SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Task Planning

Krishan Rana[1*], **Jesse Haviland**[1], **Sourav Garg**[2], **Jad Abou-Chakra**[1], **Ian Reid**[2], **Niko Sünderhauf**[1]
[1]QUT Centre for Robotics, Queensland University of Technology
[2]University of Adelaide
[*]ranak@qut.edu.au

Abstract:
Large language models (LLMs) have demonstrated impressive results in developing generalist planning agents for diverse tasks. However, grounding these plans in expansive, multi-floor, and multi-room environments presents a significant challenge for robotics. We introduce SayPlan, a scalable approach to LLM-based, large-scale task planning for robotics using 3D scene graph (3DSG) representations. To ensure the scalability of our approach, we: (1) exploit the hierarchical nature of 3DSGs to allow LLMs to conduct a *semantic search* for task-relevant subgraphs from a smaller, collapsed representation of the full graph; (2) reduce the planning horizon for the LLM by integrating a classical path planner and (3) introduce an *iterative replanning* pipeline that refines the initial plan using feedback from a scene graph simulator, correcting infeasible actions and avoiding planning failures. We evaluate our approach on two large-scale environments spanning up to 3 floors, 36 rooms and 140 objects, and show that our approach is capable of grounding large-scale, long-horizon task plans from abstract, and natural language instruction for a mobile manipulator robot to execute. We provide real robot video demonstrations and code on our project page sayplan.github.io.

1 Introduction

"Make me a coffee and place it on my desk" – The successful execution of such a seemingly straightforward command remains a daunting task for today's robots. The associated challenges permeate every aspect of robotics, encompassing navigation, perception, manipulation as well as high-level task planning. Recent advances in Large Language Models (LLMs) [1, 2, 3] have led to significant progress in incorporating common sense knowledge for robotics [4, 5, 6]. This enables robots to plan complex strategies for a diverse range of tasks that require a substantial amount of background knowledge and semantic comprehension.

For LLMs to be effective planners in robotics, they must be grounded in reality, that is, they must adhere to the constraints presented by the physical environment in which the robot operates, including the available affordances, relevant predicates, and the impact of actions on the current state. Furthermore, in expansive environments, the robot must additionally understand where it is, locate items of interest, as well comprehend the topological arrangement of the environment in order to plan across the necessary regions. To address this, recent works have explored the utilization of vision-based value functions [4], object detectors [7, 8], or Planning Domain Definition Language (PDDL) descriptions of a scene [9, 10] to ground the output of the LLM-based planner. However, these efforts are primarily confined to small-scale environments, typically single rooms with pre-encoded information on all the existing assets and objects present. The challenge lies in scaling these models. As the environment's complexity and dimensions expand, and as more rooms and entities enter the scene, pre-encoding all the necessary information within the LLMs context becomes increasingly infeasible.

To this end, we present a scalable approach to ground LLM-based task planners across environments spanning multiple rooms and floors. We achieve this by exploiting the growing body of 3D scene graph (3DSGs) research [11, 12, 13, 14, 15, 16]. 3DSGs capture a rich topological and

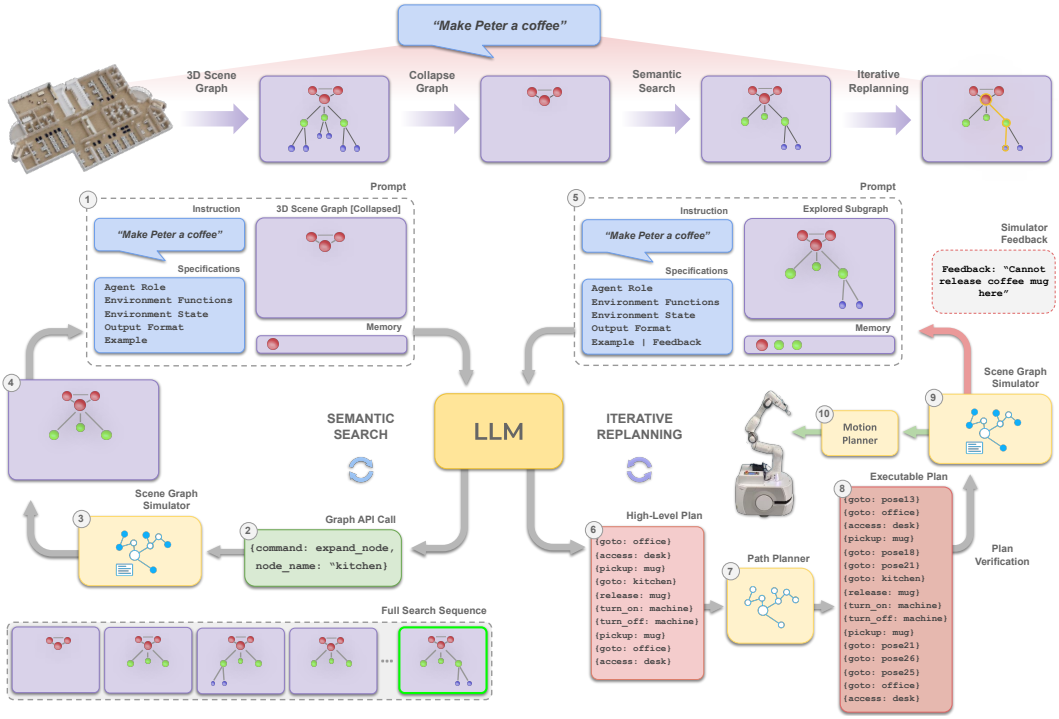


Figure 1: **SayPlan Overview (top).** SayPlan operates across two stages to ensure scalability: (left) Given a collapsed 3D scene graph and a task instruction, *semantic search* is conducted by the LLM to identify a suitable subgraph that contains the required items to solve the task; (right) The explored subgraph is then used by the LLM to generate a high-level task plan, where a classical path planner completes the navigational component of the plan; finally, the plan goes through an *iterative replanning* process with feedback from a scene graph simulator until an executable plan is identified. Numbers at the corners of the modules represent an example flow of operations.

hierarchically-organised semantic graph representation of an environment with the versatility to encode the necessary information required for task planning including object state, predicates, affordances and attributes using natural language – suitable for parsing by an LLM. We can leverage a JSON representation of this graph as input to a pre-trained LLM, however, to ensure the *scalability* of the plans to expansive scenes, we present three key innovations.

Firstly, we present a mechanism that enables the LLM to conduct a *semantic search* for a task-relevant subgraph $\mathcal{G}'$ by manipulating the nodes of a 'collapsed' graph, which exposes only the top level of the full 3DSG $\mathcal{G}$, via `expand` and `contract` API function calls – thus making it feasible to plan over increasingly large-scale environments. In doing so, the LLM maintains focus on the small, informative subgraph, $\mathcal{G}'$ during planning, without exceeding its token limit. Secondly, as the horizon of the task plans across such environments tend to grow with the complexity and range, there is an increasing tendency for LLMs to hallucinate or produce infeasible action sequences [17, 18, 7]. We counter this by relaxing the need for the LLM to generate the navigational component of the plan, and instead leverage an existing optimal path planner such as Dijkstra [19] to connect high-level nodes generated by the LLM. Finally, to ensure the feasibility of the proposed plan, we introduce an *iterative replanning* pipeline that verifies and refines the initial plan using feedback from a scene graph simulator in order to correct for any unexecutable actions, e.g., missing to open the fridge before putting something into it – thus avoiding planning failures due to inconsistencies, hallucinations, or violations of the physical constraints and predicates imposed by the environment.

Our approach SayPlan ensures feasible and grounded plan generation for a mobile manipulator robot operating in large-scale environments spanning multiple floors and rooms. We evaluate our framework across a range of 90 tasks organised into four levels of difficulty. These include semantic search tasks such as (*"Find me something non-vegetarian."*) to interactive, long-horizon tasks with ambiguous multi-room objectives that require a significant level of common-sense reasoning (*"Let's play a prank on Niko"*). These tasks are assessed in two expansive environments, including a large office floor spanning 36 rooms and 150 interactable assets and objects, and a three-storey house with 32 rooms and 121 objects. Our experiments validate SayPlan's ability to scale task planning to large-scale environments while conserving a low token footprint. By introducing a semantic search

pipeline, we can reduce full large-scale scene representations by up to 82.1% for LLM parsing and our iterative replanning pipeline allows for near-perfect executability rates, suitable for execution on a real mobile manipulator robot.[1]

2 Related Work

Task planning in robotics aims to generate a sequence of high-level actions to achieve a goal within an environment. Conventional methods employ domain-specific languages such as PDDL [20, 21, 22] and ASP [23] together with search techniques [24, 25] and complex heuristics [26] to arrive at a solution. The requirement to specify tasks and the environment via these languages as well as the need for such complex heuristics limits the versatility of the approach when scaling to larger environments and more complex tasks. Learning-based alternatives [27, 28], hierarchical and reinforcement learning [29], face challenges with data demands and scalability. Our work leverages the in-context learning capabilities of LLMs to generate feasible long-horizon task plans across 3D scene graphs, a scalable approach to representing large-scale scenes. Tasks, in this case, can be naturally expressed using natural language, and search heuristics are derived by the LLM based on its internet scale knowledge, the task's semantics and the information present within the scene graph representation.

Task planning with LLMs, that is, translating natural language prompts into task plans for robotics, is an emergent trend in the field. Earlier studies have effectively leveraged pre-trained LLMs' in-context learning abilities to generate actionable plans for embodied agents [4, 10, 9, 8, 30, 7, 31]. A key challenge for robotics is grounding these plans within the operational environment of the robot. Prior works have explored the use of object detectors [8, 7], PDDL environment representations [10, 9, 32] or value functions [4] to achieve this grounding, however, they are predominantly constrained to single-room environments, and scale poorly with the number of objects in a scene which limits their ability to plan over multi-room or multi-floor environments. In this work, we explore the use of 3D scene graphs and the ability of LLMs to generate plans over large-scale scenes by exploiting the inherent hierarchical structure of these representations.

Integrating external knowledge in LLMs has been a growing line of research combining language models with external tools to improve the reliability of their outputs. In such cases, external modules are used to provide feedback or extra information to the LLM to guide its output generation. This is achieved either through API calls to external tools [33, 34] or as textual feedback from the operating environment [35, 8]. More closely related to our work, CLAIRIFY [36] iteratively leverage compiler error feedback to re-prompt an LLM to generate syntactically valid code. Building on these ideas, we propose an iterative plan verification process with feedback from a scene graph-based simulator to ensure all generated plans adhere to the constraints and predicates captured by the pre-constructed scene graph. This ensures the direct executability of the plan on a mobile manipulator robot, operating in the corresponding real-world environment.

3 SayPlan

3.1 Problem Formulation

We aim to address the challenge of long-range planning for an autonomous agent, such as a mobile manipulator robot, in a large-scale environment based on natural language instructions. This requires the robot to comprehend abstract and ambiguous instructions, understand the scene and generate task plans involving both navigation and manipulation of a mobile robot within an environment. Existing approaches lack the ability to reason over scenes spanning multiple floors and rooms. Our focus is on integrating large-scale scenes into planning agents based on Language Models (LLMs) and solving the scalability challenge. We aim to tackle two key problems: 1) representing large-scale scenes within LLM token limitations, and 2) mitigating LLM hallucinations and erroneous outputs when generating long-horizon plans in large-scale environments.

[1] sayplan.github.io

Algorithm 1: SayPlan

Given: scene graph simulator ψ, classical path planner ϕ, large language model LLM
Inputs: prompt $\mathcal{P}$, scene graph $\mathcal{G}$, instruction $\mathcal{I}$

$\mathcal{G}' \leftarrow \texttt{collapse}_{\psi}(\mathcal{G})$ ▷ collapse scene graph
Stage 1: Semantic Search ▷ search scene graph for all relevant items
while command != *terminate* **do**
  command, node_id $\leftarrow LLM(\mathcal{P}, \mathcal{G}', \mathcal{I})$
  if command == 'expand' **then**
    $\mathcal{G}' \leftarrow \texttt{expand}_{\psi}$(node_id) ▷ expand node to reveal objects and assets
  else if command == 'contract' **then**
    $\mathcal{G}' \leftarrow \texttt{contract}_{\psi}$(node_id) ▷ contract node if nothing relevant found
Stage 2: Causal Planning ▷ generate a feasible plan
while feedback != *success* **do**
  plan $\leftarrow LLM(\mathcal{P}, \mathcal{G}', I,$ feedback$)$ ▷ high level plan
  full_plan $\leftarrow \phi($plan$, \mathcal{G}')$ ▷ compute optimal navigational path between nodes
  feedback $\leftarrow \texttt{verify\_plan}_{\psi}$(full_plan) ▷ forward simulate the full plan
return full_plan ▷ executable plan

3.2 Preliminaries

Here, we describe the 3D scene graph representation of an environment and the components of the scene graph API which we leverage throughout our approach.

Scene Representation: The 3D Scene Graph (3DSG) [11, 12, 14] have recently emerged as an actionable world representation for robots [13, 15, 16, 37, 38, 39], which hierarchically abstracts the environment at multiple levels through spatial semantics and object relationships while capturing relevant states, affordances and predicates of the entities present in the environment. Formally, a 3DSG is a hierarchical multigraph $G = (V, E)$ in which the set of vertices V comprises $V_1 \cup V_2 \cup \ldots \cup V_K$, with each V_k signifying the set of vertices at a particular level of the hierarchy k. Edges stemming from a vertex $v \in V_k$ may only terminate in $V_{k-1} \cup V_k \cup V_{k+1}$, i.e. edges connect nodes within the same level, or one level higher or lower.

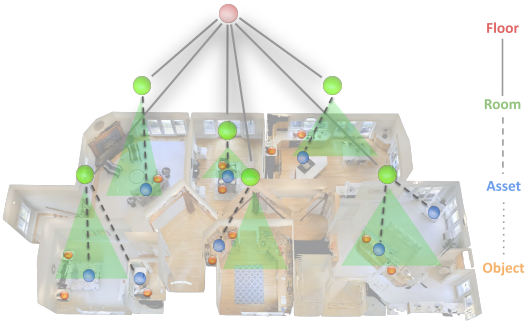


Figure 2: **Hierarchical Structure of a 3D Scene Graph.** This graph consists of 4 layers. Notes that the room nodes are connected to one another via sequences of pose nodes which capture the topological arrangement of a scene.

We assume a pre-constructed 3D scene graph representation of a large-scale environment generated using existing techniques [15, 13, 11]. The entire 3D scene graph can be represented as a NetworkX `Graph` object [40] and text-serialised into a JSON data format that can be parsed directly by a pre-trained LLM. An example of a single asset node from the 3D scene graph is represented as: `{name: coffee_machine, type: asset, location: kitchen, affordances: [turn_on, turn_off, release], state: off, attributes: "red", position: [2.34, 0.45, 2.23]}`. The 3D Scene Graph (3DSG) is organized in a hierarchical manner with four primary layers: floors, rooms, assets, and objects as shown in Figure 2. The top layer contains floors, each of which branches out to several rooms. These rooms are interconnected through pose nodes to represent the environment's topological structure. Within each room, we find assets (immovable entities) and objects (movable entities). Both asset and object nodes encode particulars such as state, affordances, additional attributes such as colour or weight, and 3D pose. The graph also incorporates a dynamic agent node, denoting a robot's location within the scene.

Scene Graph API: The LLM is given access to an external API which provides it with a set of tools required to manipulate and operate over 3DSGs. It enables the LLM to manipulate scene graphs through `expand` and `contract` functions, revealing connected nodes in a lower layer, or reversing the process respectively. Furthermore, generated plans can be verified through a task-agnostic

`scene graph simulator` which consists of a set of rules which verify if actions performed on the nodes adhere to the physical constraints, predicates and affordances present in the corresponding environment.

3.3 Approach

Given a 3D scene graph representations $\mathcal{G}$ and a task instruction $\mathcal{I}$ defined in natural language, we can view our framework SayPlan as a high-level task planner $\pi(\boldsymbol{a}|\mathcal{I}, \mathcal{G})$, capable of generating long-horizon plans $\boldsymbol{a}$ grounded in the large scale environment within which a mobile manipulator robot operates. The plan can then be fed to a low-level visually grounded motion planner for real-world execution. An overview of the SayPlan pipeline is illustrated in Figure 1 and the corresponding pseudo-code is given in Algorithm 1.

We address the challenges that arise when planning across these large-scale scenes by decomposing the planning pipeline into two key stages: *semantic search* and *iterative replanning*. During semantic search, the LLM explores from a collapsed representation of the full scene graph for a suitable *subgraph* $\mathcal{G}'$ that contains the necessary items required to solve the given task. This relaxes the need for providing the entire scene to the LLM, which would typically contain items that are not required for the given task, consuming unnecessary tokens from the LLM's input context. Once the desired subgraph is identified, the LLM switches to the iterative planning phase. Here, we reduce the planning horizon for an LLM by integrating a classical path planner to fill in optimal navigation paths within the high-level plan generated by the LLM. To ensure the plan adheres to the specific constraints and predicates imposed by the environment, it iteratively goes through a plan verification stage where it is executed within a scene graph environment. In the event of a failed plan, a feedback message is returned and appended to the output plan before being fed back to the LLM for replanning. This process is repeated until an executable plan is achieved. We provide more detail on each component in the following sections.

Semantic Graph Search: The semantic search phase begins with a collapsed representation of the full 3D scene graph $\mathcal{G}$, exposing only the highest level of the hierarchy to the LLM eg. room layer as shown in Figure 6. Given a natural language task description $\mathcal{I}$, the goal of this phase is to conduct a search, for a sub-graph $\mathcal{G}'$ which contains all the asset and object nodes necessary for solving the task. The search is governed by the LLM's common sense reasoning capabilities and in-context learning from a set of input-output examples [41, 42]. We leverage Chain-of-Thought (CoT) [18] reasoning to help the LLM decompose complex tasks into intermediate steps to facilitate its ability to decide on the appropriate nodes to `expand` or `contract` using the available API calls. At each step, the subgraph $\mathcal{G}'$ in the LLM's previous input is updated and passed again to the LLM until a suitable $\mathcal{G}'$ is identified. The ability to `contract` nodes that are not required for solving the task reduces the token footprint over the course of long search sequences (see Fig. 3). To avoid expanding already-contracted nodes, we maintain a list of expanded nodes, passed as an additional **`Memory`** input to the LLM. This leads to a fully Markovian decision-making process, where the current subgraph $\mathcal{G}'$ and the history of expanded nodes are the only state inputs required for the LLM to make its next decision. This allows it to scale to long search sequences, unlike [5] which has to maintain the full history of interactions. Once the LLM agent identifies that the current subgraph has visibility over all the assets and objects required to solve the task, it autonomously switches to the planning phase. An example of the LLM-scene graph interaction during semantic search is provided in Appendix I.

Iterative replanning: Given the identified subgraph $\mathcal{G}'$, we generate correct and feasible long-horizon task plans, via two key mechanisms. First, we shorten the LLM's planning horizon by delegating pose-level path planning to an optimal path planner, such as Dijkstra. For example, a typical plan output such as `[goto(meeting_room), goto(pose13), goto(pose14), goto(pose8), ..., goto(kitchen), access(fridge), open(fridge)]` is simplified to `[goto(meeting_room), goto(kitchen), access(fridge), open(fridge)]`. The path planner handles finding the optimal route between high-level locations, allowing the LLM to focus on essential manipulation components of the task. Secondly, we utilise the `scene graph simulator` to evaluate if the generated plan complies with the scene graph's predicates, state, and affordances. For instance, a `pick(banana)` action might fail if the robot is already holding something, if it is not in the correct location or if the fridge was not opened beforehand. Such failures are transformed into textual feedback (e.g., *"cannot pick banana"*), appended to the LLM's input, and used to generate an updated, executable plan. This iterative process, involving planning, validation,

Instruction Family	Num	Explanation	Example Instruction
		Semantic Search	
Simple Search	30	Queries focussed on evaluating the basic semantic search capabilities of SayPlan	Find me a ripe banana.
Complex Search	30	Abstract semantic search queries which require complex reasoning	Find the room where people are playing board games.
		Causal Planning	
Simple Planning	15	Queries which require the agent to perform search, causal reasoning and environment interaction in order to solve a task.	Refrigerate the orange left on the kitchen bench.
Long-Horizon Planning	15	Long Horizon planning queries requiring multiple interactive steps	Tobi spilt soda on his desk. Help him clean up.

