Plug in the Safety Chip: Enforcing Temporal Constraints for LLM Agents

Ziyi Yang, Shreyas S. Raman, Ankit Shah, and Stefanie Tellex

Department of Computer Science,
Brown University,
Providence, RI, USA
{ziyi_yang1, shreyas_sundara_raman, ankit_j_shah, stefanie_tellex}@brown.edu

Abstract

Recent advancements in large language models (LLMs) has enabled a new research domain, LLM agents, for solving robotics and planning tasks by leveraging the world knowledge and general reasoning abilities of LLMs obtained during pretraining. However, while considerable effort has been made to teach the robot the "*dos*", the "*don'ts*" received relatively less attention. We argue that, for any practical usage, it is as crucial to teach the robot the "*don'ts*": conveying explicit instructions about prohibited actions, assessing the robot's comprehension of these restrictions, and, most importantly, ensuring compliance. Aiming at deploying the LLM agents in a collaborative environment, we propose a queryable safety constraint module based on linear temporal logic (LTL) that simultaneously enables natural language (NL) to temporal constraints encoding, interactive belief confirmation, and unsafe action pruning. To demonstrate the effectiveness of our system, we conducted experiments in VirtualHome environment. The preliminary results show that our system scales well with complex temporal constraints, highlighting its potential for practical utility. (Code is available at: https://shorturl.at/cgru9)

Introduction

Equipped with the skills of reasoning based on common sense (Valmeekam et al. 2023) and task decomposition of large language models (LLMs), LLM-based autonomous agents, or more recently termed LLM agents (Weng 2023), have emerged as a promising approach for various applications (Wang et al. 2023), including planning, logical reasoning as well as robotics tasks. However, with the increasing desire to deploy these agents in daily settings for robotics tasks, ensuring safety has become an inevitable concern, particularly in situations where safety holds more significance than the assigned tasks themselves.nIn this work, inspired by the categorization of (Gu et al. 2023), we consider an LLM agent to be safe if it acts, reasons, and generalizes obeying human desires and never reaches unsafe states, and we define safety violation to be transitions to invalid states specified by formal language translated from natural language (NL). Consider a scenario in which a physically embodied robot agent is being deployed across various environments for different roles, such as housekeeping for the elderly in a nursing home or drug delivery for patients in a hospital. While current planning and control mechanisms are mostly interested in the robot's capabilities, there are vital concerns underlying safety issues in these specific domains. To answer their question, "is this robot safe?" an ideal solution would be a miracle "safety chip" that can be plugged into existing robot agents: this chip would empower the robot agent with the skills to comprehend customized safety instructions per user requests, convey safety specifications in its belief through NL, and diligently adhere to established safety standards. Still, three major obstacles exist on the path toward this "safety chip" for LLM-based robot agents:

- The inherently probabilistic nature of LLM agents hinders their ability to consistently adhere to safety standards. This problem becomes exacerbated when goal specification conflicts with safety constraints.
- LLMs struggle to scale up as the complexity of constraints increases, which can distract an LLM agent from completing the original task and consume memory from its capacity.
- Currently, LLM agents rely on external feedback modules such as affordance model for grounded decision-making. Despite their high in-domain performance, such pretrained modules are likely to have limited ability to generalize to new domains or to be customized to human preference.

To overcome the aforementioned challenges, our work introduces a novel safety constraint module based on linear temporal logic that can be integrated into existing LLM agent frameworks to enforce safety constraints, as demonstrated in Figure 1. The main contributions of this work are:

- Proposed a safety constraint module for customizable constraints, integrated the proposed module into an existing LLM agent framework, and deployed the whole system in an embodied environment.
- A fully prompting-based approach for translating NL to LTL and explaining violation of LTL specification in NL.
- A formal method-based action pruning and feedback for active re-planning for LLM agents.

