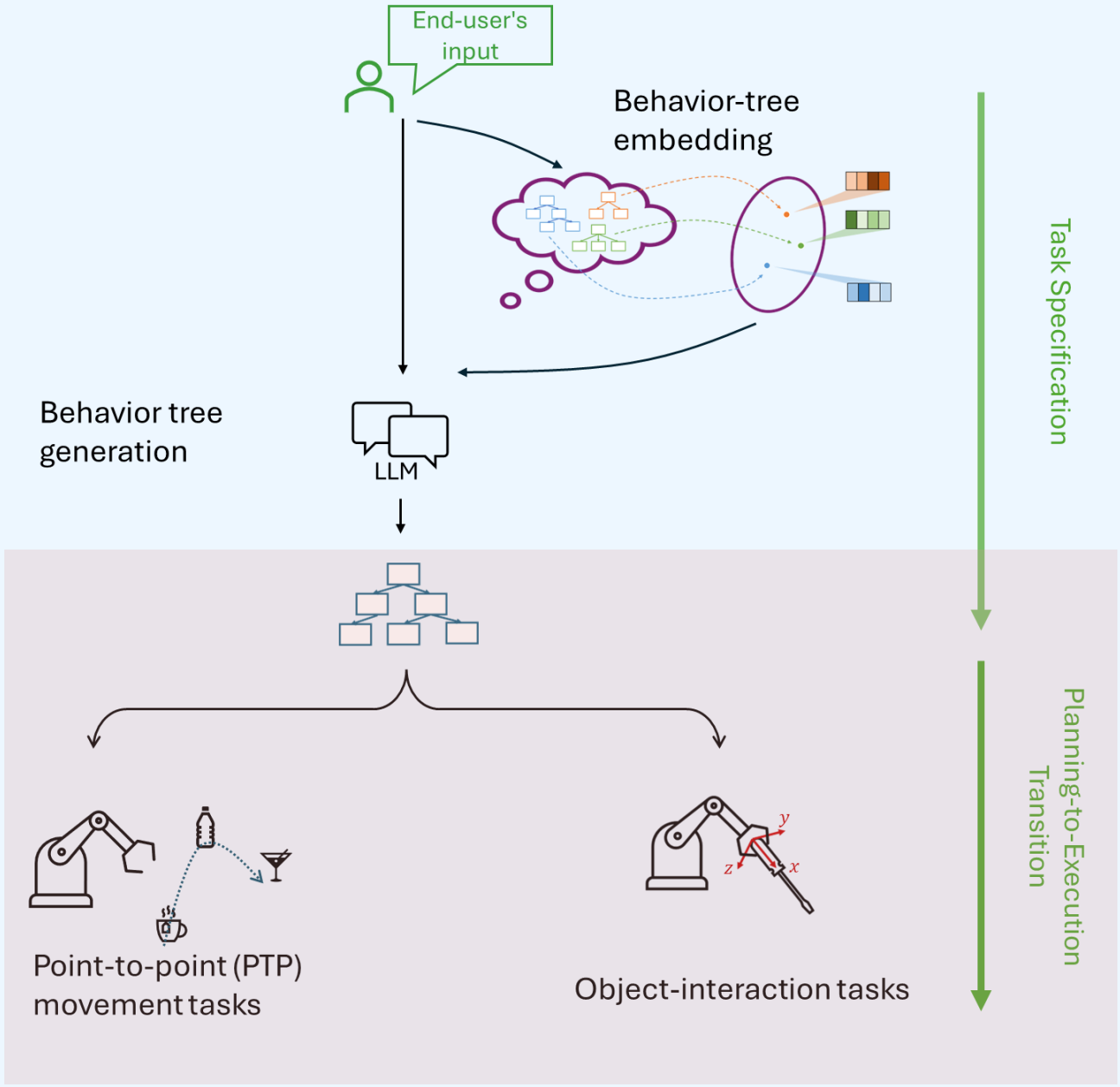
Research Overview

- Part 1: Task Specification
  - LLM-based behavior tree generation
  - Behavior-tree-embedding-augmented generation
- Contributions:
  - Handle natural language input
  - No need for fixed primitive-task libraries
  - Generate into new task domains



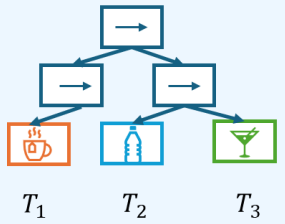
Research Overview

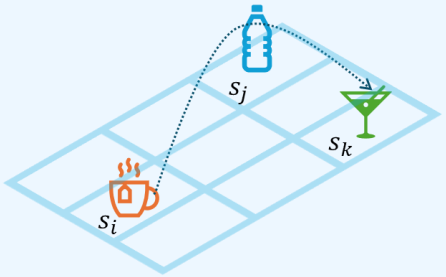
- Part 2: Planning-to-Execution Transition
  - Behavior-rewards for point-to-point (PTP) movement tasks
  - LLM-based approach for object-interaction tasks



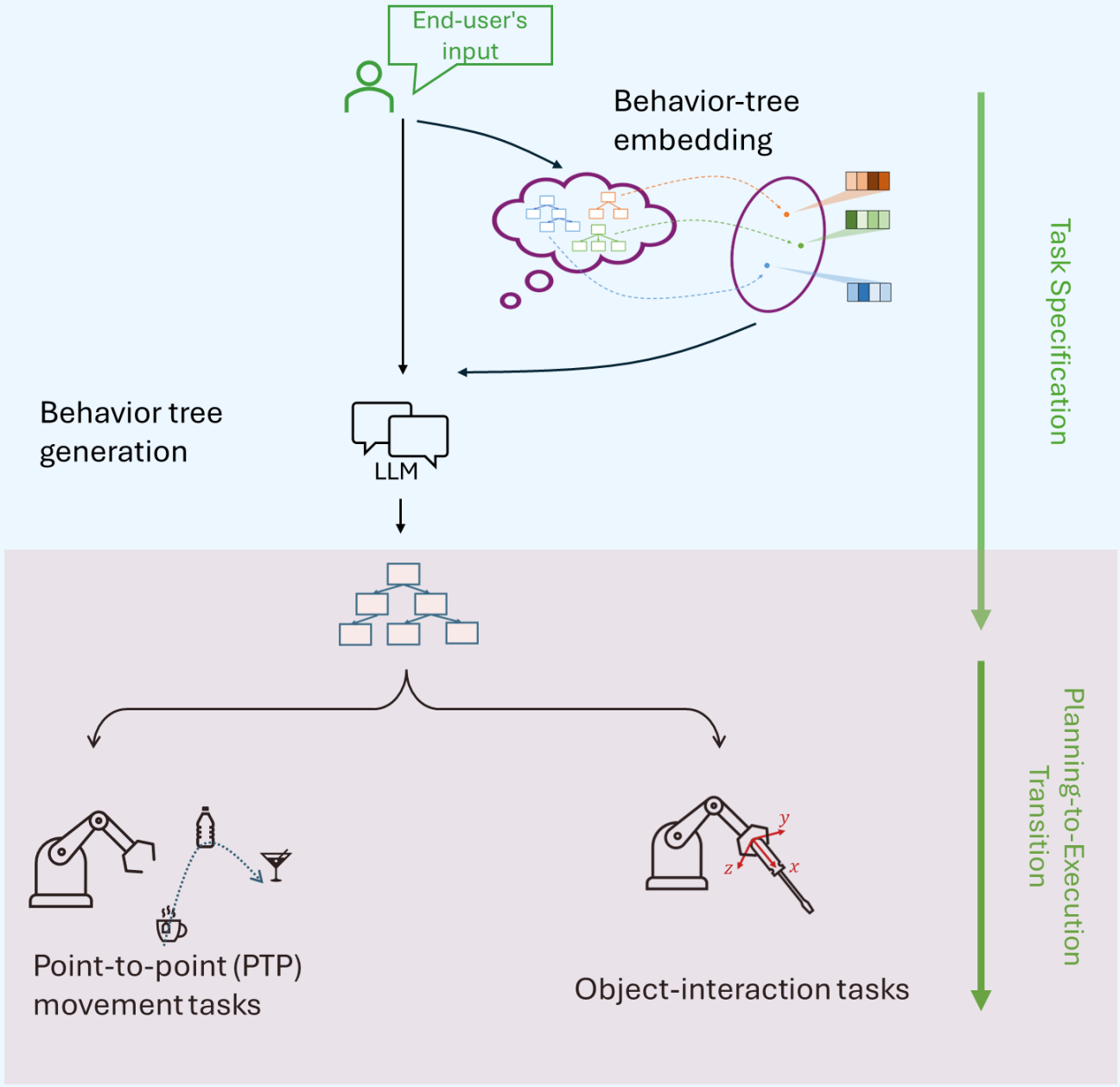
Research Overview

- Part 2: Planning-to-Execution Transition
  - Behavior-rewards for point-to-point (PTP) movement tasks
  - LLM-based approach for object-interaction tasks