Table 1: **List of evaluation task instructions.** We evaluate SayPlan on 90 instructions, grouped to test various aspects of the planning capabilities across large-scale scene graphs. The full instruction set is given in Appendix B.

and feedback integration, continues until a feasible plan is obtained. This plan is then passed to a low-level motion planner for robotic execution. An example of the LLM-scene graph interaction during iterative replanning is provided in Appendix J.

Implementation Details: We utilise `GPT-4` [3] as the underlying LLM agent unless otherwise stated. We follow a similar prompting structure to Wake et al. [5] as shown in Appendix H. We define the agent's role, details pertaining to the scene graph environment, the desired output structure and a set of input-output examples which together form the static prompt used for in-context learning. This static prompt is both task- and environment-agnostic and takes up approximately 3900 tokens of the LLMs input. During semantic search, both the **`3D Scene Graph`** and **`Memory`** components of the input prompt get updated, while during iterative planning only the **`Feedback`** component gets updated with information from the `scene graph simulator`.

4 Experimental Setup

We design our experiments to evaluate the 3D scene graph reasoning capabilities of LLMs with a particular focus on high-level task planning pertaining to a mobile manipulator robot. The plans adhere to a particular embodiment consisting of a 7-degree-of-freedom robot arm with a two-fingered gripper attached to a mobile base. We use two large-scale environments, shown in Figure 4, which exhibit multiple rooms and multiple floors which the LLM agent has to plan across. To better ablate and showcase the capabilities of SayPlan, we decouple its semantic search ability from the overall causal planning capabilities using the following two evaluation settings:

Semantic Search: Here, we focus on queries which test the semantic search capabilities of an LLM provided with a collapsed 3D scene graph. This requires the LLM to reason over the room and floor node names and their corresponding attributes in order to aid its search for the relevant assets and objects required to solve the given task instruction. We evaluate against a human baseline to understand how the semantic search capabilities of an LLM compare to a human's thought process. Furthermore, to gain a better understanding of the impact different models have on this graph-based reasoning, we additionally compare against a variant of SayPlan using `GPT-3.5`.

Causal Planning: In this experiment, we evaluate the ability of SayPlan to generate feasible plans to solve a given natural language instruction. The evaluation metrics are divided into two components: *1) Correctness*, which primarily validates the overall goal of the plan and its alignment to what a human would do to solve the task and *2) Executability*, which evaluates the alignment of the plan to the constraints of the scene graph environment and its ability to be executed by a mobile manipulator robot. We note here that for a plan to be executable, it does not necessarily have to be correct and vice versa. We evaluate SayPlan against two baseline methods that integrate an LLM for task planning:

LLM-As-Planner, which generates a full plan sequence in an open-loop manner; the plan includes the full sequence of both navigation and manipulation actions that the robot must execute to complete

a task, and **LLM+P**, an ablated variant of SayPlan, which only incorporates the path planner to allow for shorter horizon plan sequences, without any iterative replanning.

5 Results

We summarise the results for the semantic search evaluation in Table 2. SayPlan (GPT-3.5) consistently failed to reason over the input graph representation, hallucinating nodes to explore or stagnating at exploring the same node multiple times. SayPlan(GPT-4) in contrast achieved 86.7% and 73.3% success in identifying the desired subgraph across both the simple and complex search tasks respectively, demonstrating significantly better graph-based reasoning than GPT-3.5.

	Office			Home		
Subtask	Human	SayPlan (GPT-3.5)	SayPlan (GPT-4)	Human	SayPlan (GPT-3.5)	SayPlan (GPT-4)
Simple Search	100%	6.6%	86.7%	100%	0.0%	86.7%
Complex Search	100%	0.0%	73.3%	100%	0.0%	73.3%

Table 2: **Evaluating the semantic search capabilities of GPT-4.** The table shows the semantic search success rate in finding a suitable subgraph for planning.

While as expected the human baseline achieved 100% on all sets of instructions, we are more interested in the qualitative assessment of the common-sense reasoning used during semantic search. More specifically we would like to identify the similarity in the semantic search heuristics utilised by humans and that used by the underlying LLM based on the given task instruction.

5.1 Semantic Search

We present the full sequence of explored nodes for both SayPlan (GPT-4) and the human baseline in Appendix E. As shown in the tables, SayPlan (GPT-4) demonstrates remarkably similar performance to a human's commonsense reasoning for most tasks, exploring a similar sequence of nodes given a particular instruction. For example when asked to *"find a ripe banana"*, the LLM first explores the kitchen followed by the next most likley location, the cafeteria. In the case where no semantics are present in the instruction such as "find me object K31X", we note that the LLM agent is capable of conducting a breadth-first-like search across all the unexplored nodes.

An odd failure case in the simple search instructions involved negation, where the agent consistently failed when presented with questions such as *"Find me an office that does not have a cabinet"* or *"Find me a bathroom with no toilet"*. Other failure cases noted across the complex search instructions included the LLM's failure to conduct simple distance-based and count-based reasoning over graph nodes. While trivial to a human, this does require the LLM agent to reason over multiple nodes simultaneously, where it tended to hallucinate or miscount connected nodes.

	Full Graph (Token Count)	Collapsed Graph (Token Count)	Compression Ratio
Office	4962	888	82.1%
Home	4602	1827	60.4%

Table 3: **3D Scene Graph Token Count** Number of tokens required for the full graph vs. collapsed graph.

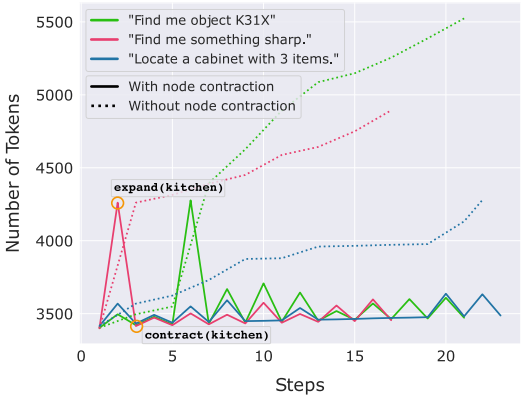


Figure 3: **Scene Graph Token Progression During Semantic Search.** This graph illustrates the scalability of our approach to large-scale 3D scene graphs.

Scalability Analysis: We additionally analyse the scalability of SayPlan during semantic search. Table 3 illustrates the impact of exploiting the hierarchical nature of 3D scene graphs and allowing the LLM to explore the graph from a collapsed initial state. This allows for a reduction of 82.1% in the input tokens required to represent the Office environment and a 60.4% reduction for the Home environment. In Figure 3, we illustrate

	Simple		Long Horizon		Types of Errors				
	Corr	Exec	Corr	Exec	Missing Action	Missing Pose	Wrong Action	Incomplete Search	Hallucinated Nodes
LLM+P	93.3%	13.3%	33.3%	0.0%	26.7%	0.0%	10.0%	3.33%	10.0%
LLM-As-Planner	93.3%	80.0%	66.7%	13.3%	20.0%	60.0%	0.17%	0.03%	10.0%
SayPlan	93.3%	100.0%	73.3%	86.6%	0.0%	0.0%	0.0%	0.0%	6.67%

Table 4: **Causal Planning Results.** *Left:* **Corr**ectness and **Exec**utability on Simple and Long Horizon planning tasks and *Right:* Types of execution errors encountered when planning using LLMs. Note that SayPlan corrects the majority of the errors faced by LLM-based planners.

how endowing the LLM with the ability to contract explored nodes which it deems unsuitable for solving the task allows it to maintain near-constant input memory from a token perspective across the entire semantic search process. Note that the initial number of tokens already present represents the input prompt tokens as given in Appendix H.

5.2 Causal Planning

The results of causal planning across simple and long-horizon instructions are summarised in Table 4 (left). We compared SayPlan's performance against two baselines: LLM-As-Planner and LLM+P. All three methods displayed consistent correctness in simple planning tasks at 93%, given that this metric is more a function of the underlying LLMs reasoning capabilities. However, it is interesting to note that in the long-horizon tasks, both the path planner and iterative replanning play a role in improving this correctness metric by reducing the planning horizon and allowing the LLM to reflect on its previous output.

The results illustrate that the key to ensuring the task plan's executability was iterative replanning. Both LLM-As-Planner and LLM+P exhibited poor executability, whereas SayPlan achieved near-perfect executability as a result of iterative replanning, which ensured that the generated plans were grounded to adhere to the constraints and predicated imposed by the environment. Detailed task plans and errors encountered are provided in Appendix F. We summarise these errors in Table 4 (right) which shows that plans generated with LLM+P and LLM-As-Planner entailed various types of errors limiting their executability. LLM+P mitigated navigational path planning errors as a result of the classical path planner however still suffered from errors pertaining to the manipulation of the environment - missing actions or incorrect actions which violate environment predicates. SayPlan mitigated these errors via iterative replanning, however in 6.67% of tasks, it failed to correct for some hallucinated nodes. While we believe these errors could be eventually corrected via iterative replanning, we limited the number of replanning steps to 5 throughout all experiments. We provide an illustration of the real-world execution of a generated plan using SayPlan on a mobile manipulator robot coupled with a vision-guided motion planner in Appendix G.

6 Limitations

SayPlan naturally inherits the limitations and biases inherent in current large language models, which are dependent on their training data, and can adversely affect the correctness of the generated plans causing misinterpretation of instructions; examples of such failures are illustrated in Appendix F. More specifically, SayPlan is limited by the graph-based reasoning capabilities of the underlying LLM which fails at simple distance-based reasoning, node count-based reasoning and node negation. Future work could explore fine-tuning these models for these specific tasks or alternatively incorporate existing and more complex graph reasoning tools [43] to facilitate decision-making. Secondly, SayPlan's current framework is constrained by the need for a pre-built 3D scene graph and assumes that objects remain static post-map generation, significantly restricting its adaptability to dynamic real-world environments. Future work could explore how online scene graph SLAM systems [15] could be integrated within the SayPlan framework to account for this. Lastly, a potential limitation of the current system lies in the scene graph simulator and its ability to capture the various planning failures within the environment. While this works well in the cases presented in this paper, for more complex tasks involving a diverse set of predicates and affordances, the incorporation of relevant

feedback messages for each instance may become infeasible and forms an important avenue for future work in this area.

7 Conclusion

SayPlan is a natural language-driven planning framework for robotics that integrates hierarchical 3D scene graphs and LLMs to plan across large-scale environments spanning multiple floors and rooms. We ensure the scalability of our approach by exploiting the hierarchical nature of 3D scene graphs and the semantic reasoning capabilities of LLMs to enable the agent to explore the scene graph from the highest level within the hierarchy. In this way, we are no longer required to present all lower-level entities to the LLM, resulting in a significant reduction in the tokens required to capture larger environments. Once explored, the LLM generates task plans for a mobile manipulator robot, and we ensure the plan is feasible and grounded to the state of the environment via iterative replanning using a scene graph simulator. The framework produces the highest number of correct and executable plans that a mobile robot can follow compared to existing baseline techniques. Finally, we translate a select number of the generated plans to a real-world mobile manipulator agent, capable of completing natural language-defined tasks spanning long-range, long-horizon across a wide range of rooms.

Acknowledgments

The authors would like to thank Ben Burgess-Limerick for assistance with the robot hardware setup, Nishant Rana for creating the illustrations and Norman Di Palo and Michael Milford for insightful discussions and feedback towards this manuscript. The authors also acknowledge the ongoing support of the QUT Centre for Robotics. This work was partially supported by the Australian Research Council (Project DP220102398) and by an Amazon Research Award to Niko Sünderhauf.

References

[1] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[3] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

[4] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pages 287–318. PMLR, 2023.

[5] N. Wake, A. Kanehira, K. Sasabuchi, J. Takamatsu, and K. Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application, 2023.

[6] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence. Palm-e: An embodied multimodal language model, 2023.

[7] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.

[8] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[9] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

[10] T. Silver, V. Hariprasad, R. S. Shuttleworth, N. Kumar, T. Lozano-Pérez, and L. P. Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

[11] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5664–5673, 2019.

[12] U.-H. Kim, J.-M. Park, T.-J. Song, and J.-H. Kim. 3-d scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE transactions on cybernetics*, 50(12):4921–4933, 2019.

[13] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[14] P. Gay, J. Stuart, and A. Del Bue. Visual graphs from motion (vgfm): Scene understanding with object geometry reasoning. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part III 14*, pages 330–346. Springer, 2019.

[15] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization. *Robotics: Science and Systems XIV*, 2022.

[16] C. Agia, K. M. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022.

[17] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.

[18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[19] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, pages 287–290. 2022.

[20] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. Pddl-the planning domain definition language. 1998.

[21] M. Fox and D. Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

[22] P. Haslum, N. Lipovetzky, D. Magazzeni, and C. Muise. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2):1–187, 2019.

[23] M. Gelfond and Y. Kahl. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press, 2014.

[24] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the national conference on artificial intelligence*, pages 1194–1201, 1996.

[25] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.

[26] M. Vallati, L. Chrpa, M. Grześ, T. L. McCluskey, M. Roberts, S. Sanner, et al. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3):90–98, 2015.

[27] R. Chitnis, T. Silver, B. Kim, L. Kaelbling, and T. Lozano-Perez. Camps: Learning context-specific abstractions for efficient planning in factored mdps. In *Conference on Robot Learning*, pages 64–79. PMLR, 2021.

[28] T. Silver, R. Chitnis, A. Curtis, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Planning with learned object importance in large problem instances using graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11962–11971, 2021.

[29] F. Ceola, E. Tosello, L. Tagliapietra, G. Nicola, and S. Ghidoni. Robot task planning via deep reinforcement learning: a tabletop object sorting application. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 486–492, 2019. doi:10.1109/SMC.2019.8914278.

[30] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[31] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[32] Y. Xie, C. Yu, T. Zhu, J. Bai, Z. Gong, and H. Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

[33] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen, et al. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*, 2023.

[34] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.

[35] R. Liu, J. Wei, S. S. Gu, T.-Y. Wu, S. Vosoughi, C. Cui, D. Zhou, and A. M. Dai. Mind's eye: Grounded language model reasoning through simulation. *arXiv preprint arXiv:2210.05359*, 2022.

[36] M. Skreta, N. Yoshikawa, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, K. Darvish, A. Aspuru-Guzik, F. Shkurti, and A. Garg. Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv preprint arXiv:2303.14100*, 2023.

[37] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9272–9279. IEEE, 2022.

[38] A. Kurenkov, R. Martín-Martín, J. Ichnowski, K. Goldberg, and S. Savarese. Semantic and geometric modeling with neural message passing in 3d scene graphs for hierarchical mechanical search. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11227–11233. IEEE, 2021.

[39] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T.-J. Chin, I. Reid, S. Gould, et al. Semantics for robotic mapping, perception and interaction: A survey. *Foundations and Trends® in Robotics*, 8(1–2):1–224, 2020.

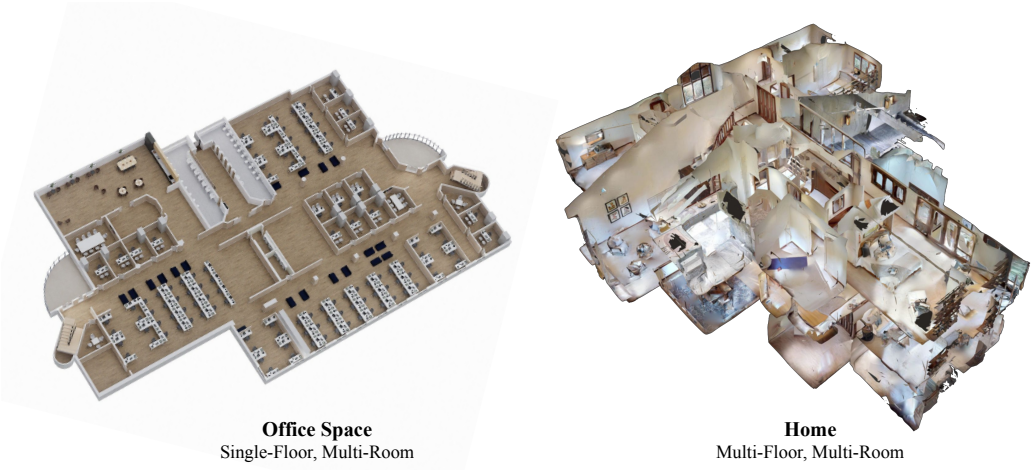
[40] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[41] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[42] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[43] J. Zhang. Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt. *arXiv preprint arXiv:2304.11116*, 2023.

A Environments



Office Space
Single-Floor, Multi-Room

Home
Multi-Floor, Multi-Room

Figure 4: **Large-scale environments used to evaluate SayPlan.**

We evaluate SayPlan across a set of two large-scale environments spanning multiple rooms and floors as shown in Figure 4. We provide details of each of these environments below:

Office: A large-scale office floor, spanning 36 rooms and 150 assets and objects which the agent can interact with. This scene graph represents a real-world office floor within which a mobile manipulator robot is present. This allows us to embody the plans generated using SayPlan and evaluate their feasibility in the corresponding environment. A full and collapsed 3D scene graph representation of the office floor is provided in Appendix C and D respectively.

Home: An existing 3D scene graph from the Stanford 3D Scene Graph dataset [11] which consists of a family home environment (`Klickitat`) spanning 32 rooms across 3 floors and contains 121 assets and objects which the agent can interact with. A 3D visual of this environment can be viewed at the 3D Scene Graph project website.[2]

B Tasks

We evaluate SayPlan across 4 instruction sets which are classified to evaluate different aspects of its 3D scene graph reasoning and planning capabilities:

Simple Search: Focused on evaluating the semantic search capabilities of the LLM based on queries which directly reference information in the scene graph as well as the basic graph-based reasoning capabilities of the LMM.
Complex Search: Abstract semantic search queries which require complex reasoning. The information required to solve these search tasks is not readily available in the graph and has to be inferred by the underlying LLM.
Simple Planning: Task planning queries which require the agent to perform graph search, causal reasoning and environment interaction in order to solve the task. Typically requires shorter horizon plans over single rooms.
Long Horizon Planning: Long Horizon planning queries require multiple interactive steps. These queries evaluate SayPlan's ability to reason over temporally extended instructions to investigate how well it scales to such regimes. Typically requires long horizon plans spanning multiple rooms.

The full list of instructions used and the corresponding aspect the query evaluates are given in the following tables.

[2] 3dscenegraph.stanford.edu/Klickitat

B.1 Simple Search

B.1.1 Office Environment

Instruction	
Find me object K31X.	▷ unguided search with no semantic cue
Find me a carrot.	▷ semantic search based on node name
Find me anything purple in the postdoc bays.	▷ semantic search with termination conditioned on attribute
Find me a ripe banana.	▷ semantic search with termination conditioned on attribute
Find me something that has a screwdriver in it.	▷ unguided search with termination conditioned on children
One of the offices has a poster of the Terminator. Which one is it?	▷ semantic search with termination conditioned on children
I printed a document but I don't know which printer has it. Find the document.	▷ semantic search based on parent
I left my headphones in one of the meeting rooms. Locate them.	▷ semantic search based on parent
Find the PhD bay that has a drone in it.	▷ semantic search with termination conditioned on children
Find the kale that is not in the kitchen.	▷ semantic search with termination conditioned on a negation predicate on parent
Find me an office that does not have a cabinet.	▷ semantic search with termination conditioned on a negation predicate on children
Find me an office that contains a cabinet, a desk, and a chair.	▷ semantic search with termination conditioned on a conjunctive query on children
Find a book that was left next to a robotic gripper.	▷ semantic search with termination conditioned on a sibling
Luis gave one of his neighbours a stapler. Find the stapler.	▷ semantic search with termination conditioned on a sibling
There is a meeting room with a chair but no table. Locate it.	▷ semantic search with termination conditioned on a conjunctive query with negation

Table 5: **Simple Search Instructions.** Evaluated in Office Environment.

