Dynamic planning with an LLM

Dynamic Planning with a LLM

Gautier Dagan **Frank Keller** **Alex Lascarides**
School of Informatics
University of Edinburgh, UK
gautier.dagan@ed.ac.uk, {keller, alex}@inf.ed.ac.uk

Abstract

While Large Language Models (LLMs) can solve many NLP tasks in zero-shot settings, applications involving embodied agents remain problematic. In particular, complex plans that require multi-step reasoning become difficult and too costly as the context window grows. Planning requires understanding the likely effects of one's actions and identifying whether the current environment satisfies the goal state. While symbolic planners find optimal solutions quickly, they require a complete and accurate representation of the planning problem, severely limiting their use in practical scenarios. In contrast, modern LLMs cope with noisy observations and high levels of uncertainty when reasoning about a task. Our work presents LLM Dynamic Planner (LLM-DP): a neuro-symbolic framework where an LLM works hand-in-hand with a traditional planner to solve an embodied task. Given action-descriptions, LLM-DP solves Alfworld faster and more efficiently than a naive LLM ReAct baseline.

1 Introduction

Large Language Models (LLMs), like GPT-4 (OpenAI, 2023), have proven remarkably effective at various natural language processing tasks, particularly in zero-shot or few-shot settings (Brown et al., 2020). However, employing LLMs in embodied agents, which interact with dynamic environments, presents substantial challenges. LLMs tend to generate incorrect or spurious information, a phenomenon known as hallucination, and their performance is brittle to the phrasing of prompts (Ji et al., 2022). Moreover, LLMs are ill-equipped for naive long-term planning since managing an extensive context over multiple steps is complex and resource-consuming (Silver et al., 2022; Liu et al., 2023).

Various approaches have aimed to mitigate some of these limitations. For instance, methods like Chain-of-Thought (Wei et al., 2022) and Self-Consistency (Wang et al., 2023b) augment the context with reasoning traces. Other, agent-based approaches, such as ReAct (Yao et al., 2023), integrate feedback from the environment iteratively, giving the agent the ability to take 'thinking' steps or to augment its context with a reasoning trace. However, these approaches frequently involve high computational costs due to the iterated invocations of LLMs and still face challenges dealing with the limits of the context window and recovering from hallucinations, which can compromise the quality of the plans.

Conversely, traditional symbolic planners, such as the Fast-Forward planner (Hoffmann and Nebel, 2001) or the BFS(f) planner(Lipovetzky et al., 2014), excel at finding optimal plans efficiently. But symbolic planners require problem and domain descriptions as prerequisites (McDermott, 2000), which hampers their applicability in real-world scenarios where it may be infeasible to achieve these high informational demands. For instance, knowing a complete and accurate description of the goal may not be possible before exploring the environment through actions.

Previous work by (Liu et al., 2023) has shown that LLMs can generate valid problem files in the Planning Domain Definition Language (PDDL) for many simple examples. Yet, the problem of incomplete information remains: agents often need to interact with the world to discover their surroundings before optimal planning can be applied. Some versions of PDDL have been proposed in the past to deal with probabilities or Task and Motion Planning, such as PPDDL and PDDLStream (Younes and Littman, 2004; Garrett et al., 2018), but these still assume a human designer encoding the agent's understanding of the domain and the planning problem, rather than the agent learning from interactions. Therefore, where modern LLMs need minimal information to figure out a task, e.g. through Few-shot or In-Context Learning (Honovich et al.,