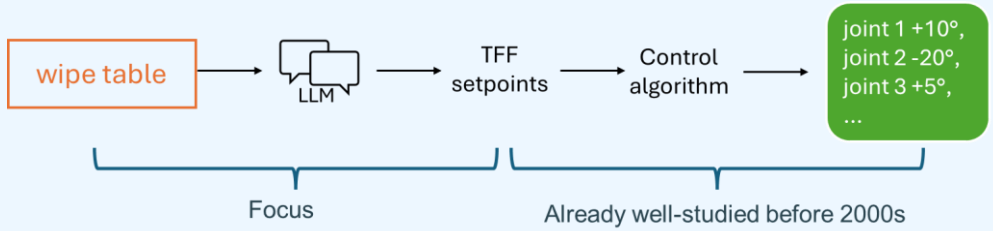


- Contributions:
  - Enable reinforcement learning
  - Simplify end-user's work

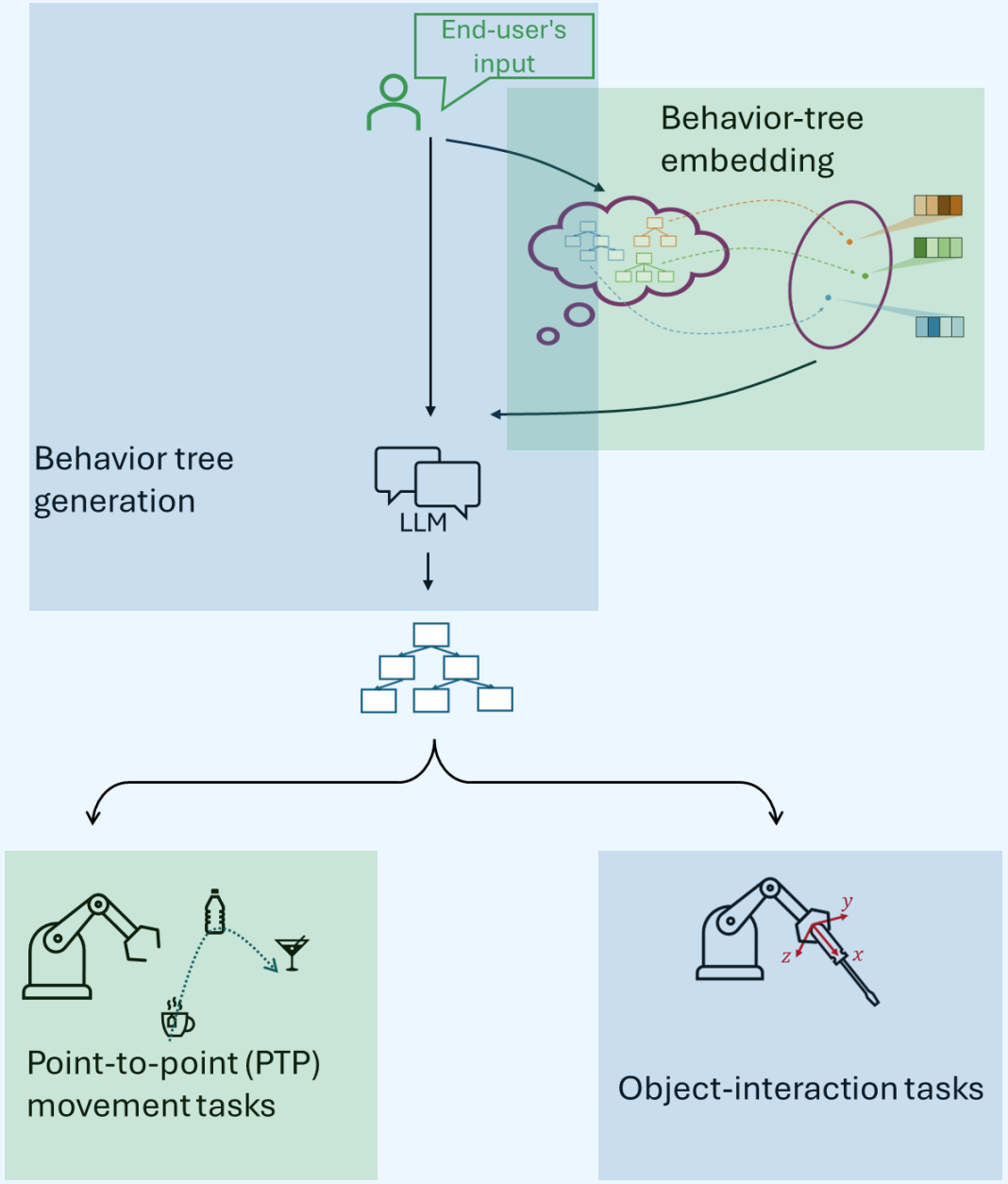


Research Overview

- Part 2: Planning-to-Execution Transition
  - Behavior-rewards for point-to-point (PTP) movement tasks
  - LLM-based approach for object-interaction tasks



- Contributions:
  - Automatic conversion
  - No need for fixed primitive-task libraries



Research Progress

- Prelim (July 2022)
  Title: Machine Learning Empowered Behavior Trees for Robot Tasks

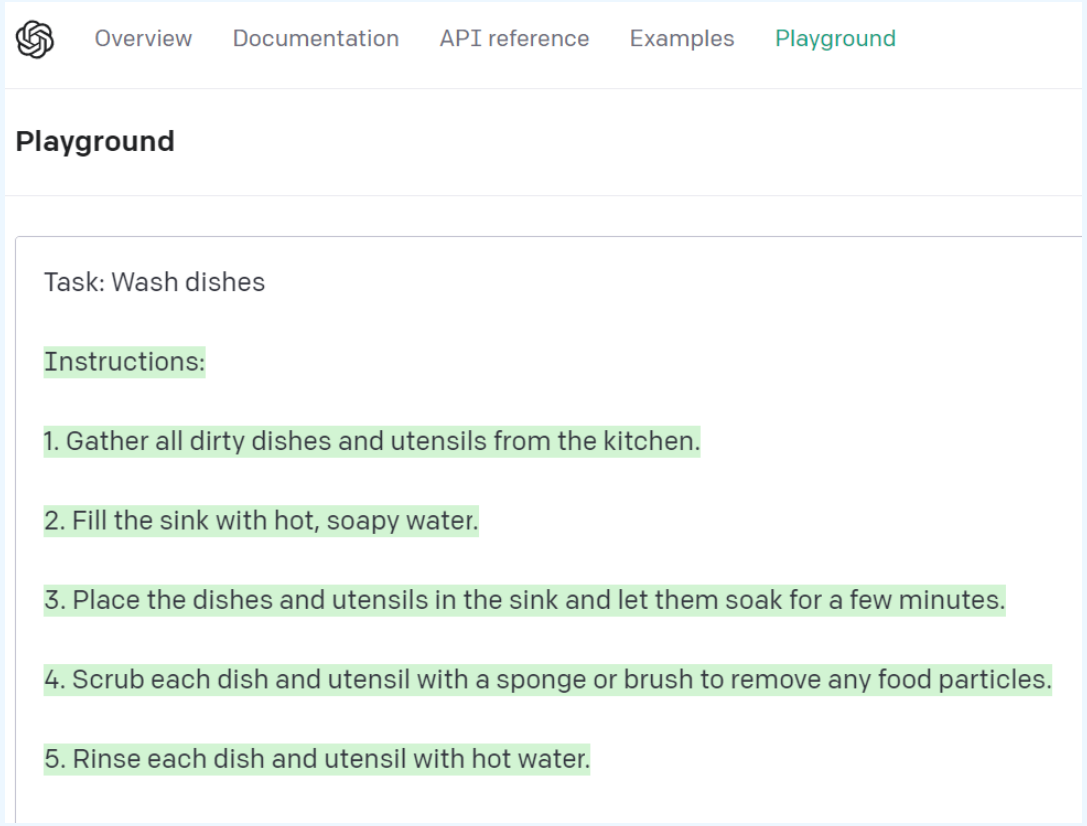
  Works in green

- Final (June 2024)

  Works in blue

* ChatGPT was release in November 2022.

Part 1: Task Specification

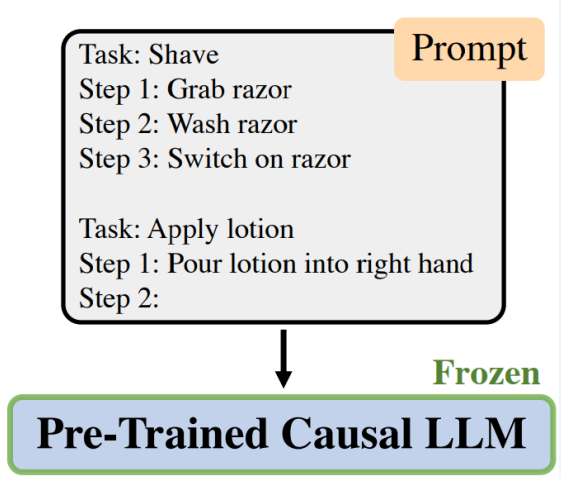


GPT can decompose a high-level task into a sequence of sub-tasks.