B.1.2 Home Environment

Instruction	
Find me a FooBar.	▷ unguided search with no semantic cue
Find me a bottle of wine.	▷ semantic search based on node name
Find me a plant with thorns.	▷ semantic search with termination conditioned on attribute
Find me a plant that needs watering.	▷ semantic search with termination conditioned on attribute
Find me a bathroom with no toilet.	▷ semantic search with termination conditioned on a negation predicate
The baby dropped their rattle in one of the rooms. Locate it.	▷ semantic search based on node name
I left my suitcase either in the bedroom or the living room. Which room is it in.	▷ semantic search based on node name
Find the room with a ball in it.	▷ semantic search based on node name
I forgot my book on a bed. Locate it.	▷ semantic search based on node name
Find an empty vase that was left next to sink.	▷ semantic search with termination conditioned on sibling
Locate the dining room which has a table, chair and a baby monitor.	▷ semantic search with termination conditioned on conjuctive query
Locate a chair that is not in any dining room.	▷ semantic search with termination conditioned on negation predicate
I need to shave. Which room has both a razor and shaving cream.	▷ semantic search with termination conditioned on children
Find me 2 bedrooms with pillows in them.	▷ semantic search with multiple returns
Find me 2 bedrooms without pillows in them.	▷ semantic search with multiple returns based on negation predicate

Table 6: **Simple Search Instructions.** Evaluated in Home Environment.

B.2 Complex Search

B.2.1 Office Environment

Instruction	
Find object J64M. J64M should be kept at below 0 degrees Celsius.	▷ semantic search guided by implicit world knowledge (knowledge not directly encoded in graph)
Find me something non vegetarian.	▷ semantic search with termination conditioned on implicit world knowledge
Locate something sharp.	▷ unguided search with termination conditioned on implicit world knowledge
Find the room where people are playing board games.	▷ semantic search with termination conditioned on ability to deduce context from node children using world knowledge ("board game" is not part of any node name or attribute in this graph)
Find an office of someone who is clearly a fan of Arnold Schwarzenegger.	▷ semantic search with termination conditioned on ability to deduce context from node children using world knowledge
There is a postdoc that has a pet Husky. Find the desk that's most likely theirs.	▷ semantic search with termination conditioned on ability to deduce context from node children using world knowledge
One of the PhD students was given more than one complimentary T-shirts. Find his desk.	▷ semantic search with termination conditioned on the number of children
Find me the office where a paper attachment device is inside an asset that is open.	▷ semantic search with termination conditioned on node descendants and their attributes
There is an office which has a cabinet containing exactly 3 items in it. Locate the office.	▷ semantic search with termination conditioned on the number of children
There is an office which has a cabinet containing a rotten apple. The cabinet name contains an even number. Locate the office.	▷ semantic search guided by numerical properties
Look for a carrot. The carrot is likely to be in a meeting room but I'm not sure.	▷ semantic search guided by user provided bias
Find me a meeting room with a RealSense camera.	▷ semantic search that has no result (no meeting room has a realsense camera in the graph)
Find the closest fire extinguisher to the manipulation lab.	▷ search guided by node distance
Find me the closest meeting room to the kitchen.	▷ search guided by node distance
Either Filipe or Tobi has my headphones. Locate it.	▷ evaluating constrained search, early termination once the two office are explored

Table 7: **Complex Search Instructions.** Evaluated in Office Environment.

B.2.2 Home Environment

Instruction	
I need something to access ChatGPT. Where should I go?	▷ semantic search guided by implicit world knowledge
Find the livingroom that contains the most electronic devices.	▷ semantic search with termination conditioned on children with indirect information
Find me something to eat with a lot of potassium.	▷ semantic search with termination conditioned on implicit world knowledge
I left a sock in a bedroom and one in the living room. Locate them. They should match.	▷ semantic search with multiple returns
Find me a potted plant that is most likely a cactus.	▷ semantic search with termination implicitly conditioned on attribute
Find the dining room with exactly 5 chairs.	▷ semantic search with termination implicitly conditioned on quantity of children
Find me the bedroom closest to the home office.	▷ semantic search with termination implicitly conditioned on node distance
Find me a bedroom with an unusual amount of bowls.	▷ semantic search with termination implicitly conditioned on quantity of children
Which bedroom is empty.	▷ semantic search with termination implicitly conditioned on quantity of children
Which bathroom has the most potted plants.	▷ semantic search with termination implicitly conditioned on quantity of children
The kitchen is flooded. Find somewhere I can heat up my food.	▷ semantic search guided by negation
Find me the room which most likely belongs to a child	▷ semantic search with termination conditioned on ability to deduce context from node children using world knowledge
15 guests are arriving. Locate enough chairs to seat them.	▷ semantic search with termination implicitly conditioned on the quantity of specified node
A vegetarian dinner was prepared in one of the dining rooms. Locate it.	▷ semantic search with selection criteria based on world knowledge
My tie is in one of the closets. Locate it.	▷ evaluating constrained search that has no result, termination after exploring closets

Table 8: **Complex Search Instructions.** Evaluated in Home Environment.

B.3 Simple Planning

Instruction
Close Jason's cabinet.
Refrigerate the orange left on the kitchen bench.
Take care of the dirty plate in the lunchroom.
Place the printed document on Will's desk.
Peter is working hard at his desk. Get him a healthy snack.
Hide one of Peter's valuable belongings.
Wipe the dusty admin shelf.
There is coffee dripping on the floor. Stop it.
Place Will's drone on his desk.
Move the monitor from Jason's office to Filipe's.
My parcel just got delivered! Locate it and place it in the appropriate lab.
Check if the coffee machine is working.
Heat up the chicken kebab.
Something is smelling in the kitchen. Dispose of it.
Throw what the agent is holding in the bin.

Table 9: **Simple Planning Instructions.** Evaluated in Office Environment.

B.4 Long Horizon Planning

Instruction
Heat up the noodles in the fridge, and place it somewhere where I can enjoy it.
Throw the rotting fruit in Dimity's office in the correct bin.
Wash all the dishes on the lunch table. Once finished, place all the clean cutlery in the drawer.
Safely file away the freshly printed document in Will's office then place the undergraduate thesis on his desk.
Make Niko a coffee and place the mug on his desk.
Someone has thrown items in the wrong bins. Correct this.
Tobi spilt soda on his desk. Throw away the can and take him something to clean with.
I want to make a sandwich. Place all the ingredients on the lunch table.
A delegation of project partners is arriving soon. We want to serve them snacks and non-alcoholic drinks. Prepare everything in the largest meeting room. Use items found in the supplies room only.
Serve bottled water to the attendees who are seated in meeting room 1. Each attendee can only receive a single bottle of water.
Empty the dishwasher. Place all items in their correct locations
Locate all 6 complimentary t-shirts given to the PhD students and place them on the shelf in admin.
I'm hungry. Bring me an apple from Peter and a pepsi from Tobi. I'm at the lunch table.
Let's play a prank on Niko. Dimity might have something.
There is an office which has a cabinet containing a rotten apple. The cabinet name contains an even number. Locate the office, throw away the fruit and get them a fresh apple.

Table 10: **Long-Horizon Planning Instructions.** Evaluated in Office Environment.

C Full 3D Scene Graph: Office Environment

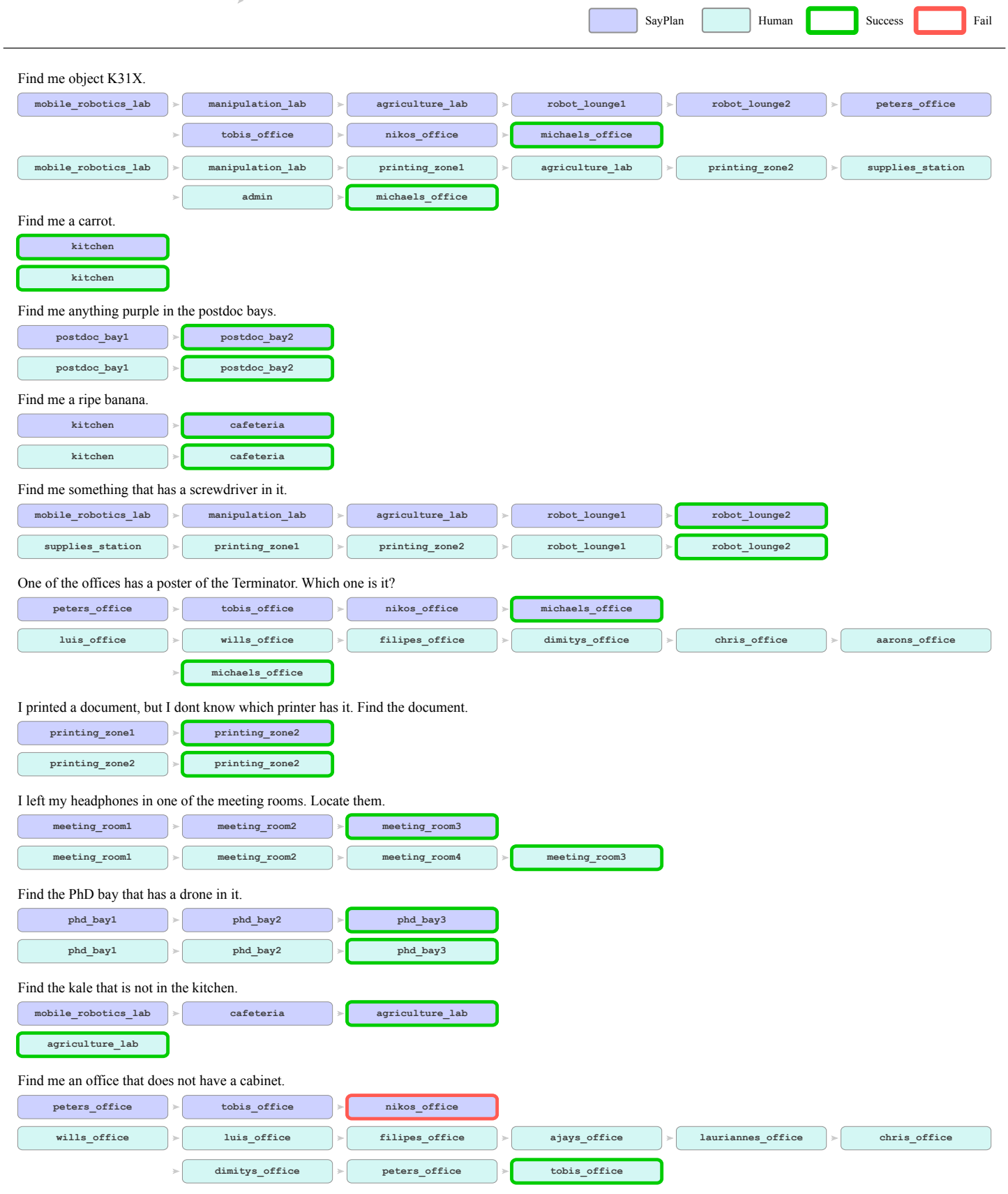
Figure 5: **3D Scene Graph - Fully Expanded Office Environment.** Full 3D scene graph exposing all the rooms, assets and objects available in the scene. Note that the LLM agent never sees all this information unless it chooses to expand every possible node without contraction.

D Contracted 3D Scene Graph: Office Environment

Figure 6: **3D Scene Graph - Contracted Office Environment.** Contracted 3D scene graph exposing only the highest level within the hierarchy - room nodes. This results in an 82.1% reduction in the number of tokens required to represent the scene before the semantic search phase.

E Semantic Search Evaluation Results

- Full listings of the generated semantic search sequences for the evaluation instruction sets are provided on the following page -


SayPlan
Human
Success
Fail
Find me object K31X.
mobile_robotics_lab
manipulation_lab
agriculture_lab
robot_lounge1
robot_lounge2
peters_office
tobis_office
nikos_office
michaels_office
mobile_robotics_lab
manipulation_lab
printing_zone1
agriculture_lab
printing_zone2
supplies_station
admin
michaels_office
Find me a carrot.
kitchen
kitchen
Find me anything purple in the postdoc bays.
postdoc_bay1
postdoc_bay2
postdoc_bay1
postdoc_bay2
Find me a ripe banana.
kitchen
cafeteria
kitchen
cafeteria
Find me something that has a screwdriver in it.
mobile_robotics_lab
manipulation_lab
agriculture_lab
robot_lounge1
robot_lounge2
supplies_station
printing_zone1
printing_zone2
robot_lounge1
robot_lounge2
One of the offices has a poster of the Terminator. Which one is it?
peters_office
tobis_office
nikos_office
michaels_office
luis_office
wills_office
filipes_office
dimitys_office
chris_office
aarons_office
michaels_office
I printed a document, but I dont know which printer has it. Find the document.
printing_zone1
printing_zone2
printing_zone2
printing_zone2
I left my headphones in one of the meeting rooms. Locate them.
meeting_room1
meeting_room2
meeting_room3
meeting_room1
meeting_room2
meeting_room4
meeting_room3
Find the PhD bay that has a drone in it.
phd_bay1
phd_bay2
phd_bay3
phd_bay1
phd_bay2
phd_bay3
Find the kale that is not in the kitchen.
mobile_robotics_lab
cafeteria
agriculture_lab
agriculture_lab
Find me an office that does not have a cabinet.
peters_office
tobis_office
nikos_office
wills_office
luis_office
filipes_office
ajays_office
lauriannes_office
chris_office
dimitys_office
peters_office
tobis_office

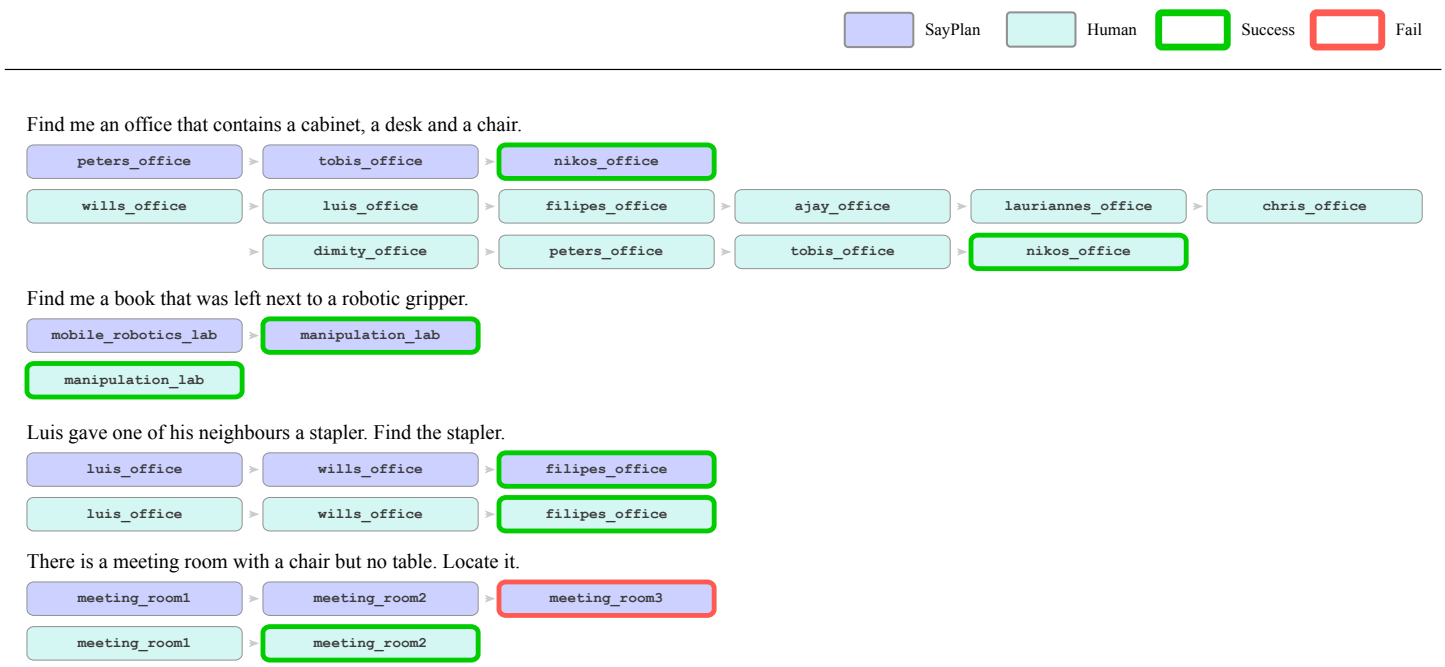
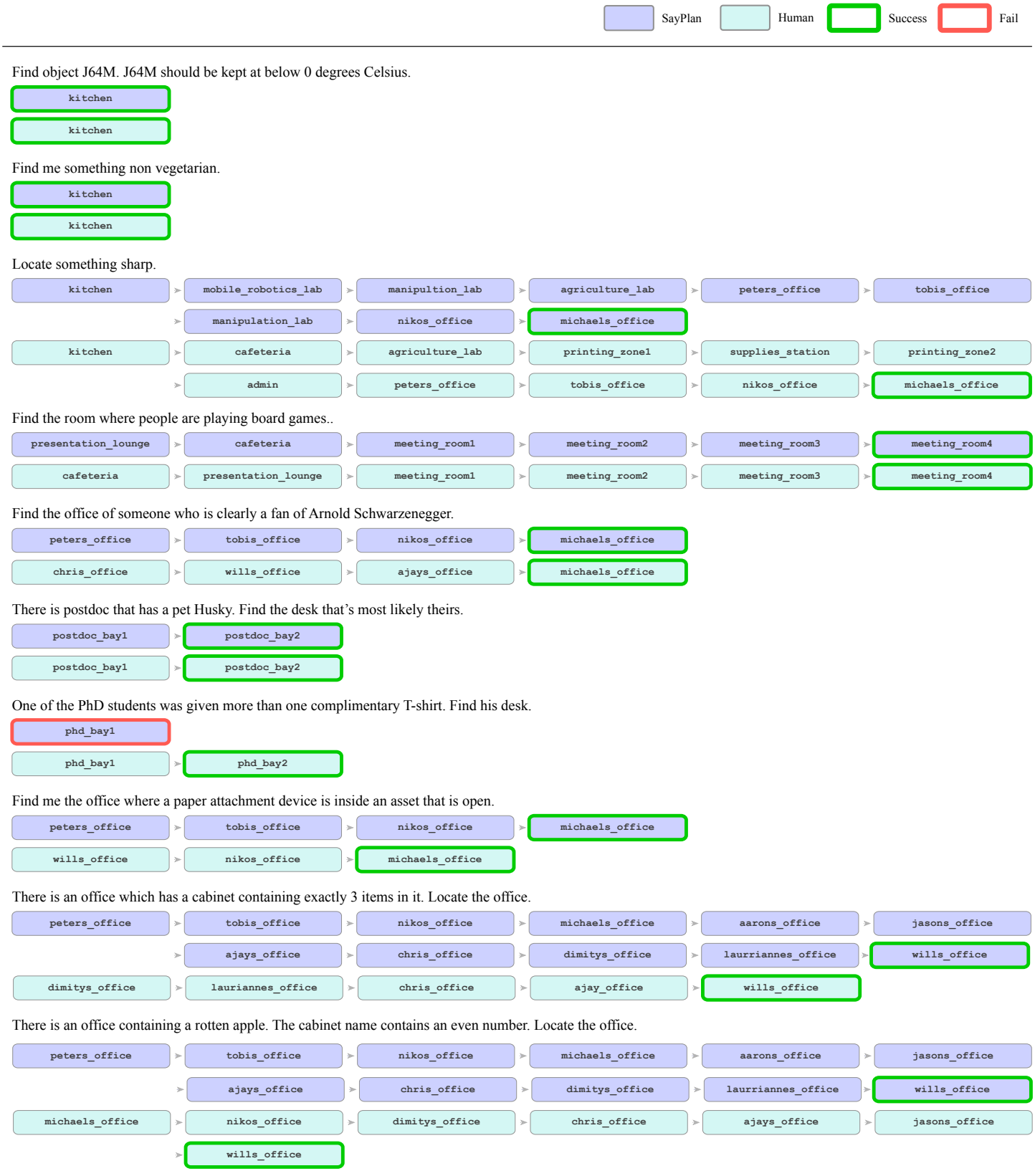


Table 11: **Simple Search Office Environment Evaluation.** Sequence of Explored Nodes for Simple Search Office Environment Instructions.