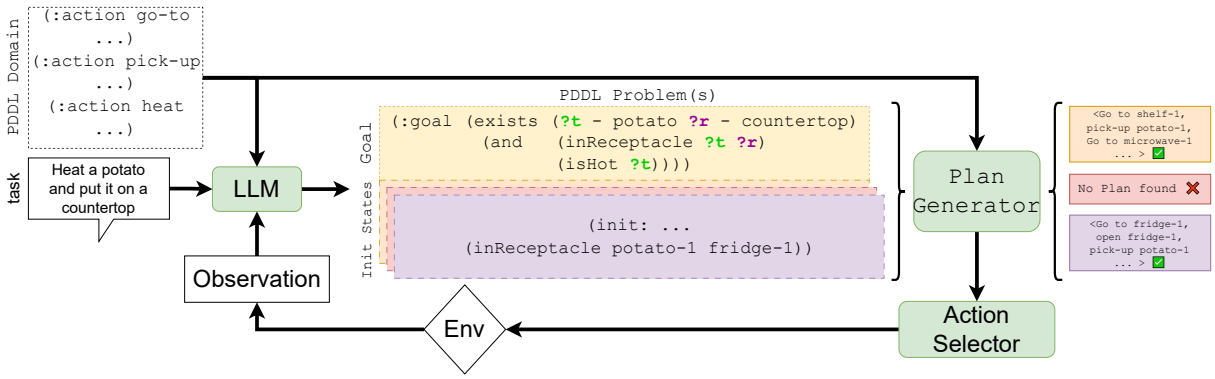

Figure 1: LLM Dynamic Planner (LLM-DP). The LLM grounds observations and processes natural language instructions into PDDL to use with a symbolic planner. This model can solve plans for unobserved or previously unknown objects because the LLM generates plausible predicates for relevant objects through semantic and pragmatic inference. Through sampling possible predicates, multiple plans can be found, and an Action Selector decides whether to act, review its understanding of the problem, or ask clarification questions.

2022; Chen et al., 2022; Min et al., 2022), traditional planners need maximal information.

In this work, **we introduce the LLM Dynamic Planner (LLM-DP), a neuro-symbolic framework that integrates an LLM with a symbolic planner to solve embodied tasks.**[1] LLM-DP capitalises on the LLM's ability to understand actions and their impact on their environment and combines it with the planner's efficiency in finding solutions. Using domain knowledge, LLM-DP solves the Alfworld test set faster and more efficiently than a LLM-only (ReAct) approach. The remainder of this paper explores the architecture of LLM-DP, discusses how to combine the strengths of LLMs and symbolic planning and presents potential research avenues for future work in LLM-driven agents.

2 Related Work

Symbolic Planners Symbolic planners have been a cornerstone in automated planning and artificial intelligence for decades (Fikes and Nilsson, 1971). Based on formal logic, they operate over symbolic representations of the world to find a sequence of actions that transition from an initial state to a goal state. Since the introduction of PDDL (McDermott, 2000), the AI planning community has developed an array of efficient planning algorithms. For example, the Fast-Forward planner (FF) (Hoffmann and Nebel, 2001) employs heuristics derived from a relaxed version of the planning problem. Similarly, the BFS(f) planner (Lipovetzky et al., 2014) combines breadth-first search and specialised heuristics. These planners find high-quality or optimal solutions quickly in well-defined domains. However, their up-front requirement for comprehensive problem and domain descriptions limits their applicability in complex real-world settings where complete information may not be available.

LLMs in Planning and Reasoning In contrast to symbolic planners, LLMs have shown promise in adapting to noisy planning and reasoning tasks through various methods. Some general approaches such as Chain-of-Thought (Wei et al., 2022), Self-Consistency (Wang et al., 2023b), and Reasoning via Planning (Hao et al., 2023) augment the context with a reasoning trace that the LLM generates to improve its final prediction. Alternatively, giving access to tools/APIs (Schick et al., 2023; Patil et al., 2023), outside knowledge or databases (Peng et al., 2023; Hu et al., 2023), code (Surís et al., 2023), and even symbolic reasoners (Yang et al., 2023) to enrich an LLM's context and ability to reason. The LLM can trigger these external sources of information or logic (through fine-tuning or prompting) to obtain additional context and improve its downstream performance.

