Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J. 8

Yue Wu, So Yeon Min, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Yuanzhi Li, Tom Mitchell, and Shrimai Prabhumoye. Plan, eliminate, and track – language models are good teachers for embodied agents, 2023a. 9

Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models, 2023b. 9

Danfei Xu, Roberto Martín-Martín, De-An Huang, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Regression planning networks. *arXiv: Artificial Intelligence,arXiv: Artificial Intelligence*, Sep 2019. 1

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities, 2023. 9

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. 8, 9

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. 9

Andy Zeng, Maria Attarian, brian ichter, Krzysztof Marcin Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael S Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=G2Q2Mh3avow. 9

Zirui Zhao, Wee Sun Lee, and David Hsu. Large language models as commonsense knowledge for large-scale task planning, 2023. 9

A Environment

A.1 Action Space

The action types in the Virtual Home Environment are listed as follows:

1. Sleep
2. StandUp
3. WakeUp
4. Walk
5. Find
6. Grab
7. Wash
8. Wipe
9. Pull
10. Push
11. Pour
12. TurnTo
13. PointAt
14. Watch
15. Touch
16. Open
17. Close
18. Run
19. Sit
20. Read
21. PutOn
22. Drop
23. Lie
24. SwitchOn
25. SwitchOff
26. Drink
27. PutIn
28. PutBack

A.2 Partial Observation

We implemented partial observation based on Li et al. (2022a). The official definition of partial observation is that when the agent is situated in a room, its observation consists of all the objects in the room that are not located inside enclosed objects. For instance, if an apple is placed inside a closed refrigerator, it will not appear in the observation of the agent.

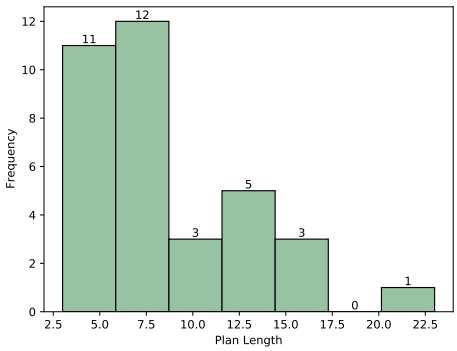

Figure 6: Distribution of plan length.

A.3 Observation Representation

In the VirtualHome Environment (Puig et al., 2018), observations primarily consist of two components: *object states* and *inter-object relationships*. The *object states* describes the state in which an object exists. For example, the state of television can either be "on" or "off", i.e., "*On(TV)*" or "*Off(TV)*". The *inter-object relationships* are represented using predicates to express the relationships between objects. For example, when a character is in close proximity to a television, there may be a predicate such as: "*Close(character, TV)*". We convert the observations from VH into English sentences using a rule-based approach. For example, the predicate "*Close(character, TV)*" is transformed into "character is close to TV", and the predicate "*"Off(TV)*" is transformed into "TV is off".

A.4 Basic Statistics

In the gold plan annotated in the dataset, the plan with the longest length consists of 23 actions, while the average length is 8.13. The frequency distribution histogram regarding the length of the plan is shown in Figure 6. Furthermore, we have also computed the frequency histograms for actions and objects, which are depicted in Figure 7 and Figure 8.

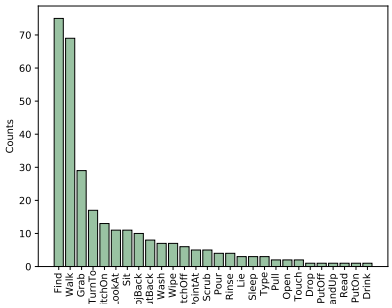

Figure 7: Histogram of action frequencies. Certain actions exhibit a significantly higher frequency than others, such as **Find** and **Walk**.

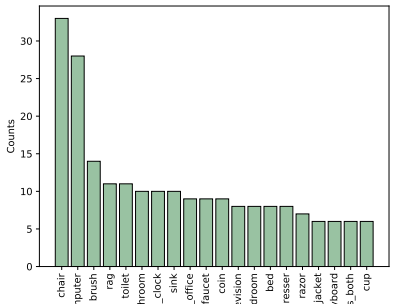

Figure 8: Histogram of object frequencies. Only objects with a frequency in the top 20 are included for better visualization.

B More Implementation Details

B.1 Hyper-parameter Search

For both Grounded Deciding and ITERATIVE-PLANNER, we conducted a grid search over the sampling parameters of OpenAI APIs. The search range for temperature was set from 0.1 to 1.0 in increments of 0.1, while the search range for top-p was set to 0.85, 0.9, 0.95, and 1.0.