SayPlan
Human
Success
Fail
Find object J64M. J64M should be kept at below 0 degrees Celsius.
kitchen
kitchen
Find me something non vegetarian.
kitchen
kitchen
Locate something sharp.
kitchen
mobile_robotics_lab
manipultion_lab
agriculture_lab
peters_office
tobis_office
manipulation_lab
nikos_office
michaels_office
kitchen
cafeteria
agriculture_lab
printing_zone1
supplies_station
printing_zone2
admin
peters_office
tobis_office
nikos_office
michaels_office
Find the room where people are playing board games..
presentation_lounge
cafeteria
meeting_room1
meeting_room2
meeting_room3
meeting_room4
cafeteria
presentation_lounge
meeting_room1
meeting_room2
meeting_room3
meeting_room4
Find the office of someone who is clearly a fan of Arnold Schwarzenegger.
peters_office
tobis_office
nikos_office
michaels_office
chris_office
wills_office
ajays_office
michaels_office
There is postdoc that has a pet Husky. Find the desk that's most likely theirs.
postdoc_bay1
postdoc_bay2
postdoc_bay1
postdoc_bay2
One of the PhD students was given more than one complimentary T-shirt. Find his desk.
phd_bay1
phd_bay1
phd_bay2
Find me the office where a paper attachment device is inside an asset that is open.
peters_office
tobis_office
nikos_office
michaels_office
wills_office
nikos_office
michaels_office
There is an office which has a cabinet containing exactly 3 items in it. Locate the office.
peters_office
tobis_office
nikos_office
michaels_office
aarons_office
jasons_office
ajays_office
chris_office
dimitys_office
laurriannes_office
wills_office
dimitys_office
lauriannes_office
chris_office
ajay_office
wills_office
There is an office containing a rotten apple. The cabinet name contains an even number. Locate the office.
peters_office
tobis_office
nikos_office
michaels_office
aarons_office
jasons_office
ajays_office
chris_office
dimitys_office
laurriannes_office
wills_office
michaels_office
nikos_office
dimitys_office
chris_office
ajays_office
jasons_office
wills_office

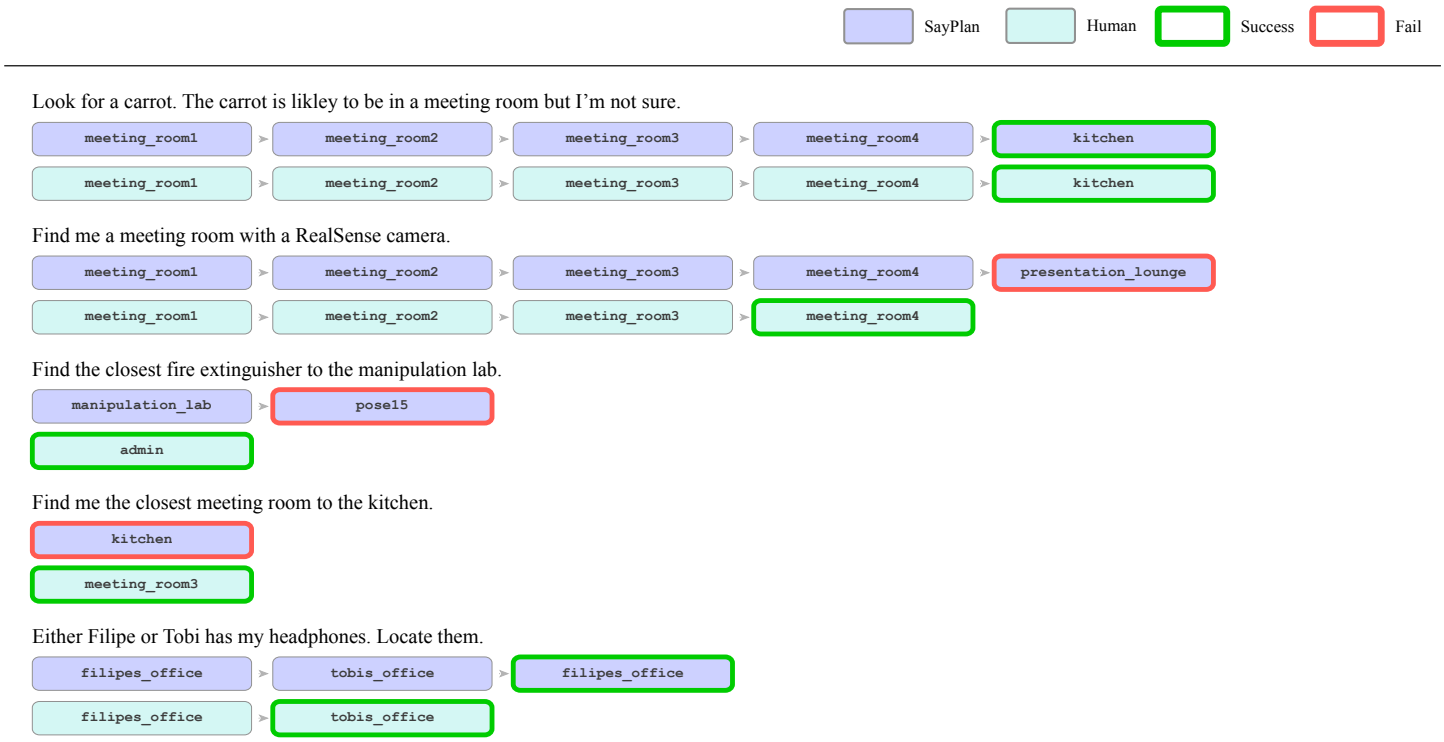
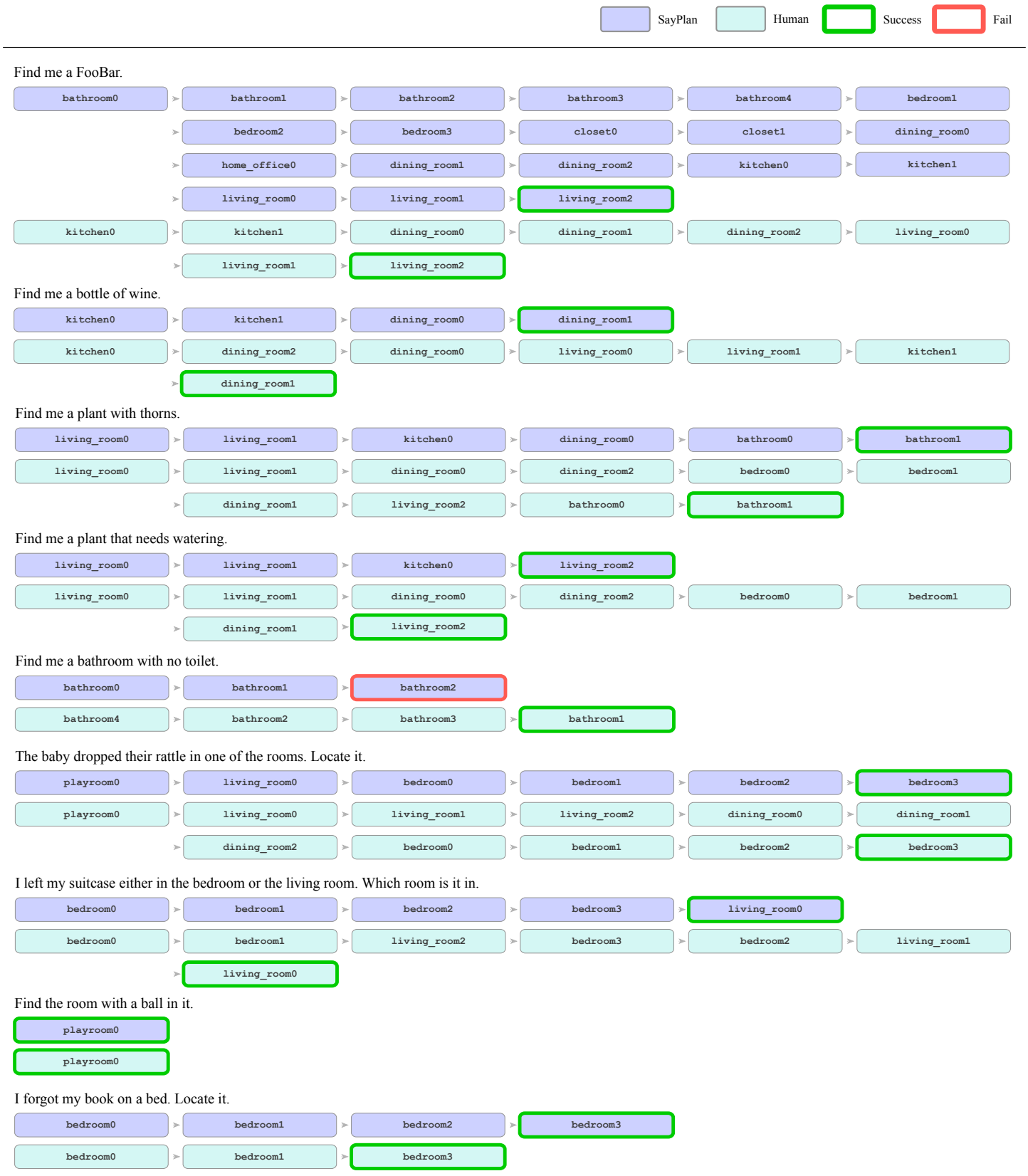


Table 13: **Complex Search Office Environment Evaluation.** Sequence of Explored Nodes for Complex Search Office Environment Instructions.


SayPlan
Human
Success
Fail
Find me a FooBar.
bathroom0 bathroom1 bathroom2 bathroom3 bathroom4 bedroom1
bedroom2 bedroom3 closet0 closet1 dining_room0
home_office0 dining_room1 dining_room2 kitchen0 kitchen1
living_room0 living_room1 living_room2
kitchen0 kitchen1 dining_room0 dining_room1 dining_room2 living_room0
living_room1 living_room2
Find me a bottle of wine.
kitchen0 kitchen1 dining_room0 dining_room1
kitchen0 dining_room2 dining_room0 living_room0 living_room1 kitchen1
dining_room1
Find me a plant with thorns.
living_room0 living_room1 kitchen0 dining_room0 bathroom0 bathroom1
living_room0 living_room1 dining_room0 dining_room2 bedroom0 bedroom1
dining_room1 living_room2 bathroom0 bathroom1
Find me a plant that needs watering.
living_room0 living_room1 kitchen0 living_room2
living_room0 living_room1 dining_room0 dining_room2 bedroom0 bedroom1
dining_room1 living_room2
Find me a bathroom with no toilet.
bathroom0 bathroom1 bathroom2
bathroom4 bathroom2 bathroom3 bathroom1
The baby dropped their rattle in one of the rooms. Locate it.
playroom0 living_room0 bedroom0 bedroom1 bedroom2 bedroom3
playroom0 living_room0 living_room1 living_room2 dining_room0 dining_room1
dining_room2 bedroom0 bedroom1 bedroom2 bedroom3
I left my suitcase either in the bedroom or the living room. Which room is it in.
bedroom0 bedroom1 bedroom2 bedroom3 living_room0
bedroom0 bedroom1 living_room2 bedroom3 bedroom2 living_room1
living_room0
Find the room with a ball in it.
playroom0
playroom0
I forgot my book on a bed. Locate it.
bedroom0 bedroom1 bedroom2 bedroom3
bedroom0 bedroom1 bedroom3

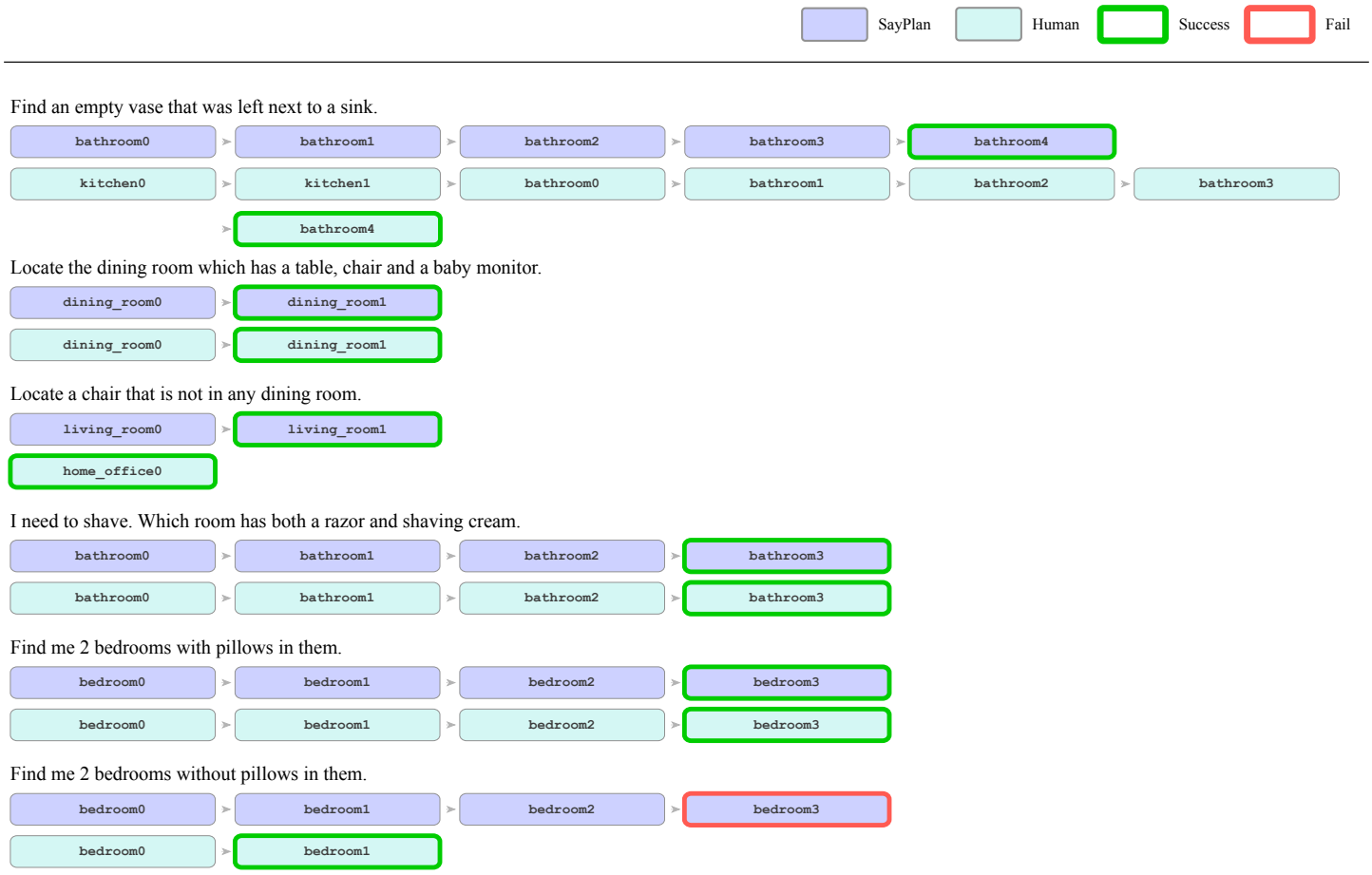
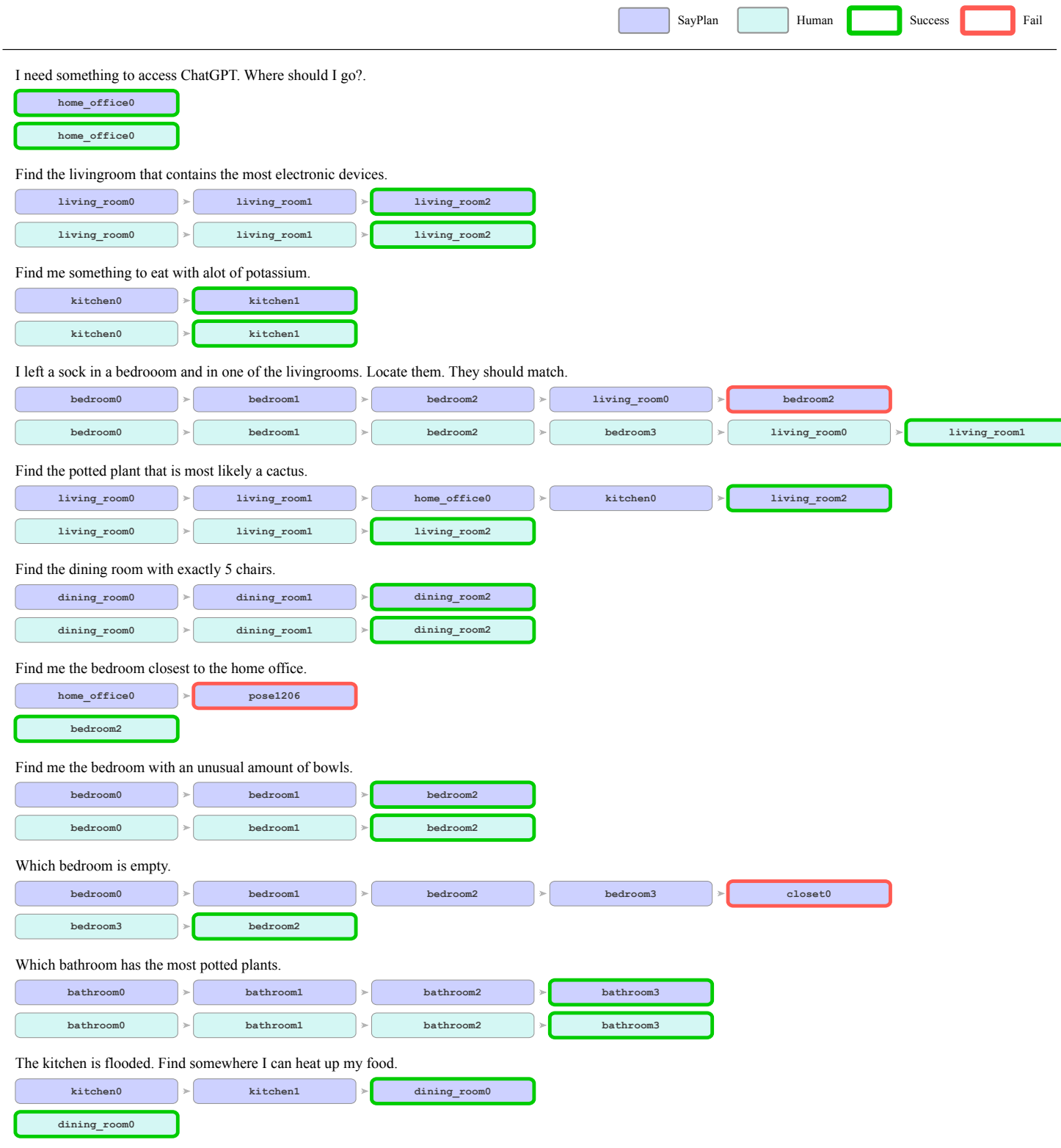


Table 15: **Simple Search Home Environment Evaluation.** Sequence of Explored Nodes for Simple Search Home Environment Instructions.


