

# Leveraging 3D Scene Graphs in Large Language Models for Task Planning

Krishan Rana\*, Sourav Garg<sup>†</sup>, Jesse Haviland\*, Jad Abou-Chakra\*, Ian Reid<sup>†</sup> and Niko Sünderhauf\*  
Email: ranak@qut.edu.au

\*QUT Centre for Robotics, Queensland University of Technology, Brisbane

<sup>†</sup>Australian Institute for Machine Learning, University of Adelaide

**Abstract**—Advancements in Large Language Models (LLMs) have paved the way for versatile planning agents capable of tackling a myriad of tasks. Nevertheless, grounding these plans in complex, multi-room, multi-floor environments poses a formidable challenge in robotics. To address this, we propose SayPlan: a scalable method for LLM-based task planning in large-scale environments by leveraging 3D Scene Graph (3DSG) representations. Our method ensures scalability by (1) utilizing the hierarchical structure of 3DSGs to perform a *semantic search* for task-relevant subgraphs from a collapsed representation of the full graph; (2) integrating a classical path planner to decrease the planning horizon for the LLM, and (3) implementing an *iterative replanning* mechanism, which refines initial plans based on feedback from a scene graph simulator correcting for unexecutable action sequences, and preventing planning failures. We validate our approach in two extensive environments spanning up to 3 floors, 36 rooms and 140 objects. Our results demonstrate that SayPlan effectively translates large-scale, long-horizon task plans from natural language instructions suitable for execution on a mobile manipulator robot. We provide real robot demonstrations and code on our project page [sayplan.github.io](https://sayplan.github.io)

## I. INTRODUCTION

Advances in Large Language Models (LLMs) [1, 2, 3] have shown potential for common sense reasoning in robotics, allowing for complex planning strategies for various tasks. Yet, for these LLMs to function as effective planners, they must adapt to the constraints of the physical environment, comprehend the topological arrangements and understand where they are. Existing research has employed vision-based value functions [4], object detectors [5, 6], and Planning Domain Definition Language (PDDL) [7, 8] for grounding these planners, but these are often limited to small-scale environments. Our work aims to scale these methods, proposing a framework to ground LLM-based task planners across multiple rooms and floors through 3D scene graph (3DSG) representations [9, 10, 11, 12, 13, 14] which capture a rich, hierarchically-organised semantic graph representation of an environment.

We propose three key innovations to ensure scalability. Firstly, we enable the LLM to perform a semantic search for a task-relevant subgraph by manipulating nodes of a *collapsed* graph, allowing planning over large environments

without exceeding token limits. Secondly, we relax the need for LLMs to generate path-level navigational plans and instead employ an existing optimal path planner to connect high-level nodes generated by the LLM. Lastly, we introduce an iterative replanning pipeline that refines the plan using feedback from a scene graph simulator to avoid inconsistencies and hallucinations.

Our approach, SayPlan, ensures grounded and feasible plan generation for robots operating in expansive environments. We evaluate our framework across 90 tasks of varying difficulty in two large environments, including a large office floor and a three-storey house. Our experiments demonstrate SayPlan’s ability to scale task planning to large-scale environments while conserving a low token footprint, providing promising implications for real-world robotics.<sup>1</sup>

## II. RELATED WORK

The domain of **robotic task planning** traditionally employs languages such as PDDL [15, 16, 17] and search methods [18, 19], supplemented by complex heuristics [20] to formulate a solution. Despite their effectiveness, these techniques lack versatility when expanding to more complex tasks in larger environments. Learning-based alternatives, which include hierarchical and reinforcement learning methods [21, 22, 23], have data-intensive demands and struggle with scalability. Recent works leverage **LLMs for task planning** and demonstrate their ability to generate executable plans for embodied agents by grounding them using various strategies such as object detectors [5], value functions [4] or PDDL environment descriptions [8]. However, these strategies are primarily limited to single-room environments, often scaling poorly with an increased number of objects or lacking a comprehensive environment-level prior [5, 4]. In this work, we propose the integration of hierarchical 3D scene graph representations for grounding LLM plans in large-scale environments.

## III. SAYPLAN

### A. Problem Formulation

We aim to address the challenge of long-range planning for an autonomous agent, such as a mobile manipulator

The authors 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 NS.

<sup>1</sup>[sayplan.github.io](https://sayplan.github.io)

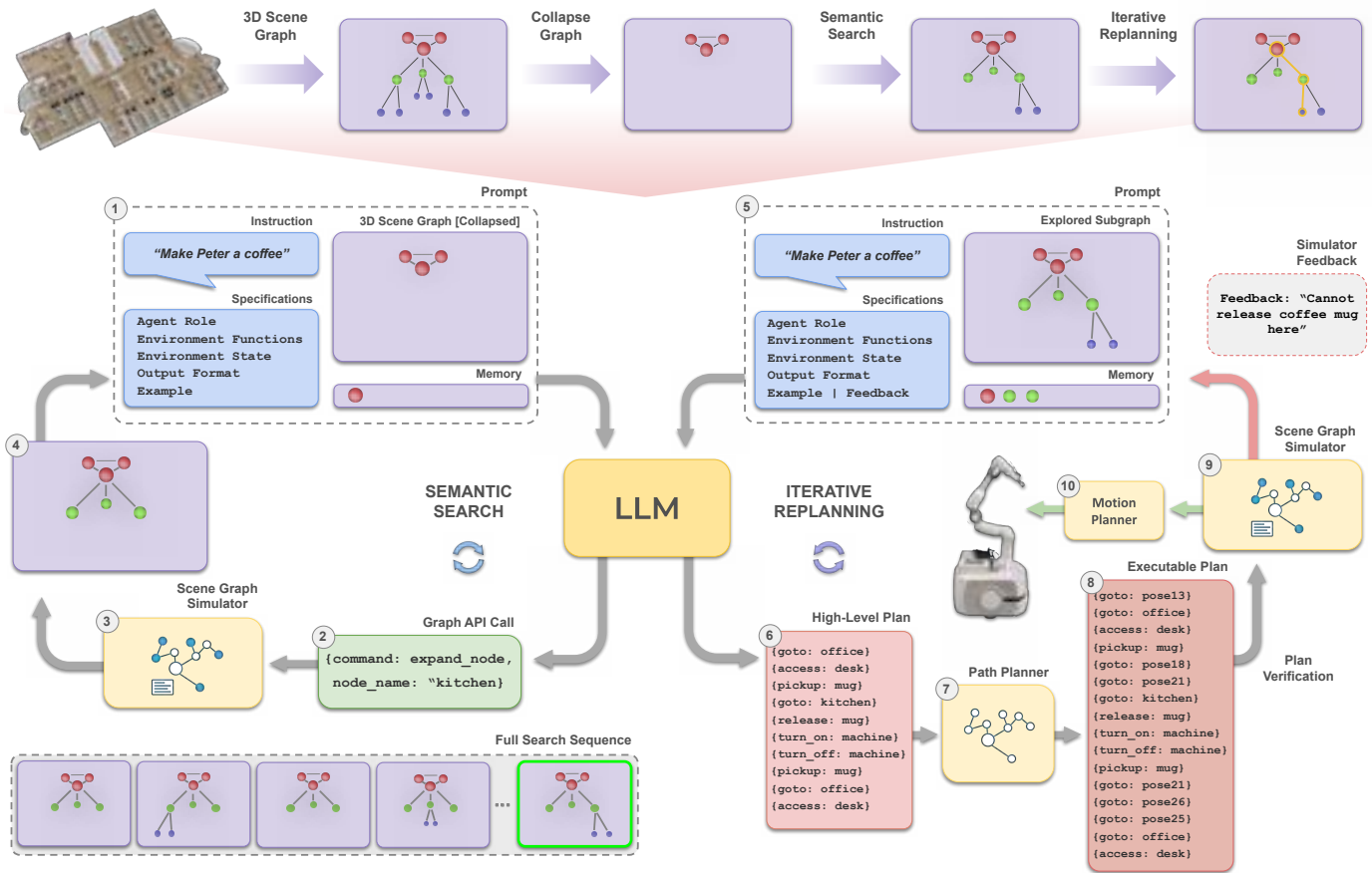


Fig. 1: **SayPlan Overview.** SayPlan operates across two stages to ensure scalability: (left) Given a collapsed 3D scene graph and a task instruction, *semantic search* is conducted by the LLM to identify a suitable subgraph that contains the desired items for the task; (right) The explored subgraph 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 the order of operation.

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.

**B. Preliminaries**

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

**Scene Representation:** The 3D Scene Graph (3DSG) [9, 10, 12] has recently emerged as an actionable world representation for robots [11, 13, 14, 24, 25, 26], which hierarchically

abstracts the environment at multiple levels through spatial semantics, object relationships, affordances, attributes, as well as 3D pose information. Formally, a 3DSG is a hierarchical multigraph  $G = (V, E)$  in which the set of vertices  $V$  comprises  $V_1 \cup V_2 \cup \dots \cup V_K$ , 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. A visual depiction of this structure is given in Figure 2.

We assume a pre-constructed 3D scene graph representation of a large-scale environment generated using existing techniques [13, 11, 9]. The entire 3D scene graph can be represented as a NetworkX Graph object [27] and text-serialised into a JSON data format which 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, 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.

### C. Approach

Using 3D scene graph representations  $\mathcal{G}$  and by defining a task instruction  $\mathcal{I}$  in natural language, we can view our framework SayPlan as a high-level task planner  $\pi(a|\mathcal{I}, \mathcal{G})$ , capable of generating long-horizon plans  $a$  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*.

**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. 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 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 [2, 28]. We leverage Chain-of-Thought (CoT) [29] 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 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 [30] 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 VIII.

**Iterative Re-planning:** Given the 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 query, and used to generate an updated, executable plan. This iterative process, involving planning, validation, 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 re-planning is provided in Appendix IX.

## IV. 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. 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:

*a) 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 nodes 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.

*b) 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 pose-level 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 navigation plan sequences, however, without any iterative re-planning.

## V. RESULTS

We summarise the results for the semantic search evaluation in Table III. 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.

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.

### A. Semantic Search

We present the full sequence of explored nodes for both SayPlan (GPT-4) and the human baseline in Appendix IV. 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. Additional ablation studies for the semantic search pipeline are given in Appendix III.

### B. Causal Planning

The results of causal planning across simple and long-horizon instructions (Table IV, left) reveal the effectiveness of SayPlan. We compared its performance with two baselines: LLM-As-Planner and LLM+P. All three methods displayed consistent correctness in simple planning tasks (93%). However, in long-horizon tasks, LLM-As-Planner had a significant performance drop to 33.3%. LLM+P, which incorporates a path planner, performed better at 66.7%, highlighting the planner’s importance. SayPlan, benefitting from both a path planner and iterative re-planning, achieved the highest correctness rate (73.3%). The key to ensuring plan executability is iterative re-planning. Both LLM-As-Planner and LLM+P showed poor executability, whereas SayPlan achieved near-perfect executability as a result of iterative re-planning, ensuring plans are grounded to match the environment’s feasibility.

Detailed task plans and errors encountered are provided in Appendix V. We summarise these errors in Table IV (right) which shows that plans generated with LLM+P and LLM-As-Planner entailed various types of errors limiting their executability. LLM+P mitigated path planning errors as a result of the classical path planner however still suffered from errors pertaining to the manipulation of the environment. 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 Figure 7.

## VI. CONCLUSION

SayPlan is a natural language-driven planning framework for robotics that integrates hierarchical 3D scene graphs to ground LLMs when planning across large-scale environments spanning multiple floors and rooms. To ensure the scalability of our approach we introduce a semantic search pipeline which exploits the heirarchical nature of 3DSGs which significantly reduces the number of tokens required to represent a full environment. The feasibility of generated plans are guaranteed using a combination of classical path planners and iterative feedback from a scene graph simulator. Compared to existing techniques, SayPlan yields the most correct and executable plans suitable for direct deployment on a real world mobile manipulator robot.

## REFERENCES

- [1] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [3] OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- [4] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan 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] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. *arXiv preprint arXiv:2212.04088*, 2022.
- [6] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.
- [7] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.
- [8] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.
- [9] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio 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.
- [10] Ue-Hwan Kim, Jin-Man Park, Taek-Jin Song, and Jong-Hwan 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.
- [11] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.
- [12] Paul Gay, James Stuart, and Alessio 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.
- [13] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization. *Robotics: Science and Systems XIV*, 2022.
- [14] Christopher Agia, Krishna Murthy Jatavallabhula, Mohamed Khodeir, Ondrej Miksik, Vibhav Vineet, Mustafa Mukadam, Liam Paull, and Florian Shkurti. Taskography: Evaluating robot task planning over large 3d scene graphs. In *Conference on Robot Learning*, pages 46–58. PMLR, 2022.
- [15] Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. Pddl—the planning domain definition language. 1998.
- [16] Maria Fox and Derek Long. Pddl2. 1: An extension to pddl for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.
- [17] Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, and Christian Muise. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2):1–187, 2019.
- [18] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the national conference on artificial intelligence*, pages 1194–1201, 1996.
- [19] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1-2):5–33, 2001.
- [20] Mauro Vallati, Lukas Chrpá, Marek Grześ, Thomas Leo McCluskey, Mark Roberts, Scott Sanner, et al. The 2014 international planning competition: Progress and trends. *Ai Magazine*, 36(3):90–98, 2015.
- [21] Rohan Chitnis, Tom Silver, Beomjoon Kim, Leslie Kaelbling, and Tomás Lozano-Pérez. Camps: Learning context-specific abstractions for efficient planning in factored mdps. In *Conference on Robot Learning*, pages 64–79. PMLR, 2021.
- [22] Tom Silver, Rohan Chitnis, Aidan Curtis, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack 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.
- [23] Federico Ceola, Elisa Tosello, Luca Tagliapietra, Giorgio Nicola, and Stefano 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.
- [24] Zachary Ravichandran, Lisa Peng, Nathan Hughes, J Daniel Griffith, and Luca 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.
- [25] Andrey Kurenkov, Roberto Martín-Martín, Jeff Ichnowski, Ken Goldberg, and Silvio 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.
- [26] Sourav Garg, Niko Sünderhau, Feras Dayoub, Douglas Morrison, Akansel Cosgun, Gustavo Carneiro, Qi Wu, Tat-Jun Chin, Ian Reid, Stephen Gould, et al. Semantics for robotic mapping, perception and interaction: A survey. *Foundations and Trends® in Robotics*, 8(1–2):1–224, 2020.
- [27] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.
- [28] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.
- [29] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- [30] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application, 2023.

## APPENDIX

### I. IMPLEMENTATION DETAILS

We utilise GPT-4 [3] as the underlying LLM agent unless otherwise stated. We follow a similar prompting structure to [30] as shown in Appendix VII. 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.