Embodied Agents with LLMs In a parallel direction, recent works such as ReAct (Yao et al., 2023), Reflexion (Shinn et al., 2023), AutoGPT (Significant-Gravitas, 2023), and Voyager (Wang et al., 2023a), take an agent-based approach and augment the reasoning process through a closed 'while' loop that feeds environment observations back to the LLM. ReAct (Yao et al., 2023) allows the LLM agent to either take an action or a 'thinking' step. This allows the LLM to augment its context with its reasoning, which can be seen as

[1] Our code is available at github.com/itl-ed/llm-dp

agent-driven Chain-of-Thought prompting. Voyager (Wang et al., 2023a) incrementally builds an agent's capabilities from its interactions with the environment and an accessible memory component (skill library). While many of these works show promising results in building general executable agents in embodied environments (Wang et al., 2023a), they still require many expensive calls to the LLMs, are limited by the LLM's context window, and do not guarantee optimal plans.

3 Alfworld

Alfworld (Shridhar et al., 2020) is a text-only home environment where an agent is tasked with seven possible tasks, such as interacting with one or more objects and placing them in a specific receptacle. At the start of each episode, the goal is given in natural language, and the initial observation does not include the location of any objects. Therefore an agent must navigate the environment to search for the relevant objects and perform the correct actions. The possible locations of the environment are known, and the agent can navigate to any receptacle by using a 'go to' action. However, since none of the objects' locations are initially observed, the agent must be able to plan around uncertainty, estimate where objects are likely to be observed and adjust accordingly.

4 LLM-DP

To tackle an embodied environment like Alfworld, we introduce the Large Language Model Dynamic Planner (LLM-DP), which operates as a closed-loop agent. LLM-DP uses a combination of language understanding and symbolic reasoning to plan and solve tasks in the simulated environment. The model tracks a World State $\mathcal{W}$ and beliefs $\mathcal{B}$ about predicates in the environment, uses an LLM to translate the task description into an executable goal state and samples its beliefs to generate plausible world states. We describe the working of the LLM-DP agent as pseudo-code in Algorithm 1.

4.1 Assumptions

We make several simplifying assumptions when applying the LLM-DP framework to Alfworld:

1. **Known action-descriptions and predicates**: Our input to the planner and the LLM requires the `PDDL` domain file, which describes what actions can be taken, their pre- and post-conditions, and what predicates exist.

Algorithm 1 LLM-DP Pseudo-code

Require: LLM, PG, AS, Domain, $task$, obs_0
  $goal \leftarrow$ LLM(Domain, $task$)
  $\mathcal{W}, \mathcal{B} \leftarrow$observe($goal, obs_0$)
  while $goal$ not reached **do**
    $plans \leftarrow \emptyset$
    for i in N **do**
      $w_{belief} \leftarrow$LLM($\mathcal{B}, \mathcal{W}$)
      $plans \leftarrow$PG($w_{belief} \bigcup \mathcal{W}$)
    end for
    $action \leftarrow$AS($plans$)
    $obs \leftarrow$Env($action$)
    $\mathcal{W}, \mathcal{B} \leftarrow$observe($action, obs$)
  end while

2. **Perfect observations**: The Alfworld environment provides a perfect textual description of the current location. This observation also contains the intrinsic attributes of observed objects and receptacles, such as whether or not a given receptacle can be opened.
3. **Causal Environment**: changes in the environment are entirely caused by the agent.
4. ***Valid* actions always succeed**

4.2 Generating the Goal State

LLM-DP uses an LLM to generate a `PDDL` goal, given the natural language instruction ($task$) and the valid predicates defined by the `PDDL` domain file. Figure 1 shows an example task converted to a valid `PDDL` goal. For each episode, we use a set of three in-context examples that are fixed for the entire evaluation duration. We use the OpenAI `gpt-3.5-turbo-0613` LLM model with a temperature of 0 in all our LLM-DP experiments.