LLM for Task Generation

Work 1. LLM Planner

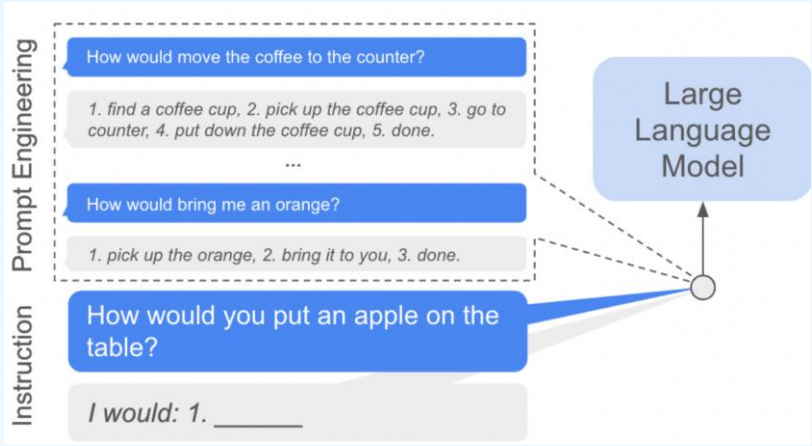
Published in: Jan 2022
LLM: GPT-3



[Huang 2022]

Work 2. SayCan

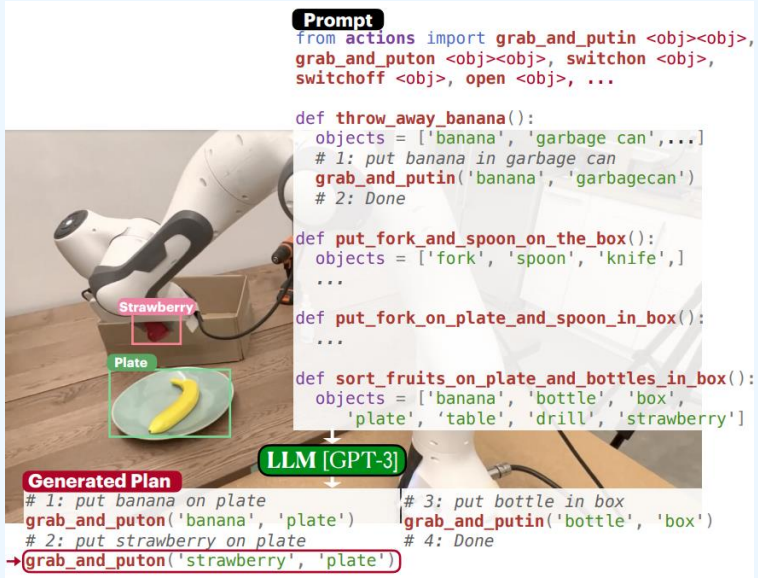
Published in: Apr 2022
LLM: Google PaLM



[Ahn 2022]

Work 3. ProgPrompt

Published in: Sep 2022
LLM: GPT-3



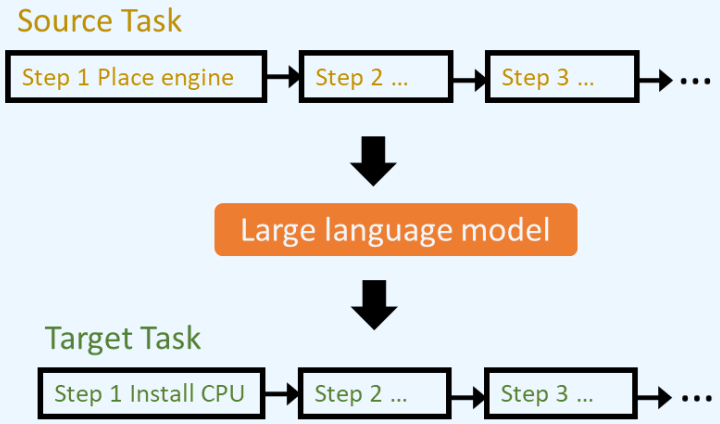
[Singh 2022]

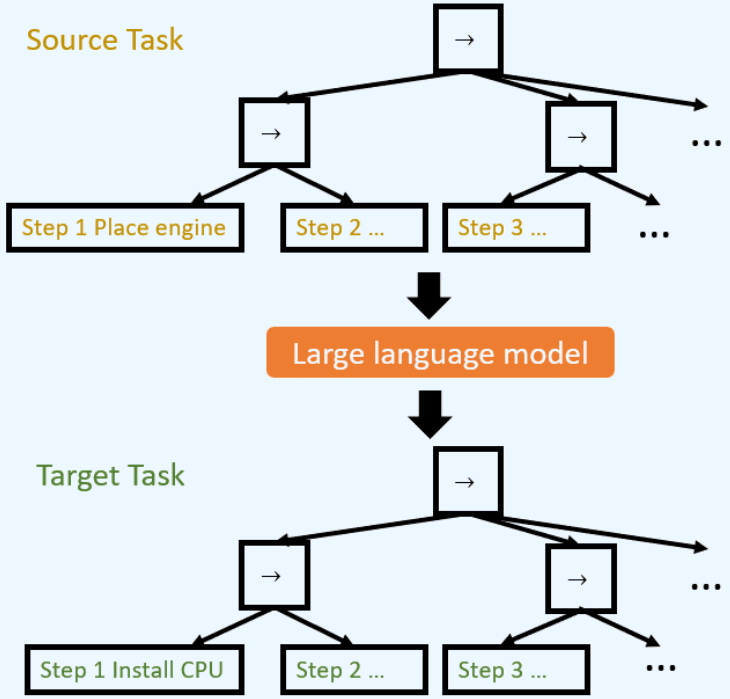
These 3 pictures are captured from the referred papers.

LLM for Task Generation

Previous works:
Generated task are sequential

Can we generate modular tasks, such as behavior trees?
Better reusability and readability



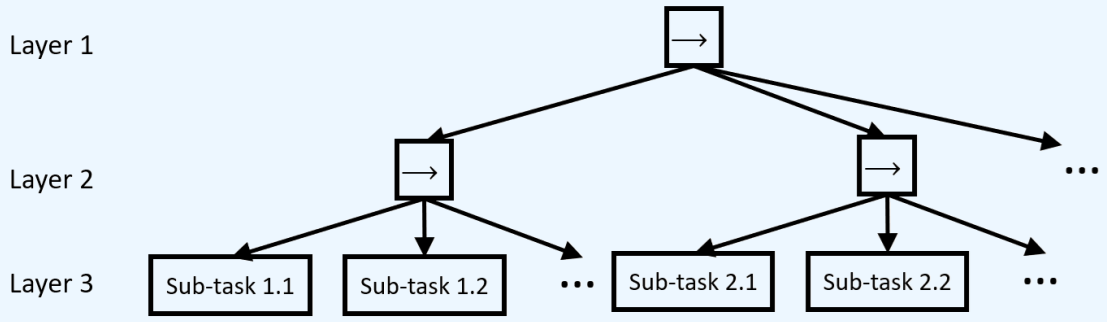


Yue Cao and C. S. George Lee, "Robot Behavior-Tree-Based Task Generation with Large Language Models", AAAI Spring Symposium, 2023

LLM for Behavior Tree Generation

In LLMs, the prompt guides the text generation.

Prompt: text that converts the original input into a template string, leaving two types of slots unfilled: input [ ]$_X$ and output [ ]$_Y$



A sample behavior tree

Prompt Design: Phase-Step Prompt

Source Task
Procedures:
Phase 1.
Step 1. [Sub-task 1.1] $_X$; Step 2. [Sub-task 1.2] $_X$; ...
Phase 2.
Step 1. [Sub-task 2.1] $_X$; Step 2. [Sub-task 2.2] $_X$; ...
...
Target Task: Task Description
Procedures:
[Phase 1.
Step 1. Sub-task ?; Step 2. Sub-task ?; ...
Phase 2.
Step 1. Sub-task ?; Step 2. Sub-task ?; ...
...] $_Y$

Layer 2 node: Phase; Layer 3 node: Step

Prompt Evaluations

- Can LLMs generate modular tasks without our Phase-Step prompt?
- Rarely.

without prompt →

	GPT-3 text-davinci-003		ChatGPT	
	Avg. R	Avg. N_{total}	Avg. R	Avg. N_{total}
PS-none prompt	0.12	5.67	0.22	6.90
PS-wheel prompt	0.65	7.80	0.60	9.77
PS-desktop prompt	0.93	8.80	0.66	9.13

- Does the Phase-Step prompt affect the quality of generated tasks?
- Yes. Prompts containing more details tend to generate more informative tasks.

more detailed prompt →

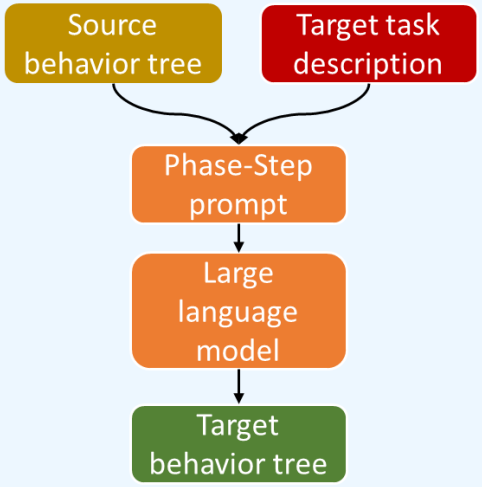
	GPT-3 text-davinci-003		ChatGPT	
	Avg. N_{mate}	Avg. N_{total}	Avg. N_{mate}	Avg. N_{total}
PS-wheel prompt	1.87	7.80	3.20	10.80
PS-desktop prompt	5.33	8.87	5.73	10.00

Tired of Setting Prompt?