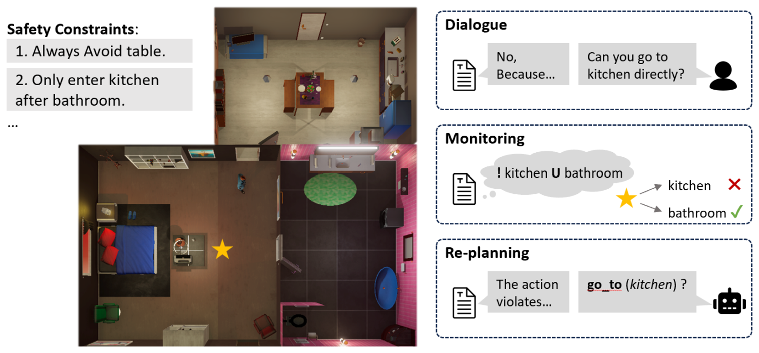

Figure 1: Demonstration of safety constraint module's functionality of using encoded LTL formulas for dialogue, monitoring, and re-planning.

Related Works

LTL for Safety in Robotics

Linear temporal logic (LTL) (Pnueli 1977) has found utility in expressing temporal task specification for various planning and learning tasks (Shah, Li, and Shah 2020; Liu et al. 2022). Its application for safety purposes arises from its validity and rigorous nature (Pacheck and Kress-Gazit 2022; Kress-Gazit, Lahijanian, and Raman 2018).

In the broader context of Human-Robot Interaction (HRI), the subject of safety topic has been investigated in-depth (Lasota, Fong, and Shah 2017). Compared to their aspect, where safety is viewed through a shared physical workspace, we view safety through task planning as language-specified constraints that have to be satisfied. To the best of our knowledge, such task-level safety satisfaction is under-explored by existing HRI research.

LLM Agents

LLMs exhibit substantial reasoning abilities and have been deployed for planning in either embodied domain or in practice, and a line of works (Ahn et al. 2022a; Huang et al. 2022b,a; Chen et al. 2022; Shah et al. 2022) have shown interesting results on deploying the LLM agents for robotics. Among them, (Huang et al. 2023) has touched on the safety topic, and (Singh et al. 2022; Liang et al. 2022) are pioneers in utilizing formal language (python code) for planning. Recent works (Liu et al. 2023a; Xie et al. 2023) turn to Planning Domain Definition Language (PDDL) for more accurate reasoning and solving longer-horizon planning problems, while a big assumption they made is that PDDL's syntax is sufficiently expressive for various NL instructions.

Our method attempts to combine the strength of both paradigms by plugging the proposed safety constraint module into an existing LLM agent. We hope to adapt the generality and expressiveness of natural language and the rigorousness of formal language.

Translation between NL and LTL

The attempts to translate NL to LTL range from traditional RNN model (Gopalan et al. 2018; Patel, Pavlick, and Tellex 2020) to the recent LLM-based works (Chen et al. 2023; Pan, Chou, and Berenson 2023; Liu et al. 2023b). However, a shared challenge in these studies is the limited availability of training data. While LLMs show great improvement in translation, their performance deteriorates with increased complexity, posing difficulties in generalizing to out-of-distribution formulas.

LTL to NL works such as (Cherukuri, Ferrari, and Spoletini 2022) are still in an early stage, likely due to the challenge of learning representations of high-level abstractions of LTL and deterministic finite automatons (DFAs). Our approach tackles the problem by providing state information as knowledge representation of the DFA. We believe such backward translation could serve as a valuable interface between humans and LTL-based robot systems.

Method

Problem Definition

For an existing LLM agent making decisions based on in-context knowledge l_{agent}, we enforce a series of temporal constraints $\{c_i\}$ concerning safety aspects by encoding each of them into LTL formulas $\{\varphi_i\}$ and composing them together into one single formula $\boldsymbol{\varphi} = \bigwedge_{i=0}^{\infty} \varphi_i$, which is then stored as a DFA A . Responding to a safety constraint query Q, our module generates a response W via a language model based on $P(W|l_{env}, l_A)$, where l_{env} is the domain knowledge, l_A is an explanation of constraint violation generated from $P(l_A|l_{agent}, Q, T_{0:i})$. To monitor the LLM agent executing a high-level task T, which ought to be decomposed by the agent into a set of subtasks or actions $\{t_i\}$, we mask unsafe action t_i at time step i by progressing its partial trajectory $\{t_0, t_1, ..., t_i\}$ with A, and regenerate the action t'_i after modifying its distribution from $P(t'_i|l_{agent}, T_{0:i-1})$ to $P(t'_i|l_{agent}, T_{0:i-1}, l_A)$ with the explanation l_A.