---

### Algorithm 1: SayPlan

---

**Given:** scene graph simulator  $\psi$ , classical path planner  $\phi$ , large language model  $LLM$

**Inputs:** prompt  $\mathcal{P}$ , scene graph  $\mathcal{G}$ , instruction  $\mathcal{I}$

```

1:  $\mathcal{G}' \leftarrow \text{collapse}_{\psi}(\mathcal{G})$  ▷ collapse scene graph
   Stage 1: Semantic Search ▷ search scene graph for all relevant items
2: while command  $\neq$  terminate do
3:   command  $\leftarrow LLM(\mathcal{P}, \mathcal{G}', \mathcal{I})$ 
4:   if command == 'expand' then
5:      $\mathcal{G}' \leftarrow \text{expand}_{\psi}(\text{node\_id})$  ▷ reveal objects and assets
6:   else if command == 'contract' then
7:      $\mathcal{G}' \leftarrow \text{contract}_{\psi}(\text{node\_id})$  ▷ nothing relevant found
   Stage 2: Causal Planning ▷ generate a feasible plan
8:   while feedback  $\neq$  success do
9:     plan  $\leftarrow LLM(\mathcal{P}, \mathcal{G}', \mathcal{I}, \text{feedback})$  ▷ high level plan
10:    full_plan  $\leftarrow \phi(\text{plan}, \mathcal{G}')$  ▷ compute optimal path between nodes
11:    feedback  $\leftarrow \text{verify\_plan}_{\psi}(\text{full\_plan})$  ▷ forward simulate plan
12: return full_plan ▷ executable plan

```

---

## II. EXPERIMENTAL SETUP

### Environments

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 visual 3D scene graph representation of the office floor is provided in Figure 5 and 6. We utilise generic names across our scene to maintain anonymity throughout the review process.

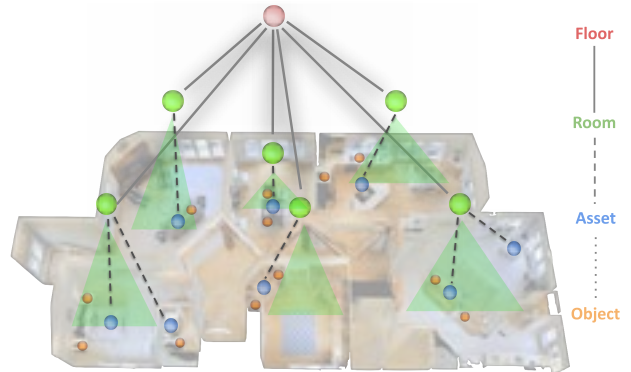


Fig. 2: **Hierarchical Structure of a 3D Scene Graph.** This graph consists of 4 layers. Room nodes are connected to one another via sequences of pose nodes which capture the topological arrangement of a scene.

**Home:** An existing 3D scene graph from the Stanford 3D Scene Graph dataset [9] which consists of a family home environment (Klickitat) spanning 32 rooms across 3 floors and contains 121 assets and objects which the agent can



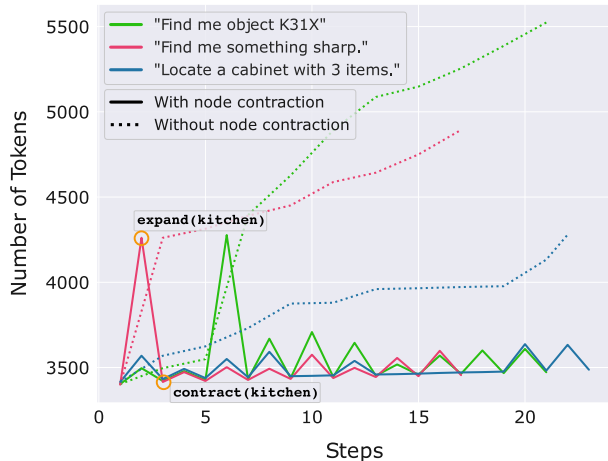


Fig. 3: **Scene Graph Token Progression During Semantic Search.** This graph illustrates the scalability of our approach to large-scale 3D scene graphs

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

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

interact with. A 3D visual of this environment can be viewed at the 3D Scene Graph project website.<sup>2</sup>

### 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 for each of the above sets are given in Tables V to X.

<sup>2</sup>3dscenegraph.stanford.edu/Klickitat

### III. SEMANTIC SEARCH ABLATION

We additionally analyse the scalability of SayPlan during semantic search. Table I 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 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 described in Section III-C.

Instruction Family	Num	Explanation	Example Instruction
<b>Semantic Search</b>			
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.
<b>Causal Planning</b>			
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 II: **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 III.

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

	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 IV: **Causal Planning Results.** *Left:* Correctness and Executability 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.

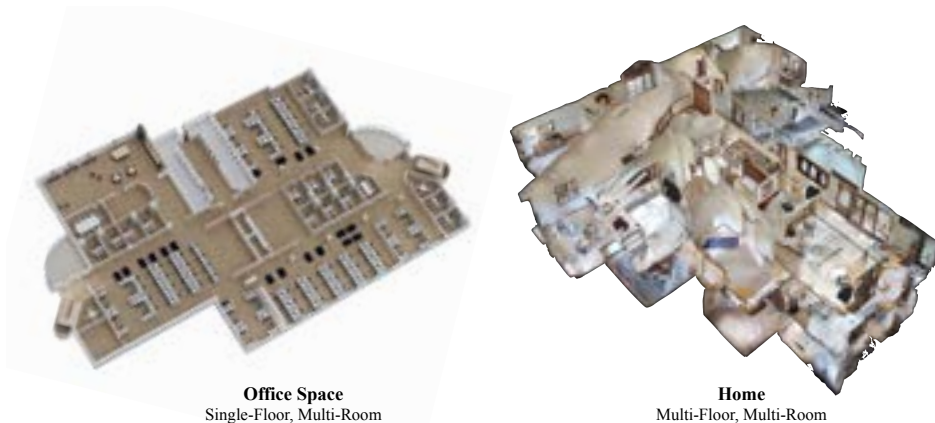


Fig. 4: **Large-scale environments used to evaluate SayPlan.** The environments exhibit multiple rooms and floors each containing various assets and objects that the agent can interact with.



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

### A. Simple Search

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 V: Simple Search Instructions. Evaluated in Office 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 conjunctive 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 VI: Simple Search Instructions. Evaluated in Home Environment.

## B. Complex Search

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 used 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 VII: Complex Search Instructions. Evaluated in Office 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 multiple returns
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, termination after exploring closets

TABLE VIII: Complex Search Instructions. Evaluated in Home Environment.

### C. 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 IX: **Simple Planning Instructions.** Evaluated in Office Environment.

### D. 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 X: **Long-Horizon Planning Instructions.** Evaluated in Office Environment.

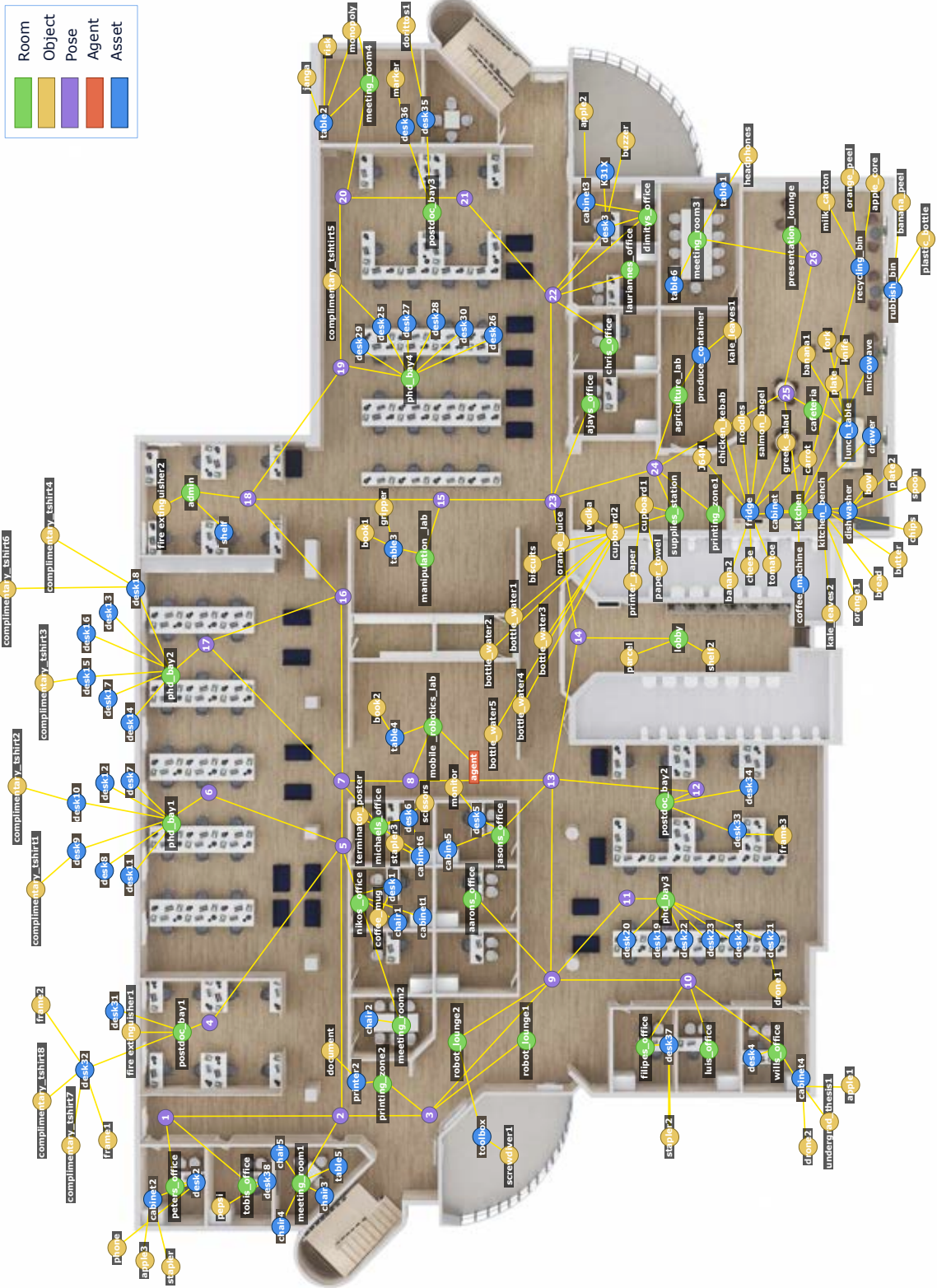


Fig. 5: **3D Scene Graph - Fully Expanded Office Environment.** Full 3D scene graph exposing all the rooms, assets and objects available in the scene. Note that the LLM agent never sees all this information unless it chooses to expand every possible node without contraction.



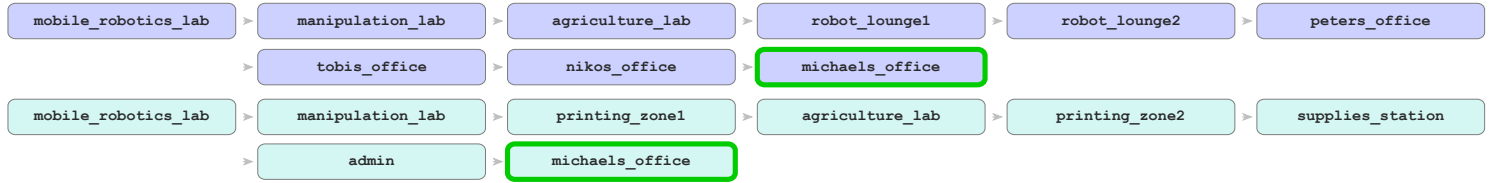
Fig. 6: **3D Scene Graph - Contracted Office Environment.** Contracted 3D scene graph exposing only the highest level within the hierarchy - room nodes. This results in an 82.1% reduction in the number of tokens required to represent the scene before the exploration phase.

#### IV. SEMANTIC SEARCH EVALUATION RESULTS

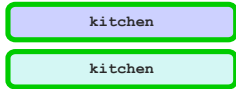
- Full listings of the generated semantic search sequences for the evaluation instruction sets are provided on the following page -



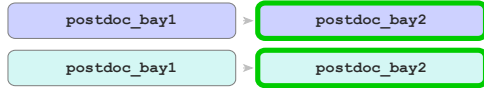
Find me object K31X.



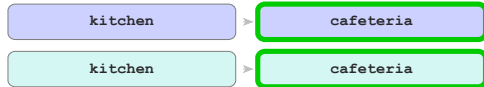
Find me a carrot.



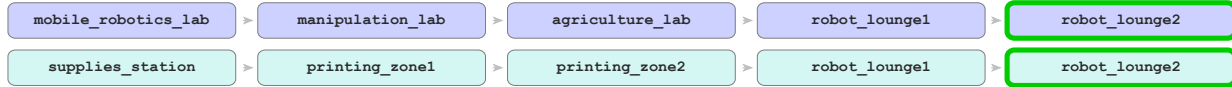
Find me anything purple in the postdoc bays.



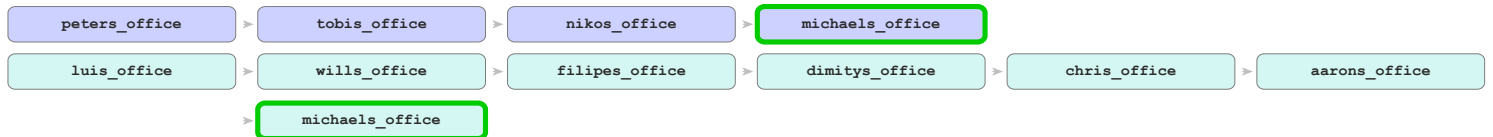
Find me a ripe banana.



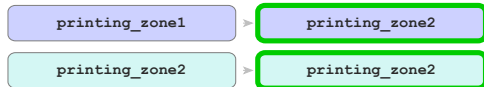
Find me something that has a screwdriver in it.



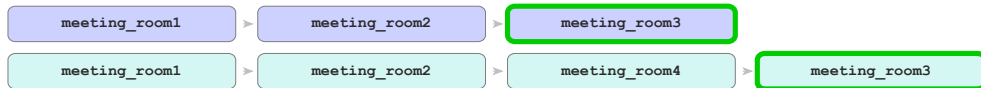
One of the offices has a poster of the Terminator. Which one is it?



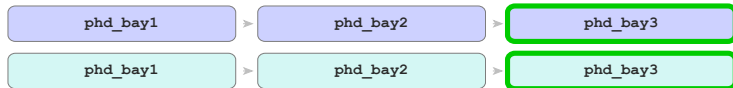
I printed a document, but I dont know which printer has it. Find the document.



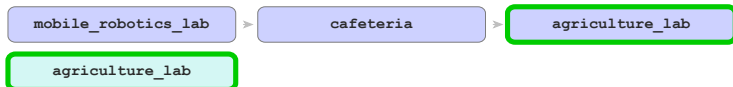
I left my headphones in one of the meeting rooms. Locate them.



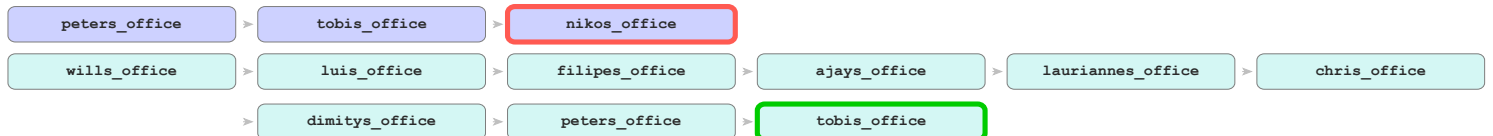
Find the PhD bay that has a drone in it.