Source Task
Procedures:
Phase 1.
Step 1. Put car at a conveyor;
Step 2. Lift the car.
Phase 2.
Step 1. Pick the wheel;
Step 2. Approach conveyor;
Step 3. Align wheel with wheel hub.
Phase 3.
Step 1. Insert screws;
Step 2. Fasten screws;
Step 3. Leave the conveyor.

Target Task: Desktop assembly
Procedures:
To be generated …

Standard way



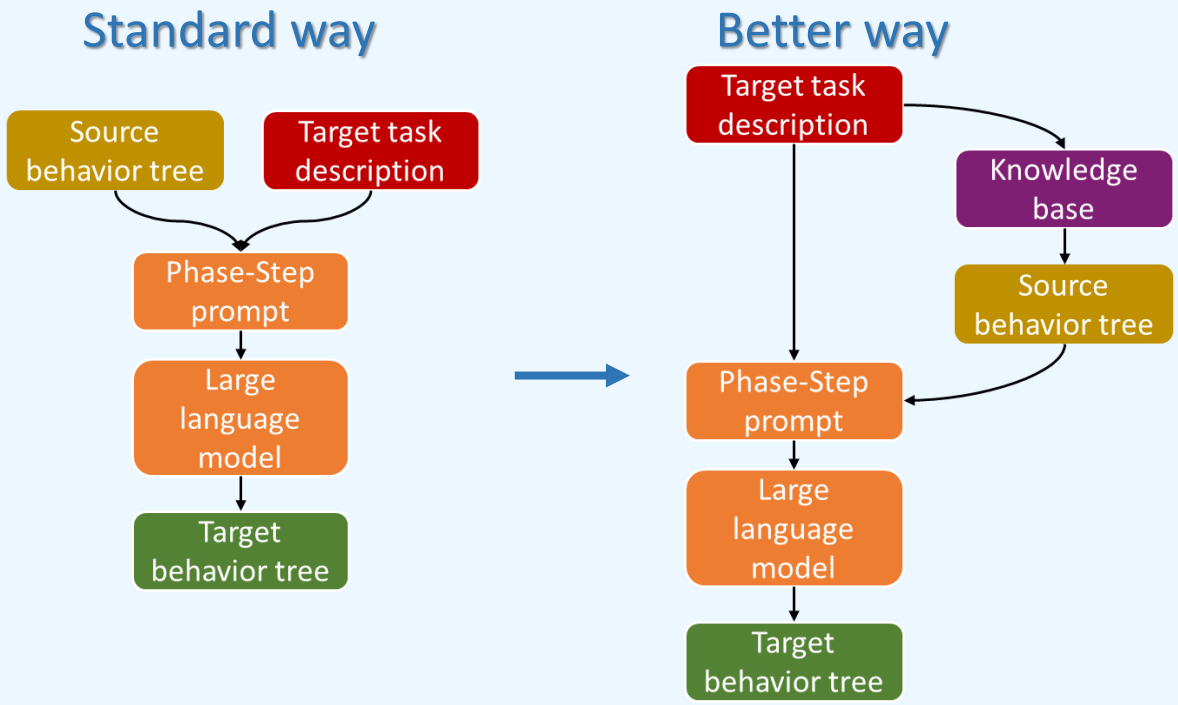
Tired of Setting Prompt?

no need to write this part now

end-user just inputs this line

Source Task
Procedures:
Phase 1.
Step 1. Put car at a conveyor;
Step 2. Lift the car.
Phase 2.
Step 1. Pick the wheel;
Step 2. Approach conveyor;
Step 3. Align wheel with wheel hub.
Phase 3.
Step 1. Insert screws;
Step 2. Fasten screws;
Step 3. Leave the conveyor.

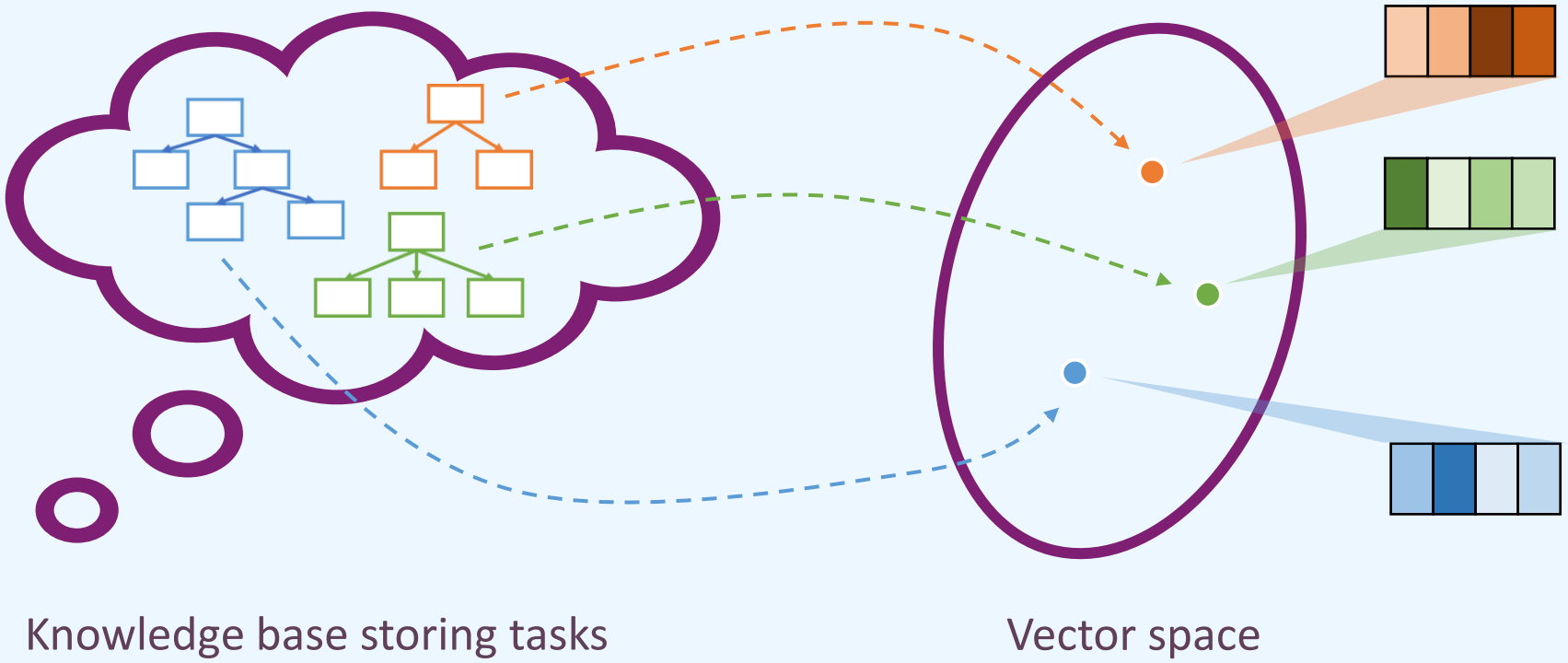
Target Task: Desktop assembly
Procedures:
To be generated …



Retrieval-Augmented Generation (RAG)

Knowledge Base Retrieval (In Prelim)

- Vector Embeddings for behavior-tree tasks



Knowledge base storing tasks

Vector space

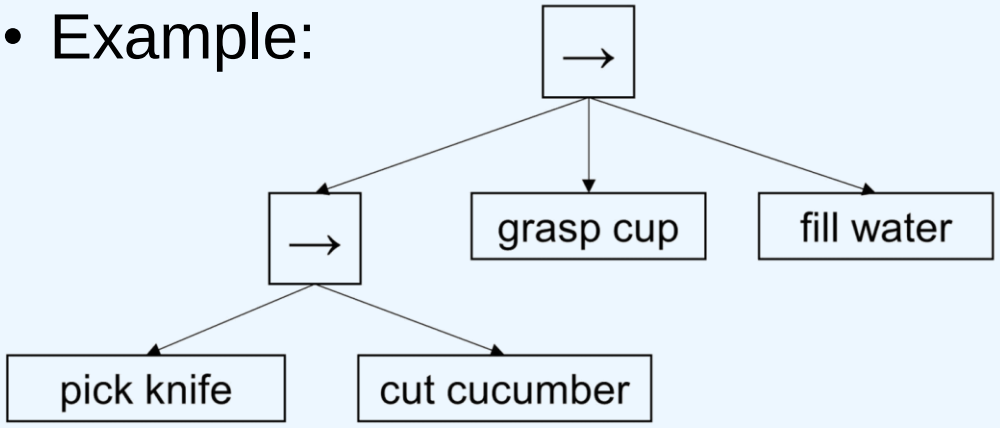
Yue Cao and C. S. George Lee, “Behavior-Tree Embeddings for Robot Task-Level Knowledge”, IROS, 2022

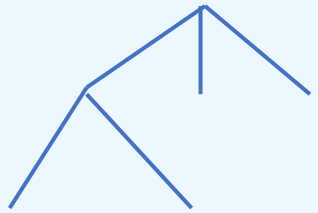
Behavior-Tree Embedding

- Principle: In the vector space, similar tasks should be close, while distinct tasks should be far away.