Overview

As Figure 2 shows, our system consists of a translator system for NL to LTL translation, a dialogue system for queryable safety constraints, and a constraint monitor system for validating and pruning actions generated by the LLM agent. The DFA serves as the central part of the safety constraints module: it represents the safety constraints encoded from the LTL formula in a validatable form, reasons the violation of constraints with state changes of propositions, and validates the agent's proposed action by progressing the proposition-level trajectory. In addition, the output

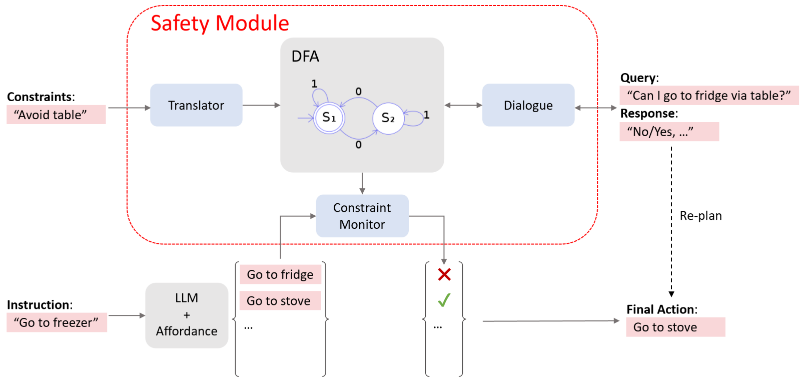

Figure 2: System Diagram. The safety constraint module infers specification of constraints from NL and stores it as DFA. The DFA is then used for monitoring the decision-making process of an LLM agent, and responses of queries from human or agent can be generated accordingly to confirm the encoded constraints or help the LLM agent with decision-making

of the dialogue system is also used to assist the agent in performing re-planning.

NL to LTL Translation

We adopted the modular framework of Lang2LTL (Liu et al. 2023b) as NL to domain-specific LTL formula **translator** using a predefined vocabulary. This involves extracting referring expressions, grounding referring expressions to propositions within the vocabulary, translating lifted utterances to formulas, and finally producing grounded formulas by replacing placeholders with grounded propositions (see Figure 3). The noteworthy distinction is that our translation module relies solely on in-context learning and necessitates no fine-tuning due to the compositionality nature of our approach: the safety constraints are assumed to be provided as a series of basic segments. This enables separate translation and assembly into a single LTL formula with logical operators, which is usually logical AND (&) since the constraints commonly apply simultaneously. For more details, we recommend referring to the original Lang2LTL paper.

Queryable Safety Constraints

The **dialogue system** serves two purposes: confirming the module's belief regarding safety constraints and assisting LLM agents in effective re-planning. Similar to the translation system, the query system relies entirely on prompting. As recent research (Ji et al. 2023) shows, LLMs are trapped

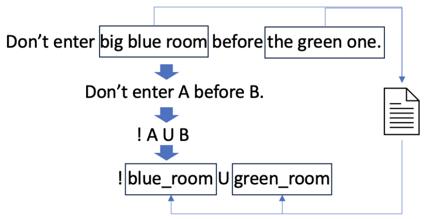

Figure 3: Lang2LTL translation example. Referring expressions are extracted and grounded into predefined propositions, and lifted NL utterance is translated into a lifted formula. The two are combined to produce the final result.

by hallucination and generally unreliable for classification tasks, making them unsuitable to be directly applied for validation detection in our case. Fortunately, the DFA can provide action validity through trajectory progression, thereby reducing LLMs' role to explanation. The dialogue system handles two query types:

Agent query is only sent to the dialogue system when the LLM agent violates given safety constraints in its next action. After a violation is detected by the constraint monitor, safety constraints, valid actions and invalid actions, and its corresponding truth value changes are provided in the prompt together with task specifications, which explicitly instructs the language model to explain the reason for violation for robot agent to re-plan (see the action pruning section).