Find the kale that is not in the kitchen.

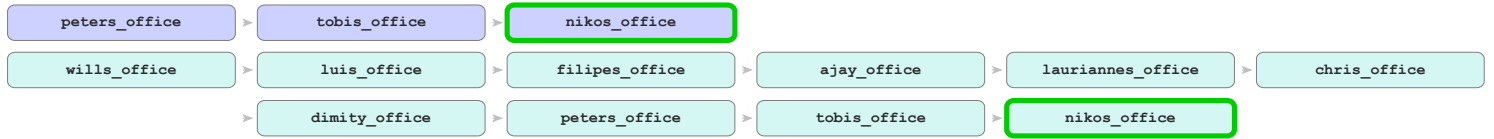


Find me an office that does not have a cabinet.

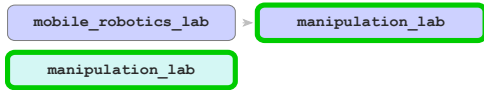




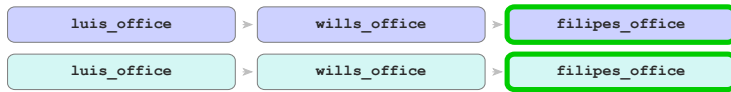
Find me an office that contains a cabinet, a desk and a chair.



Find me a book that was left next to a robotic gripper.



Luis gave one of his neighbours a stapler. Find the stapler.



There is a meeting room with a chair but no table. Locate it.

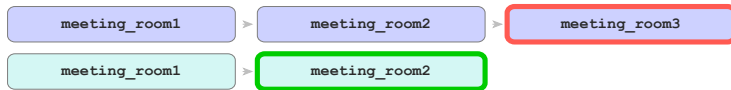
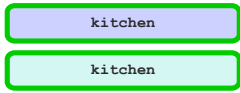
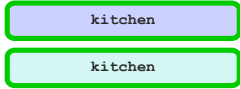


TABLE XI: **Simple Search Office Environment Evaluation.** Sequence of Explored Nodes for Simple Search Office Environment Instructions.

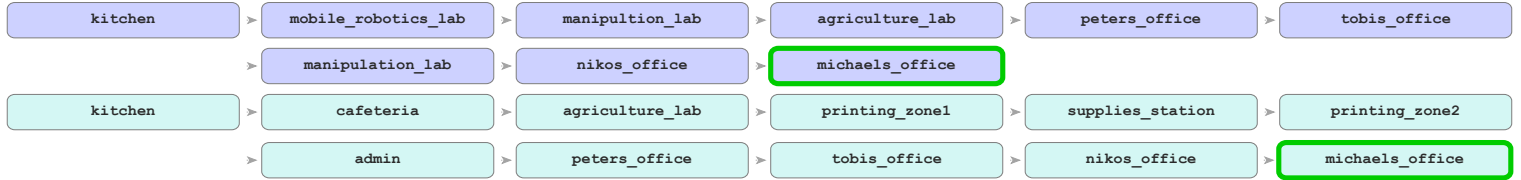
Find object J64M. J64M should be kept at below 0 degrees Celsius.



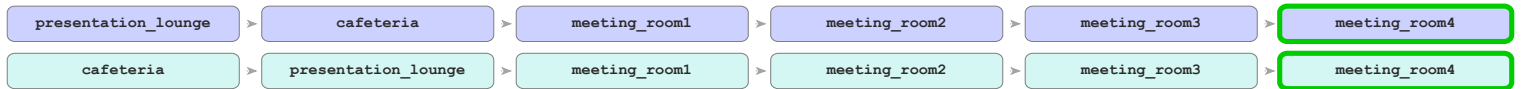
Find me something non vegetarian.



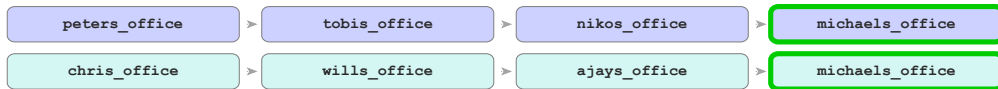
Locate something sharp.



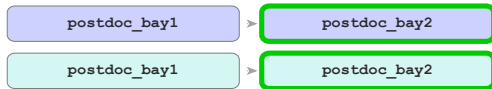
Find the room where people are playing board games..



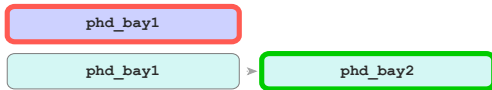
Find the office of someone who is clearly a fan of Arnold Schwarzenegger.



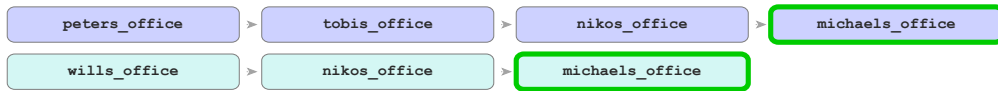
There is postdoc that has a pet Husky. Find the desk that's most likely theirs.



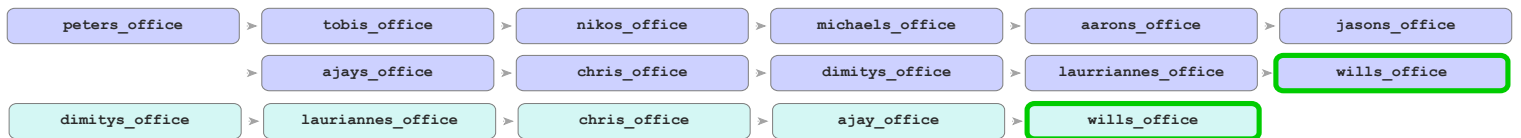
One of the PhD students was given more than one complimentary T-shirt. Find his desk.



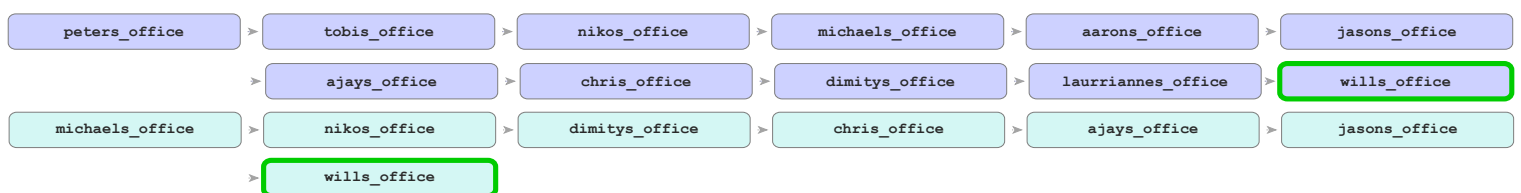
Find me the office where a paper attachment device is inside an asset that is open.



There is an office which has a cabinet containing exactly 3 items in it. Locate the office.

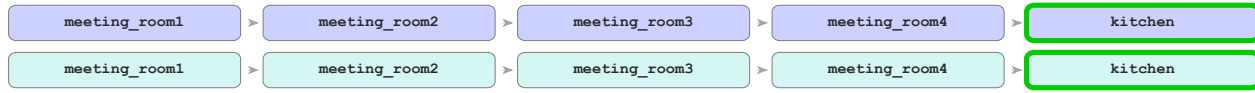


There is an office containing a rotten apple. The cabinet name contains an even number. Locate the office.

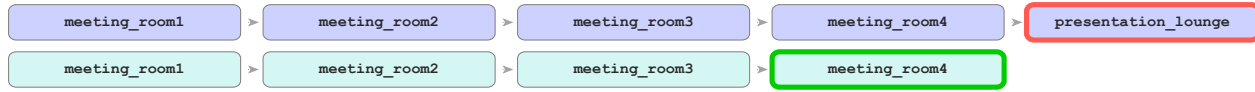




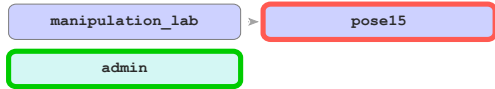
Look for a carrot. The carrot is likely to be in a meeting room but I'm not sure.



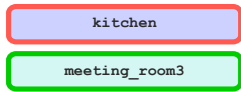
Find me a meeting room with a RealSense camera.



Find the closest fire extinguisher to the manipulation lab.



Find me the closest meeting room to the kitchen.



Either Filipe or Tobi has my headphones. Locate them.

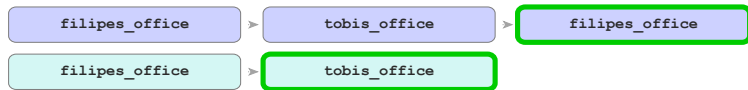
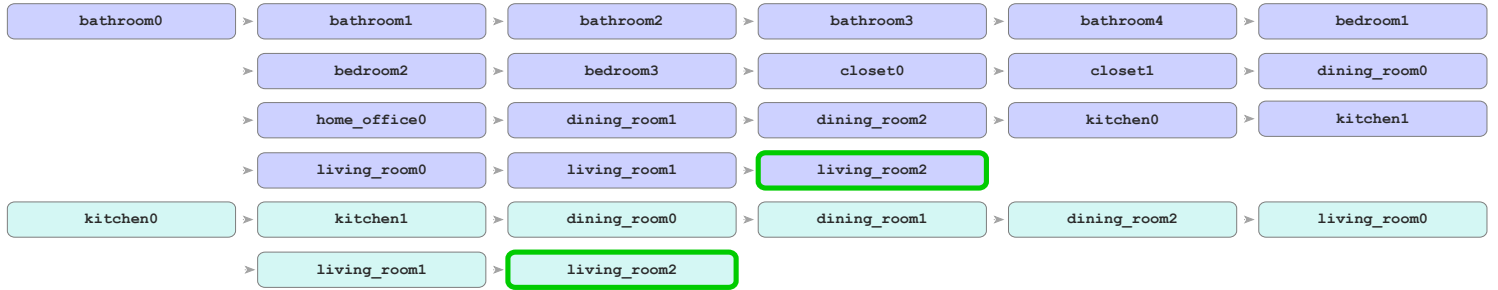
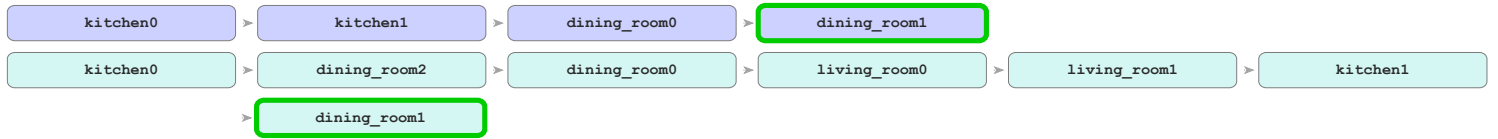


TABLE XIII: **Complex Search Office Environment Evaluation.** Sequence of Explored Nodes for Complex Search Office Environment Instructions.

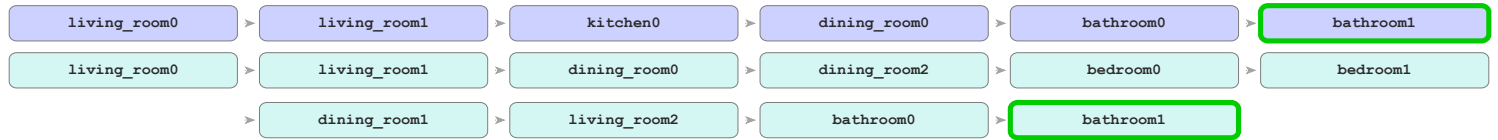
Find me a FooBar.



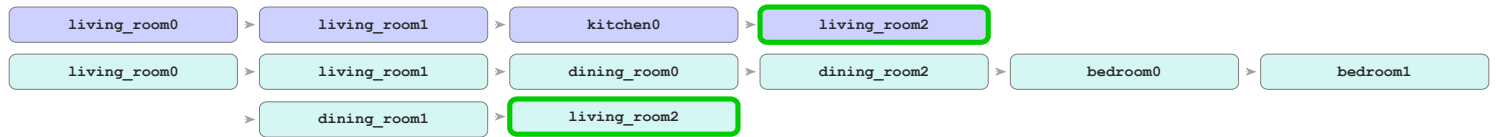
Find me a bottle of wine.



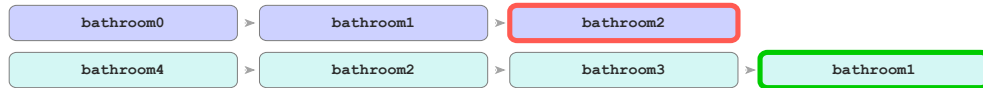
Find me a plant with thorns.



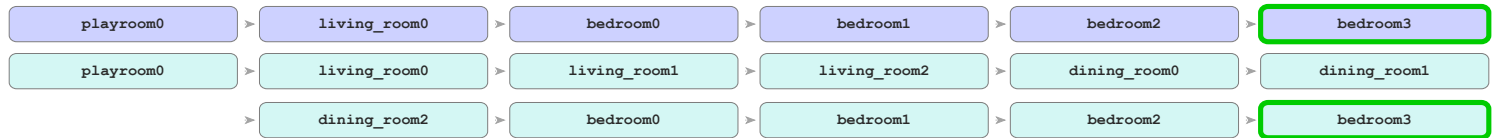
Find me a plant that needs watering.



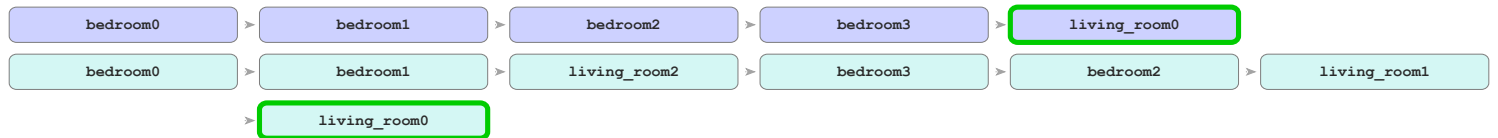
Find me a bathroom with no toilet.



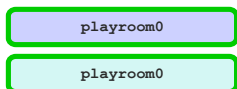
The baby dropped their rattle in one of the rooms. Locate it.



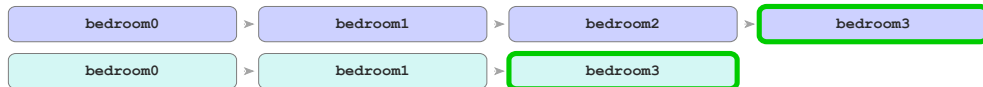
I left my suitcase either in the bedroom or the living room. Which room is it in.



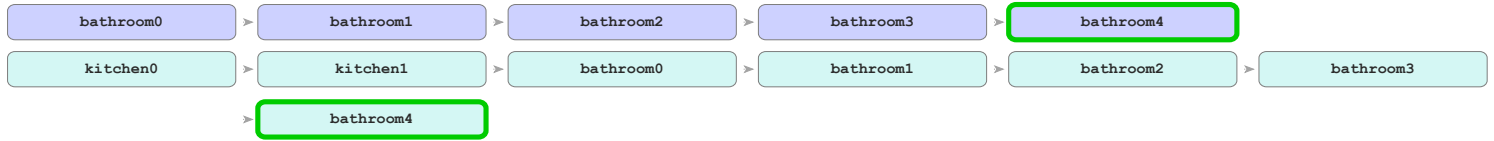
Find the room with a ball in it.