→ Define similarity

- Example:





Structural characteristics

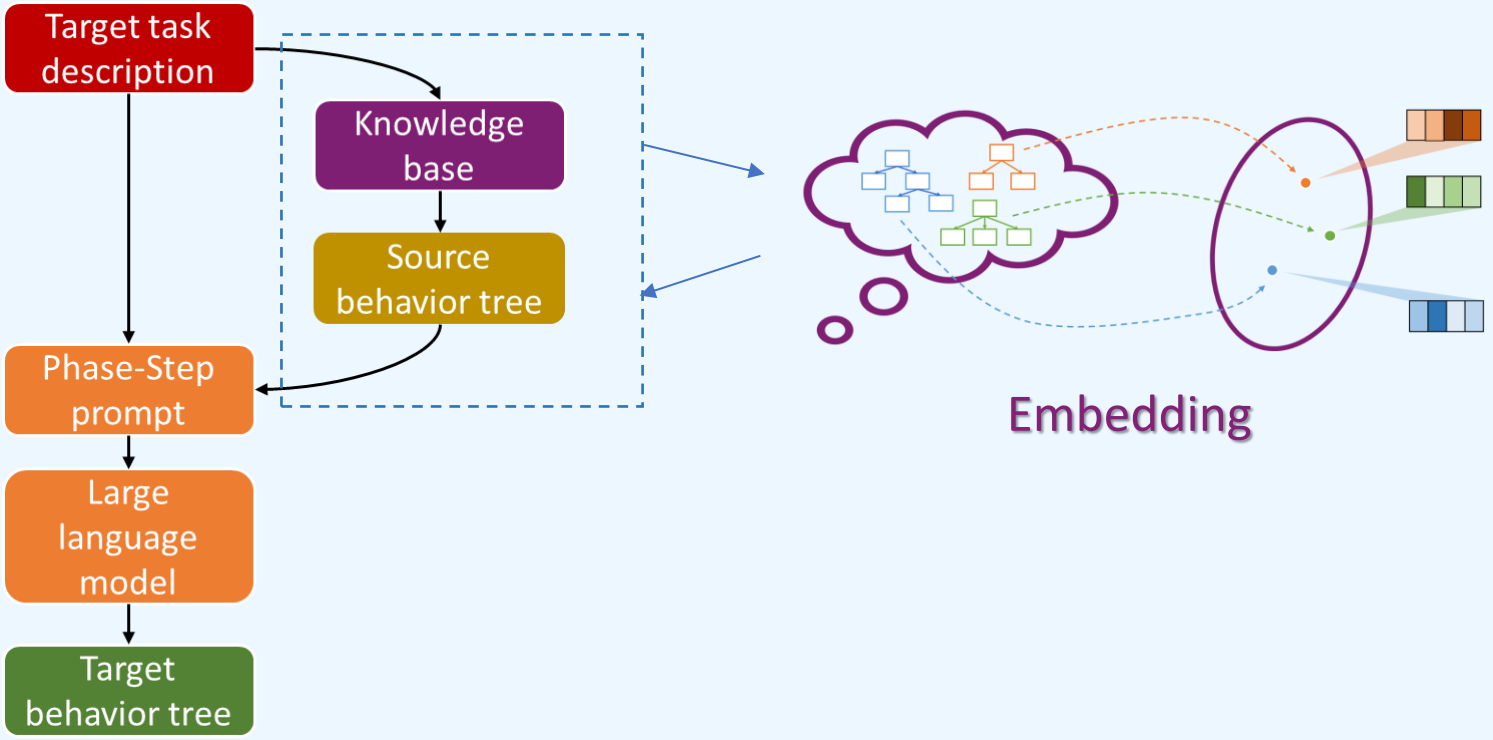
+

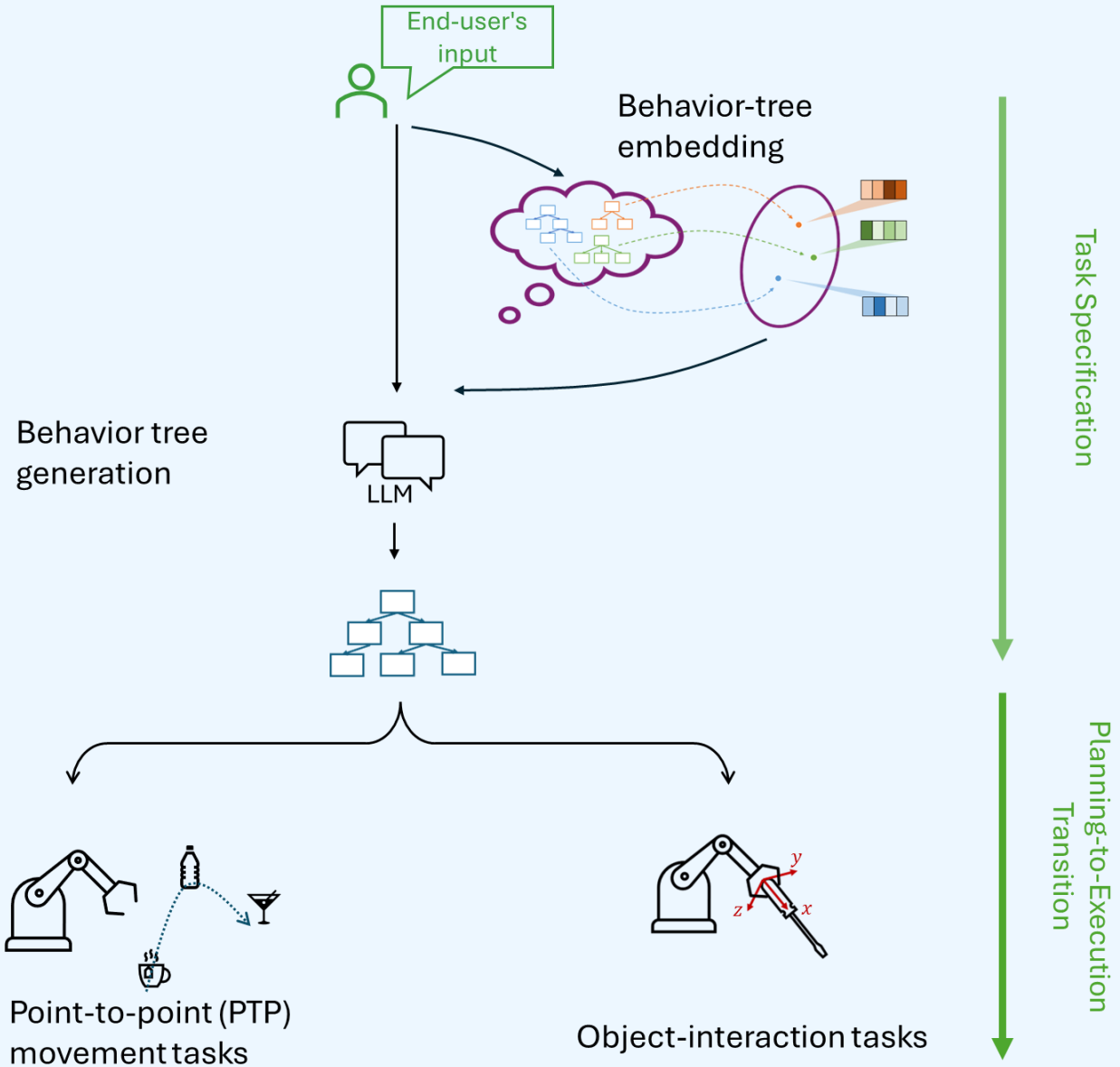
grasp cup | pick knife

pick knife | cut cucumber

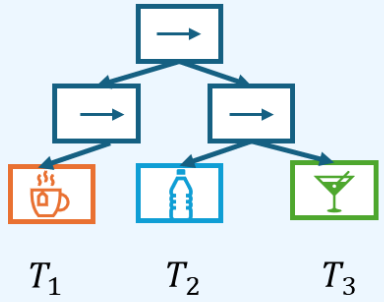
Semantic characteristics

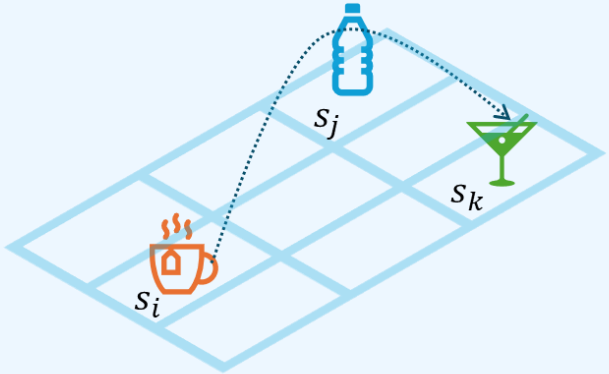
Retrieval-Augmented Generation

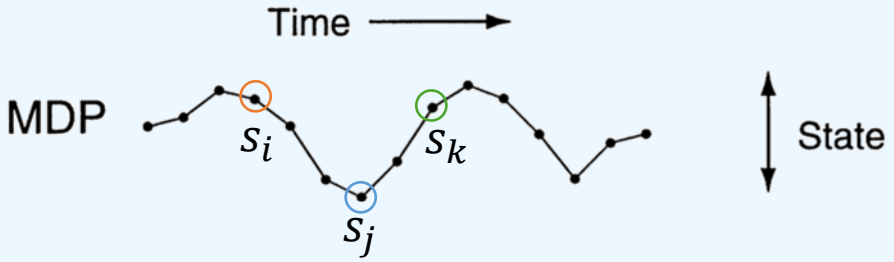



End-user's input
Behavior-tree embedding
Behavior tree generation
LLM
Task Specification
Planning-to-Execution Transition
Point-to-point (PTP) movement tasks
Object-interaction tasks
y
z
x

