Forgetful Large Language Models:
Lessons Learned from Using LLMs in Robot Programming

Juo-Tung Chen

Chien-Ming Huang

Intuitive Computing Laboratory
Department of Computer Science

JOHNS HOPKINS UNIVERSITY

AAAI Fall Symposium 2023

Overview

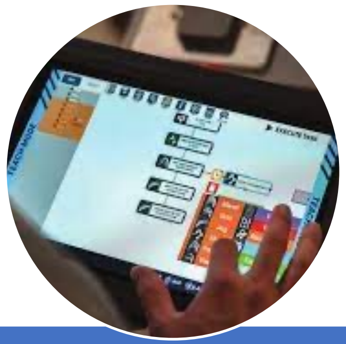
- Common errors in LLM-generated code for robot programming
- Proposal of prompt engineering strategies to reduce execution errors
- Demonstration of tactics' effectiveness with ChatGPT, Bard, and LLaMA-2
- Key takeaways and lessons learned from using LLMs in robot programming

Robot Application Development paradigms

Traditional RAD

Time consuming,
requires expertise

→



End-user RAD

Lower barriers
Still requires programming specifications

⇢

AI-Aided RAD

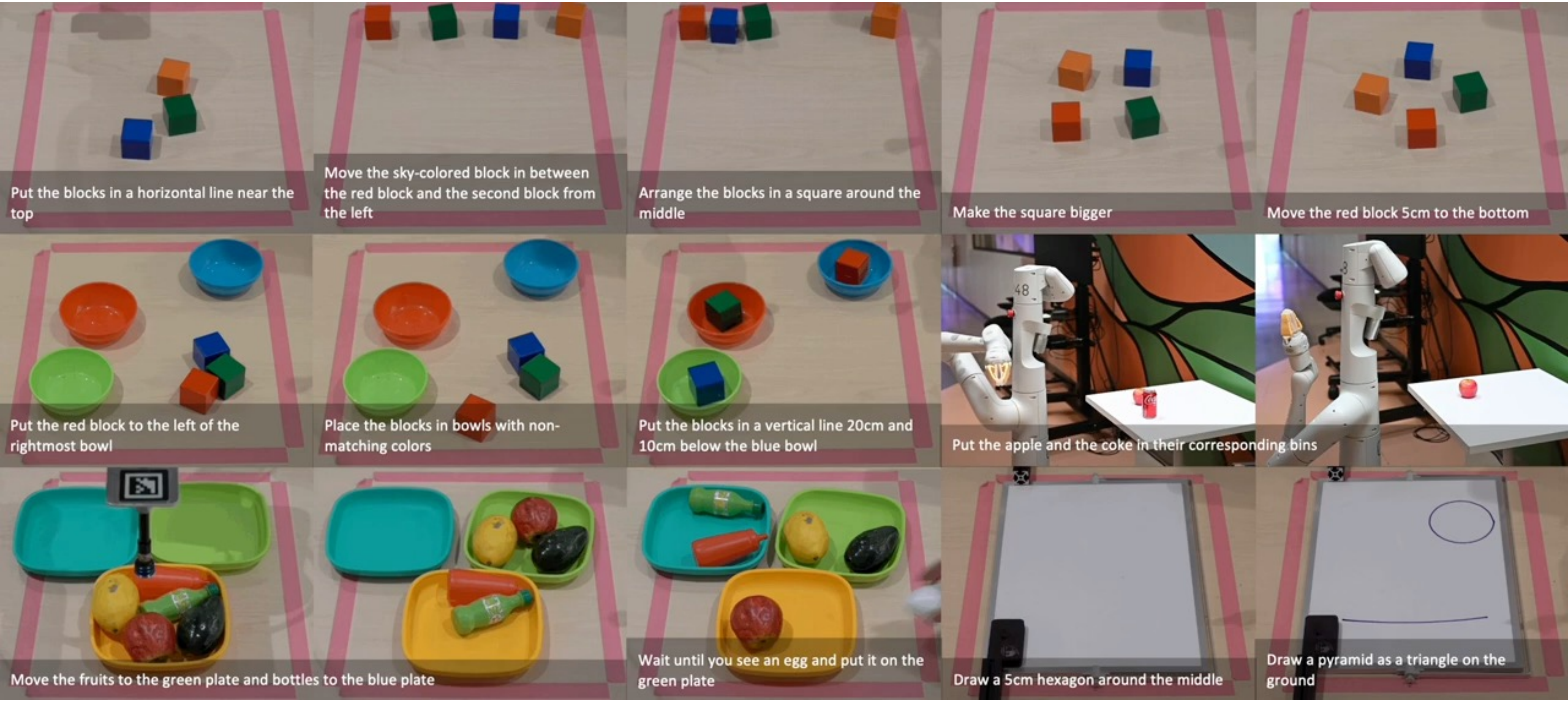
Natural language (prompt)
Collaborative interaction