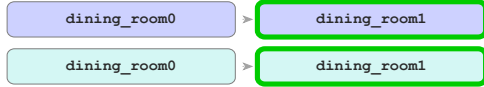
I forgot my book on a bed. Locate it.



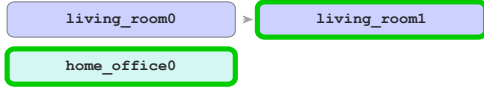
Find an empty vase that was left next to a sink.



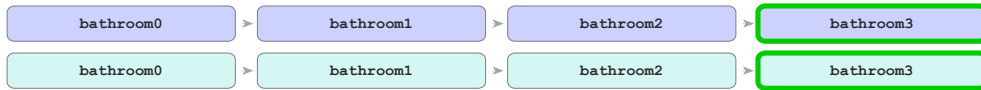
Locate the dining room which has a table, chair and a baby monitor.



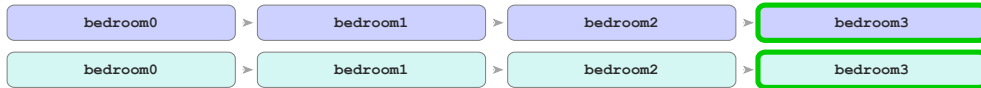
Locate a chair that is not in any dining room.



I need to shave. Which room has both a razor and shaving cream.



Find me 2 bedrooms with pillows in them.



Find me 2 bedrooms without pillows in them.

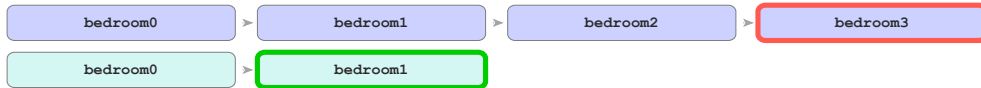
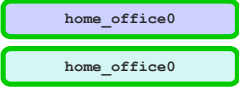


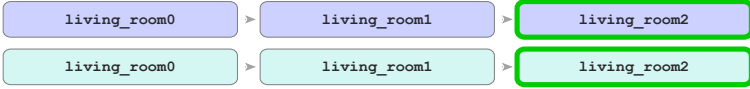
TABLE XV: **Simple Search Home Environment Evaluation.** Sequence of Explored Nodes for Simple Search Home Environment Instructions.



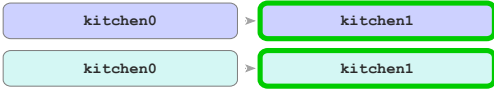
I need something to access ChatGPT. Where should I go?.



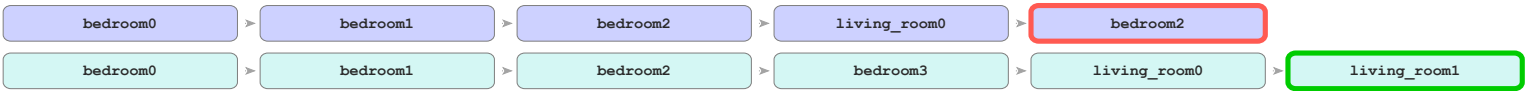
Find the livingroom that contains the most electronic devices.



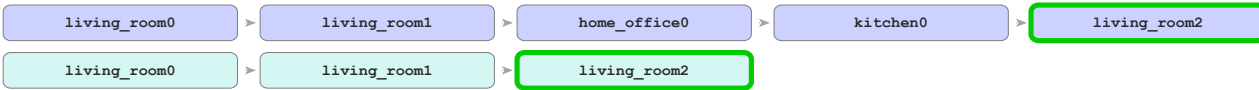
Find me something to eat with alot of potassium.



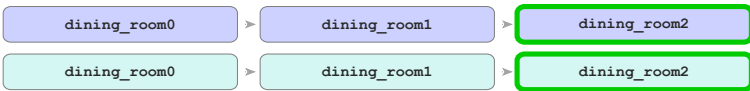
I left a sock in a bedroom and in one of the livingrooms. Locate them. They should match.



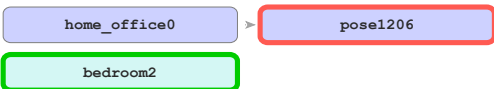
Find the potted plant that is most likely a cactus.



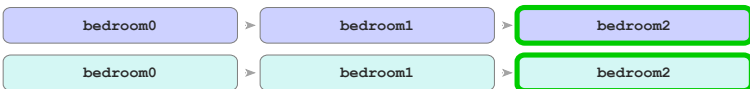
Find the dining room with exactly 5 chairs.



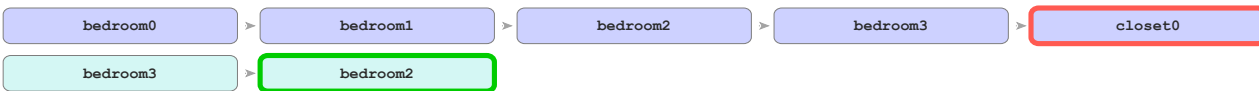
Find me the bedroom closest to the home office.



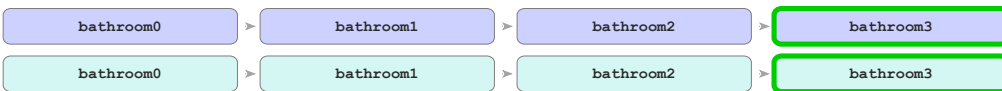
Find me the bedroom with an unusual amount of bowls.



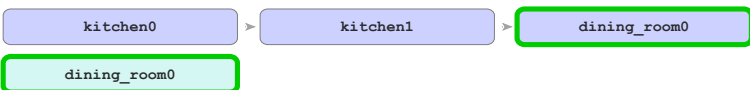
Which bedroom is empty.



Which bathroom has the most potted plants.

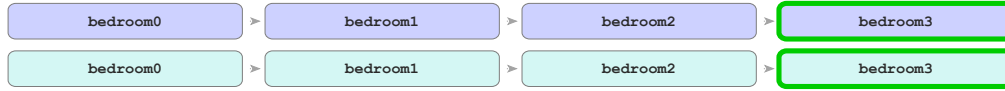


The kitchen is flooded. Find somewhere I can heat up my food.

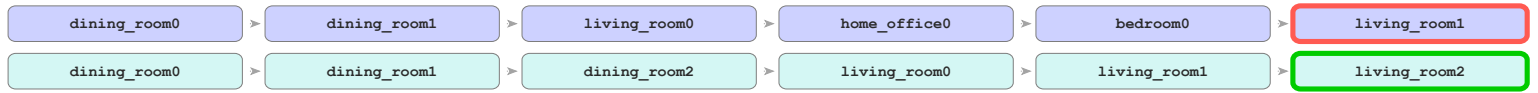




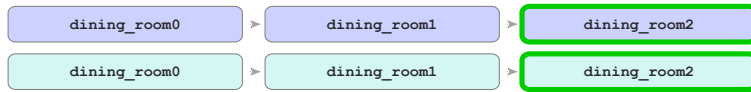
Find me the room which most likley belongs to a child.



15 guests are arriving. Locate enough chairs to seat them.



A vegetarian dinner was prepared in one of the dining rooms. Locate it.



My tie is in one of the closests. Locate it.

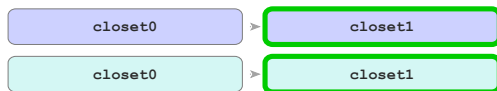
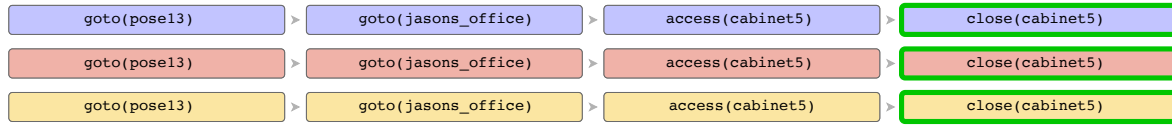


TABLE XVII: **Complex Search Home Environment Evaluation.** Sequence of Explored Nodes for Complex Search Home Environment Instructions.

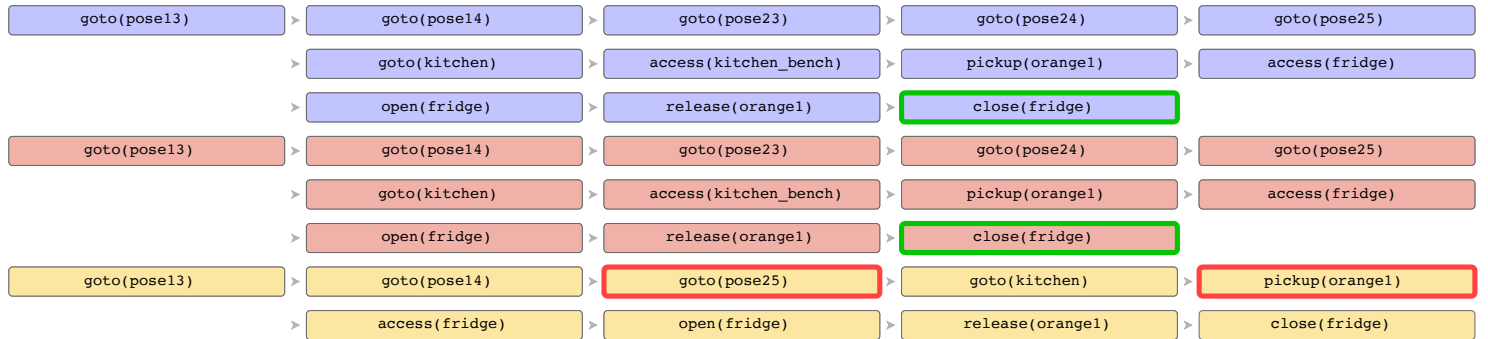
## V. CAUSAL PLANNING EVALUATION RESULTS

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

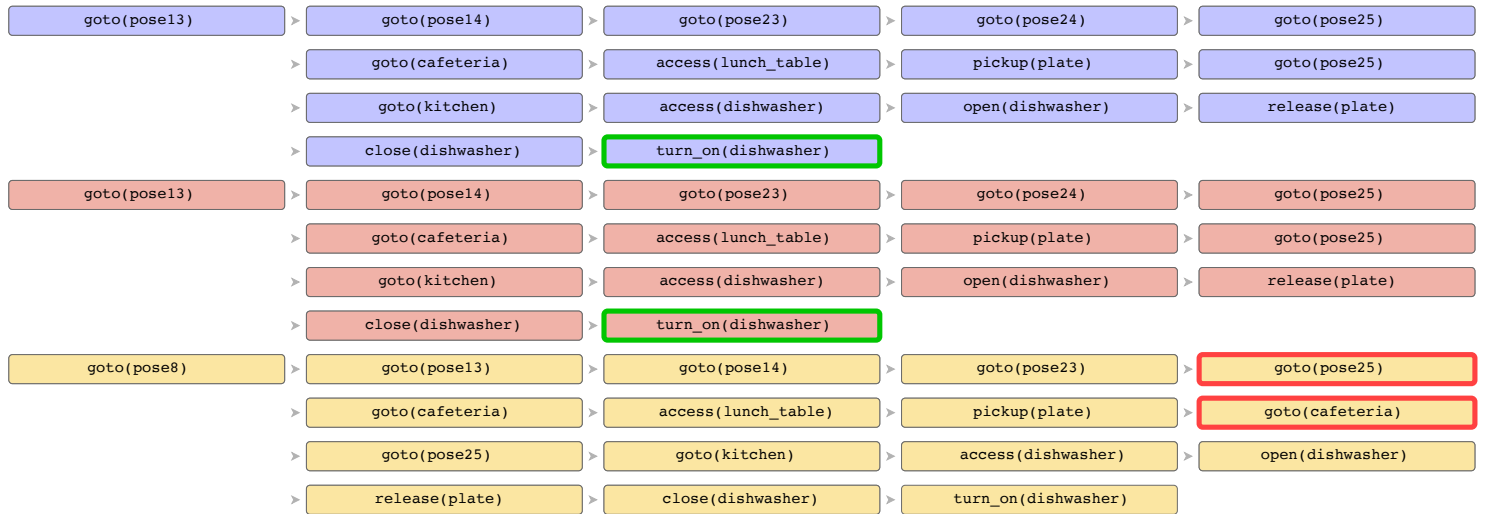
Close Jason's cabinet.



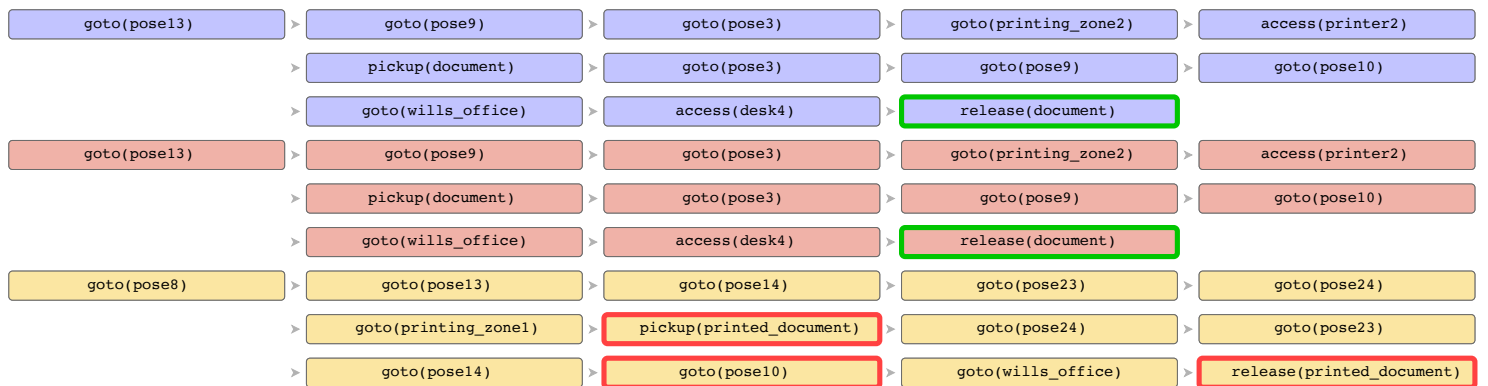
Refrigerate the orange left on the kitchen bench.