Part 2: Planning-to-Execution Transition – PTP Movement Tasks (In Prelim)



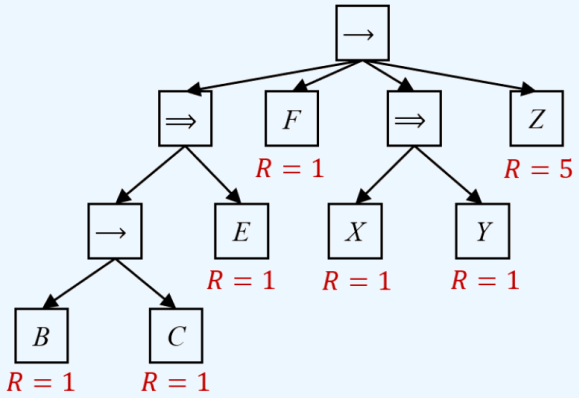




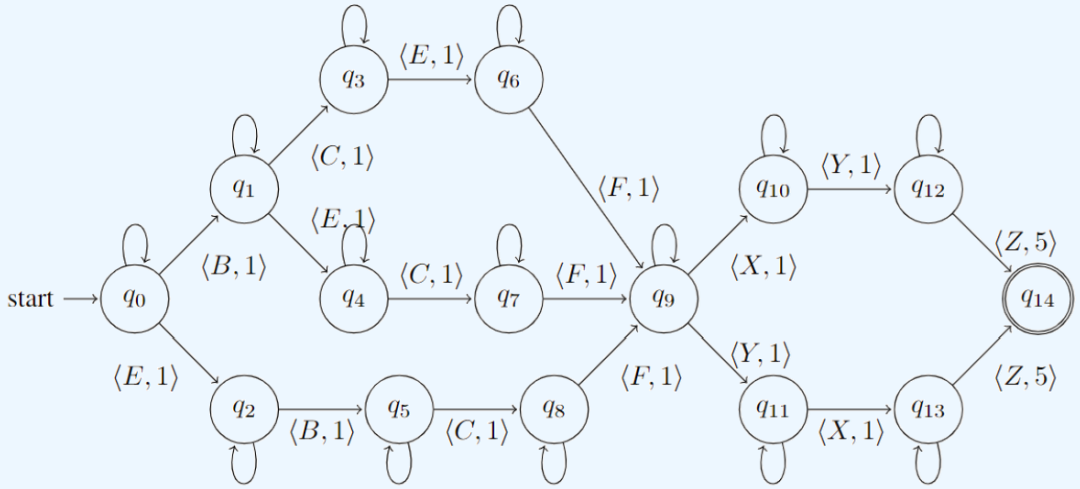
Need to visit s_i, s_j, s_k in order But MDP is memory-less

Yue Cao and C. S. George Lee, "Behavior Tree for Precedence-Constrained Task Modeling and Policy Learning", in revision

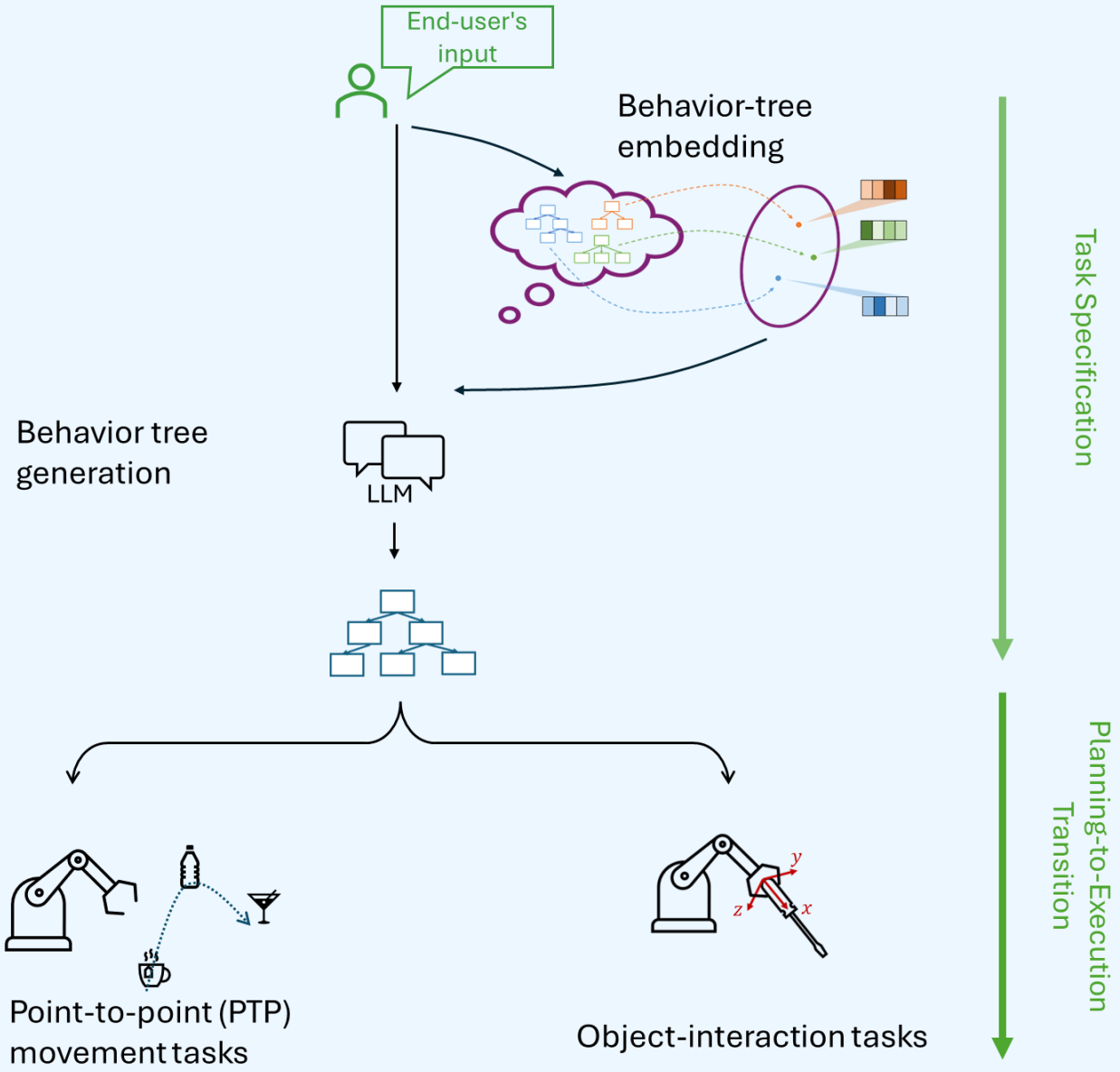
Behavior-Tree Reward



Behavior-Tree Reward



Büchi Automaton Reward


End-user's input
Behavior-tree embedding
Behavior tree generation
LLM
Task Specification
Planning-to-Execution Transition
Point-to-point (PTP) movement tasks
Object-interaction tasks
y
z
x