4.3 Sampling Beliefs

We parse the initial scene description into a structured representation of the environment $\mathcal{W}$ and a set of beliefs $\mathcal{B}$. The internal representation of the world $\mathcal{W}$ contains all *known* information, for instance, all receptacles (possible locations) in the scene from the initial observation and their intrinsic attributes are known (i.e. a fridge holds the `isFridge` predicate). Whereas the set of beliefs $\mathcal{B}$ are a set of possible valid predicates that can be true or false and which the model does not have enough information to disambiguate. In Alfworld, the objects' locations are unknown; therefore, the set of possible predicates for each object includes all possible locations.

Model	Average Accuracy (%) clean	cool	examine	heat	put	puttwo	overall (↑)	LLM Tokens (↓)
LLM-DP	0.94	1.00	1.00	0.87	1.00	0.94	0.96	633k
LLM-DP-random	0.94	1.00	1.00	0.87	0.96	1.00	0.96	67k
ReAct (Yao et al., 2023)	0.61	0.81	0.89	0.30	0.79	0.47	0.64	—*
ReAct (ours)	0.35	0.90	0.33	0.65	0.71	0.29	0.54	9.16M

(a) The average accuracy and number of LLM Tokens processed (context + generation) for each model. *Not reported.

Model	Average Episode Length clean	cool	examine	heat	put	puttwo	overall (↓)
LLM-DP	12.00	13.67	12.06	12.30	12.75	17.59	13.16
LLM-DP-random	15.06	17.14	10.56	14.04	14.62	18.94	15.02
ReAct (ours)	25.10	9.86	21.67	14.70	15.33	24.94	18.69

(b) The average episode length for each model, where the length of an episode denotes how many actions the agent has taken or attempted to take to complete a task. We do not count the 'thinking' action of ReAct as an action in this metric.

Table 1: Summary of model performance on the Alfword test set. LLM-DP and LLM-DP-random differ in the sampling strategy of the belief. LLM-DP uses an LLM to generate $n = 3$ plausible world states, while LLM-DP-random randomly samples $n = 3$ plausible world states.

LLM-DP uses stored observations $\mathcal{W}$, beliefs $\mathcal{B}$ and an LLM to construct different planning problem files in PDDL . A PDDL problem file includes the objects observed (:objects), a representation of the current state (:init) of the world and the object attributes, and the goal to be achieved (:goal). The goal is derived from the LLM (Section 4.2), while the objects and their attributes are obtained from $\mathcal{W}$ (observations) and the beliefs the $\mathcal{B}$ has about the objects.

Since $\mathcal{B}$ includes possible predicates which are unknown, we sample from $\mathcal{B}$ using an LLM to obtain w_{belief}. For instance, our belief could be that (inReceptacle tomato ?x) where ?x can be countertop, cabinet, fridge, etc. Since we want to condition the sampling of where the tomato can appear, we pass the known world state $\mathcal{W}$ along with the predicate (in this case inReceptacle) and its options to the LLM.This sampling leverages the LLM to complete a world state and is extendable to any unknown predicate from which a set of beliefs can be deduced. We also compare LLM sampling with random sampling (llmdp-random).

We describe our likely world state as the union between a sampled set of beliefs and the known world state $w_{belief} \bigcup \mathcal{W}$. Then sampling $i = 1, .., N$ different sets of beliefs during the planning loop, we obtain N likely world states. Finally, we convert each likely world state to lists of predicates to interface with the PDDL planner.

4.4 Plan Generator

Upon constructing the different PDDL problems, the agent uses a Plan Generator (PG) to solve each problem and obtain a plan. We use the BFS(f) solver (Lipovetzky et al., 2014) implemented as an executable by LAPKT (Ramirez et al., 2015). A generated plan is a sequence of actions, where each action is represented in a symbolic form, which, if executed, would lead to the goal state from the initial state.

4.5 Action Selector

The Action Selector (AS) module decides the agent's immediate next action. It takes the planner's output, a set of plans, and selects an action from them. In our Alfworld experiments, the Action Selector simply selects the shortest plan returned. If no valid plans are returned, all sampled states were satisfying goal states, there is a mistake with the constructed domain/problem files, or the planner has failed to find a path to the goal. In the first case, we re-sample random world states and re-run the planners once.