Ajaykumar, Gopika, Maureen Steele, and Chien-Ming Huang. "A survey on end-user robot programming." *ACM Computing Surveys (CSUR)* 54.8 (2021): 1-36.

AR

Awesome! I want now to use the blocks to build the logo from Microsoft on top of the white pad. It consists of four colors forming a square, blue on the bottom left, yellow on the bottom right, red on the top left and green on the top right.

Vemprala, Sai, et al. "Chatgpt for robotics: Design principles and model abilities." *Microsoft Auton. Syst. Robot. Res* 2 (2023): 20.


Put the blocks in a horizontal line near the top
Move the sky-colored block in between the red block and the second block from the left
Arrange the blocks in a square around the middle
Make the square bigger
Move the red block 5cm to the bottom
Put the red block to the left of the rightmost bowl
Place the blocks in bowls with non-matching colors
Put the blocks in a vertical line 20cm and 10cm below the blue bowl
Put the apple and the coke in their corresponding bins
Move the fruits to the green plate and bottles to the blue plate
Wait until you see an egg and put it on the green plate
Draw a 5cm hexagon around the middle
Draw a pyramid as a triangle on the ground

Liang, Jacky, et al. "Code as policies: Language model programs for embodied control." *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023.

Key Differences from Existing Research

1. Exploration vs. Resolution

2. Identifying Errors

3. Mitigation Strategies

4. Sharing Lessons

The stochastic nature of LLMs, which purposefully introduce **randomness** to increase response diversity, can pose a significant challenge in robot programming where **correctness** often hinges on finding a specific, rare solution