SayPlan
Human
Success
Fail
I need something to access ChatGPT. Where should I go?.
home_office0
home_office0
Find the livingroom that contains the most electronic devices.
living_room0
living_room1
living_room2
living_room0
living_room1
living_room2
Find me something to eat with alot of potassium.
kitchen0
kitchen1
kitchen0
kitchen1
I left a sock in a bedrooom and in one of the livingrooms. Locate them. They should match.
bedroom0
bedroom1
bedroom2
living_room0
bedroom2
bedroom0
bedroom1
bedroom2
bedroom3
living_room0
living_room1
Find the potted plant that is most likely a cactus.
living_room0
living_room1
home_office0
kitchen0
living_room2
living_room0
living_room1
living_room2
Find the dining room with exactly 5 chairs.
dining_room0
dining_room1
dining_room2
dining_room0
dining_room1
dining_room2
Find me the bedroom closest to the home office.
home_office0
pose1206
bedroom2
Find me the bedroom with an unusual amount of bowls.
bedroom0
bedroom1
bedroom2
bedroom0
bedroom1
bedroom2
Which bedroom is empty.
bedroom0
bedroom1
bedroom2
bedroom3
closet0
bedroom3
bedroom2
Which bathroom has the most potted plants.
bathroom0
bathroom1
bathroom2
bathroom3
bathroom0
bathroom1
bathroom2
bathroom3
The kitchen is flooded. Find somewhere I can heat up my food.
kitchen0
kitchen1
dining_room0
dining_room0

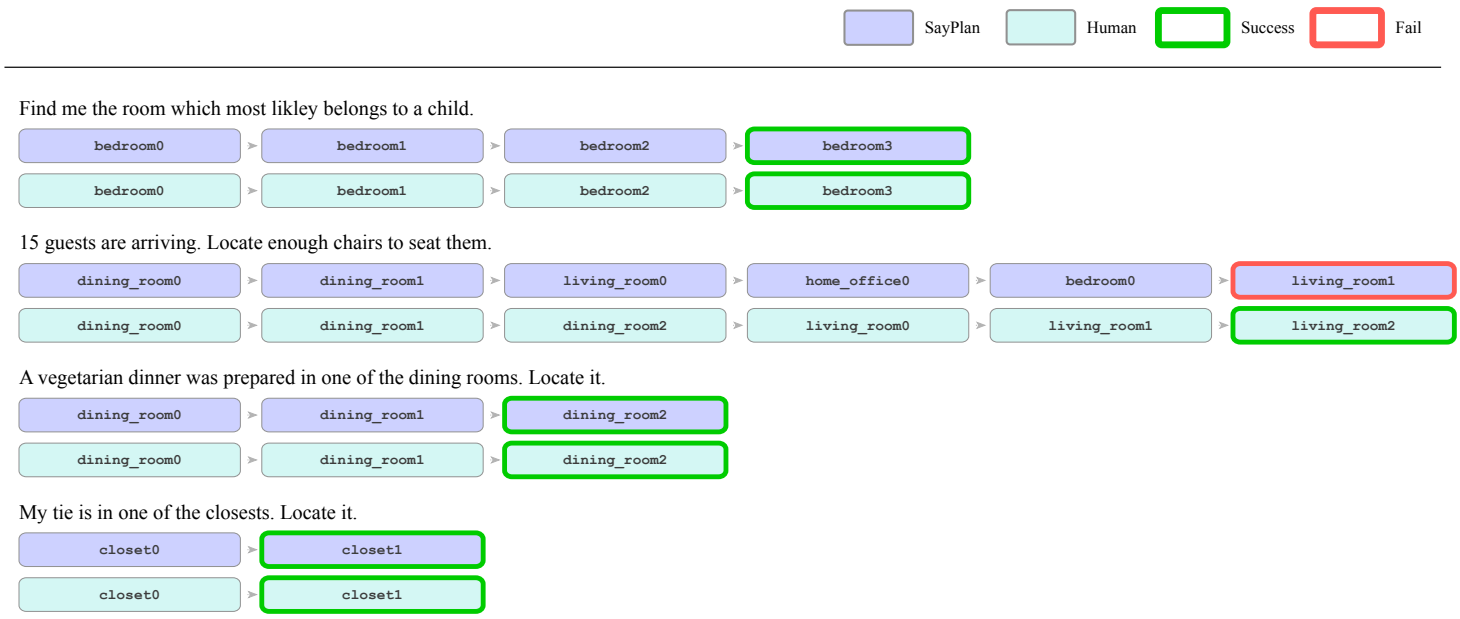
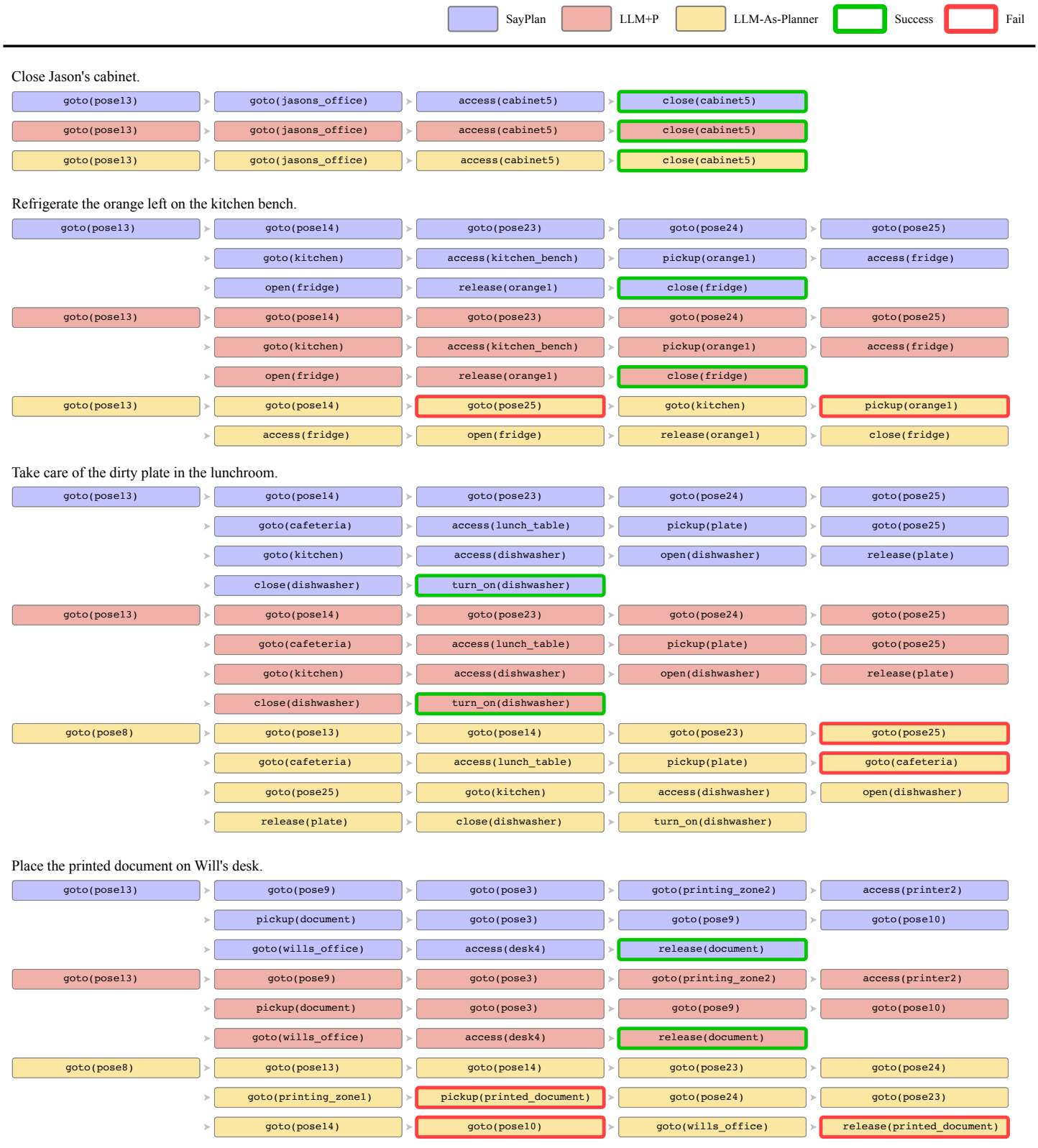
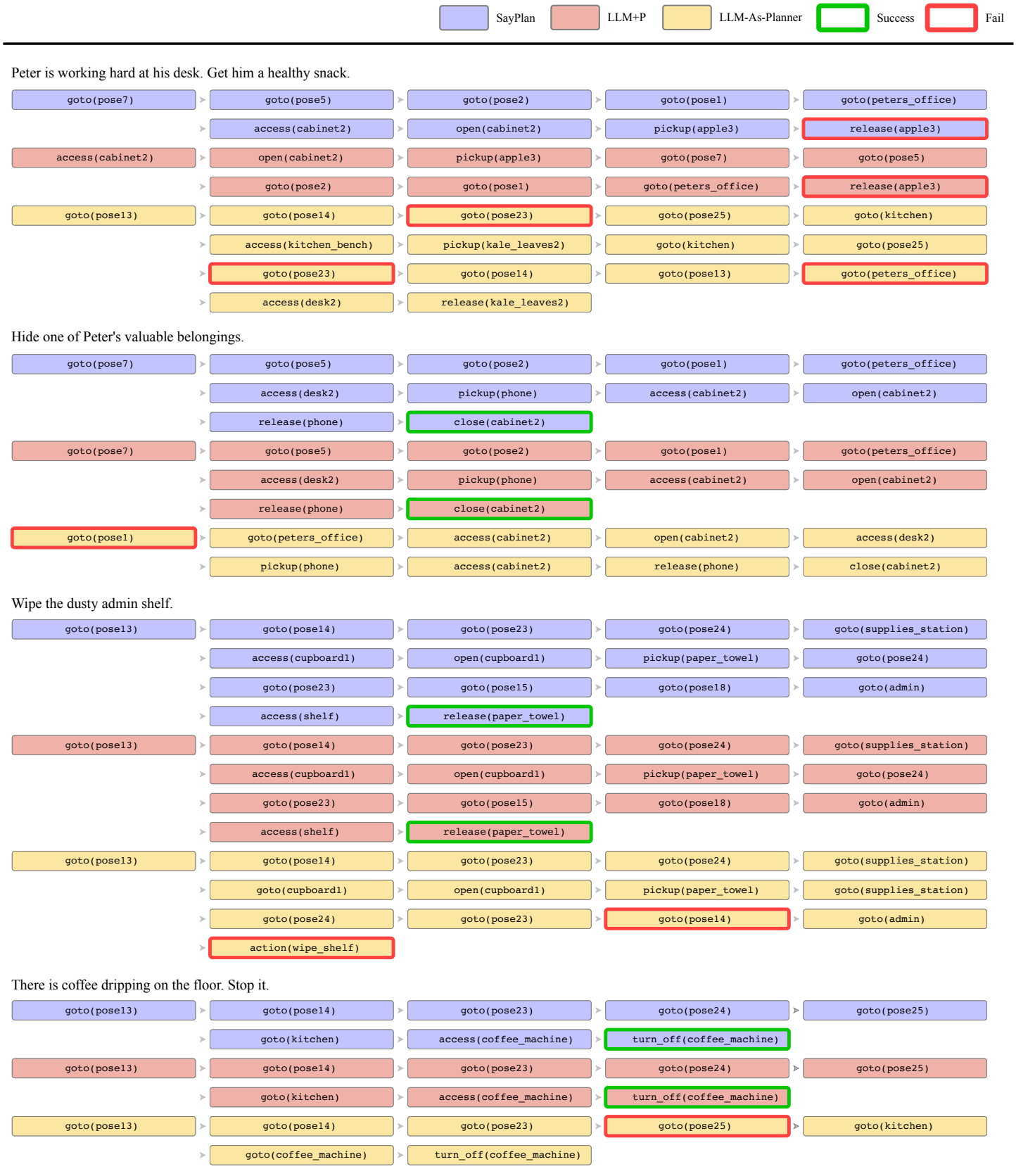


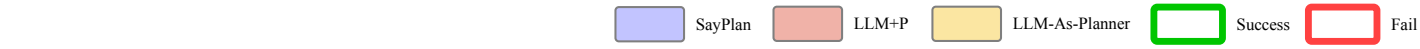
Table 17: **Complex Search Home Environment Evaluation.** Sequence of Explored Nodes for Complex Search Home Environment Instructions.

F Causal Planning Evaluation Results

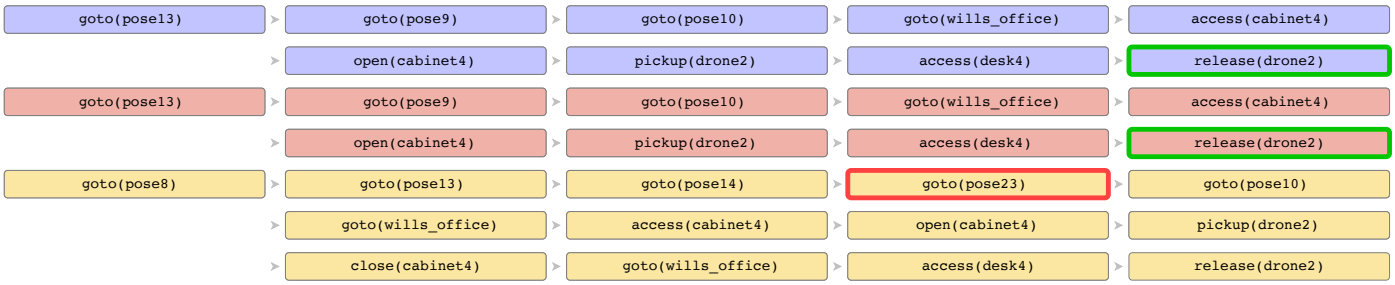
- Full listings of the generated planning sequences for the evaluation instruction sets are provided on the following page -


SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Close Jason's cabinet.
goto(pose13) goto(jasons_office) access(cabinet5) close(cabinet5)
goto(pose13) goto(jasons_office) access(cabinet5) close(cabinet5)
goto(pose13) goto(jasons_office) access(cabinet5) close(cabinet5)
Refrigerate the orange left on the kitchen bench.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(kitchen_bench) pickup(orange1) access(fridge)
open(fridge) release(orange1) close(fridge)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(kitchen_bench) pickup(orange1) access(fridge)
open(fridge) release(orange1) close(fridge)
goto(pose13) goto(pose14) goto(pose25) goto(kitchen) pickup(orange1)
access(fridge) open(fridge) release(orange1) close(fridge)
Take care of the dirty plate in the lunchroom.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(cafeteria) access(lunch_table) pickup(plate) goto(pose25)
goto(kitchen) access(dishwasher) open(dishwasher) release(plate)
close(dishwasher) turn_on(dishwasher)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(cafeteria) access(lunch_table) pickup(plate) goto(pose25)
goto(kitchen) access(dishwasher) open(dishwasher) release(plate)
close(dishwasher) turn_on(dishwasher)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose25)
goto(cafeteria) access(lunch_table) pickup(plate) goto(cafeteria)
goto(pose25) goto(kitchen) access(dishwasher) open(dishwasher)
release(plate) close(dishwasher) turn_on(dishwasher)
Place the printed document on Will's desk.
goto(pose13) goto(pose9) goto(pose3) goto(printing_zone2) access(printer2)
pickup(document) goto(pose3) goto(pose9) goto(pose10)
goto(wills_office) access(desk4) release(document)
goto(pose13) goto(pose9) goto(pose3) goto(printing_zone2) access(printer2)
pickup(document) goto(pose3) goto(pose9) goto(pose10)
goto(wills_office) access(desk4) release(document)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(printing_zone1) pickup(printed_document) goto(pose24) goto(pose23)
goto(pose14) goto(pose10) goto(wills_office) release(printed_document)

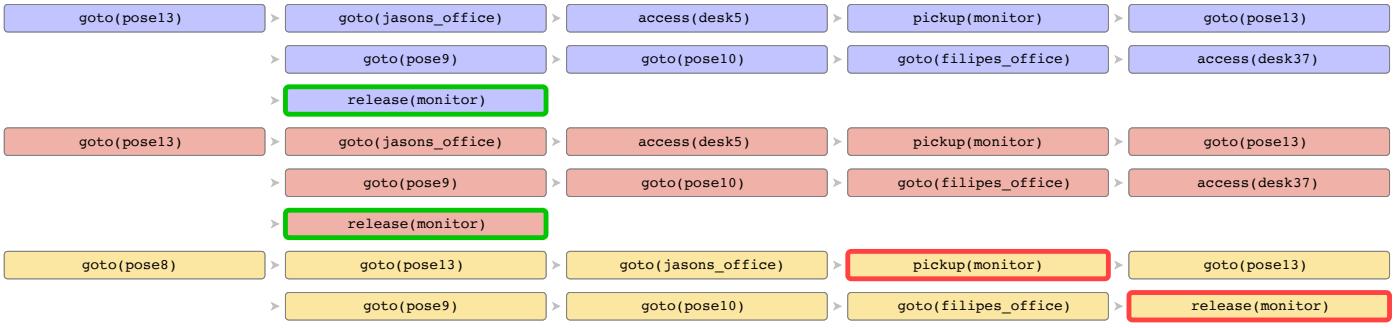

SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Peter is working hard at his desk. Get him a healthy snack.
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(peters_office)
access(cabinet2) open(cabinet2) pickup(apple3) release(apple3)
access(cabinet2) open(cabinet2) pickup(apple3) goto(pose7) goto(pose5)
goto(pose2) goto(pose1) goto(peters_office) release(apple3)
goto(pose13) goto(pose14) goto(pose23) goto(pose25) goto(kitchen)
access(kitchen_bench) pickup(kale_leaves2) goto(kitchen) goto(pose25)
goto(pose23) goto(pose14) goto(pose13) goto(peters_office)
access(desk2) release(kale_leaves2)
Hide one of Peter's valuable belongings.
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(peters_office)
access(desk2) pickup(phone) access(cabinet2) open(cabinet2)
release(phone) close(cabinet2)
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(peters_office)
access(desk2) pickup(phone) access(cabinet2) open(cabinet2)
release(phone) close(cabinet2)
goto(pose1) goto(peters_office) access(cabinet2) open(cabinet2) access(desk2)
pickup(phone) access(cabinet2) release(phone) close(cabinet2)
Wipe the dusty admin shelf.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
access(cupboard1) open(cupboard1) pickup(paper_towel) goto(pose24)
goto(pose23) goto(pose15) goto(pose18) goto(admin)
access(shelf) release(paper_towel)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
access(cupboard1) open(cupboard1) pickup(paper_towel) goto(pose24)
goto(pose23) goto(pose15) goto(pose18) goto(admin)
access(shelf) release(paper_towel)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
goto(cupboard1) open(cupboard1) pickup(paper_towel) goto(supplies_station)
goto(pose24) goto(pose23) goto(pose14) goto(admin)
action(wipe_shelf)
There is coffee dripping on the floor. Stop it.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(coffee_machine) turn_off(coffee_machine)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(coffee_machine) turn_off(coffee_machine)
goto(pose13) goto(pose14) goto(pose23) goto(pose25) goto(kitchen)
goto(coffee_machine) turn_off(coffee_machine)



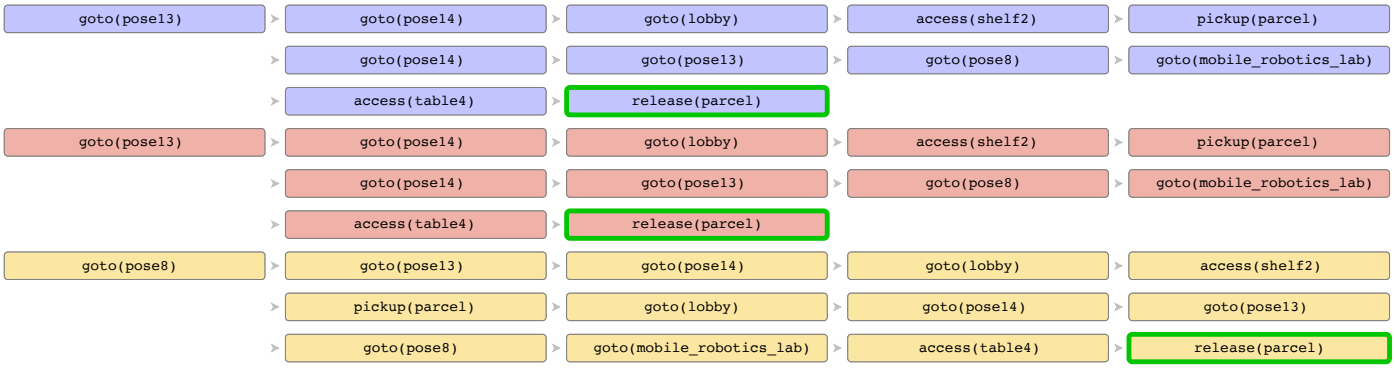
Place Will's drone on his desk.