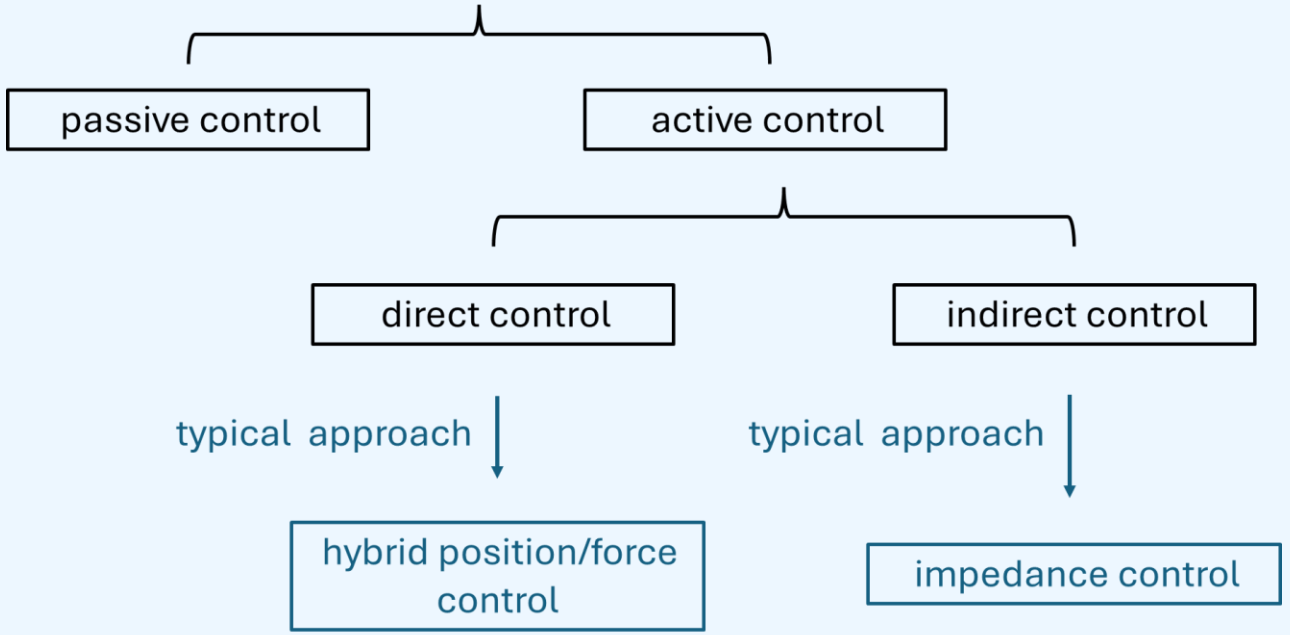
Part 2: Planning-to-Execution Transition – Object-Interaction Tasks

- **Scope:** manipulator object-interaction task.
- **Input:** primitive task, described in natural language.
- **Output:** motor actions.

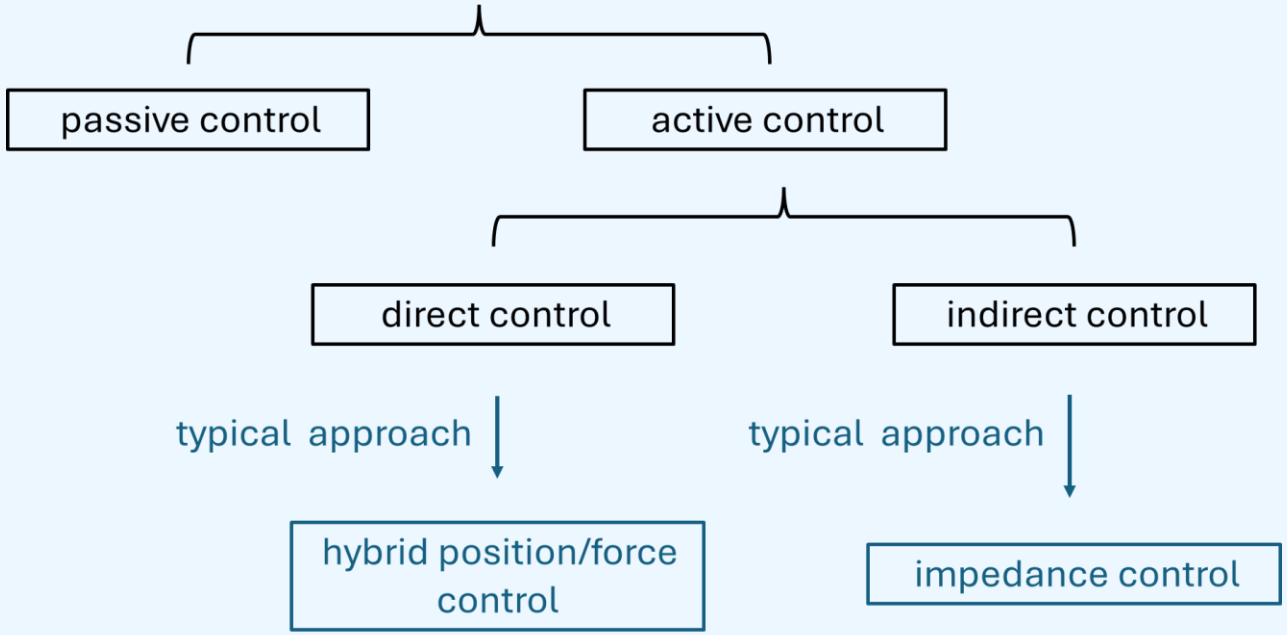
Can we use LLMs to solve this problem?

Yue Cao and C. S. George Lee, "Ground Manipulator Primitive Tasks to Executable Actions using Large Language Models", AAAI Fall Symposium, 2023

Interaction Control Recap



Ideas



[Raibert 1981]

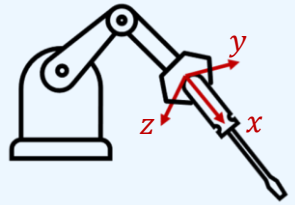
- Task-oriented setting
- Straightforward physics

[Hogan 1984]

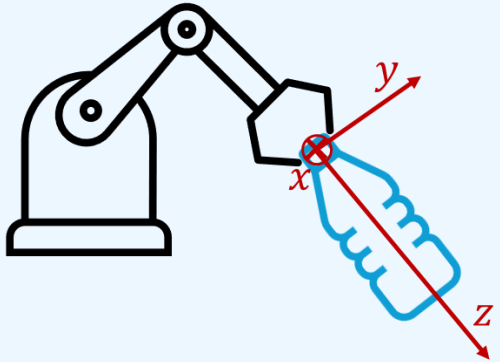
- Robot-oriented setting
- Resemble the interaction as a mass-spring-damper model

Intro on Task Frame Formalism

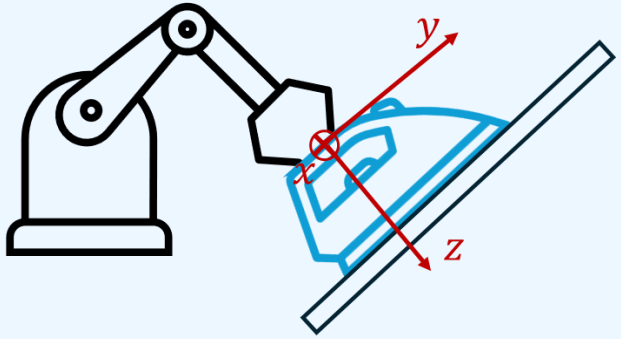
Task Frame Formalism (TFF):
A concept associated with hybrid position/force control
[Mason 1981]



Position $\boldsymbol{X} = [x_x, x_y, x_z, \omega_x, \omega_y, \omega_z]^T$
Force $\boldsymbol{F} = [f_x, f_y, f_z, g_x, g_y, g_z]^T$

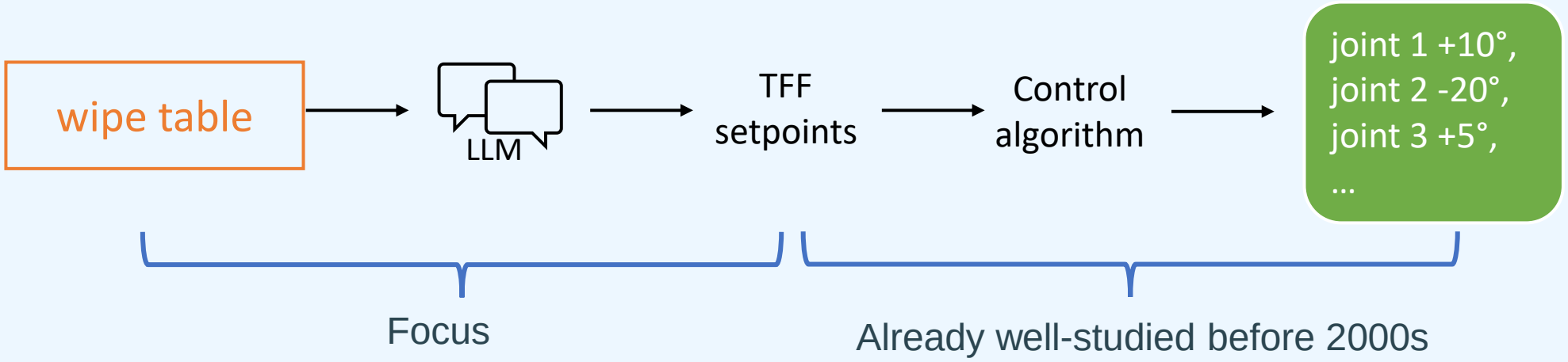


Position $\boldsymbol{X} = [0\ 0\ 0,0,0,0]^T$
Force $\boldsymbol{F} = [0\ 0,0,0,0,1]^T$