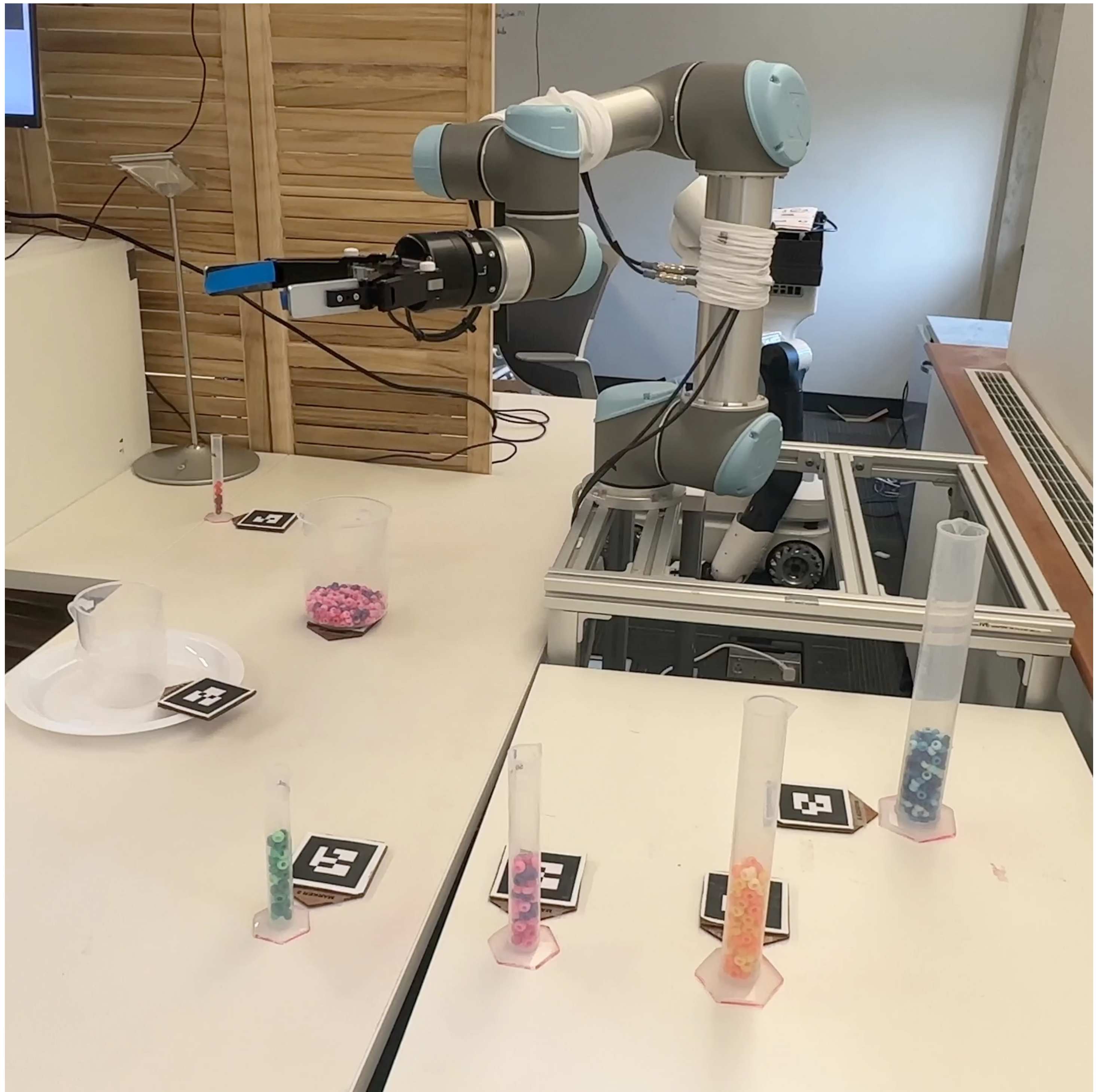
Research Questions

1. What are the **common errors** generated by LLMs in robot programming?

2. How can we develop **practical strategies** to reduce these errors?

Experiment 1: Identify Common Errors

Sequential Manipulation Task



Develop a standard prompt to assess LLM-generated errors

Baseline Prompt

System

Description of API library

Solution Example

Objective

You are an assistant helping me with the UR5 robot arm. This is a 6 degrees of freedom robot manipulator that has a gripper as its end effector. The gripper is in the open position in the beginning. When I ask you to do something, you are supposed to give me **Python code** that is needed to achieve the task using the UR5 robot arm and then an explanation of what that code does. You are only allowed to use the functions I have defined for you. You are not to use any other hypothetical functions that you think might exist. You can use simple Python functions from libraries such as math and numpy.

…

In the environment, the following items might be present:
beaker 1L: radius = 6.5 cm and height = 15 cm ,
beaker 500mL: radius = 5.5 cm and height = 12 cm,
beaker 250mL: radius = 4.75 cm and height = 10 cm,
…
Use the dimensions provided above when I don't specifically tell you the dimensions of the objects.

(Please refer to our appendix to see the full baseline prompt)

Develop a standard prompt to assess LLM-generated errors

Baseline Prompt

System

Description of API library

Solution Example

Objective

At any point, you have access to the following functions, which are accessible after initializing a function library. **You are not to use any hypothetical functions.** All units are in the SI system.
lib=FunctionLib(): Initializes all functions; access any of the following functions by using lib.
move_to_home_position(): Moves the robot to a neutral home position
go(x,y,z,roll,pitch,yaw): Moves the robot arm to the x, y, z position in meters, and roll, pitch, yaw in degrees with respect to the base frame of the robot.
pour(target_container_name): The robot will go to near the target container and rotate its wrist to pour the contents inside the object that is grasped by the gripper into the target container.
…