In the case of Grounded Deciding, the optimal hyperparameter combination was found to be a temperature of 0.7 and topp of 1.0. As for ITERATIVE-PLANNER, the optimal hyperparameter combination was a temperature of 0 and topp of 1.0.

B.2 Baseline Models

ZERO-SHOT PLANNER (Huang et al., 2022a) propose to translate each unstructured action generated by LLM into an admissible action via another pre-trained masked language model. The translated action is then appended to the prompt used for generating the remaining steps. We utilize the official implementation provided by Huang et al. (2022a), which employs a dynamically retrieved plan as an exemplar in the prompt. Moreover, concerning the hyper-parameters, we configure the maximum number of steps as 20 and set the early stopping threshold to 0.5 to achieve optimal performance.

PROGPROMPT (Singh et al., 2022) proposes a programming language-inspired prompt with an assert statement. These assert statements provide a mechanism for incorporating environment feedback into the plan generation process, ensuring that preconditions for each action are met;

C More Experimental Results

C.1 Results by Plan Length

	EXEC. ↑	SR ↑	GCR ↑	NO.EC ↓
$0 < \|a\| \leq 5$	100.00 ± 0.00	64.72±5.20	77.12±4.13	0.30±0.15
$5 < \|a\| \leq 10$	83.59±4.03	35.10±5.95	52.25±3.33	2.47±0.35
$10 < \|a\| \leq 15$	88.09±8.48	26.98±4.89	55.98±7.00	3.20±1.12
$15 < \|a\|$	66.67±0.00	22.22±0.00	36.11±0.00	4.09±0.21

Table 4: The performance of TREE-PLANNER$_{N=50}$ across different plan sequence lengths.

Table 4 above presents the performance of TREE-PLANNER$_{N=50}$, categorized by different plan sequence lengths. In general, as the plan sequence length increases, the performance of the model tends to decrease. Specifically, for tasks with plan lengths smaller than 5, the SR can reach 64.72% and even 100% for EXEC.. However, for tasks with plan lengths larger than 15, the SR decreases to 22.22% (-42.5%), and EXEC. decreases to 66.67% (-33.33%). Furthermore, as the plan length increases, the number of corrections in the model also increases. This is due to the higher likelihood of errors accumulating in longer sequential task planning, thus necessitating a greater need for error correction. The above experimental results provide a more comprehensive analysis of the performance of our approach, emphasizing potential directions for future enhancements.

C.2 Results by Scene

As is shown in Table 5, TREE-PLANNER exhibits consistent performance across various scenes, thereby further illustrating its robustness.

C.3 Diversity of Sampled Plans

As shown in Figure 9, the number of different plans varies approximately linearly with the change in the sampling n. Therefore, when conducting plan sampling, the issue of homogenization due to the sampled plans will not arise.

	EXEC. ↑	SR ↑	GCR ↑	NO.EC ↓
ENV-1	92.42±2.14	45.45±3.71	64.72±3.25	1.74±0.44
ENV-2	91.30±4.10	36.75±4.10	52.43±7.13	2.33±0.47
ENV-3	85.18±2.62	37.04±2.62	50.80±3.19	1.83±0.39
ENV-4	88.89±3.14	48.89±3.14	66.74±1.72	2.35±0.58

Table 5: The performance of TREE-PLANNER$_{N=50}$ across different scenes.

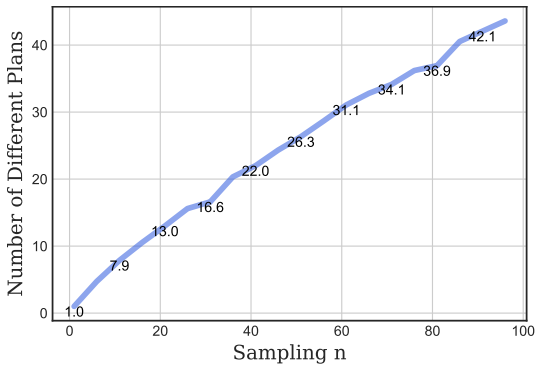

Figure 9: The diversity of plans generated. The x-axis represents the number of programs sampled, while the y-axis represents the average number of different plans generated.