Position $\boldsymbol{X} = [1,1,0,0,0,0]^T$
Force $\boldsymbol{F} = [0,0,1,0,0,0]^T$

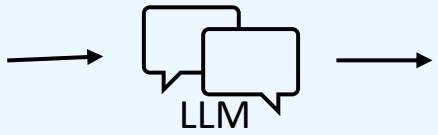
LLM-based Approach



LLM-based Approach

```python
# Source function 1
def turn_screw(translational_x, translational_y, translational_z,
               angular_x, angular_y, angular_z):
  # Coordinate setting: Z axis as the direction of screw
  translational_x=0
  translational_y=0
  translational_z=-5 # N

  angular_x=0
  angular_y=0
  angular_z=5 # rad/sec
  return translational_x, translational_y, translational_z, \
  angular_x, angular_y, angular_z

# Source function 2
def wipe_table(...)
  ...
  return

# Source function 3
def open_door_from_doorknob(...)
  ...
  return

# Target function
def turn_steering_wheel(translational_x, translational_y, translational_z,
                        angular_x, angular_y, angular_z):
```

A 3-shot prompt example




```python
# Coordinate setting: Z axis as the direction of the steering wheel
translational_x=0
translational_y=0
translational_z=0

angular_x=0
angular_y=0
angular_z=5 # rad/sec
return translational_x, translational_y, translational_z, \
angular_x, angular_y, angular_z
```

LLM Output

- Program-function-like prompt
- Task-frame-formalism-based representation
- Few-shot inference

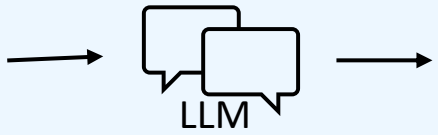
Approach in Details

primitive task name

```python
# Source function 1
def turn_screw(translational_x, translational_y, translational_z,
               angular_x, angular_y, angular_z):
  # Coordinate setting: Z axis as the direction of screw
  translational_x=0
  translational_y=0
  translational_z=-5 # N

  angular_x=0
  angular_y=0
  angular_z=5 # rad/sec
  return translational_x, translational_y, translational_z, \
  angular_x, angular_y, angular_z

# Source function 2
def wipe_table(...)
  ...
  return

# Source function 3
def open_door_from_doorknob(...)
  ...
  return

# Target function
def turn_steering_wheel(translational_x, translational_y, translational_z,
                        angular_x, angular_y, angular_z):
```

coordinate in task frame formalism

activate directions in task frame formalism




```python
# Coordinate setting: Z axis as the direction of the steering wheel
translational_x=0
translational_y=0
translational_z=0

angular_x=0
angular_y=0
angular_z=5 # rad/sec
return translational_x, translational_y, translational_z, \
angular_x, angular_y, angular_z
```

LLM Output

generate for a new primitive task

A 3-shot prompt example

Preliminary Evaluation

- Evaluated in 30 manipulator primitive tasks in July 2023.

1. cut pizza	11. rasp wood	21. open door from hinge
2. scrub desk with bench brush	12. scrape substance from surface	22. slide block over vertical surface
3. spear cake with fork	13. peel potato	23. turn steering wheel
4. fasten screw with screwdriver	14. slice cucumber	24. shake cocktail bottle
5. loosen screw with screwdriver	15. flip bread	25. cut banana
6. unlock lock with key	16. shave object	26. crack egg
7. fasten nut with wrench	17. use roller to roll out dough	27. press button
8. loosen nut with wrench	18. insert peg into pegboard	28. insert GPU into socket
9. spread paint with brush	19. brush across tray	29. open bottle
10. hammer in nail	20. insert straw through cup lid	30. open childproof bottle

- Task correctness in 5-shot test. blue: correct, red: incorrect

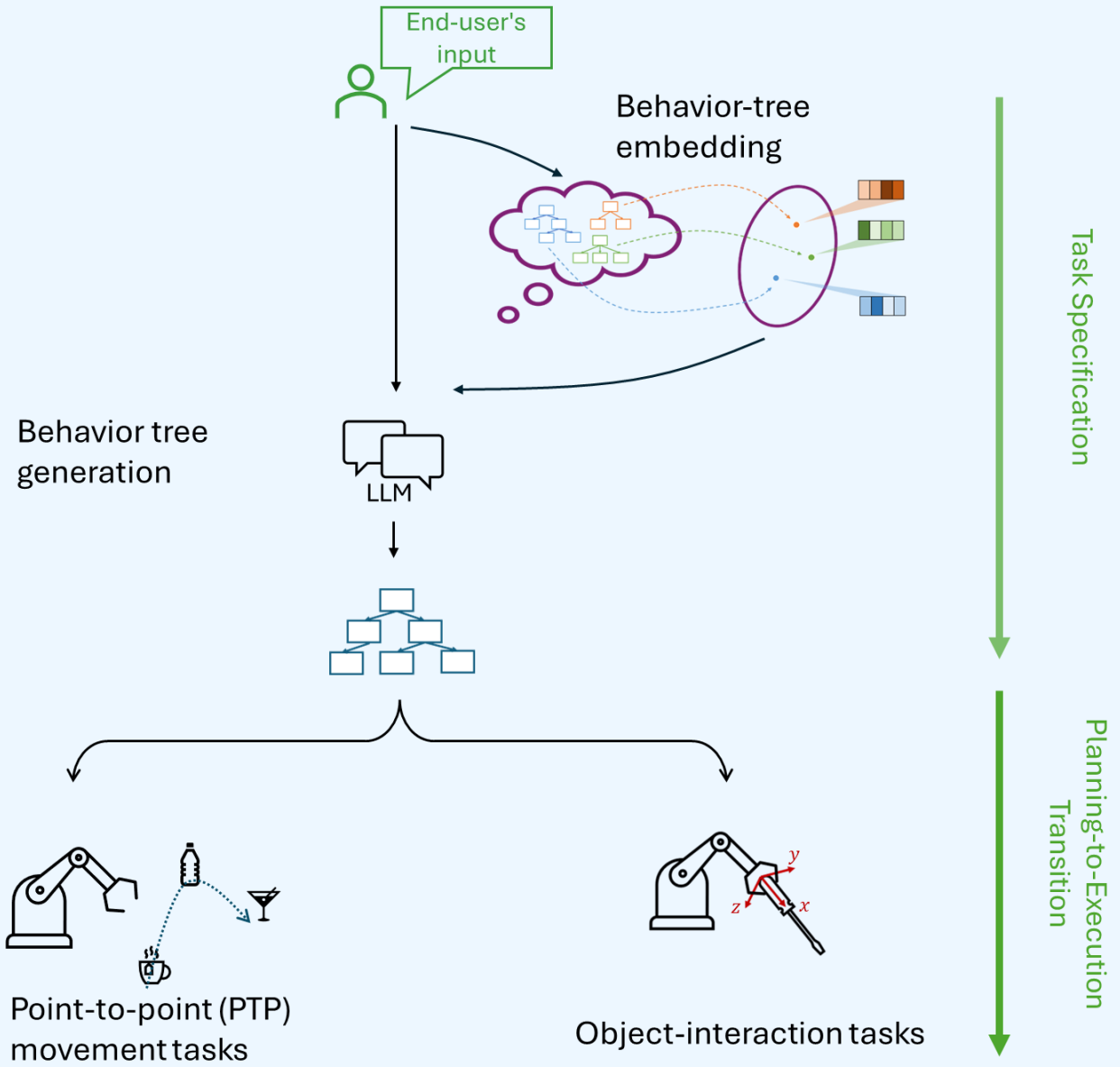
	Task No.	Correctness
GPT-3.5-turbo	1-10	
	11-20	
	21-30	
GPT-4	1-10	
	11-20	
	21-30	
Bard	1-10	
	11-20	
	21-30	
LLaMA-2-70B	1-10	
	11-20	
	21-30	

Conclusions

Challenge 1: The Missing of Natural Languages
→ Natural languages as system input

Challenge 2: The Dilemma of Primitive-Task Libraries
→ Go beyond fixed primitive-task libraries

Challenge 3: The “Mechanical Turk” Like System
→ Minimize human intervention



Towards Manipulator Task-Oriented Programming: Automating Behavior-Tree Configuration

Thanks!

Yue Cao

yuecao@purdue.edu

This work was supported in part by the National Science Foundation under Grant IIS-1813935.

ARTLAB
Assistive Robotics Technology

PURDUE UNIVERSITY
Elmore Family School of Electrical and Computer Engineering