A few useful things: **Always start your code by importing FunctionLib and also make sure to always init a node with rospy**. If you are uncertain about something, you can ask me a clarification question, as long as you specifically identify it by saying "Question"
…

(Please refer to our appendix to see the full baseline prompt)

Develop a standard prompt to assess LLM-generated errors

Baseline Prompt

System

Description of API library

Solution Example

Objective

The following is an **example** of writing the code. If the user asked, “There is a 100mL graduated cylinder on Marker 6 and a 1L beaker on Marker 9. Pick up the graduated cylinder and pour its contents into the beaker. After pouring, place the graduated cylinder at Marker 5,” then you should write a code like the following:

```
from Lib.ur5.FunctionLibrary import FunctionLib
import rospy

# Initialize rospy node called gpt
rospy.init_node(`gpt')

# Initialize function library
lib = FunctionLib()

# Move the robot back to home position
lib.move_to_home_position()
rospy.sleep(2)
…
```

(Please refer to our appendix to see the full baseline prompt)

Develop a standard prompt to assess LLM-generated errors

Baseline Prompt

System

Description of API library

Solution Example

Objective

Please write a **Python function** to pick up a 25mL graduated cylinder at Marker 15 and pour its contents into a 500mL beaker at Marker 7.
After that, put the cylinder back to where it was.

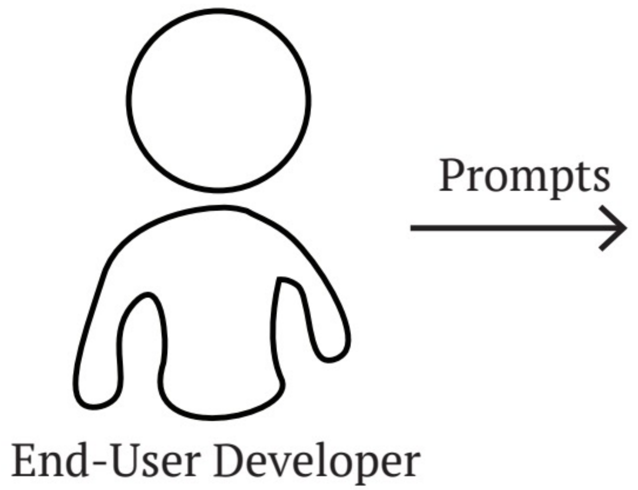
Language Models

LLaMA-2
(13B parameters)

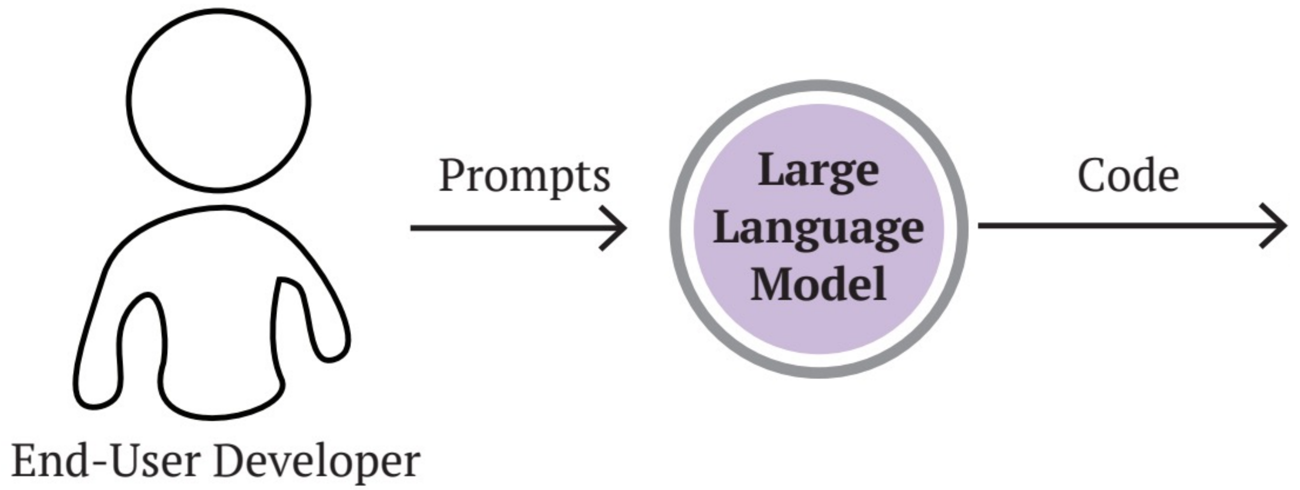
GPT-3.5
(154B parameters)