D DETAILS ON TOKEN EFFICIENCY

The detailed derivation of the boundary conditions for N is as follows:

$$
\begin{aligned}
&\textit{to prove} \quad \mu_{ours} < \mu_{sbs} \\
\Rightarrow \quad & \rho_{ps} + MN|a| + M \cdot (\rho_{gd} + N) < M \cdot (\rho_{ps} + \rho_{gd} + |a|) \\
\Rightarrow \quad & MN|a| + M \cdot N < (M-1) \cdot \rho_{ps} + M \cdot |a| \\
\Rightarrow \quad & M \cdot (|a|+1) \cdot N < (M-1) \cdot \rho_{ps} + M \cdot |a| \\
\Rightarrow \quad & N < \frac{1-1/M}{1+1/|a|} \cdot \frac{\rho_{ps}}{|a|} + \frac{|a|}{|a|+1}
\end{aligned}
$$

E QUALITATIVE ANALYSIS ON ERRORS

E.1 DECIDING ERROR

Task: *Hang up jacket*

Action Tree: See Figure 10

Explanation: As depicted in Figure 10, the model made an incorrect decision during the second step. This is due to the failure of TREE-PLANNER to comprehend the sequential order of searching for a jacket and searching for a hanger.

E.2 ENVIRONMENT MISUNDERSTANDING

Task: *Clean toilet*

Example Plan:

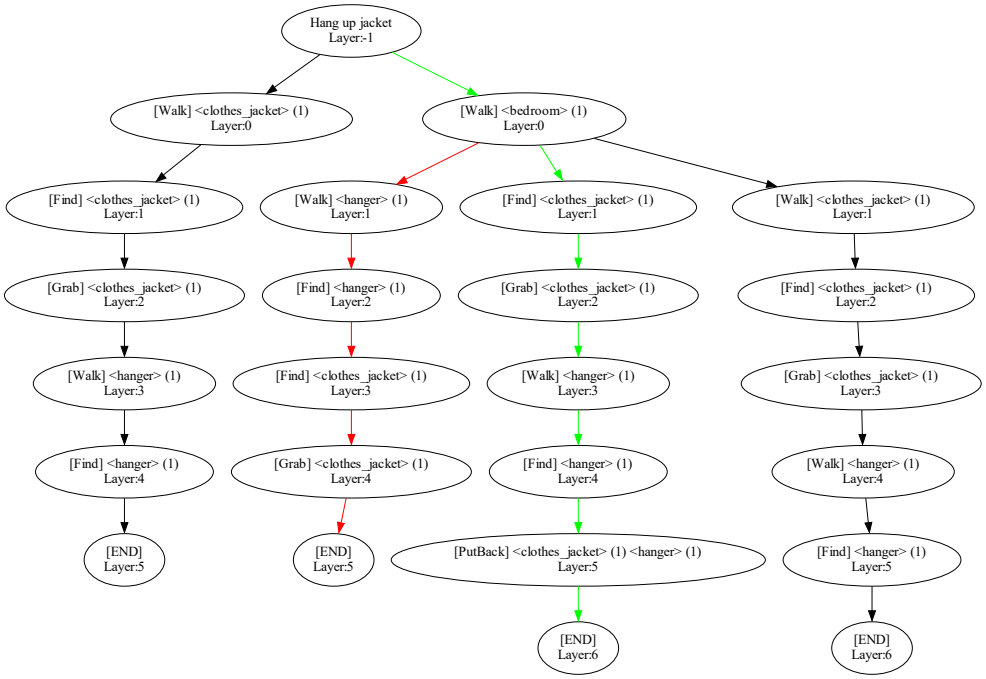

Figure 10: Visualization of the action tree for *Hang up jacket*. The red path is the plan chosen by LLM, while the green path is the correct plan.

1. **[Walk]** <bathroom>(1)
2. **[Walk]** <toilet>(1)
3. **[Pull]** <toilet>(1)
4. **[Wash]** <toilet>(1)
5. **[Wipe]** <toilet>(1)
6. **[Push]** <toilet>(1)
7. **[Wash]** <toilet>(1)
8. **[Find]** <mop_bucket>(1)
9. **[Walk]** <detergent>(1)
10. **[Grab]** <detergent>(1)
11. **[Pour]** <detergent>(1) <mop_bucket>(1)

Explanation: The toilet is not movable, so the execution of step 3 would be failed.

E.3 Incomplete Plan

Task: *Put alarm clock in bedroom*

Example Plan:

1. **[Walk]** <bedroom>(1)
2. **[Walk]** <alarm_clock>(1)
3. **[Find]** <alarm_clock>(1)
4. **[Grab]** <alarm_clock>(1)
5. **[Find]** <dresser>(1)
6. **[Open]** <dresser>(1)
7. **[SwitchOn]** <alarm_clock>(1)

Explanation: After executing this plan, the alarm clock will be in the hand of the character. However, a step is missing: **[PutBack]** the <alarm_clock>(1) on the <dresser>(1) to place the alarm clock *in bedroom*.