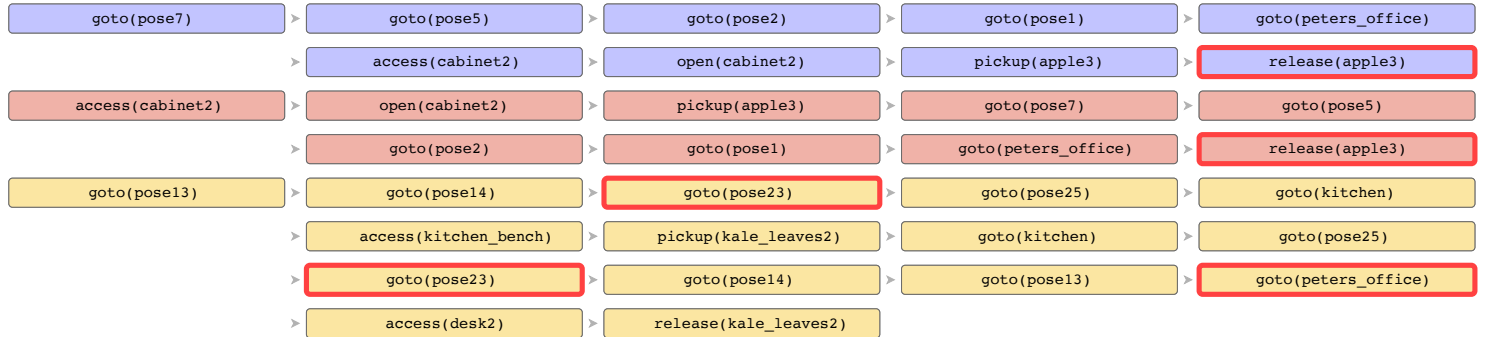
Take care of the dirty plate in the lunchroom.



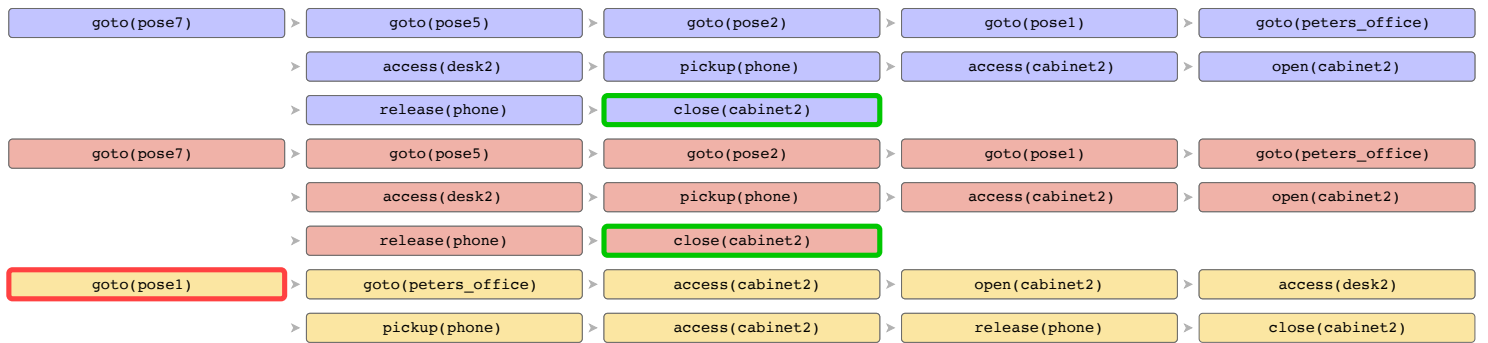
Place the printed document on Will's desk.



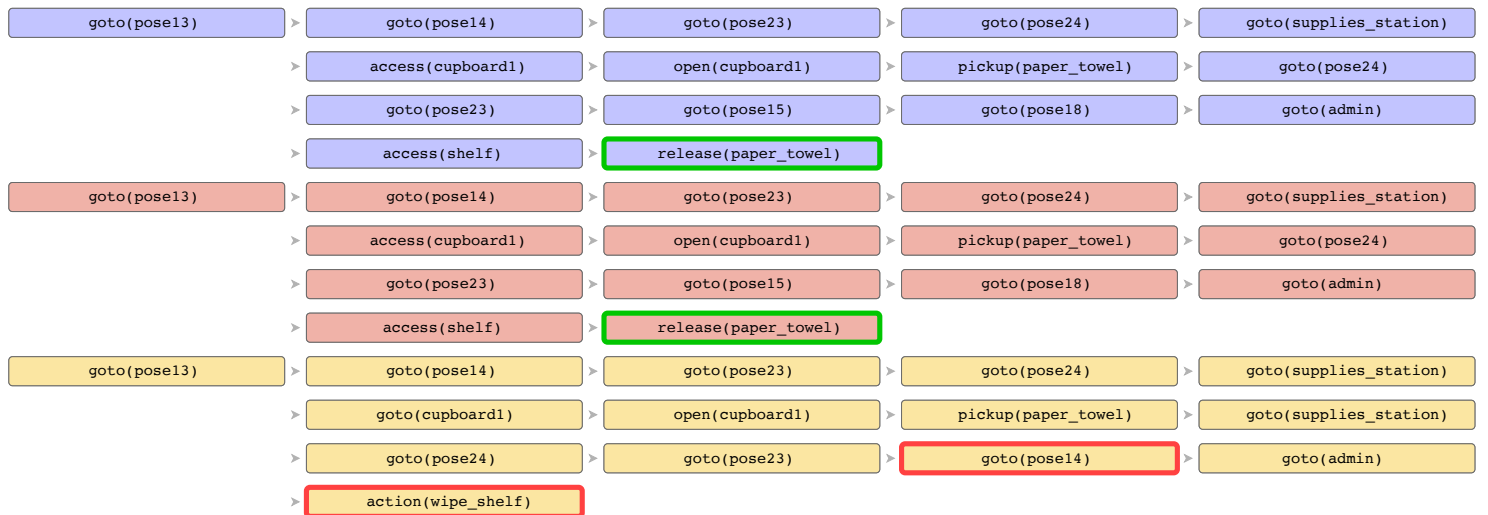
Peter is working hard at his desk. Get him a healthy snack.



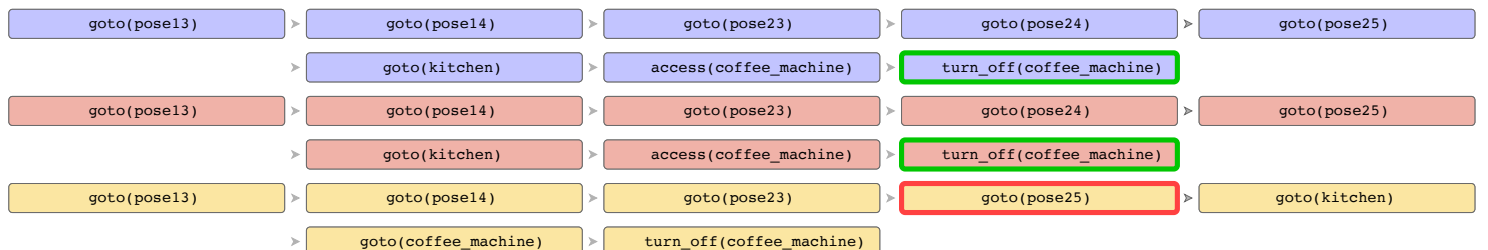
Hide one of Peter's valuable belongings.



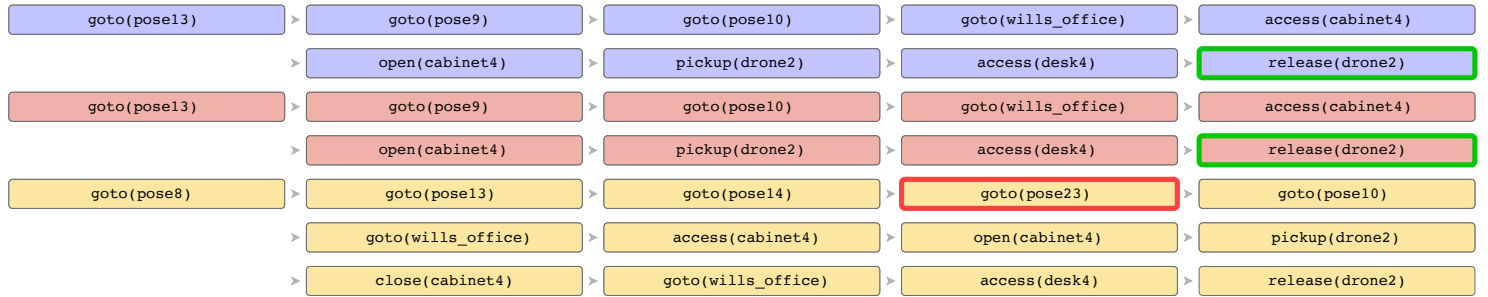
Wipe the dusty admin shelf.



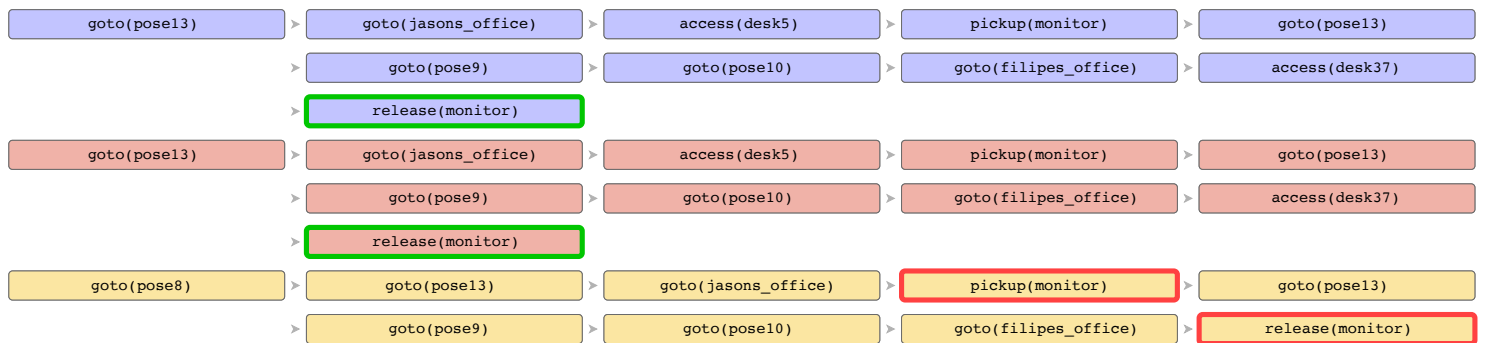
There is coffee dripping on the floor. Stop it.



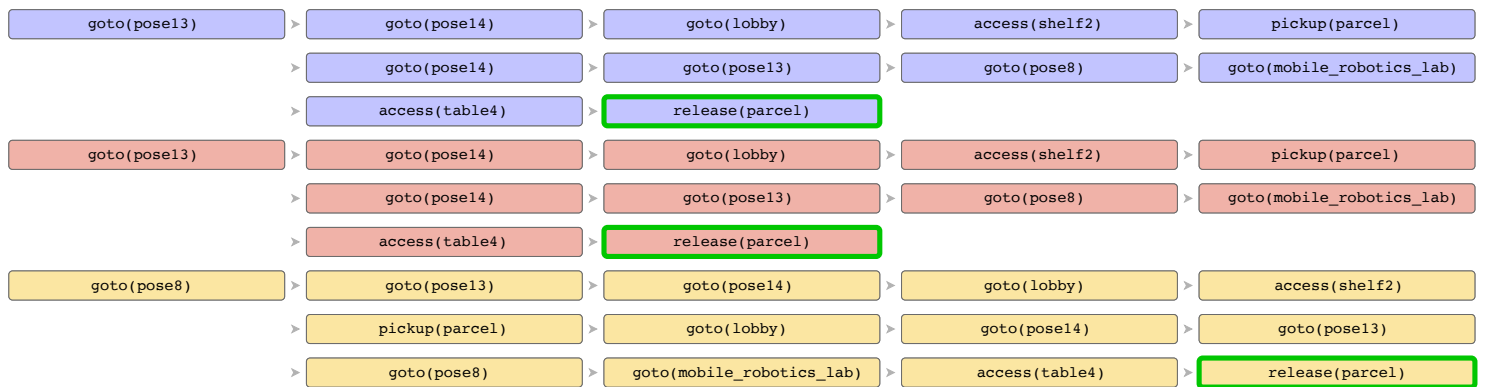
Place Will's drone on his desk.



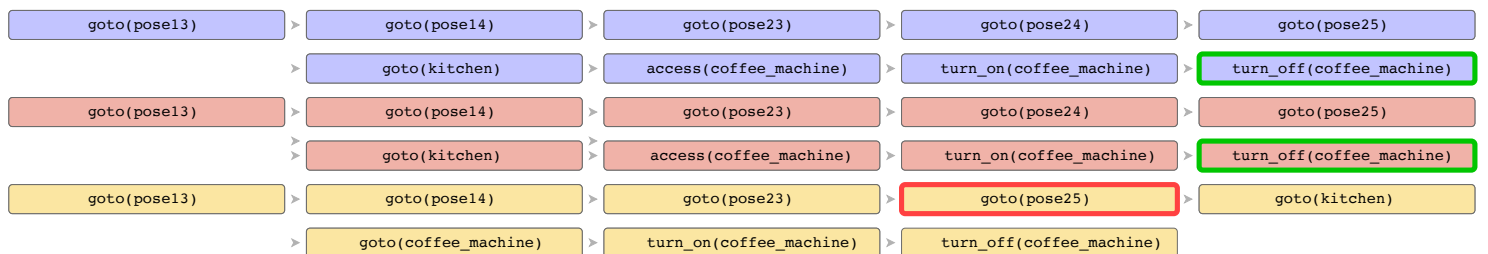
Move the monitor from Jason's office to Filipe's.



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

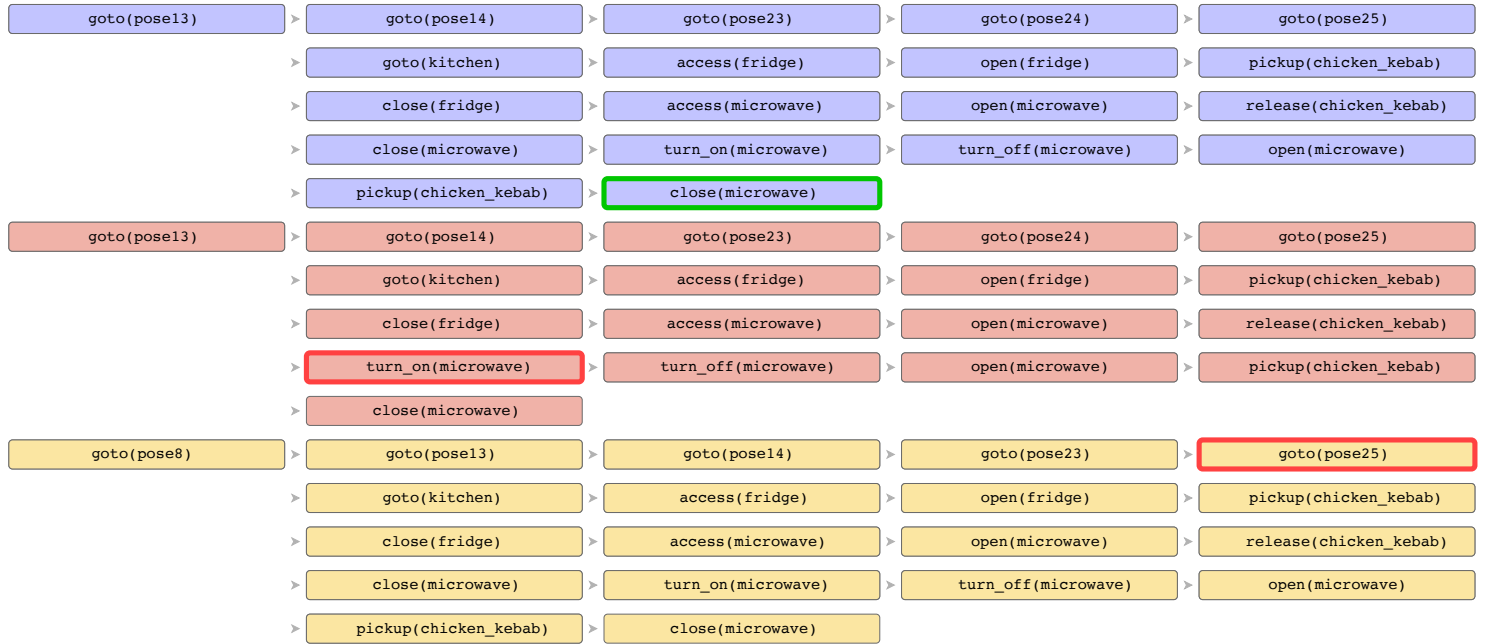


Check if the coffee machine is working.