We also propose exploring different strategies when valid plans cannot be found. For instance, similarly to self-reflection (Shinn et al., 2023), the Action Selector could prompt an update in the agent's belief about the world state if none of generated problem descriptions are solvable. The Action Selector could also interact with a human teacher

or oracle to adjust its understanding of the environment (problem) or its logic (domain).

4.6 Observation Processing

LLM-DP uses the result of each action to update its internal state representation. It uses the symbolic effects of the action to infer changes in the state of the objects and receptacles. Then it integrates the information from the new observation, which might reveal additional details not directly inferred from the action itself. For instance, opening an unseen drawer might reveal new objects inside. Observing also updates the beliefs – if an object is observed at a location, it cannot be elsewhere, but if an object is not observed at a location, it cannot be there. Observations incorporate beliefs into $\mathcal{W}$.

If the agent detects new information from the scene - such as discovering new objects - it triggers a re-planning process. The agent then generates a new set of possible PDDL problems using the updated state representation and corresponding plans using the Plan Generator. This approach is similar to some Task and Motion Planning (TAMP) methods (Garrett et al., 2018; Chen et al., 2023), enabling the agent to adapt to environmental changes and unexpected outcomes of actions.

5 Results

We contrast the LLM-DP approach with ReAct (LLM-only baseline) from the original implementation by Yao et al. (2023). Since we use a different backbone LLM model (`gpt-3.5-turbo` rather than `text-davinci-002`) than the ReAct baseline for cost purposes, we also reproduce their results using `gpt-3.5-turbo` and adapt the ReAct prompts to a chat format.

As shown in Table 1, LLM-DP solves Alfworld almost perfectly (96%) compared to our baseline reproduction of ReAct (53%). The LLM-DP can translate the task description into an executable PDDL goal 97% of the time, but sampling reduces the accuracy further when it fails to select a valid set of possible world states – for instance, by sampling states where the goal is already satisfied.

We note, that the ReAct baseline makes different assumptions about the problem; while it does not require a domain file containing the action-descriptions and object predicates, it uses two separate human-annotated episodes per example to bootstrap its in-context logic. ReAct also switches out which examples to use in-context based on the type of task, such that two examples of the same type of task being solved are always shown. We also find that our reproduction of ReAct is worse than the original and attribute this to the `gpt-3.5-turbo` model being more conversational than `text-davinci-002`, and thus less likely to output valid actions as it favours fluency over following the templated action language.

We also measure the length of each successful episode and find that LLM-DP reaches the goal state faster on average (13.16 actions) versus ReAct (18.69 actions) and a random search strategy (15.02 actions). The Average Episode Length measures the number of actions taken in the environment and how efficient the agent is.

6 Conclusion

The LLM-DP agent effectively integrates language understanding, symbolic planning, and state tracking in a dynamic environment. It uses the language model to understand tasks and scenes expressed in natural language, constructs and solves planning problems to decide on a course of action, and keeps track of the world state to adapt to changes and make informed decisions. This workflow enables the agent to perform complex tasks in the Alfworld environment, making it a promising approach for embodied tasks that involve language understanding, reasoning, and decision-making.

LLM-DP offers a cost and efficiency trade-off between a wholly symbolic solution and an LLM-only model. The LLM's semantic knowledge of the world is leveraged to translate the problem into PDDL while guiding the search process through belief instantiation. We find that not only is LLM-DP cheaper, on a per-token comparison, but it is also faster and more successful at long-term planning in an embodied environment. LLM-DP validates the need for LLM research to incorporate specialised tools, such as PDDL solvers, in embodied agents to promote valid

Despite these promising results, numerous topics and unresolved issues remain open for future investigation. Key among these is devising strategies to encode the world model and belief, currently handled symbolically, and managing uncertain observations — particularly from an image model — along with propagating any uncertainty to the planner and Action Selector. We intentionally kept the Action Selector simple for our experiments, but future work may also explore different strategies to

encourage self-reflection within the agent loop. For instance, if all plans prove invalid, beliefs may be updated, or it might indicate an incorrect domain definition. Such instances may necessitate agents to interact with an instructor who can provide insights about action pre-conditions and effects. This direction could lead us from a static domain file towards an agent truly adaptable to new environments, fostering continual learning and adaptation.