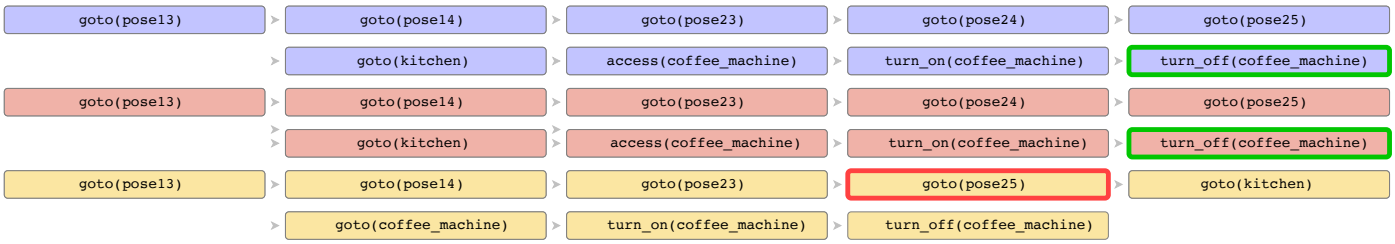
Move the monitor from Jason's office to Filipe's.

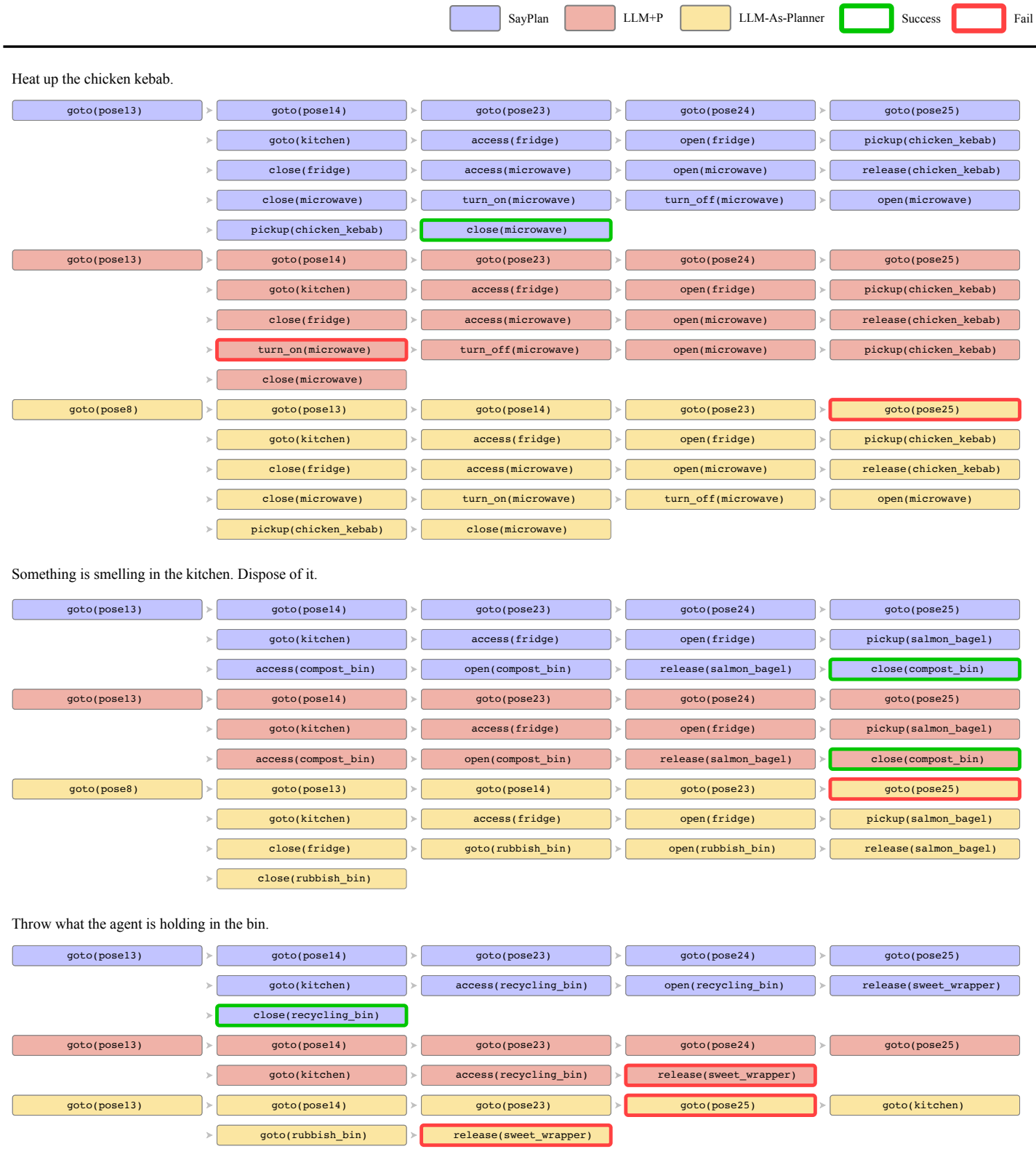


My parcel just got delivered! Locate it and place it in the appropriate lab.



Check if the coffee machine is working.



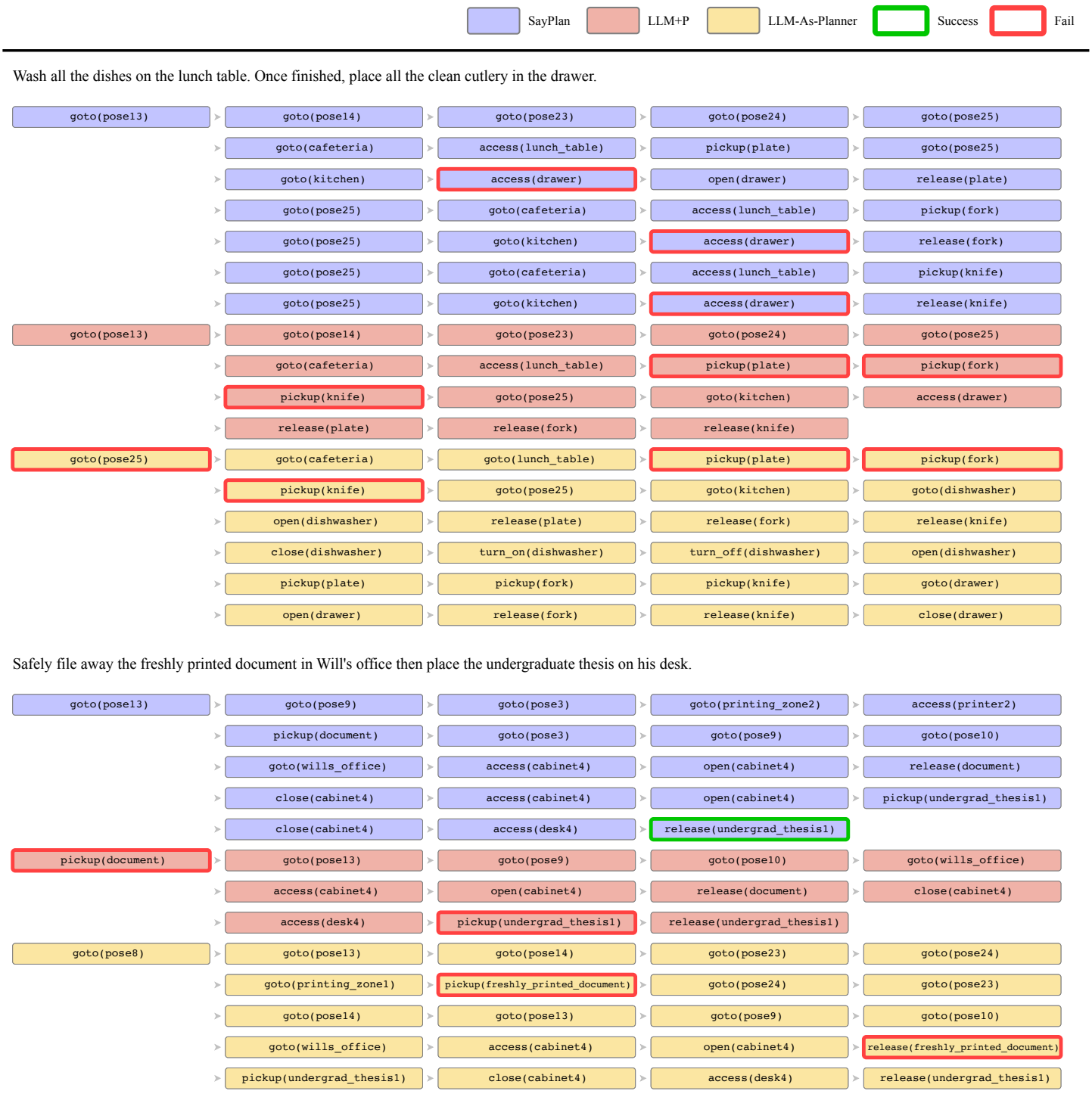

SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Heat up the chicken kebab.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(fridge) open(fridge) pickup(chicken_kebab)
close(fridge) access(microwave) open(microwave) release(chicken_kebab)
close(microwave) turn_on(microwave) turn_off(microwave) open(microwave)
pickup(chicken_kebab) close(microwave)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(fridge) open(fridge) pickup(chicken_kebab)
close(fridge) access(microwave) open(microwave) release(chicken_kebab)
turn_on(microwave) turn_off(microwave) open(microwave) pickup(chicken_kebab)
close(microwave)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose25)
goto(kitchen) access(fridge) open(fridge) pickup(chicken_kebab)
close(fridge) access(microwave) open(microwave) release(chicken_kebab)
close(microwave) turn_on(microwave) turn_off(microwave) open(microwave)
pickup(chicken_kebab) close(microwave)
Something is smelling in the kitchen. Dispose of it.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(fridge) open(fridge) pickup(salmon_bagel)
access(compost_bin) open(compost_bin) release(salmon_bagel) close(compost_bin)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(fridge) open(fridge) pickup(salmon_bagel)
access(compost_bin) open(compost_bin) release(salmon_bagel) close(compost_bin)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose25)
goto(kitchen) access(fridge) open(fridge) pickup(salmon_bagel)
close(fridge) goto(rubbish_bin) open(rubbish_bin) release(salmon_bagel)
close(rubbish_bin)
Throw what the agent is holding in the bin.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(recycling_bin) open(recycling_bin) release(sweet_wrapper)
close(recycling_bin)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(recycling_bin) release(sweet_wrapper)
goto(pose13) goto(pose14) goto(pose23) goto(pose25) goto(kitchen)
goto(rubbish_bin) release(sweet_wrapper)

Heat up the noodles in the fridge, and place it somewhere where I can enjoy it.

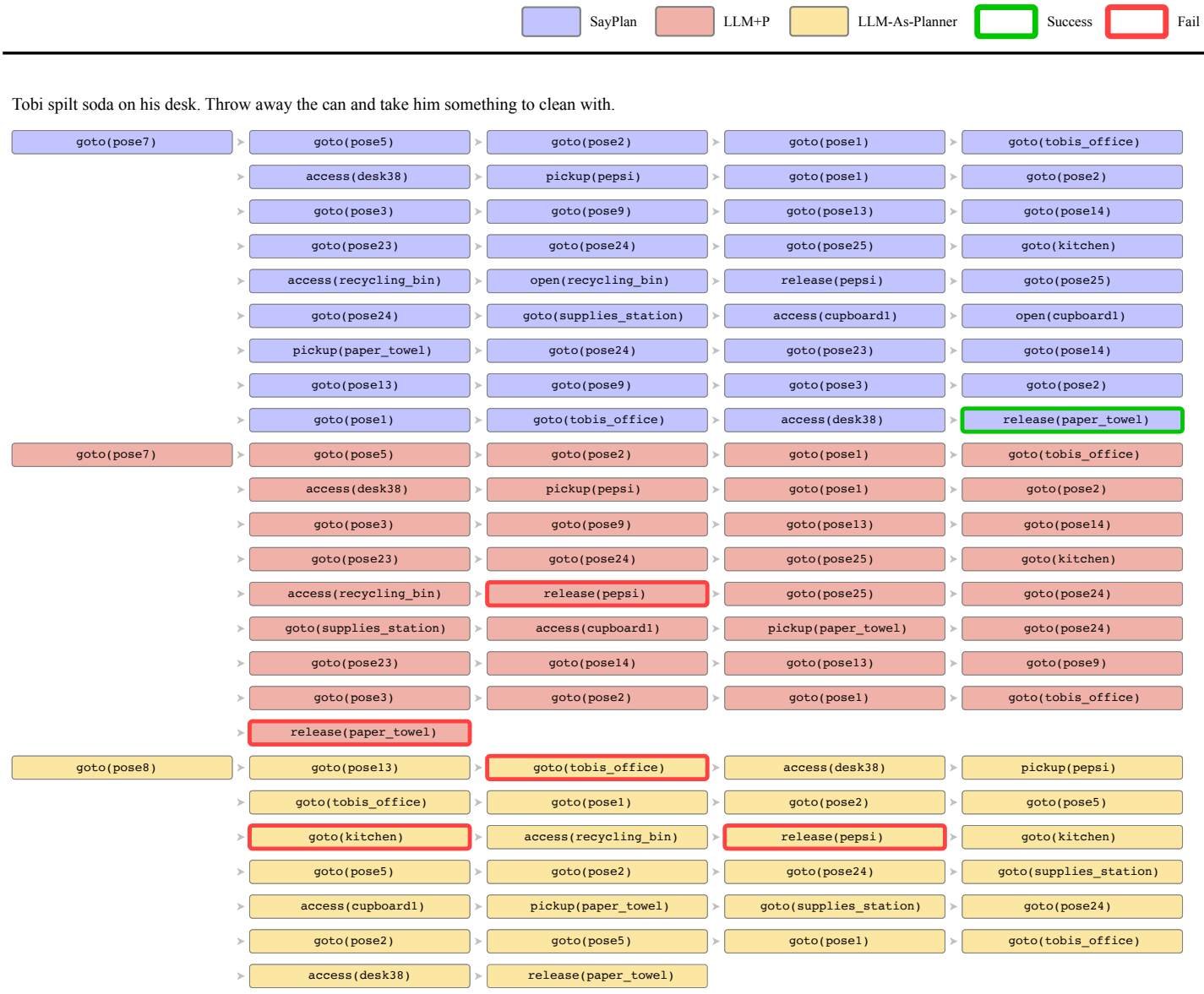
goto(pose13)	goto(pose14)	goto(pose23)	goto(pose24)	goto(pose25)
	goto(kitchen)	access(fridge)	open(fridge)	pickup(noodles)
	access(microwave)	open(microwave)	release(noodles)	close(microwave)
	turn_on(microwave)	turn_off(microwave)	open(microwave)	pickup(noodles)
	goto(pose25)	goto(cafeteria)	access(lunch_table)	release(noodles)
goto(pose13)	goto(pose14)	goto(pose23)	goto(pose24)	goto(pose25)
	goto(kitchen)	access(fridge)	open(fridge)	pickup(noodles)
	access(microwave)	open(microwave)	release(noodles)	turn_on(microwave)
	turn_off(microwave)	open(microwave)	pickup(noodles)	goto(pose25)
	goto(cafeteria)	access(lunch_table)	release(noodles)	
goto(pose25)	goto(kitchen)	access(fridge)	open(fridge)	pickup(noodles)
	close(fridge)	access(microwave)	open(microwave)	release(noodles)
	close(microwave)	turn_on(microwave)	turn_off(microwave)	open(microwave)
	pickup(noodles)	close(microwave)	goto(cafeteria)	release(noodles)

Throw the rotting fruit in Dimity's office in the correct bin.

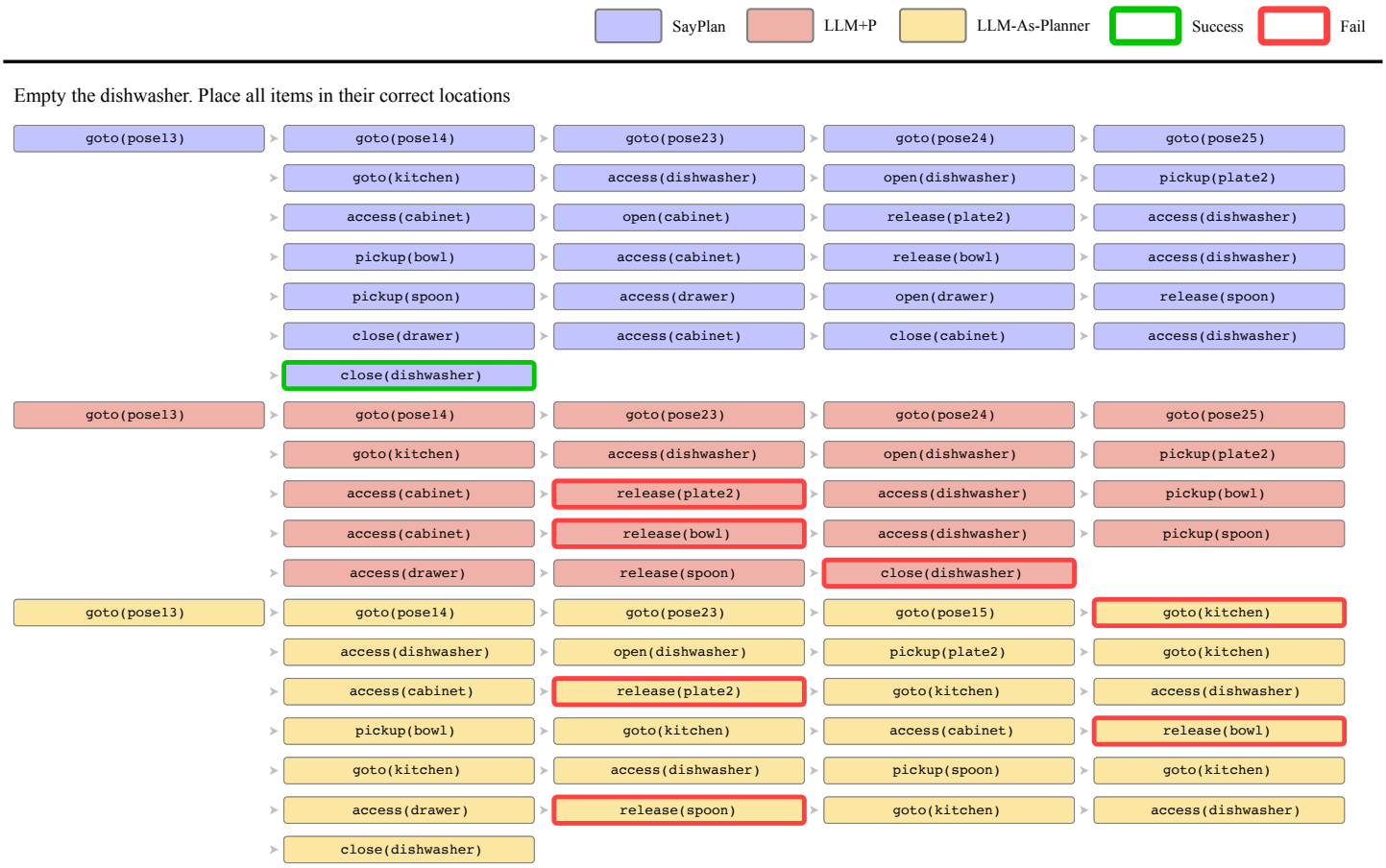
goto(pose13)	goto(pose14)	goto(pose23)	goto(pose22)	goto(dimitys_office)
	access(cabinet3)	open(cabinet3)	pickup(apple2)	goto(pose22)
	goto(pose23)	goto(pose24)	goto(pose25)	goto(kitchen)
	access(compost_bin)	open(compost_bin)	release(apple2)	
goto(pose13)	goto(pose14)	goto(pose23)	goto(pose22)	goto(dimitys_office)
	access(cabinet3)	open(cabinet3)	pickup(apple2)	goto(pose22)
	goto(pose23)	goto(pose24)	goto(pose25)	goto(kitchen)
	access(compost_bin)	release(apple2)		
goto(pose13)	goto(pose14)	goto(pose23)	goto(pose22)	goto(dimitys_office)
	access(cabinet3)	pickup(apple2)	goto(dimitys_office)	goto(pose22)
	goto(pose23)	goto(pose25)	goto(kitchen)	access(recycling_bin)
	release(apple2)			