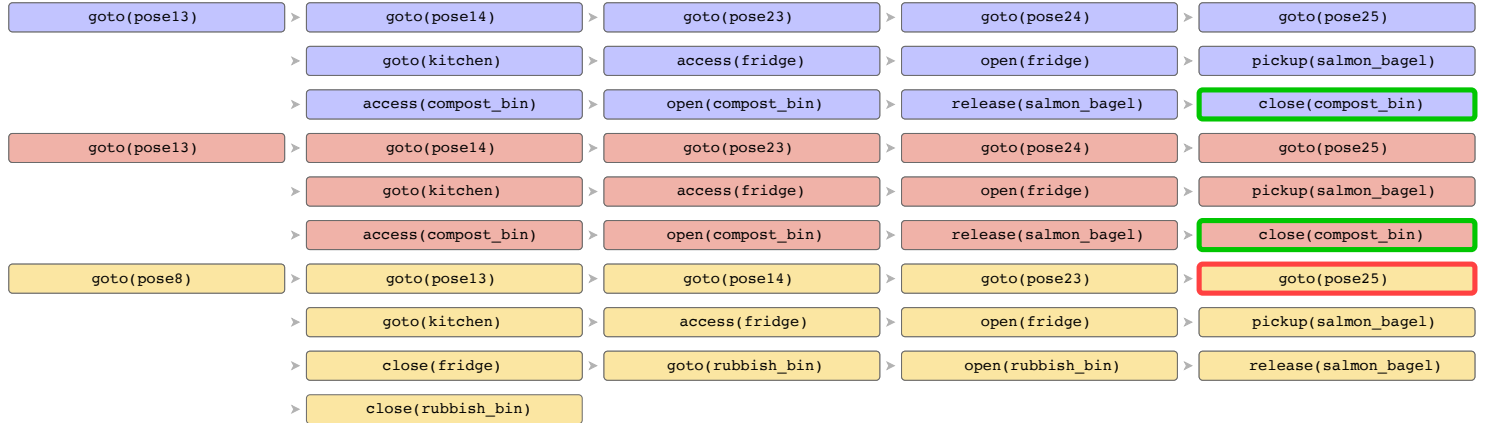




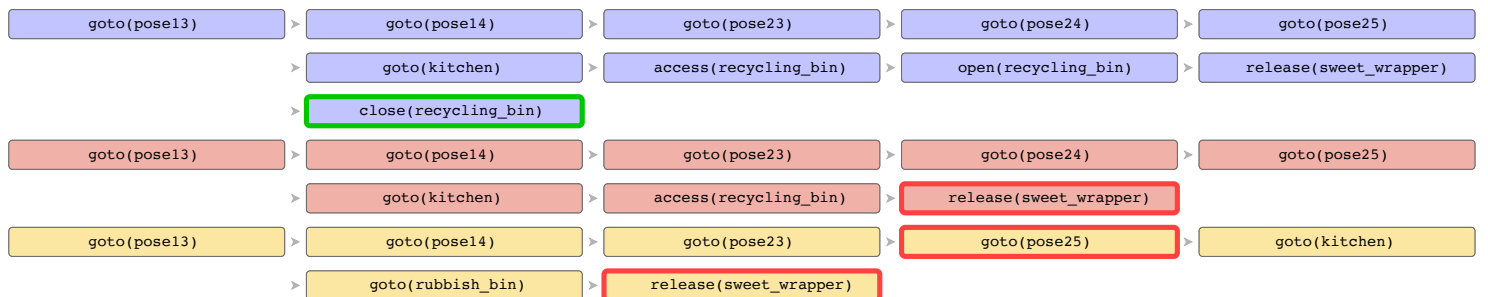
Heat up the chicken kebab.



Something is smelling in the kitchen. Dispose of it.

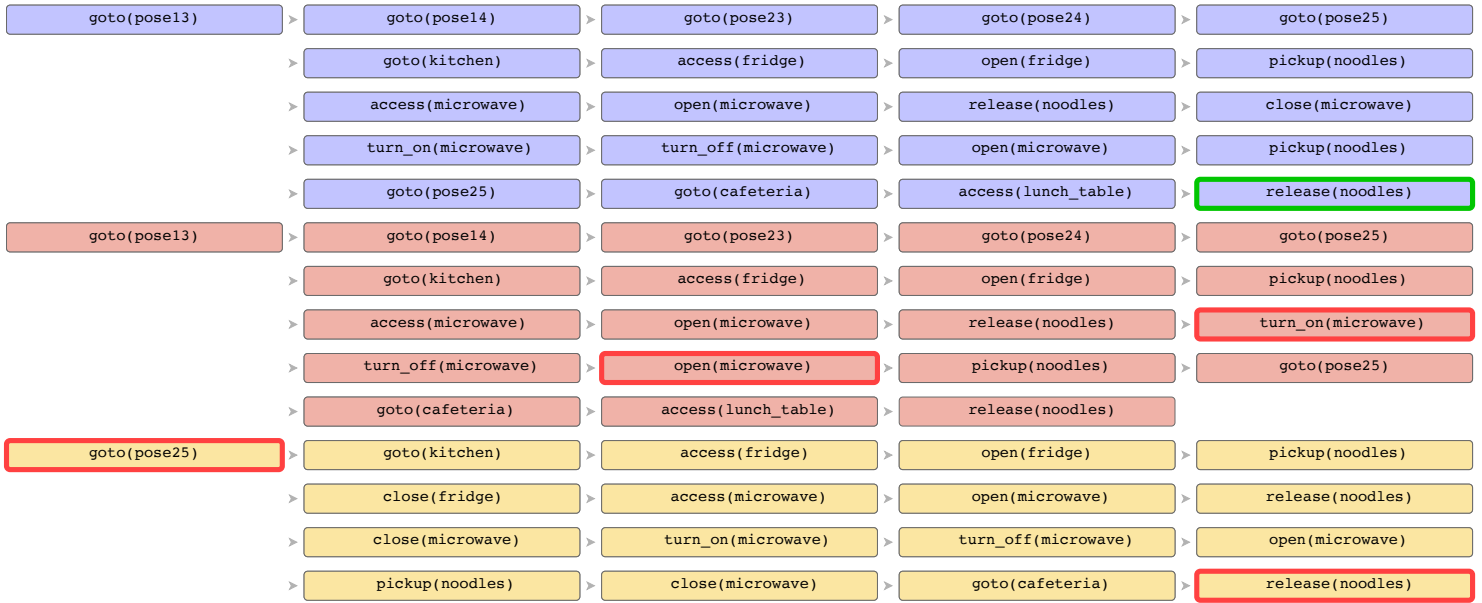


Throw what the agent is holding in the bin.

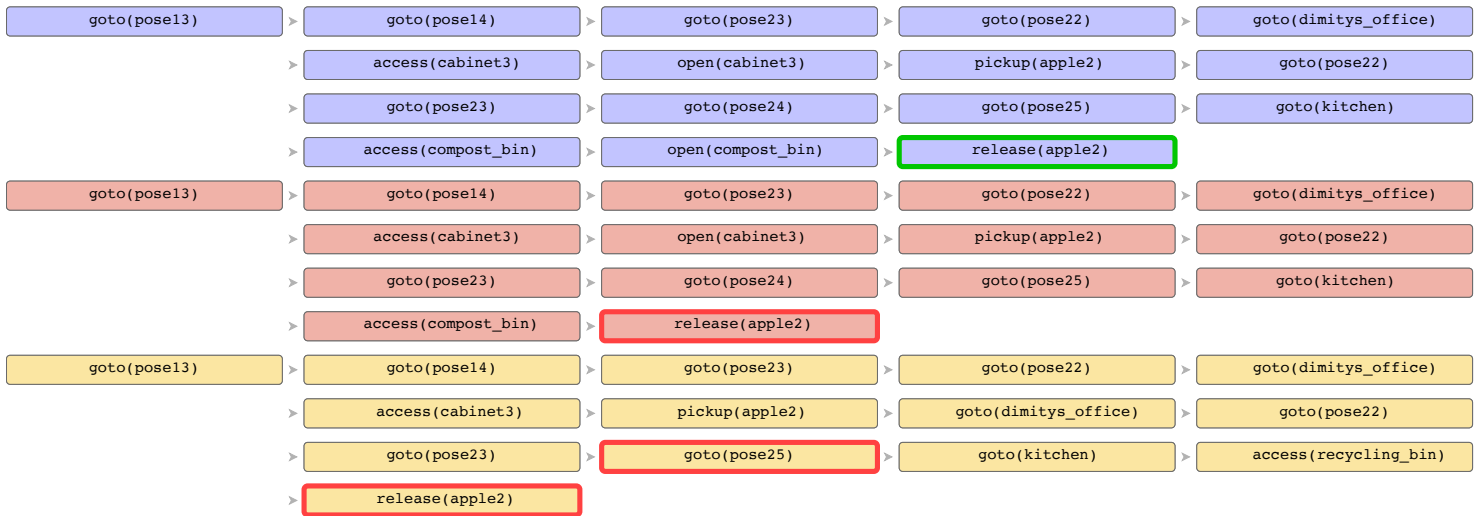




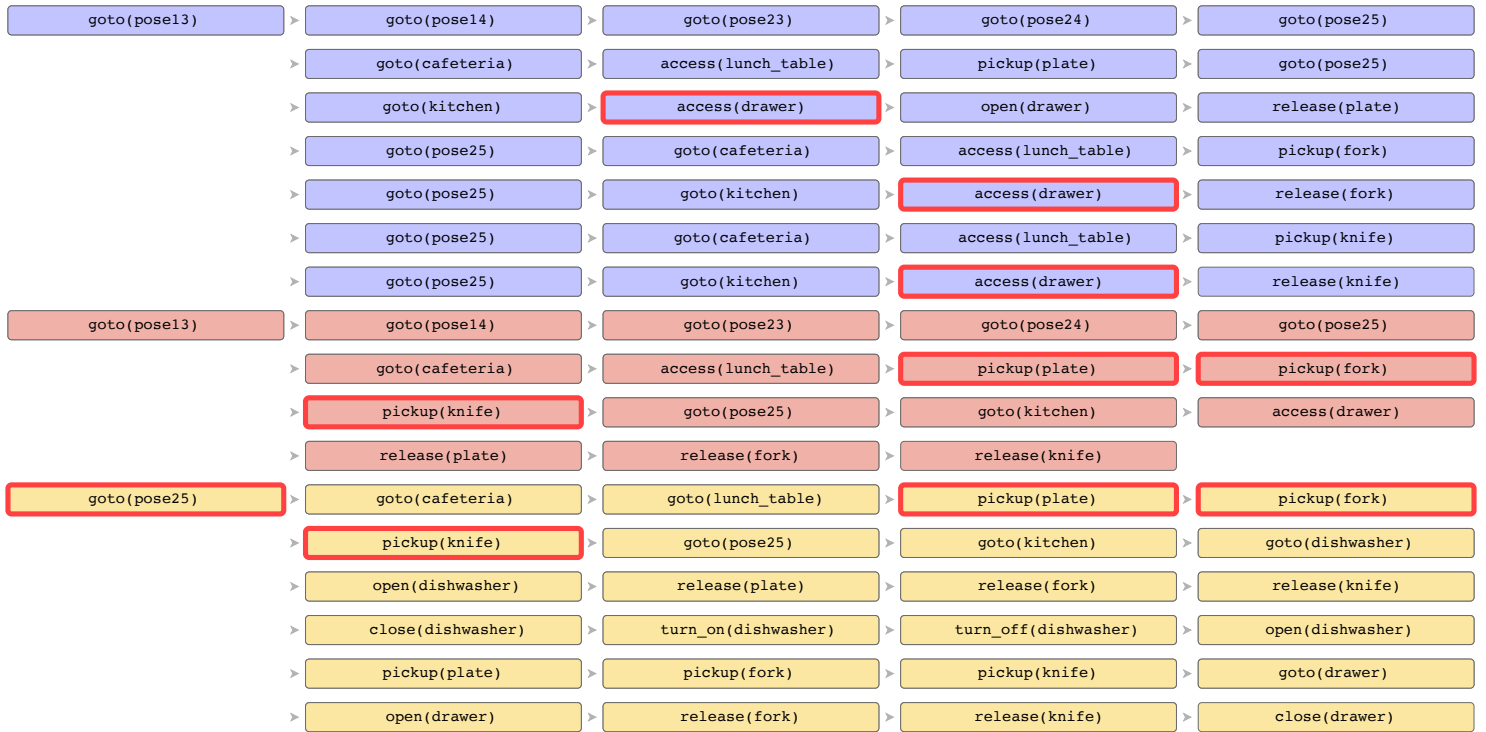
Heat up the noodles in the fridge, and place it somewhere where I can enjoy it.



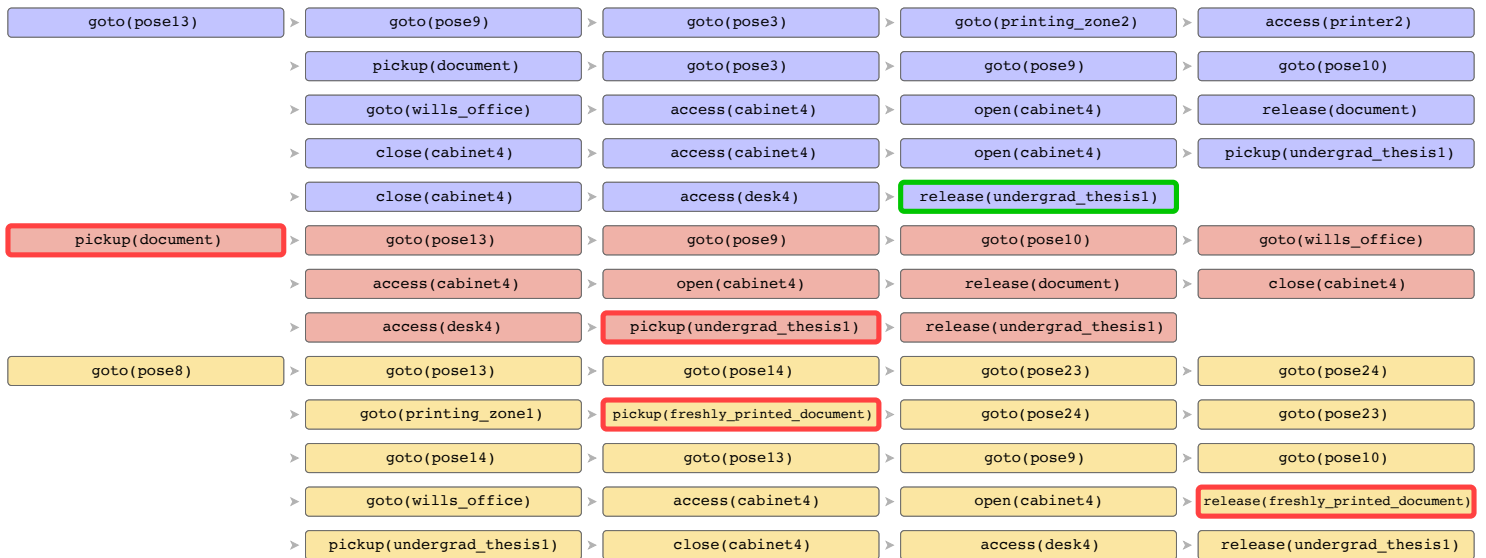
Throw the rotting fruit in Dimity's office in the correct bin.