Human query could be sent anytime to confirm the agent's belief of encoded constraints are aligned with human users through queries such as "Will you go to the kitchen before the bathroom?". In order to get trajectory for validation, the original LLM agent roll-outs the full trajectory (no execution) without monitored by the safety chip, and the trajectory is then progressed by the DFA. If any violation is detected, a query regarding validity and state changes will be sent to the dialogue system similar to the agent query. Here is an example prompt of dialogue system to handle **agent query**:

```
Constraints: ['you have to enter living room before bathroom']
Invalid action: walk to bathroom
State change:
Safe: !bathroom & bedroom & !kitchen & !living_room
Violated: bathroom & bedroom & !kitchen & !living_room
Reason of violation:
```

Unsafe Action Pruning

Pruning unsafe actions performed by the **constraint monitor** module constitutes the core functionality of the safety chip. Since we keep tracking the truth values of each proposition in the LTL formula as the LLM agent is planning and executing, state transition in the DFA can also be monitored. Thus, assuming we have perfect truth value functions and transition model, any violation of the safety constraints can be detected in advance, and we can mask out the unsafe action and ask the LLM agent to re-generate the plan from the last step. However, masking alone doesn't provide any information to the LLM agent, causing the agent to likely choose an action within the same distribution of the masked unsafe action since its distribution remains unchanged. Inspired by (Raman et al. 2022; Yao et al. 2023), in addition to masking out unsafe actions, we utilize the dialogue system to provide feedback to the agent regarding reasons for violation. This approach prompts the LLM agent to reason over explanations for validation results, introducing an inductive bias from the dialogue system.

An example implementation of re-planning for correction in the generation loop is shown as follows (task specification and examples are removed for simplicity):

Go to toilet
Description: Travel through the house and locate the bathroom, and proceed to the toilet.
0. walk to bathroom
Error: The user is trying to enter the bathroom before entering the living room, which violates the constraint "you have to enter living room before bathroom". The correct plan would be:
0. walk to living room
Error: The action "walk to livingroom" violates the constraint "you have to enter bedroom before going into living room". According to the state change, the user is currently in the kitchen and has not entered the bedroom yet. The correct plan would be:
0. walk to bedroom
1. walk to living room
2. walk to bathroom
3. walk to toilet
4. DONE

Experiments

Experiments are conducted in VirtualHome environment. In the experiment section, we aim to draw comparisons between the proposed method and baseline method where both instruction and constraints are fed together to LLM agents.

VirtualHome Environment & Tasks

VirtualHome (Puig et al. 2018) is a multi-agent platform that supports simulated household tasks in embodied environments. We adopted five household tasks from its original dataset, e.g., "Put salmon into the fridge", and five satisfiable constraints are incrementally applied to each task, e.g., "Go to living room means you have to go to kitchen in the future." All constraints fall in avoidance and trigger patterns formulated by (Menghi et al. 2021). Task difficulty is carefully controlled as all tasks can be completed successfully by both our approach and the baseline method when no constraint applies, and all tasks are ensured to be achievable with any constraint set. Through the experiment, we access environmental information such as locations and states of objects from the simulator for truth values.

LLM Agent

The LLM agent for both the baseline and the foundation framework for the proposed safety module is developed on

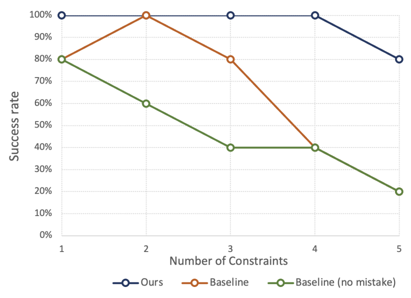

Figure 4: Average success rate versus number of constraints

SayCan (Ahn et al. 2022b) and LLM-Planner (Liu et al. 2023a). During execution, the agent generates one action at a time and then appends it to the prompt for the subsequent generation loop. In its prompt, environmental information and available actions are provided together with an example of planning under constraints.