E.4 ILLOGICAL ERROR

Task: *Shave*

Example Plan:

1. **[Walk]** <bathroom>(1)
2. **[Find]** <shaving_cream>(1)
3. **[Grab]** <shaving_cream>(1)
4. **[Walk]** <after_shave>(1)
5. **[Find]** <after_shave>(1)
6. **[Grab]** <after_shave>(1)
7. **[Walk]** <razor>(1)
8. **[Find]** <razor>(1)
9. **[Grab]** <razor>(1)
10. **[PutOn]** <shaving_cream>(1)
11. **[Wash]** <headset>(1)
12. **[PutOn]** <after_shave>(1)

Explanation: The character does not have a free hand when executing step 9. This is because, in steps 3 and 6, the character has already picked up two items.

E.5 FALSE NEGATIVE

Task: *Use computer*

Example Plan:

1. **[Walk]** <home_office>(1)
2. **[Walk]** <computer>(1)
3. **[Find]** <chair>(1)
4. **[Sit]** <chair>(1)
5. **[SwitchOn]** <computer>(1)
6. **[Type]** <keyboard>(1)

Explanation: The error information from the environment is as follows: "<keyboard >" cannot be found when executing "**[Type]** <keyboard >(1)". However, at this moment, the character is seated in front of the computer, indicating that the keyboard should be located near the character.

F PROMPTS

F.1 ITERATIVE-PLANNER

(Instruction) You need to act as a task planner who decomposes a high-level household task into several sub-tasks. The temporal relationship between subtask sequences must adhere to common-sense logic. Each sub-task can be one of the following form: 1. [action_name]; 2. [action_name] <object name 1>(object id 1). 3. [action_name] <object name 1>(object id 1) <object name 2>(object id 2). The number of arguments depends on the action type. The (object id) is used to tell the simulator that the actions should be done on the same object instance. For example a program as: [Walk] <glass>(1) [Grab] <glass>(1) Indicates that the agent should first walk to a glass and then grab that same glass. If you think your task has been successful, you can output [END], which is action type 1.

(Global Information) For action type 1, the available actions are: [Sleep], [StandUp], [WakeUp] For action type 2, the available actions are: [Walk], [Find], [Grab], [Wash], [Wipe], [Pull], [Push], [Pour], [TurnTo], [PointAt], [Watch], [Touch], [Open], [Close], [Run], [Sit], [Read], [PutOn], [Drop], [Lie], [SwitchOn], [SwitchOff], [Drink] For action type 3, the available actions are: [PutIn], [PutBack] All action_name of the sub-tasks must be chosen from the above actions, and follow the corresponding format. You are in a house that consists of four rooms. These rooms are bathroom, dining_room, bedroom, home_office. Available objects in the house are : clothes_hat, ceilinglamp, cpuscreen, orchid, couch, trashcan, dresser, dishwasher, centerpiece, phone, toaster, measuring_cup, stereo, mat, computer, envelope, oven_mitts, piano_bench, box, photoframe, shower, ceiling, wall,

window, freezer, faucet, detergent, light, desk, napkin, food_rice, kitchen_counter, folder, stovefan, walllamp, food_food, coffee_pot, food_steak, jelly, vacuum_cleaner, powersocket, filing_cabinet, alcohol, bathroom, door, bathroom_counter, clothes_gloves, microwave, oven, sink, milk, ice, bedroom, laptop, doorjamb, food_cake, bills, tea_bag, television, laser_pointer, toilet, board_game, sponge, food_carrot, table, tray, cupboard, mousepad, picture, tvstand, tablelamp, hanger, pot, dry_pasta, floor, knifeblock, curtain, chair, food_bread, drawing, creditcard, check, coffe_maker, character, pasta, bag, food_bacon, bookshelf, toothbrush_holder, cutting_board, home_office, dining_room, nail_polish, pillow, tape, nightstand, bathroom_cabinet, bench, conditioner, cat, bed, keyboard, mouse All object names must be chosen from the above object list