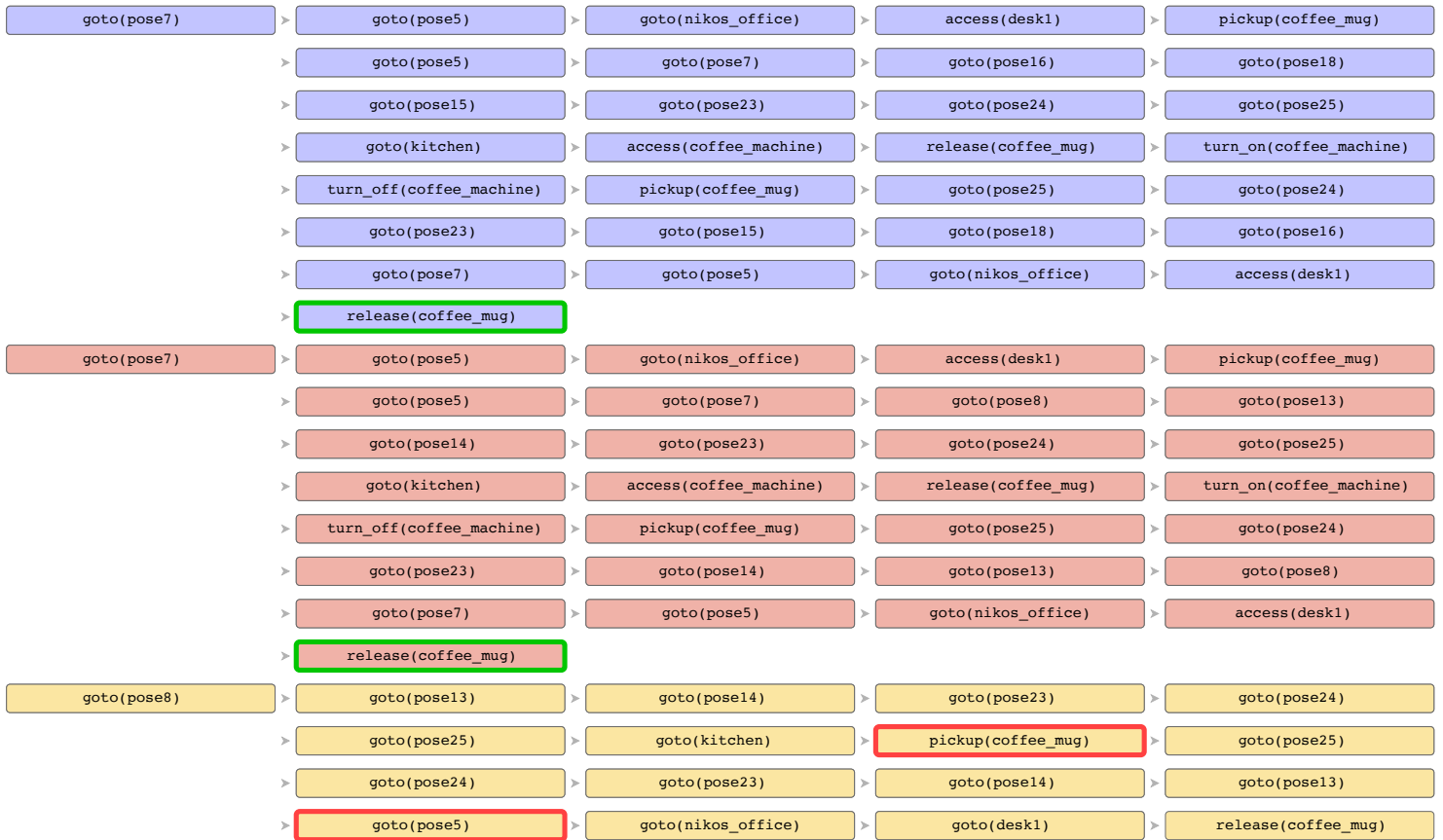
Wash all the dishes on the lunch table. Once finished, place all the clean cutlery in the drawer.



Safely file away the freshly printed document in Will's office then place the undergraduate thesis on his desk.



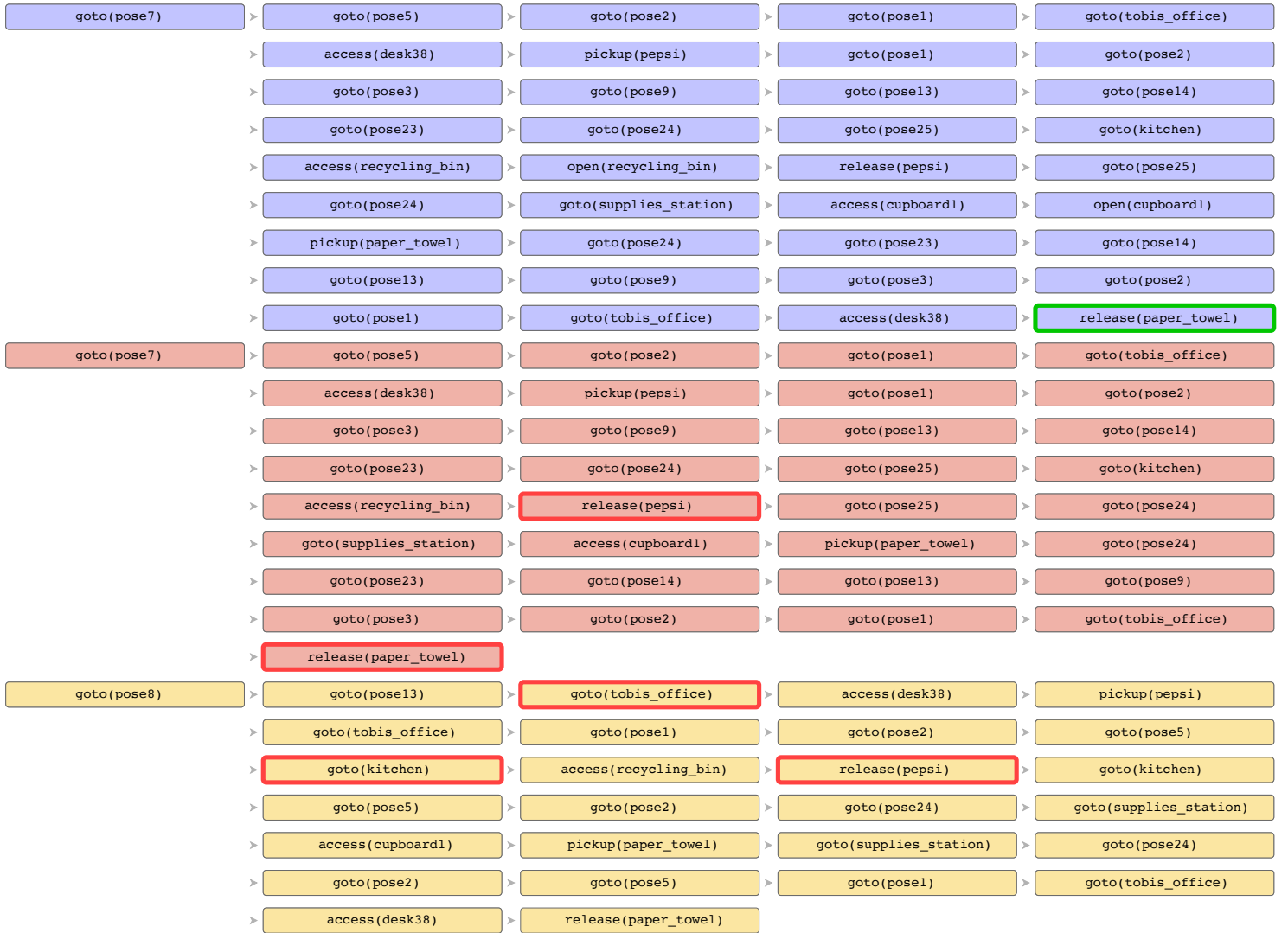
Make Niko a coffee and place the mug on his desk.



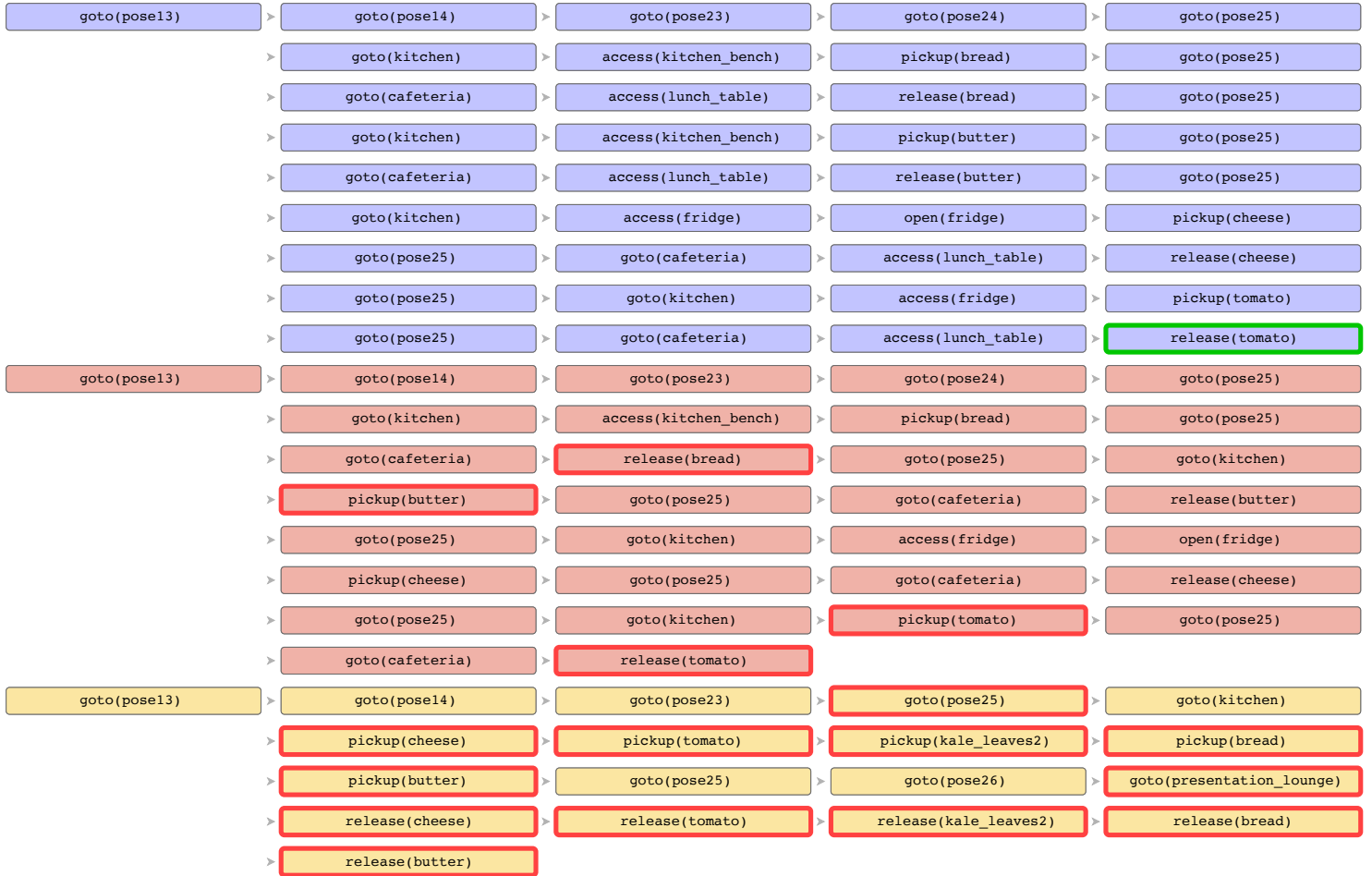
Someone has thrown items in the wrong bins. Correct this.



Tobi spilt soda on his desk. Throw away the can and take him something to clean with.



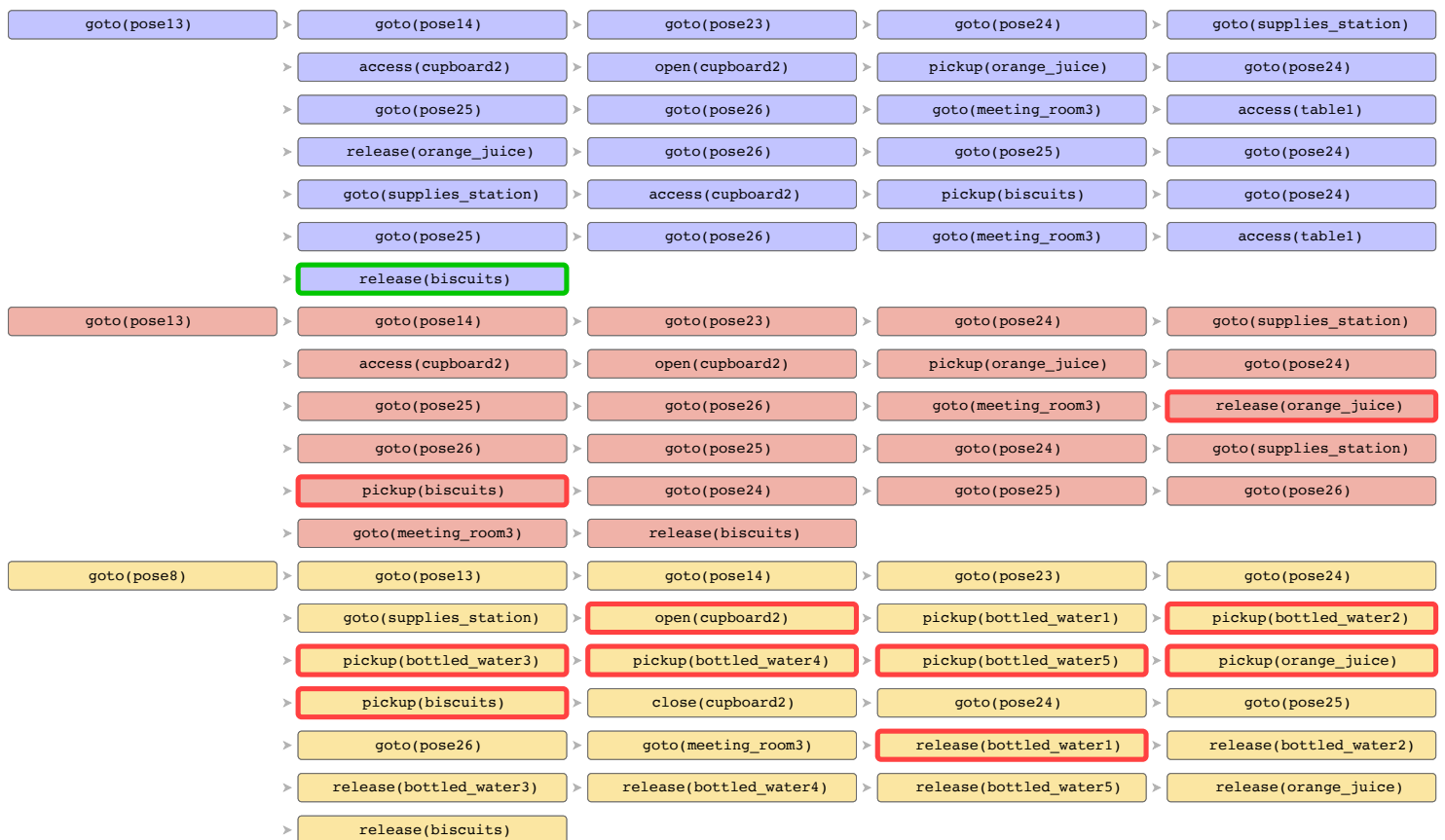
I want to make a sandwich. Place all the ingredients on the lunch table.