Acknowledgements

This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) at the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences and by the UKRI-funded TAS Governance Node (grant number EP/V026607/1).

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730, Dublin, Ireland. Association for Computational Linguistics.

Yongchao Chen, Jacob Arkin, Yang Zhang, Nicholas A. Roy, and Chuchu Fan. 2023. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. *ArXiv*, abs/2306.06531.

Richard E. Fikes and Nils J. Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208.

Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. 2018. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *International Conference on Automated Planning and Scheduling*.

Shibo Hao, Yilan Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *ArXiv*, abs/2305.14992.

Jörg Hoffmann and Bernhard Nebel. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2022. Instruction induction: From few examples to natural language task descriptions. *ArXiv*, abs/2205.10782.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Jake Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *ArXiv*, abs/2306.03901.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Wenliang Dai, Andrea Madotto, and Pascale Fung. 2022. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38.

Nir Lipovetzky, Miquel Ramirez, Christian Muise, and Hector Geffner. 2014. Width and inference based planners: Siw, bfs (f), and probe. *Proceedings of the 8th International Planning Competition (IPC-2014)*, page 43.

B. Liu, Yuqian Jiang, Xiaohan Zhang, Qian Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. Llm+p: Empowering large language models with optimal planning proficiency. *ArXiv*, abs/2304.11477.

Drew McDermott. 2000. The 1998 ai planning systems competition. *AI Magazine*, 21(2):35–55.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Conference on Empirical Methods in Natural Language Processing*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*. Computation and Language (cs.CL); Artificial Intelligence (cs.AI).

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Lidén, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *ArXiv*, abs/2302.12813.

Miquel Ramirez, Nir Lipovetzky, and Christian Muise. 2015. Lightweight Automated Planning ToolKiT. `http://lapkt.org/`. Accessed: 2020.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *ArXiv*, abs/2302.04761.

Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: An autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *CoRR*, abs/2010.03768.

Significant-Gravitas. 2023. An experimental open-source attempt to make gpt-4 fully autonomous. `https://github.com/significant-gravitas/auto-gpt`. Accessed: 2023-06-09.

Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. 2022. Pddl planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *ArXiv*, abs/2303.08128.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi (Jim) Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *ArXiv*, abs/2305.16291.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Zhun Yang, Adam Ishay, and Joohyung Lee. 2023. Coupling large language models with logic programming for robust and general reasoning from text. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 5186–5219. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Håkan LS Younes and Michael L Littman. 2004. Ppddl1. 0: An extension to pddl for expressing planning domains with probabilistic effects. *Techn. Rep. CMU-CS-04-162*, 2:99.

	SR	EL
LLM-DP (n=3)	0.96	13.16
LLM-DP (n=3) - fallback	0.92	12.80
LLM-DP (n=5)	0.96	12.54
LLM-DP (n=5) - fallback	0.94	12.24

Table 2: We compare the average Success Rate (SR) and average Episode Length (EL) for different sampling sizes n and with or without a fallback to random sampling. The random sampling fallback affects the success rate as the LLM sampler can more often sample n world states which are already satisfied. However as n increases, we see that it becomes more likely for the sampling procedure to at find at least one plan, and therefore the SR increases when no fallback (- fallback) is used.

A Prompts and Few-shot details

See Table 3 and Table 4 for LLM-DP prompts used.