Bard
(137B parameters)

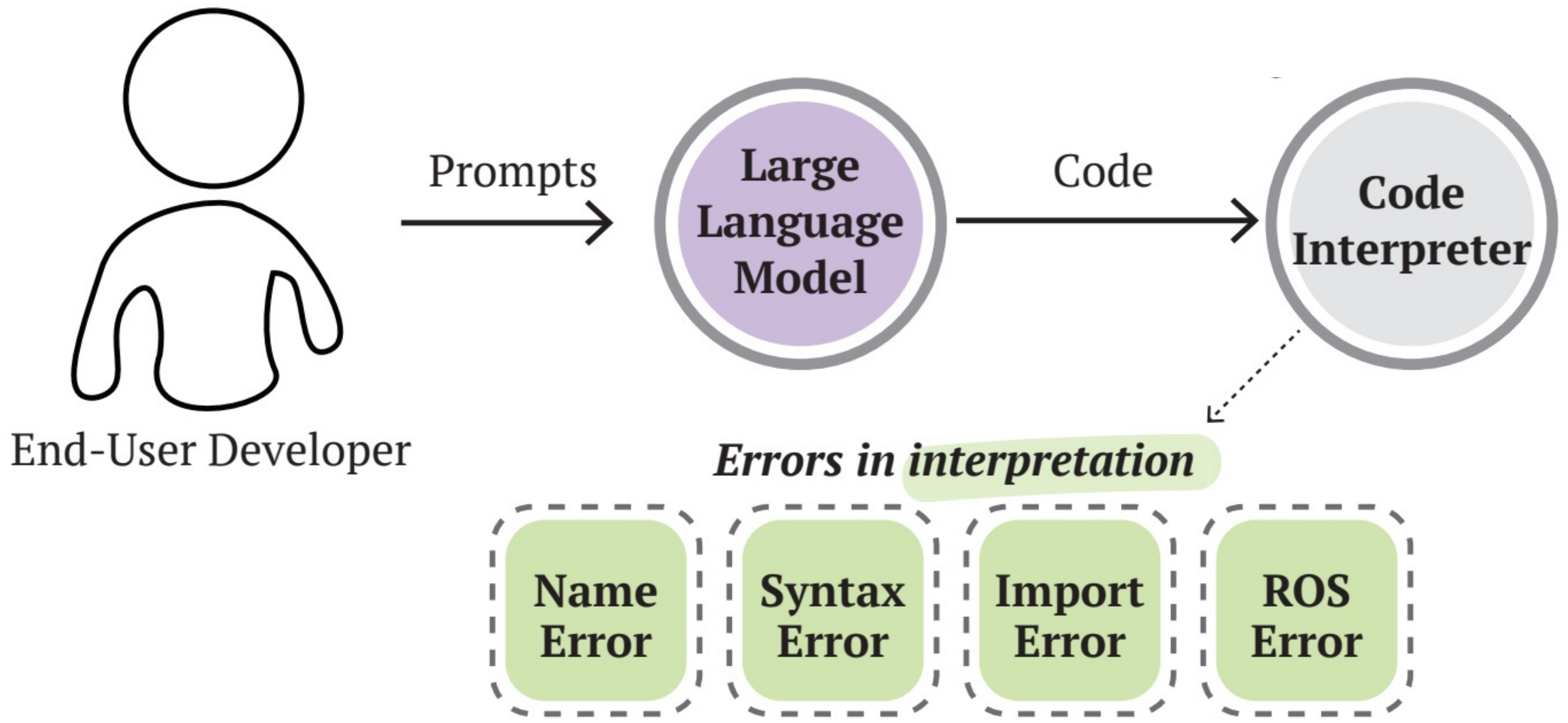
Workflow and the emergence of potential errors