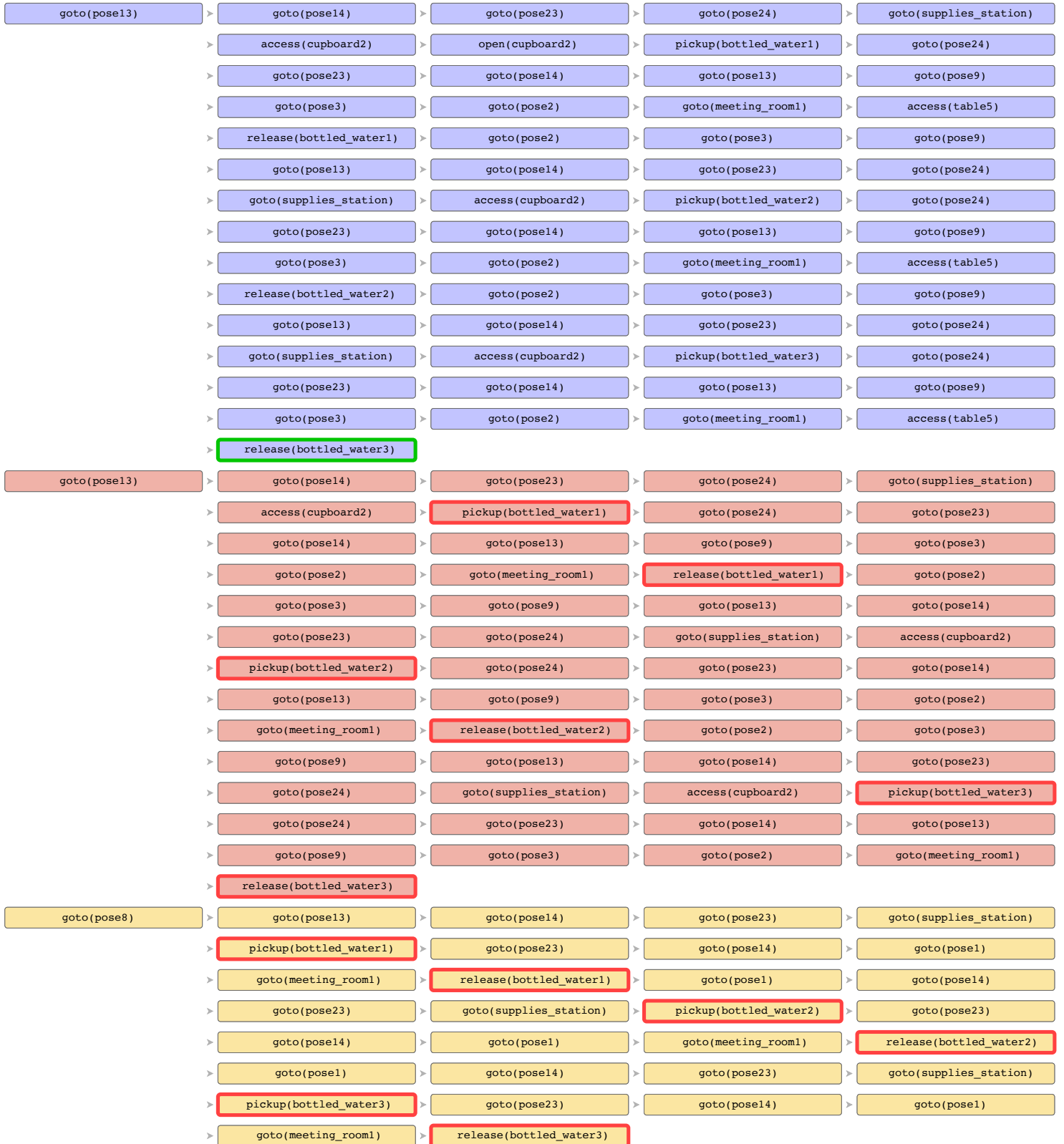
Empty the dishwasher. Place all items in their correct locations



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.

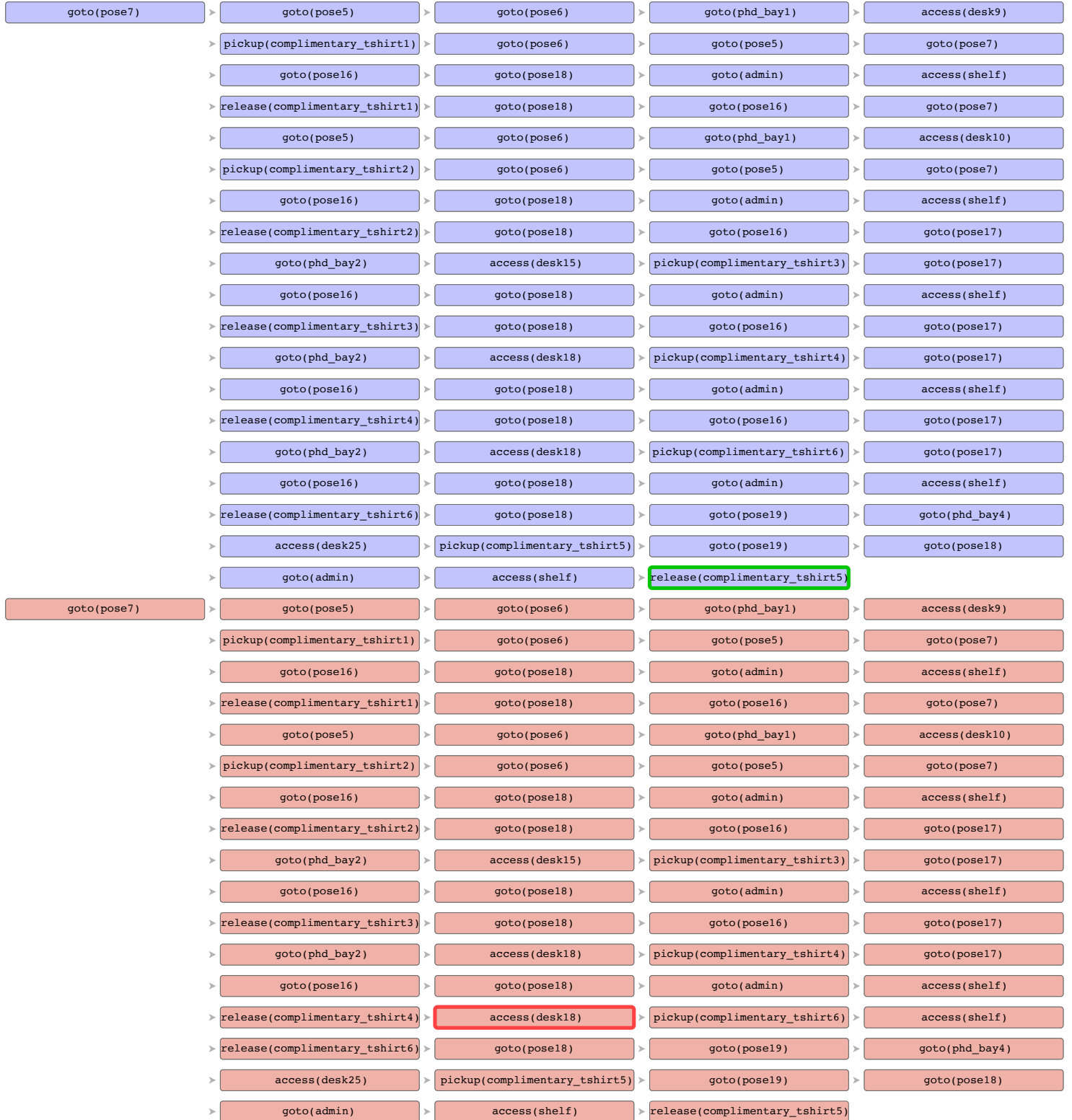


Serve bottled water to the attendees who are seated in meeting room 1. Each attendee can only receive a single bottle of water.

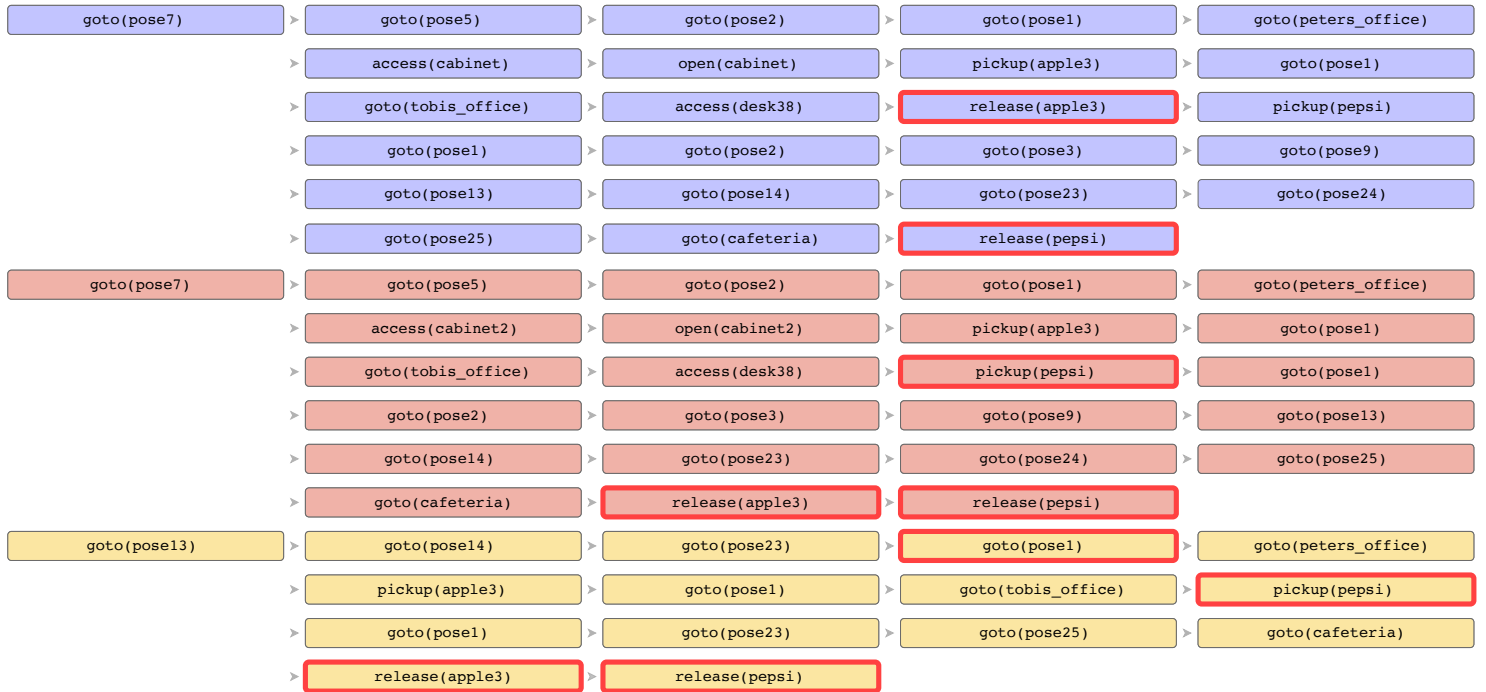




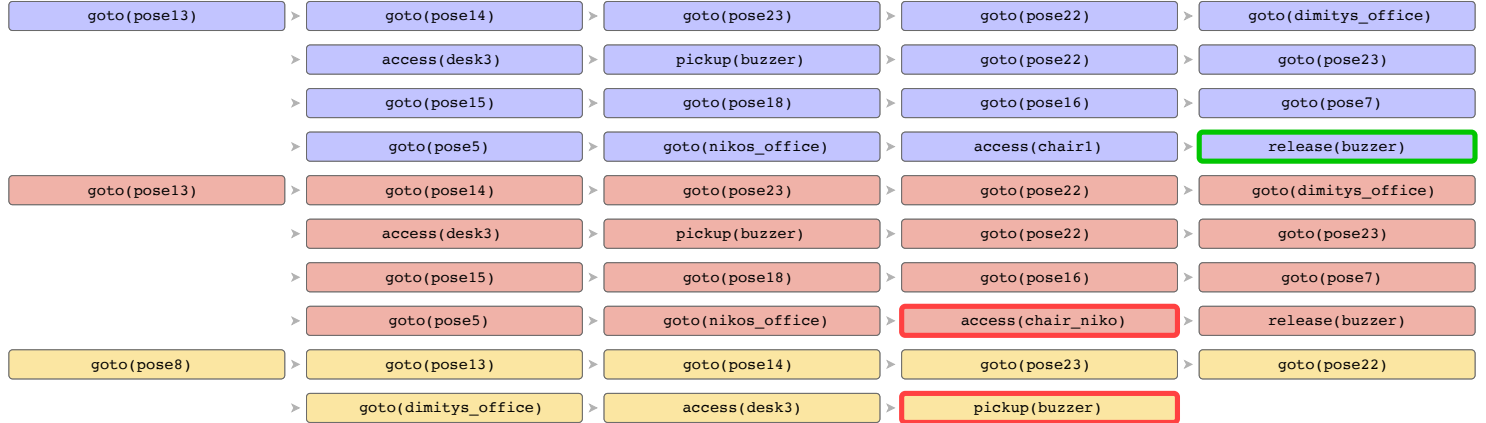
Locate all 6 complimentary t-shirts given to the PhD students and place them on the shelf in admin.



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.





## VII. 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 re-planning 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 VII and VIII.>

**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`.

## VIII. 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\_rooms, 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

## IX. EXAMPLE INPUT-OUTPUT INTERACTION BETWEEN LLM AND SCENE GRAPH ENVIRONMENT DURING ITERATIVE RE-PLANNING

**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:** "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