B ReAct

B.1 Reproduction with Chat Model

We slightly modify the 'system' prompt of the original ReAct (see Table 5) to guide the model away from its conversational tendencies. `gpt-3.5-turbo` apologises significantly more than the `text-davinci-002` model, and we found that it would often get stuck in loops of apologising. We also modify the code so that we replace all generated instances of 'in' and 'on' with 'in/on' if the model did not generate it correctly, since Alfworld expects 'in/on' but `gpt-3.5-turbo` tends to generate only the correct preposition. Without these changes, ReAct would be significantly worse than our reported metric.

C LLM-DP

C.1 Generated Goal Examples

See Table 6 for examples of generated goals, both valid and invalid.

C.2 Varying n

See Table 6 for results when different varying n and fallback. Fallback is when no plans are sampled successfully through the LLM, LLM-DP resamples n plans randomly.

```
(define (domain alfred)
(:predicates
    (isReceptacle ?o - object) ; true if the object is a receptacle
    (atReceptacleLocation ?r - object) ; true if the robot is at the receptacle location
    (inReceptacle ?o - object ?r - object) ; true if object ?o is in receptacle ?r
    (openable ?r - object) ; true if a receptacle is openable
    (opened ?r - object) ; true if a receptacle is opened
    (isLight ?o - object) ; true if an object is light source
    (examined ?o - object ?l - object) ; whether the object has been looked at with light
    (holds ?o - object) ; object ?o is held by robot
    (isClean ?o - object) ; true if the object has been cleaned in sink
    (isHot ?o - object) ; true if the object has been heated up
    (isCool ?o - object) ; true if the object has been cooled
    (isSink ?o - object) ; true if the object is a sink
    (isMicrowave ?o - object) ; true if the object is a microwave
    (isFridge ?o - object) ; true if the object is a fridge
))
```

Table 3: System Prompt used by `gpt-3.5-turbo` for generating the `:goal` in LLM-DP

```
Your task is to: put a clean plate in microwave.
(:goal
(exists (?t - plate ?r - microwave)
(and (inReceptacle ?t ?r)
(isClean ?t)
)))

Your task is to: examine an alarmclock with the desklamp",
(:goal
(exists (?t - alarmclock ?l - desklamp)
(and (examined ?t ?l) (holds ?t)
)))

Your task is to: put two cellphone in bed
(:goal
(exists (?t1 - cellphone ?t2 - cellphone ?r - bed)
(and (inReceptacle ?t1 ?r)
(inReceptacle ?t2 ?r)
(not (= ?t1 ?t2))
)))
```

Table 4: Fixed Few-shot examples used by `gpt-3.5-turbo` for generating the `:goal` in LLM-DP

```
Interact with a household to solve a task.
Only reply with > followed by the action to take or 'think'.
Do not apologize.
Follow the format of the two examples below.
```

Table 5: System Prompt used by `gpt-3.5-turbo` in our reproduction of ReAct

```
task: put some peppershaker on drawer.
Generated:
(:goal
    (exists (?t - peppershaker ?r - drawer)
        (inReceptacle ?t ?r)
))

VALID ✓
```

```
task: put a clean mug in coffeemachine.
Generated:
(:goal
    (exists (?t - mug ?r - coffeemachine)
        (and (inReceptacle ?t ?r)
             (isClean ?t)
)))

VALID ✓
```

```
task: put two cd in safe.
Generated:
(:goal
    (exists (?t1 - cd ?t2 - cd ?r - safe)
        (and (inReceptacle ?t1 ?r)
             (inReceptacle ?t2 ?r)
             (not (= ?t1 ?t2))
)))
VALID ✓
```

```
task: heat some mug and put it in coffeemachine.
Generated:
(:goal
    (exists (?m - mug ?c - coffeemachine)
        (and (isReceptacle ?m)
             (isHot ?m)
             (inReceptacle ?m ?c)
)))
INVALID ✗
```

Table 6: Sample of generated PDDL goals from LLM-DP. The generation gets confused by the semantics of 'receptacle' and identifies a mug as a receptacle. While it is true that a mug is a receptacle, in our defined logic, receptacles are fixed, immovable objects which can contain other objects and therefore, a mug is not a Receptacle which leads the planning to fail subsequently.