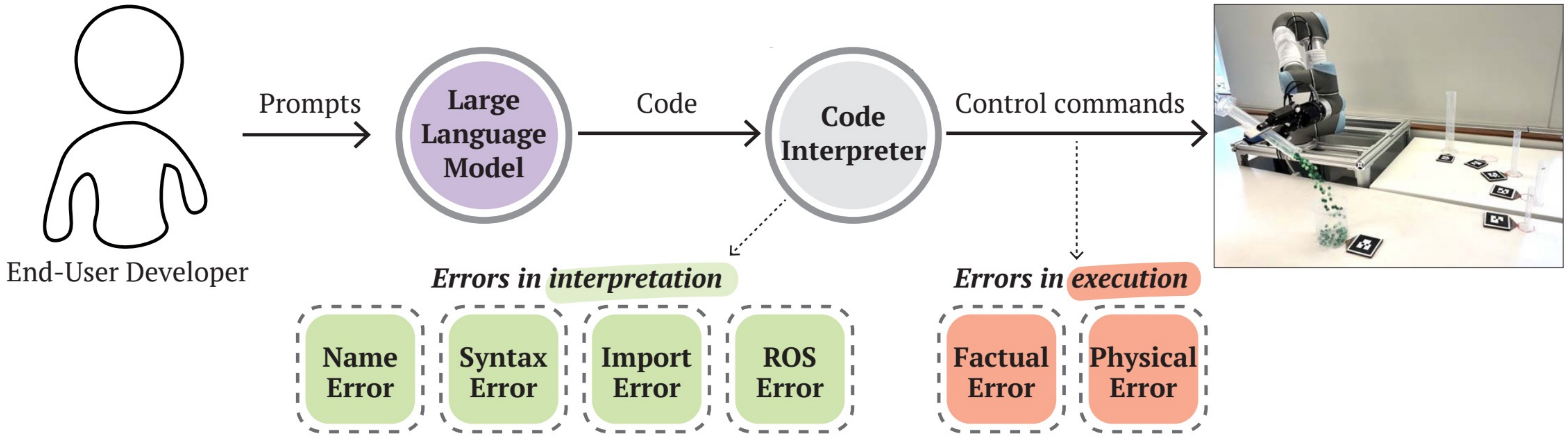
Workflow and the emergence of potential errors



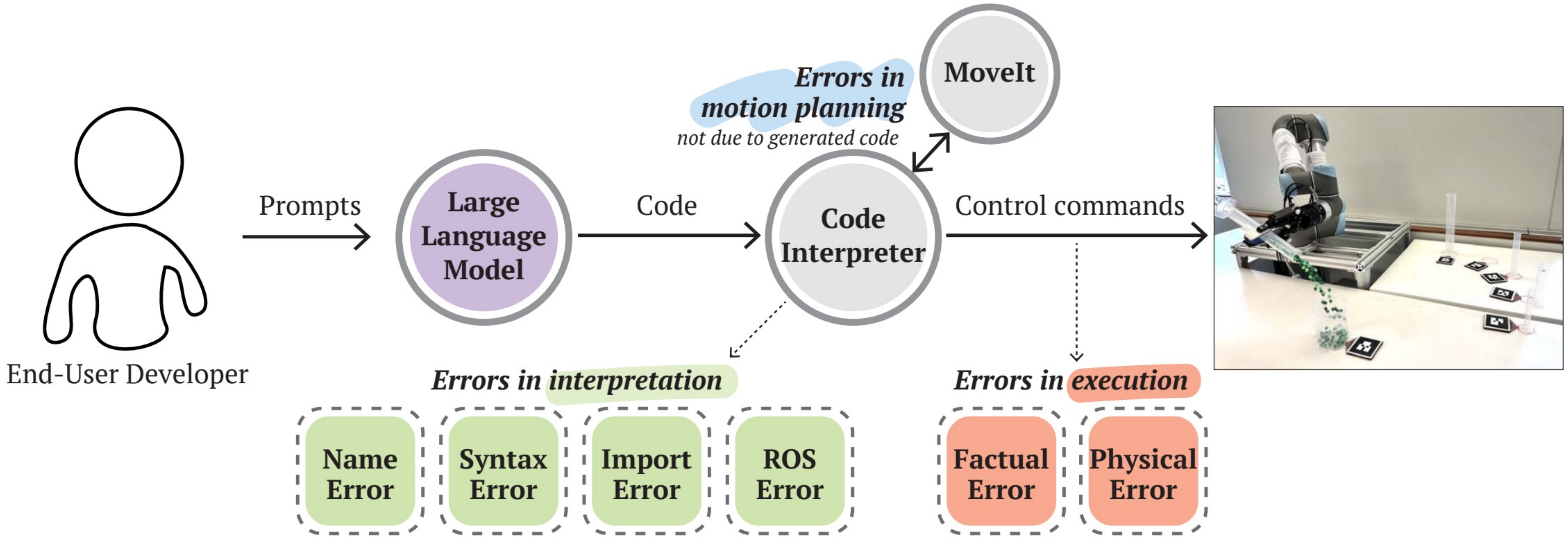
Workflow and the emergence of potential errors