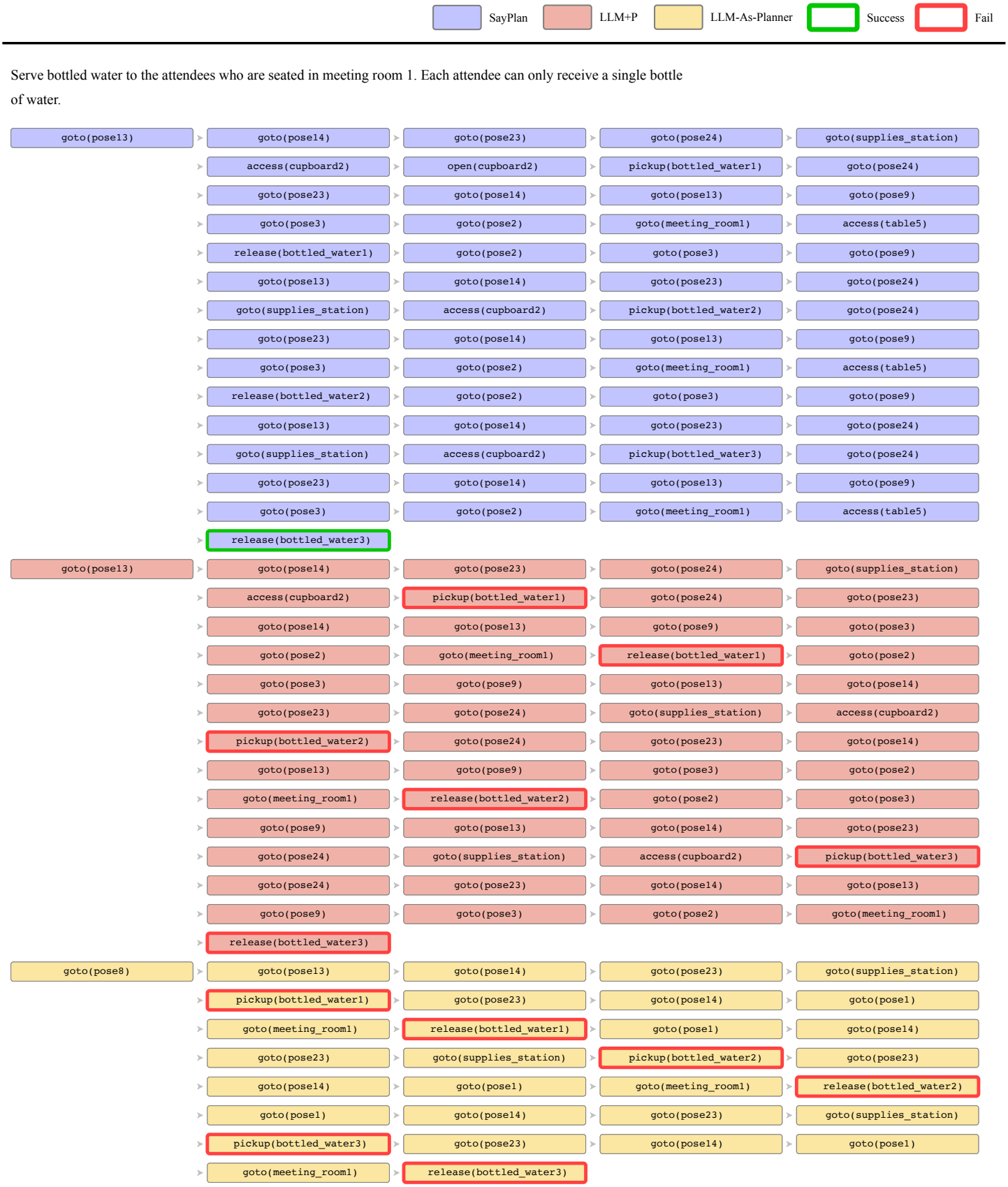

SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Wash all the dishes on the lunch table. Once finished, place all the clean cutlery in the drawer.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(cafeteria) access(lunch_table) pickup(plate) goto(pose25)
goto(kitchen) access(drawer) open(drawer) release(plate)
goto(pose25) goto(cafeteria) access(lunch_table) pickup(fork)
goto(pose25) goto(kitchen) access(drawer) release(fork)
goto(pose25) goto(cafeteria) access(lunch_table) pickup(knife)
goto(pose25) goto(kitchen) access(drawer) release(knife)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(cafeteria) access(lunch_table) pickup(plate) pickup(fork)
pickup(knife) goto(pose25) goto(kitchen) access(drawer)
release(plate) release(fork) release(knife)
goto(pose25) goto(cafeteria) goto(lunch_table) pickup(plate) pickup(fork)
pickup(knife) goto(pose25) goto(kitchen) goto(dishwasher)
open(dishwasher) release(plate) release(fork) release(knife)
close(dishwasher) turn_on(dishwasher) turn_off(dishwasher) open(dishwasher)
pickup(plate) pickup(fork) pickup(knife) goto(drawer)
open(drawer) release(fork) release(knife) close(drawer)
Safely file away the freshly printed document in Will's office then place the undergraduate thesis on his desk.
goto(pose13) goto(pose9) goto(pose3) goto(printing_zone2) access(printer2)
pickup(document) goto(pose3) goto(pose9) goto(pose10)
goto(wills_office) access(cabinet4) open(cabinet4) release(document)
close(cabinet4) access(cabinet4) open(cabinet4) pickup(undergrad_thesis1)
close(cabinet4) access(desk4) release(undergrad_thesis1)
pickup(document) goto(pose13) goto(pose9) goto(pose10) goto(wills_office)
access(cabinet4) open(cabinet4) release(document) close(cabinet4)
access(desk4) pickup(undergrad_thesis1) release(undergrad_thesis1)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(printing_zone1) pickup(freshly_printed_document) goto(pose24) goto(pose23)
goto(pose14) goto(pose13) goto(pose9) goto(pose10)
goto(wills_office) access(cabinet4) open(cabinet4) release(freshly_printed_document)
pickup(undergrad_thesis1) close(cabinet4) access(desk4) release(undergrad_thesis1)

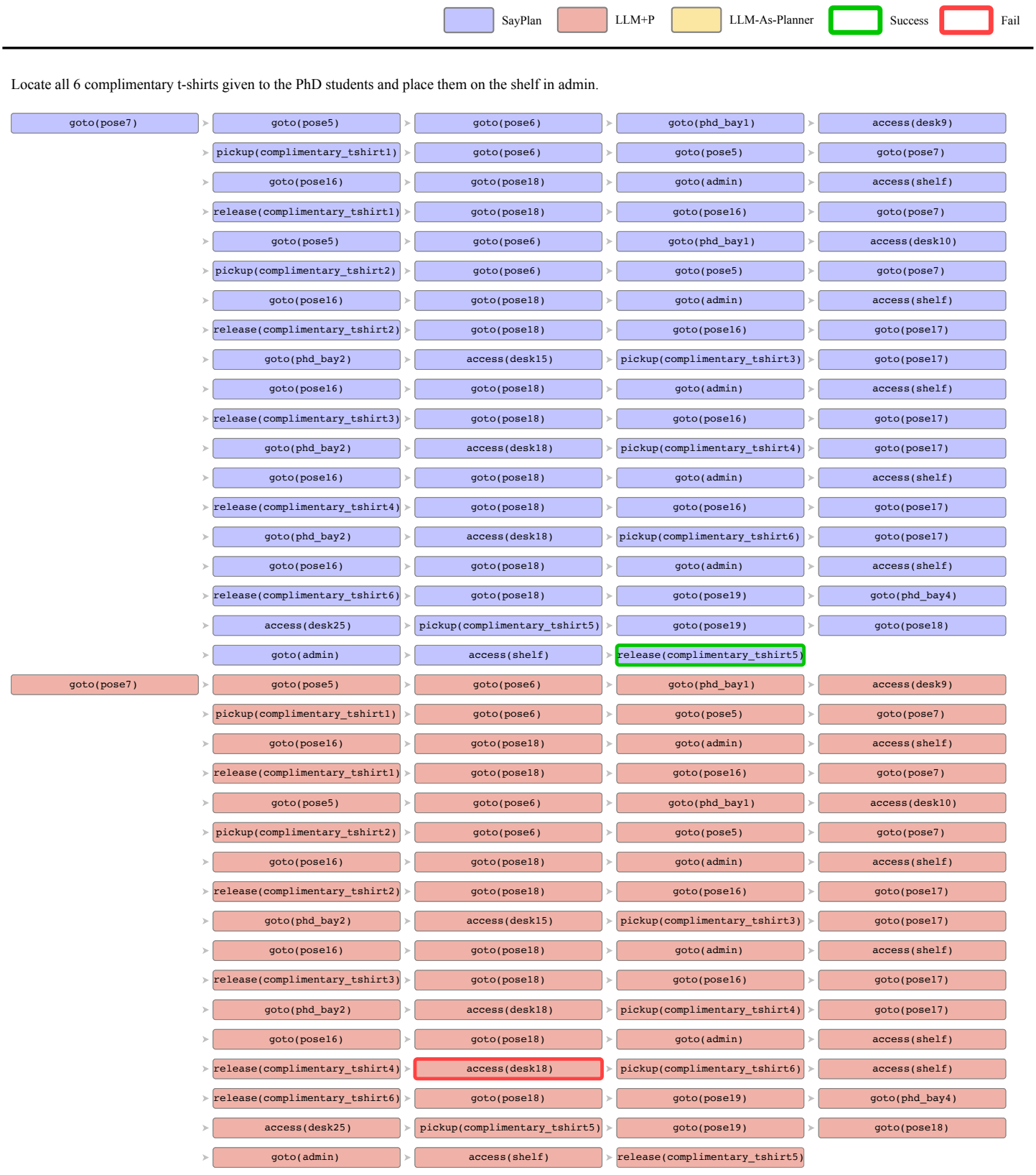

SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Make Niko a coffee and place the mug on his desk.
goto(pose7) goto(pose5) goto(nikos_office) access(desk1) pickup(coffee_mug)
goto(pose5) goto(pose7) goto(pose16) goto(pose18)
goto(pose15) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(coffee_machine) release(coffee_mug) turn_on(coffee_machine)
turn_off(coffee_machine) pickup(coffee_mug) goto(pose25) goto(pose24)
goto(pose23) goto(pose15) goto(pose18) goto(pose16)
goto(pose7) goto(pose5) goto(nikos_office) access(desk1)
release(coffee_mug)
goto(pose7) goto(pose5) goto(nikos_office) access(desk1) pickup(coffee_mug)
goto(pose5) goto(pose7) goto(pose8) goto(pose13)
goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(coffee_machine) release(coffee_mug) turn_on(coffee_machine)
turn_off(coffee_machine) pickup(coffee_mug) goto(pose25) goto(pose24)
goto(pose23) goto(pose14) goto(pose13) goto(pose8)
goto(pose7) goto(pose5) goto(nikos_office) access(desk1)
release(coffee_mug)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(pose25) goto(kitchen) pickup(coffee_mug) goto(pose25)
goto(pose24) goto(pose23) goto(pose14) goto(pose13)
goto(pose5) goto(nikos_office) goto(desk1) release(coffee_mug)
Someone has thrown items in the wrong bins. Correct this.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(recycling_bin) open(recycling_bin) pickup(milk_carton)
access(rubbish_bin) open(rubbish_bin) release(milk_carton) access(recycling_bin)
pickup(orange_peel) release(orange_peel) access(recycling_bin) pickup(apple_core)
release(apple_core) access(rubbish_bin) pickup(banana_peel) access(recycling_bin)
release(banana_peel) access(rubbish_bin) pickup(plastic_bottle) release(plastic_bottle)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(recycling_bin) open(recycling_bin) pickup(milk_carton)
access(rubbish_bin) open(rubbish_bin) release(milk_carton) access(recycling_bin)
pickup(orange_peel) release(orange_peel) access(recycling_bin) pickup(apple_core)
release(apple_core) access(rubbish_bin) pickup(banana_peel) access(recycling_bin)
release(banana_peel) access(rubbish_bin) pickup(plastic_bottle) release(plastic_bottle)
goto(pose13) goto(pose14) goto(pose23) goto(pose25) goto(kitchen)
goto(recycling_bin) pickup(milk_carton) goto(rubbish_bin) release(milk_carton)
goto(recycling_bin) pickup(orange_peel) goto(rubbish_bin) release(orange_peel)
goto(recycling_bin) pickup(apple_core) goto(rubbish_bin) release(apple_core)
goto(rubbish_bin) pickup(banana_peel) goto(recycling_bin) release(banana_peel)
goto(rubbish_bin) pickup(plastic_bottle) goto(recycling_bin) release(plastic_bottle)

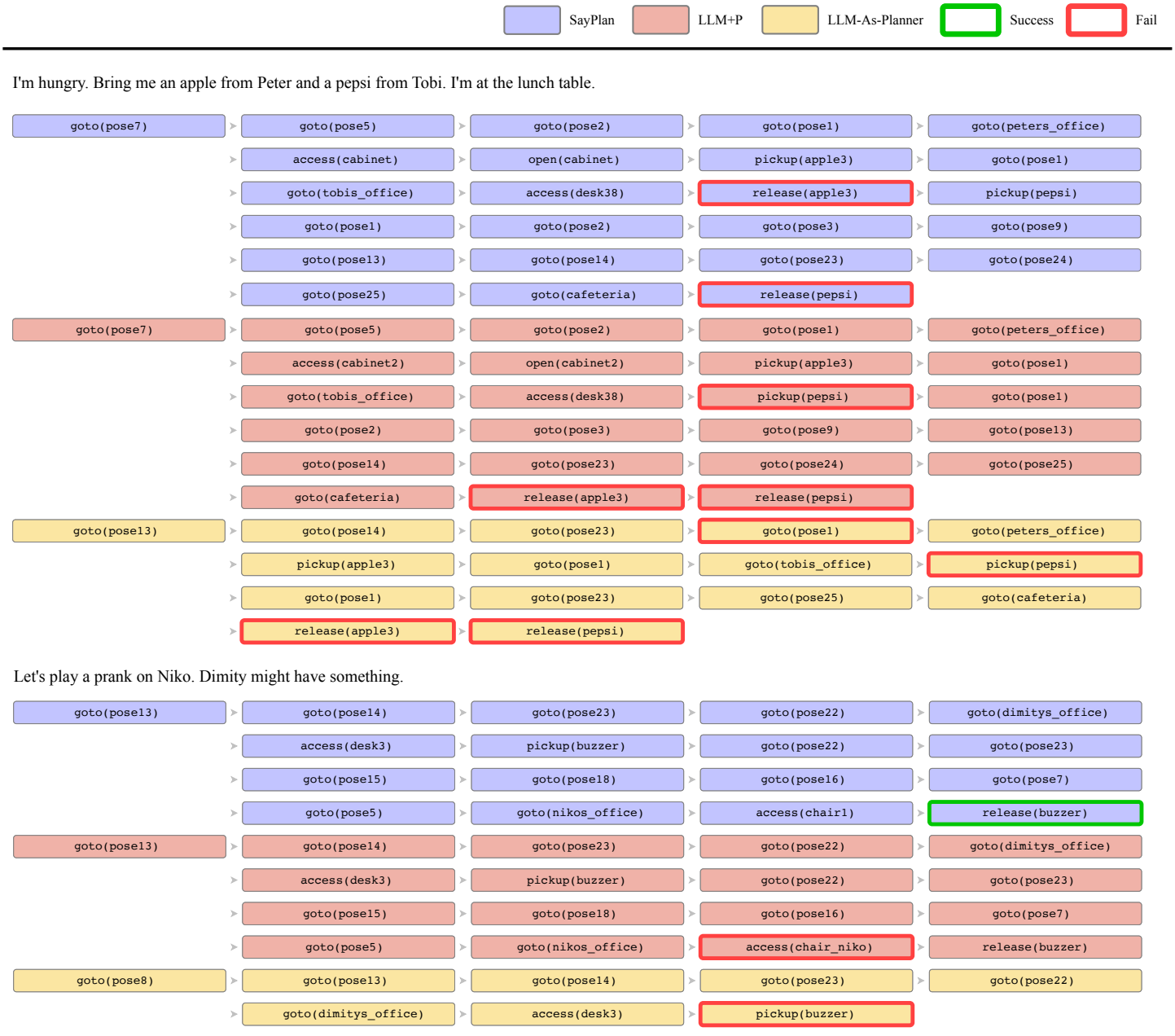

SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Tobi spilt soda on his desk. Throw away the can and take him something to clean with.
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(tobis_office)
access(desk38) pickup(pepsi) goto(pose1) goto(pose2)
goto(pose3) goto(pose9) goto(pose13) goto(pose14)
goto(pose23) goto(pose24) goto(pose25) goto(kitchen)
access(recycling_bin) open(recycling_bin) release(pepsi) goto(pose25)
goto(pose24) goto(supplies_station) access(cupboard1) open(cupboard1)
pickup(paper_towel) goto(pose24) goto(pose23) goto(pose14)
goto(pose13) goto(pose9) goto(pose3) goto(pose2)
goto(pose1) goto(tobis_office) access(desk38) release(paper_towel)
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(tobis_office)
access(desk38) pickup(pepsi) goto(pose1) goto(pose2)
goto(pose3) goto(pose9) goto(pose13) goto(pose14)
goto(pose23) goto(pose24) goto(pose25) goto(kitchen)
access(recycling_bin) release(pepsi) goto(pose25) goto(pose24)
goto(supplies_station) access(cupboard1) pickup(paper_towel) goto(pose24)
goto(pose23) goto(pose14) goto(pose13) goto(pose9)
goto(pose3) goto(pose2) goto(pose1) goto(tobis_office)
release(paper_towel)
goto(pose8) goto(pose13) goto(tobis_office) access(desk38) pickup(pepsi)
goto(tobis_office) goto(pose1) goto(pose2) goto(pose5)
goto(kitchen) access(recycling_bin) release(pepsi) goto(kitchen)
goto(pose5) goto(pose2) goto(pose24) goto(supplies_station)
access(cupboard1) pickup(paper_towel) goto(supplies_station) goto(pose24)
goto(pose2) goto(pose5) goto(pose1) goto(tobis_office)
access(desk38) release(paper_towel)