Preliminary Results

Figure 4 shows the results of the average task success rate of the five tasks with an increasing number of constraints from one to five. The performance of the baseline model worsens as the number of constraints increases from zero to five. Notably, the baseline model also makes unanticipated *mistakes* when handling instruction together with constraints, though manages to reach the specified goal state. The mistakes include redundant actions such as *touch door*, and inexecutable actions that overlook geometric features such as *walk to bedroom* when the agent is already in *bedroom*. Conversely, our proposed method produces fewer mistakes and only fails when LLM never generates a safe action due to incorrect reasoning or imperfect explanation generated by the dialogue system. Moreover, our system never executes unsafe actions, regardless of task success.

Conclusion & Future Work

To address the safety concern of deploying LLM agents in practice, we introduced a safety module that supports customizable temporal constraints by simultaneously encoding NL constraints, confirming the encoded safety constraints by interacting with human users, and monitoring and assisting the decision-making process of an LLM agent. From the preliminary results and for future works, we aim to prove three hypotheses:

- Safety constraints are challenging and distracting for LLM agents to handle as their complexity increases.
- Ablating things from different abstraction levels holds the potential to aid LLM agents in concentrating on reasoning, to effectively function as a "brain".
- Formal language like LTL that can be validated is generally more reliable and interpretable than fully end-to-end approaches, particularly concerning the safety aspects of robot agents.

Lastly, we emphasize the pressing need for more attention to safety aspects from the LLM agents community, as these considerations are indispensable for any practical utilization.

References

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Ho, D.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jang, E.; Ruano, R. J.; Jeffrey, K.; Jesmonth, S.; Joshi, N. J.; Julian, R.; Kalashnikov, D.; Kuang, Y.; Lee, K.-H.; Levine, S.; Lu, Y.; Luu, L.; Parada, C.; Pastor, P.; Quiambao, J.; Rao, K.; Rettinghouse, J.; Reyes, D.; Sermanet, P.; Sievers, N.; Tan, C.; Toshev, A.; Vanhoucke, V.; Xia, F.; Xiao, T.; Xu, P.; Xu, S.; Yan, M.; and Zeng, A. 2022a. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. arXiv:2204.01691.

Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; et al. 2022b. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

Chen, B.; Xia, F.; Ichter, B.; Rao, K.; Gopalakrishnan, K.; Ryoo, M. S.; Stone, A.; and Kappler, D. 2022. Open-vocabulary Queryable Scene Representations for Real World Planning. arXiv:2209.09874.

Chen, Y.; Gandhi, R.; Zhang, Y.; and Fan, C. 2023. NL2TL: Transforming Natural Languages to Temporal Logics using Large Language Models. *arXiv preprint arXiv:2305.07766*.

Cherukuri, H.; Ferrari, A.; and Spoletini, P. 2022. Towards Explainable Formal Methods: From LTL to Natural Language with Neural Machine Translation. In Gervasi, V.; and Vogelsang, A., eds., *Requirements Engineering: Foundation for Software Quality*, 79–86. Cham: Springer International Publishing. ISBN 978-3-030-98464-9.

Gopalan, N.; Arumugam, D.; Wong, L.; and Tellex, S. 2018. Sequence-to-Sequence Language Grounding of Non-Markovian Task Specifications. In *Proceedings of Robotics: Science and Systems*. Pittsburgh, Pennsylvania.

Gu, S.; Yang, L.; Du, Y.; Chen, G.; Walter, F.; Wang, J.; Yang, Y.; and Knoll, A. 2023. A Review of Safe Reinforcement Learning: Methods, Theory and Applications. arXiv:2205.10330.

Huang, W.; Abbeel, P.; Pathak, D.; and Mordatch, I. 2022a. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. arXiv:2201.07207.