Workflow and the emergence of potential errors



Workflow and the emergence of potential errors



Errors in interpretation

Name Error:

```
# Move and place the 100mL beaker at marker 7
 move_and_place_object("beaker_100mL", 7)
 undefined function
```

Syntax Error:

```
# Open the gripper to release the cylinder
   lib.open_gripper( )  unclosed parenthesis
```

Import Error:

```
    forgot to import libraries
 ←        import rospy
          from Lib.ur5.FunctionLibrary import FunctionLib
# Initialize rospy node called gpt
rospy.init_node('gpt')

# Initialize function library
lib = FunctionLib()
```

ROS Error:

```
import rospy
from Lib.ur5.FunctionLibrary import FunctionLib
    forgot to initialize rospy node
  ←  rospy.init_node('gpt')

# initialize function library
lib = FunctionLib()
```

➡ *Can be rectified using* ***code verifications***

Errors in Execution

Factual Error:

```
# Define the objects dimensions
cylinder_25mL_height = 0.065
```

fabricated numerical values

Physical Error:

```
...
# Pour into beaker 500mL
lib.pour("beaker 500mL")

# Move above 0.1 meters the beaker's location
success = lib.go(beaker[0], beaker[1], beaker[2] + 0.1,
          beaker[3], beaker[4], beaker[5])

# Open the gripper to release the cylinder
lib.open_gripper()
```

unnecessary steps
(will result in task failure)

→ *LLM being "forgetful"*

Experiment 2: Exploring Practical Strategies to Reduce Errors in Execution

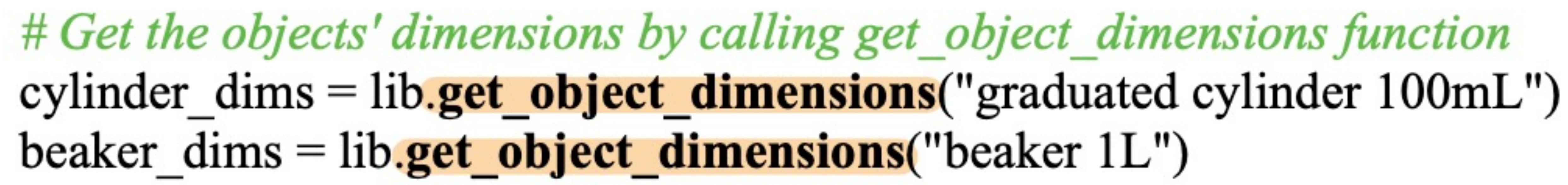
Experiment 2

Strategies:

1. Prompts involve task/context information specified in numerical form

➡ *implement dedicated functions for retrieving numerical data*

(reduce Factual Error)