SayPlan
LLM+P
LLM-As-Planner
Success
Fail
I want to make a sandwich. Place all the ingredients on the lunch table.
goto(pose13)
goto(pose14)
goto(pose23)
goto(pose24)
goto(pose25)
goto(kitchen)
access(kitchen_bench)
pickup(bread)
goto(pose25)
goto(cafeteria)
access(lunch_table)
release(bread)
goto(pose25)
goto(kitchen)
access(kitchen_bench)
pickup(butter)
goto(pose25)
goto(cafeteria)
access(lunch_table)
release(butter)
goto(pose25)
goto(kitchen)
access(fridge)
open(fridge)
pickup(cheese)
goto(pose25)
goto(cafeteria)
access(lunch_table)
release(cheese)
goto(pose25)
goto(kitchen)
access(fridge)
pickup(tomato)
goto(pose25)
goto(cafeteria)
access(lunch_table)
release(tomato)
goto(pose13)
goto(pose14)
goto(pose23)
goto(pose24)
goto(pose25)
goto(kitchen)
access(kitchen_bench)
pickup(bread)
goto(pose25)
goto(cafeteria)
release(bread)
goto(pose25)
goto(kitchen)
pickup(butter)
goto(pose25)
goto(cafeteria)
release(butter)
goto(pose25)
goto(kitchen)
access(fridge)
open(fridge)
pickup(cheese)
goto(pose25)
goto(cafeteria)
release(cheese)
goto(pose25)
goto(kitchen)
pickup(tomato)
goto(pose25)
goto(cafeteria)
release(tomato)
goto(pose13)
goto(pose14)
goto(pose23)
goto(pose25)
goto(kitchen)
pickup(cheese)
pickup(tomato)
pickup(kale_leaves2)
pickup(bread)
pickup(butter)
goto(pose25)
goto(pose26)
goto(presentation_lounge)
release(cheese)
release(tomato)
release(kale_leaves2)
release(bread)
release(butter)


SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Empty the dishwasher. Place all items in their correct locations
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(dishwasher) open(dishwasher) pickup(plate2)
access(cabinet) open(cabinet) release(plate2) access(dishwasher)
pickup(bowl) access(cabinet) release(bowl) access(dishwasher)
pickup(spoon) access(drawer) open(drawer) release(spoon)
close(drawer) access(cabinet) close(cabinet) access(dishwasher)
close(dishwasher)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(kitchen) access(dishwasher) open(dishwasher) pickup(plate2)
access(cabinet) release(plate2) access(dishwasher) pickup(bowl)
access(cabinet) release(bowl) access(dishwasher) pickup(spoon)
access(drawer) release(spoon) close(dishwasher)
goto(pose13) goto(pose14) goto(pose23) goto(pose15) goto(kitchen)
access(dishwasher) open(dishwasher) pickup(plate2) goto(kitchen)
access(cabinet) release(plate2) goto(kitchen) access(dishwasher)
pickup(bowl) goto(kitchen) access(cabinet) release(bowl)
goto(kitchen) access(dishwasher) pickup(spoon) goto(kitchen)
access(drawer) release(spoon) goto(kitchen) access(dishwasher)
close(dishwasher)
A delegation of project partners is arriving soon. We want to serve them snacks and non-alcoholic drinks. Prepare everything in the largest meeting room. Use items found in the supplies room only.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
access(cupboard2) open(cupboard2) pickup(orange_juice) goto(pose24)
goto(pose25) goto(pose26) goto(meeting_room3) access(table1)
release(orange_juice) goto(pose26) goto(pose25) goto(pose24)
goto(supplies_station) access(cupboard2) pickup(biscuits) goto(pose24)
goto(pose25) goto(pose26) goto(meeting_room3) access(table1)
release(biscuits)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
access(cupboard2) open(cupboard2) pickup(orange_juice) goto(pose24)
goto(pose25) goto(pose26) goto(meeting_room3) release(orange_juice)
goto(pose26) goto(pose25) goto(pose24) goto(supplies_station)
pickup(biscuits) goto(pose24) goto(pose25) goto(pose26)
goto(meeting_room3) release(biscuits)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(supplies_station) open(cupboard2) pickup(bottled_water1) pickup(bottled_water2)
pickup(bottled_water3) pickup(bottled_water4) pickup(bottled_water5) pickup(orange_juice)
pickup(biscuits) close(cupboard2) goto(pose24) goto(pose25)
goto(pose26) goto(meeting_room3) release(bottled_water1) release(bottled_water2)
release(bottled_water3) release(bottled_water4) release(bottled_water5) release(orange_juice)
release(biscuits)


SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Serve bottled water to the attendees who are seated in meeting room 1. Each attendee can only receive a single bottle of water.
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
access(cupboard2) open(cupboard2) pickup(bottled_water1) goto(pose24)
goto(pose23) goto(pose14) goto(pose13) goto(pose9)
goto(pose3) goto(pose2) goto(meeting_room1) access(table5)
release(bottled_water1) goto(pose2) goto(pose3) goto(pose9)
goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(supplies_station) access(cupboard2) pickup(bottled_water2) goto(pose24)
goto(pose23) goto(pose14) goto(pose13) goto(pose9)
goto(pose3) goto(pose2) goto(meeting_room1) access(table5)
release(bottled_water2) goto(pose2) goto(pose3) goto(pose9)
goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(supplies_station) access(cupboard2) pickup(bottled_water3) goto(pose24)
goto(pose23) goto(pose14) goto(pose13) goto(pose9)
goto(pose3) goto(pose2) goto(meeting_room1) access(table5)
release(bottled_water3)
goto(pose13) goto(pose14) goto(pose23) goto(pose24) goto(supplies_station)
access(cupboard2) pickup(bottled_water1) goto(pose24) goto(pose23)
goto(pose14) goto(pose13) goto(pose9) goto(pose3)
goto(pose2) goto(meeting_room1) release(bottled_water1) goto(pose2)
goto(pose3) goto(pose9) goto(pose13) goto(pose14)
goto(pose23) goto(pose24) goto(supplies_station) access(cupboard2)
pickup(bottled_water2) goto(pose24) goto(pose23) goto(pose14)
goto(pose13) goto(pose9) goto(pose3) goto(pose2)
goto(meeting_room1) release(bottled_water2) goto(pose2) goto(pose3)
goto(pose9) goto(pose13) goto(pose14) goto(pose23)
goto(pose24) goto(supplies_station) access(cupboard2) pickup(bottled_water3)
goto(pose24) goto(pose23) goto(pose14) goto(pose13)
goto(pose9) goto(pose3) goto(pose2) goto(meeting_room1)
release(bottled_water3)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(supplies_station)
pickup(bottled_water1) goto(pose23) goto(pose14) goto(pose1)
goto(meeting_room1) release(bottled_water1) goto(pose1) goto(pose14)
goto(pose23) goto(supplies_station) pickup(bottled_water2) goto(pose23)
goto(pose14) goto(pose1) goto(meeting_room1) release(bottled_water2)
goto(pose1) goto(pose14) goto(pose23) goto(supplies_station)
pickup(bottled_water3) goto(pose23) goto(pose14) goto(pose1)
goto(meeting_room1) release(bottled_water3)


SayPlan
LLM+P
LLM-As-Planner
Success
Fail
Locate all 6 complimentary t-shirts given to the PhD students and place them on the shelf in admin.
goto(pose7) goto(pose5) goto(pose6) goto(phd_bay1) access(desk9)
pickup(complimentary_tshirt1) goto(pose6) goto(pose5) goto(pose7)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt1) goto(pose18) goto(pose16) goto(pose7)
goto(pose5) goto(pose6) goto(phd_bay1) access(desk10)
pickup(complimentary_tshirt2) goto(pose6) goto(pose5) goto(pose7)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt2) goto(pose18) goto(pose16) goto(pose17)
goto(phd_bay2) access(desk15) pickup(complimentary_tshirt3) goto(pose17)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt3) goto(pose18) goto(pose16) goto(pose17)
goto(phd_bay2) access(desk18) pickup(complimentary_tshirt4) goto(pose17)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt4) goto(pose18) goto(pose16) goto(pose17)
goto(phd_bay2) access(desk18) pickup(complimentary_tshirt6) goto(pose17)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt6) goto(pose18) goto(pose19) goto(phd_bay4)
access(desk25) pickup(complimentary_tshirt5) goto(pose19) goto(pose18)
goto(admin) access(shelf) release(complimentary_tshirt5)
goto(pose7) goto(pose5) goto(pose6) goto(phd_bay1) access(desk9)
pickup(complimentary_tshirt1) goto(pose6) goto(pose5) goto(pose7)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt1) goto(pose18) goto(pose16) goto(pose7)
goto(pose5) goto(pose6) goto(phd_bay1) access(desk10)
pickup(complimentary_tshirt2) goto(pose6) goto(pose5) goto(pose7)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt2) goto(pose18) goto(pose16) goto(pose17)
goto(phd_bay2) access(desk15) pickup(complimentary_tshirt3) goto(pose17)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt3) goto(pose18) goto(pose16) goto(pose17)
goto(phd_bay2) access(desk18) pickup(complimentary_tshirt4) goto(pose17)
goto(pose16) goto(pose18) goto(admin) access(shelf)
release(complimentary_tshirt4) access(desk18) pickup(complimentary_tshirt6) access(shelf)
release(complimentary_tshirt6) goto(pose18) goto(pose19) goto(phd_bay4)
access(desk25) pickup(complimentary_tshirt5) goto(pose19) goto(pose18)
goto(admin) access(shelf) release(complimentary_tshirt5)


SayPlan
LLM+P
LLM-As-Planner
Success
Fail
I'm hungry. Bring me an apple from Peter and a pepsi from Tobi. I'm at the lunch table.
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(peters_office)
access(cabinet) open(cabinet) pickup(apple3) goto(pose1)
goto(tobis_office) access(desk38) release(apple3) pickup(pepsi)
goto(pose1) goto(pose2) goto(pose3) goto(pose9)
goto(pose13) goto(pose14) goto(pose23) goto(pose24)
goto(pose25) goto(cafeteria) release(pepsi)
goto(pose7) goto(pose5) goto(pose2) goto(pose1) goto(peters_office)
access(cabinet2) open(cabinet2) pickup(apple3) goto(pose1)
goto(tobis_office) access(desk38) pickup(pepsi) goto(pose1)
goto(pose2) goto(pose3) goto(pose9) goto(pose13)
goto(pose14) goto(pose23) goto(pose24) goto(pose25)
goto(cafeteria) release(apple3) release(pepsi)
goto(pose13) goto(pose14) goto(pose23) goto(pose1) goto(peters_office)
pickup(apple3) goto(pose1) goto(tobis_office) pickup(pepsi)
goto(pose1) goto(pose23) goto(pose25) goto(cafeteria)
release(apple3) release(pepsi)
Let's play a prank on Niko. Dimity might have something.
goto(pose13) goto(pose14) goto(pose23) goto(pose22) goto(dimitys_office)
access(desk3) pickup(buzzer) goto(pose22) goto(pose23)
goto(pose15) goto(pose18) goto(pose16) goto(pose7)
goto(pose5) goto(nikos_office) access(chair1) release(buzzer)
goto(pose13) goto(pose14) goto(pose23) goto(pose22) goto(dimitys_office)
access(desk3) pickup(buzzer) goto(pose22) goto(pose23)
goto(pose15) goto(pose18) goto(pose16) goto(pose7)
goto(pose5) goto(nikos_office) access(chair_niko) release(buzzer)
goto(pose8) goto(pose13) goto(pose14) goto(pose23) goto(pose22)
goto(dimitys_office) access(desk3) pickup(buzzer)

G Real World Execution of a Generated Long Horizon Plan.

Figure 7: **Real World Execution of a Generated Long Horizon Plan.** Execution of a generated task plan on a real-world mobile manipulator robot.

H Input Prompt Structure

Input prompt passed to the LLM for SayPlan. Note that the components highlighted in violet represent static components of the prompt that remain fixed throughout both the semantic search and iterative replanning phases of SayPlan.

Agent Role: You are an excellent graph planning agent. Given a graph representation of an environment, you can explore the graph by expanding nodes to find the items of interest. You can then use this graph to generate a step-by-step task plan that the agent can follow to solve a given instruction.
Environment Functions:
goto(<pose>): Move the agent to any room node or pose node.
access(<asset>): Provide access to the set of affordances associated with an asset node and its connected objects.
pickup(<object>): Pick up an accessible object from the accessed node.
release(<object>): Release grasped object at an asset node.
turn_on/off(<object>): Toggle object at agent's node, if accessible and has affordance.
open/close(<asset>): Open/close asset at agent's node, affecting object accessibility.
done(): Call when the task is completed.
Environment State:
ontop_of(<asset>): Object is located on <asset>
inside_of(<asset>): Object is located inside <asset>
inside_hand: Object is currently being grasped by the robot/agent
closed: Asset can be opened
open: Asset can be closed or kept open
on: Asset is currently on
off: Asset is currently off
accessible: The object is not accessible if it is inside an asset and the asset state is "closed".
Environment API:
expand_node(<node>): Reveal assets/objects connected to a room/floor node.
contract_node(<node>): Hide assets/objects, reducing graph size for memory constraints.
verify_plan(): Verify generated plan in the scene graph environment.
Output Response Format:
{*chain_of_thought:* break your problem down into a series of intermediate reasoning steps to help you determine your next command,
reasoning: justify why the next action is important
mode: "exploring" OR "planning"
command: {"command_name": Environment API call
"node_id": node to perform an operation on
"plan": task plan if in planning mode}}
Example: <see Appendix I and J>
Instruction: Natural language description of the task
3D Scene Graph: Text-serialised JSON description of a 3D scene graph
Memory: History of previously expanded nodes
Feedback: External textual feedback from scene graph simulator
Ensure the response can be parsed by Python json.loads.

I Example Input-Output Interaction Between LLM and Scene Graph Simulator During Semantic Search

Instruction: make a coffee for Tom and place it in his room
3D Scene Graph: {nodes: {room: [{id: bobs_room}, {id: toms_room}, {id: jacks_room}, {id: kitchen}, {id: livingroom}], pose: [{id: pose1}, {id: pose2}, {id: pose3}, {id: pose4}, {id: pose5}], agent: [{location: bobs_room, id: agent}]}, links: [bobs_room↔pose1, bobs_room↔agent, toms_room↔pose1, toms_room↔pose2, toms_room↔pose5, jacks_room↔pose2, jacks_room↔pose3, kitchen↔pose3, kitchen↔pose4, kitchen↔pose5, livingroom↔pose4]}
Memory: []
SayPlan [Mode->Search]: chain-of-thought: explore graph for coffee mug, coffee machine and somewhere to place the coffee -> perform common sense exploration first -> explore toms room -> explore kitchen -> if items not found then explore other nodes
reasoning: i will expand toms room
command: expand(toms_room)
Scene Graph Simulator: <Updated 3D scene graph>
Memory: [toms_room]
SayPlan [Mode->Search]: chain-of-thought: i have found a wardrobe in tom's room -> leave this node expanded -> the coffee mug is not in his room -> still have not found the coffee machine -> kitchen might have coffee machine and coffee mug -> explore this node next
reasoning: i will expand the kitchen next
command: expand(kitchen)
Scene Graph Simulator: <Updated 3D scene graph>
Memory: [toms_room, kitchen]
SayPlan [Mode->Search]: chain-of-thought: i've found the coffee machine in the kitchen -> the coffee mug however is not in the kitchen -> i will explore other unexpanded nodes now for the coffee mug -> jacks room, bobs room and the living room are unexplored yet -> i will try jack"s room next
reasoning: i will expand jacks room next
command: expand(jacks_room)
Scene Graph Simulator: <Updated 3D scene graph>
Memory: [toms_room, kitchen, jacks_room]
SayPlan [Mode->Search]: chain-of-thought: jacks room does not have anything relevant towards solving this task -> I should contract this node to save token memory -> i will explore other unexplored nodes after doing this
reasoning: contract jack's room to save memory
command: contract(jacks_room)
Scene Graph Simulator: <Updated 3D scene graph>
Memory: [toms_room, kitchen, jacks_room]
SayPlan [Mode->Search]: chain-of-thought: i have found a wardrobe in tom's room and coffee machine in the kitchen -> the coffee mug is still required for the task -> explore a new room next
reasoning: i will expand bob's room as it is unexplored
command: expand(bobs_room)
Scene Graph Simulator: <Updated 3D scene graph>
Memory: [toms_room, kitchen, jacks_room, bobs_room]
SayPlan [Mode->Search]: chain-of-thought: i have found the coffee mug, the coffee machine and tom's wardrobe on the graph -> search complete -> I will switch to planning mode-> collect coffee mug -> generate plan for making coffee -> place coffee mug on Tom's wardrobe
reasoning: I will generate a task plan using the identified subgraph

J Example Input-Output Interaction Between LLM and Scene Graph Environment During Iterative replanning

Instruction: make a coffee for Tom and place it in his room
3D Scene Graph: {nodes: {room: [{id: bobs_room}, {id: toms_room}, {id: jacks_room}, {id: kitchen}, {id: livingroom}], pose: [{id: pose1}, {id: pose2}, {id: pose3}, {id: pose4}, {id: pose5}], agent: [{location: bobs_room, id: agent}], asset: [{room: toms_room, state: free, affordances: [release], id: bed2}, {room: toms_room, state: closed, affordances: [open, close, release], id: wardrobe2}, {room: kitchen, state: closed, affordances: [open, close, release], id: fridge}, {room: kitchen, affordances: [turn_on, turn_off], state: off, id: coffee_machine}, {room: bobs_room, state: free, affordances: [release], id: bed1}, {room: bobs_room, state: closed, affordances: [open, close, release], id: wardrobe1}], object: [{affordances: [pickup], state: inside_of(wardrobe1), attributes: "blue", id: coffee_mug}]}, links: [bobs_room↔pose1, bobs_room↔agent, bobs_room↔bed1, bobs_room↔wardrobe1, toms_room↔pose1, toms_room↔pose2, toms_room↔pose5, toms_room↔bed2, toms_room↔wardrobe2, jacks_room↔pose2, jacks_room↔pose3, kitchen↔pose3, kitchen↔pose4, kitchen↔pose5, kitchen↔fridge, kitchen↔coffee_machine, livingroom↔pose4, wardrobe1↔coffee_mug]}
Memory: [toms_room, kitchen, jacks_room, bobs_room]
SayPlan [Mode->Planning]: chain-of-thought: "i have found the coffee mug, the coffee machine and tom's wardrobe on the graph -> search complete -> I will switch to planning mode-> collect coffee mug -> generate plan for making coffee -> place coffee mug on Tom's wardrobe"
reasoning: I will generate a task plan using the identified subgraph
plan: [goto(bobs_room) > access(wardrobe1) > pickup(coffee_mug) > goto(kitchen) > access(coffee_machine) > release(coffee_mug) > turn_on(coffee_machine) > turn_off(coffee_machine) > pickup(coffee_mug) > goto(toms_room) > access(wardrobe2) > release(coffee_mug) > done]
Path Planner:plan: [goto(pose3) > goto(pose4) > goto(pose1) > goto(bobs_room) > access(wardrobe1) > pickup(coffee_mug) > goto(pose2) > goto(pose5) > goto(kitchen) > access(coffee_machine) > release(coffee_mug) > turn_on(coffee_machine) > turn_off(coffee_machine) > pickup(coffee_mug) > goto(pose5) > goto(pose2) > goto(toms_room) > access(wardrobe2) > release(coffee_mug) > done]
Scene Graph Simulator (Feedback): "*coffee mug is not accessible*"
Memory: [toms_room, kitchen, jacks_room, bobs_room]
SayPlan [Mode->Planning]: chain-of-thought: "the coffee mug was not accessible -> the wardrobe it is in is currently closed -> I update the plan to open the wardrobe first -> generate plan for making coffee -> place coffee mug on Tom's wardrobe"
reasoning: I will generate a task plan using the identified subgraph and the feedback from the simulator
plan: [goto(bobs_room) > access(wardrobe1) > open(wardrobe1) > pickup(coffee_mug) > goto(kitchen) > access(coffee_machine) > release(coffee_mug) > turn_on(coffee_machine) > turn_off(coffee_machine) > pickup(coffee_mug) > goto(toms_room) > access(wardrobe2) > release(coffee_mug) > done]
Path Planner:plan: [goto(pose3) > goto(pose4) > goto(pose1) > goto(bobs_room) > access(wardrobe1) > open(wardrobe1) > pickup(coffee_mug) > goto(pose2) > goto(pose5) > goto(kitchen) > access(coffee_machine) > release(coffee_mug) > turn_on(coffee_machine) > turn_off(coffee_machine) > pickup(coffee_mug) > goto(pose5) > goto(pose2) > goto(toms_room) > access(wardrobe2) > release(coffee_mug) > done]
Scene Graph Simulator: Plan Verified