Huang, W.; Xia, F.; Shah, D.; Driess, D.; Zeng, A.; Lu, Y.; Florence, P.; Mordatch, I.; Levine, S.; Hausman, K.; and Ichter, B. 2023. Grounded Decoding: Guiding Text Generation with Grounded Models for Robot Control. arXiv:2303.00855.

Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; Sermanet, P.; Brown, N.; Jackson, T.; Luu, L.; Levine, S.; Hausman, K.; and Ichter, B. 2022b. Inner Monologue: Embodied Reasoning through Planning with Language Models. arXiv:2207.05608.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y. J.; Madotto, A.; and Fung, P. 2023. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12): 1–38.

Kress-Gazit, H.; Lahijanian, M.; and Raman, V. 2018. Synthesis for Robots: Guarantees and Feedback for Robot Behavior. *Annu. Rev. Control. Robotics Auton. Syst.*, 1: 211–236.

Lasota, P. A.; Fong, T.; and Shah, J. A. 2017. A Survey of Methods for Safe Human-Robot Interaction. *Foundations and Trends® in Robotics*, 5(4): 261–349.

Liang, J.; Huang, W.; Xia, F.; Xu, P.; Hausman, K.; Ichter, B.; Florence, P.; and Zeng, A. 2022. Code as Policies: Language Model Programs for Embodied Control. In *arXiv preprint arXiv:2209.07753*.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023a. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. arXiv:2304.11477.

Liu, J. X.; Shah, A.; Rosen, E.; Konidaris, G.; and Tellex, S. 2022. Skill Transfer for Temporally-Extended Task Specifications. *arXiv preprint arXiv:2206.05096*.

Liu, J. X.; Yang, Z.; Idrees, I.; Liang, S.; Schornstein, B.; Tellex, S.; and Shah, A. 2023b. Lang2LTL: Translating Natural Language Commands to Temporal Robot Task Specification. arXiv:2302.11649.

Menghi, C.; Tsigkanos, C.; Pelliccione, P.; Ghezzi, C.; and Berger, T. 2021. Specification Patterns for Robotic Missions. *IEEE Transactions on Software Engineering*, 47(10): 2208–2224.

Pacheck, A.; and Kress-Gazit, H. 2022. Physically-Feasible Repair of Reactive, Linear Temporal Logic-based, High-Level Tasks. arXiv:2207.02834.

Pan, J.; Chou, G.; and Berenson, D. 2023. Data-Efficient Learning of Natural Language to Linear Temporal Logic Translators for Robot Task Specification. *arXiv preprint arXiv:2303.08006*.

Patel, R.; Pavlick, E.; and Tellex, S. 2020. Grounding language to non-markovian tasks with no supervision of task specifications. In *Robotics: Science and Systems*.

Pnueli, A. 1977. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, 46–57. ieee.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8494–8502.

Raman, S. S.; Cohen, V.; Rosen, E.; Idrees, I.; Paulius, D.; and Tellex, S. 2022. Planning with Large Language Models via Corrective Re-prompting. *arXiv preprint arXiv:2211.09935*.

Shah, A.; Li, S.; and Shah, J. 2020. Planning with uncertain specifications (puns). *IEEE Robotics and Automation Letters*, 5(2): 3414–3421.

Shah, D.; Osinski, B.; Ichter, B.; and Levine, S. 2022. LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action. arXiv:2207.04429.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2022.

Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*.

Valmeekam, K.; Olmo, A.; Sreedharan, S.; and Kambhampati, S. 2023. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). arXiv:2206.10498.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; Zhao, W. X.; Wei, Z.; and Wen, J.-R. 2023. A Survey on Large Language Model based Autonomous Agents. arXiv:2308.11432.

Weng, L. 2023. LLM-powered Autonomous Agents. *lilianweng.github.io*.

Xie, Y.; Yu, C.; Zhu, T.; Bai, J.; Gong, Z.; and Soh, H. 2023. Translating Natural Language to Planning Goals with Large-Language Models. arXiv:2302.05128.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629.