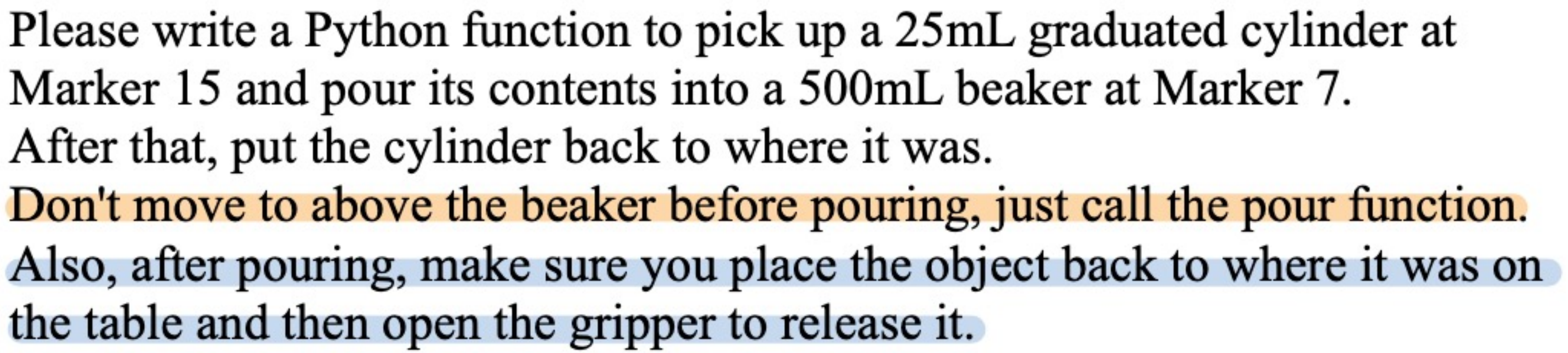
```
# Get the objects' dimensions by calling get_object_dimensions function
cylinder_dims = lib.get_object_dimensions("graduated cylinder 100mL")
beaker_dims = lib.get_object_dimensions("beaker 1L")
```

Experiment 2

Strategies:

2. Intricate functions (like the pour function in our experiment)

➡ *reinforce key constraints in the objective prompt* (reduce Physical Error)



Please write a Python function to pick up a 25mL graduated cylinder at Marker 15 and pour its contents into a 500mL beaker at Marker 7.
After that, put the cylinder back to where it was.
Don't move to above the beaker before pouring, just call the pour function.
Also, after pouring, make sure you place the object back to where it was on the table and then open the gripper to release it.

reinforce constraints (orange), articulate the physical implications (blue)

Results

Before applying the strategies

Model	Factual Error (%)	Physical Error (%)	Import Error (%)	ROS Error (%)	Name Error (%)
GPT 3.5	**100**	**90**	40	40	0
Bard	**100**	**100**	0	0	0
LLaMA-2	**90**	**90**	0	0	10

Results

After applying the strategies

Model	Factual Error (%)	Physical Error (%)	Import Error (%)	ROS Error (%)	Name Error (%)	Completion (%)	Reduction in Errors in Execution (%)
GPT 3.5	**10**	**0**	40	40	30	**60**	**94.7**
Bard	**10**	**0**	0	0	10	**70**	**95**
LLaMA-2	**0**	**30**	0	0	60	**40**	**83.3**

- Increased task completion rates.
- Reduced factual and physical errors.

Lessons Learned

- ➢ Inconsistency in LLM-Based Code Generation
  - ➡ Importance of user involvement and descriptive prompting
- ➢ Forgetful LLM can result in errors in execution
  - ➡ Importance of reinforcing constraints in objective prompt

➡ Suite of Tools for Productive LLM-Based Robot Programming

- ➢ Verification script or feedback loop (errors in interpretation)
- ➢ Data-retrieving function + reinforce constraints in objective prompt (errors in execution)
- ➢ Simulation preview tools before deployment

Forgetful Large Language Models: Lessons Learned from Using LLMs in Robot Programming

Juo-Tung Chen
jchen396@jhu.edu

Chien-Ming Huang
cmhuang@cs.jhu.edu

JOHNS HOPKINS UNIVERSITY