Deploying and Evaluating LLMs to Program Service Mobile Robots

Francesca Lucchetti[1], Zichao Hu[2], Anders Freeman[3], Sadanand Modak[2], Yash Saxena[2], Luisa Mao[2], Claire Schlesinger[1], Arjun Guha[1], Joydeep Biswas[2]

I. Introduction

Recent advancements in large language models (LLMs) have spurred interest in using them for generating robot programs from natural language, with promising initial results [1], [2], [3]. We investigate the use of LLMs to generate programs for *service mobile robots* leveraging mobility, perception, and human interaction skills, where accurate *sequencing and ordering of actions* is crucial for success. We contribute CodeBotler, an open-source tool to program service mobile robots from natural language, and RoboEval, a benchmark to evaluate the correctness and robustness of generated programs. We release our code and benchmark at https://amrl.cs.utexas.edu/codebotler/.

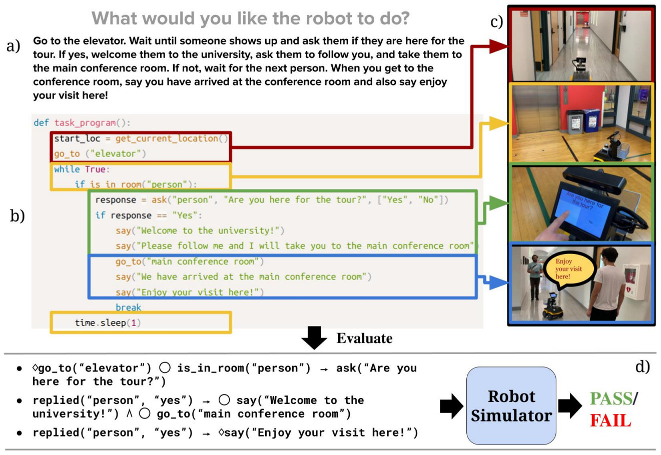
II. CodeBotler

CodeBotler is an open-source tool to generate general-purpose service robot programs from natural language, and to deploy such programs on general-purpose autonomous mobile robots. It performs program generation with an *embedded domain-specific language* (eDSL) via few-shot prompting in a programming language that LLMs are already adept in: Python. By abstracting robot skills in the eDSL, we release CodeBotler, a robot-agnostic deployment system for executing generated programs on any general-purpose mobile robot. By embedding the eDSL in Python, CodeBotler drastically reduces the number of syntax and runtime errors of generated code.

III. RoboEval

Existing code completion benchmarks tackle simple data processing functions [4] or low-level robot skills [3], which are amenable to simple input-output unit tests. However, code generation for general-purpose service robot programs cannot be evaluated just on input/output sequences. For example, given a task *"Check how many conference rooms have no markers"*, it is insufficient to just test whether the LLM-generated program executes to state the correct answer — to ensure correctness, a correct program must first visit all conference rooms and check for markers there before arriving at the result. We also observe that there are significant variations in the correctness of generated programs with small variations in the phrasing of the natural language task descriptions [5]. We thus introduce three key ideas to test for both correctness and robustness of LLM-generated robot programs: 1) we evaluate the *execution traces* of programs; 2) we check whether the execution traces satisfy *temporal logic* properties that encode correctness for each task; and 3) we *vary the prompts* and to test for robustness. Combining these ideas, we release a preview of RoboEval with five tasks, each with multiple world states, multiple prompts, and temporal logic formulas to check correctness. Fig. 2 shows the results of three LLMs (PaLM, GPT-3.5, and StarCoder) on the RoboEval benchmark.

[1]Northeastern Univ., [2]Univ. of Texas at Austin, [3]Wellesley College



Go to the elevator. Wait until someone shows up and ask them if they are here for the tour. If yes, welcome them to the university, ask them to follow you, and take them to the main conference room. If not, wait for the next person. When you get to the conference room, say you have arrived at the conference room and also say enjoy your visit here!

```
def task_program():
    start_loc = get_current_location()
    go_to ("elevator")
    while True:
        if is_in_room("person"):
            response = ask("person", "Are you here for the tour?", ["Yes", "No"])
            if response == "Yes":
                say("Welcome to the university!")
                say("Please follow me and I will take you to the main conference room")
                go_to("main conference room")
                say("We have arrived at the main conference room")
                say("Enjoy your visit here!")
                break
        time.sleep(1)
```

- ◊go_to("elevator") ◯ is_in_room("person") → ask("Are you here for the tour?")
- replied("person", "yes") → ◯ say("Welcome to the university!") ∧ ◯ go_to("main conference room")
- replied("person", "yes") → ◊say("Enjoy your visit here!")

Fig. 1: CodeBotler converts a) natural language task prompts to b) python programs and c) executes on a real robot. d) RoboEval verifies results via LTL checking of simulation traces.

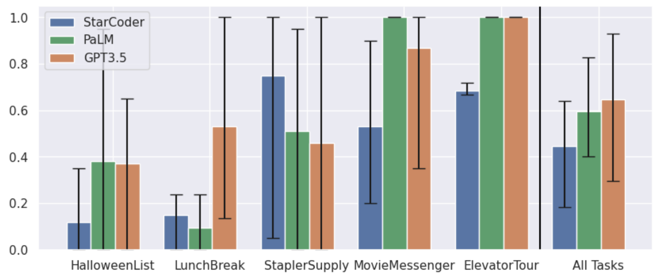


Fig. 2: RoboEval pass@1 results for PaLM (text-bison-001), StarCoder, and GPT3.5 (text-davinci-003).

References

[1] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *ICML 2022*.

[2] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "ProgPrompt: Generating Situated Robot Task Plans using Large Language Models," in *ICRA 2023*.

[3] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *arXiv:2209.07753*, 2022.

[4] M. Chen, J. Tworek, H. Jun, Q. Yuan, *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.

[5] H. M. Babe, S. Nguyen, Y. Zi, A. Guha, M. Q. Feldman, and C. J. Anderson, "StudentEval: A benchmark of student-written prompts for large language models of code," *arXiv:2306.04556*, 2023.