(Observation) Currently, you are standing in the bedroom, and holding nothing in your right hand and nothing in your left hand. pillow is clean. napkin is clean. pillow is dirty. bed is clean. mat is dirty. pillow is close to drawing. tablelamp is close to bed. mat is facing drawing. pillow is inside bedroom. mat is close to table. bed is close to drawing. table is close to mat. pillow is on floor. floor is close to bed. tablelamp is close to pillow. mat is close to curtain. bed is facing computer. mat is close to floor. pillow is facing drawing. curtain is close to mat. bed is close to tablelamp. wall is close to mat. napkin is inside bedroom. window is close to mat. pillow is close to tablelamp. bed is close to floor. pillow is close to wall. mat is close to filing_cabinet. drawing is close to bed. pillow is close to floor. bed is close to wall. filing_cabinet is close to mat. bed is close to nightstand. mat is inside bedroom. pillow is close to pillow. pillow is close to nightstand. mat is close to wall. wall is close to pillow. nightstand is close to pillow. nightstand is close to bed. drawing is close to pillow. floor is close to pillow. bed is inside bedroom. floor is close to mat. wall is close to bed. mat is close to window. pillow,napkin,pillow,mat,bed,pillow,pillow is inside bedroom.

(In-Context Examples)
Task: Watch TV
[**Find**] <remote_control>(1)
[**Find**] <television>(1)
[**SwitchOn**] <television>(1)
[**Find**] <couch>(1)
[**Sit**] <couch>(1)
[**Touch**] <remote_control>(1)
[**TurnTo**] <television>(1)
[**LookAt**] <television>(1)

Task: Turn on light
[**Walk**] <dining_room>(1)
[**Walk**] <light>(1)
[**Find**] <light>(1)
[**SwitchOn**] <light>(1)
[**Find**] <light>(2)
[**SwitchOn**] <light>(2)

Task: Go to sleep
[**Walk**] <bedroom>(1)
[**Walk**] <bed>(1)
[**Lie**] <bed>(1)
[**Sleep**]

Task: Brush teeth
[**Walk**] <bathroom>(1)
[**Walk**] <toothbrush_holder>(1)
[**Find**] <toothbrush_holder>(1)
[**Find**] <toothbrush>(1)
[**Grab**] <toothbrush>(1)
[**Walk**] <tooth_paste>(1)
[**Find**] <tooth_paste>(1)
[**Grab**] <tooth_paste>(1)
[**Pour**] <tooth_paste>(1) <toothbrush>(1)
[**Find**] <teeth>(1)

[**Scrub**] <teeth>(1)

Task: Take nap
[Walk] <bedroom>(1)

F.2 PLAN SAMPLING

(Instruction) You need to act as a task planner, who decompose a high-level household task into several sub-tasks. The temporal relationship between subtask sequences must adhere to common-sense logic. Each sub-task can be one of the following form: 1. [action_name]; 2. [action_name] <object name 1 >(object id 1). 3. [action_name] <object name 1 >(object id 1) <object name 2 >(object id 2). The number of arguments depends on the action type. The (object id) is used to tell the simulator that the actions should be done on the same object instance. For example a program as: [Walk] <glass>(1) [Grab] <glass>(1) Indicates that the agent should first walk to a glass, and then grab that same glass.

(Global Information) For action type 1, the available actions are: [Sleep], [StandUp], [WakeUp] For action type 2, the available actions are: [Walk], [Find], [Grab], [Wash], [Wipe], [Pull], [Push], [Pour], [TurnTo], [PointAt], [Watch], [Touch], [Open], [Close], [Run], [Sit], [Read], [PutOn], [Drop], [Lie], [SwitchOn], [SwitchOff], [Drink] For action type 3, the available actions are: [PutIn], [PutBack] All action_name of the sub-tasks must be chosen from the above actions, and follow the corresponding format. You are in a house that consists of four rooms. These rooms are bathroom, dining_room, bedroom, home_office. Available objects in the house are : clothes_hat, ceilinglamp, cpuscreen, orchid, couch, trashcan, dresser, dishwasher, centerpiece, phone, toaster, measuring_cup, stereo, mat, computer, envelope, oven_mitts, piano_bench, box, photoframe, shower, ceiling, wall, window, freezer, faucet, detergent, light, desk, napkin, food_rice, kitchen_counter, folder, stovefan, walllamp, food_food, coffee_pot, food_steak, jelly, vacuum_cleaner, powersocket, filing_cabinet, alcohol, bathroom, door, bathroom_counter, clothes_gloves, microwave, oven, sink, milk, ice, bedroom, laptop, doorjamb, food_cake, bills, tea_bag, television, laser_pointer, toilet, board_game, sponge, food_carrot, table, tray, cupboard, mousepad, picture, tvstand, tablelamp, hanger, pot, dry_pasta, floor, knifeblock, curtain, chair, food_bread, drawing, creditcard, check, coffe_maker, character, pasta, bag, food_bacon, bookshelf, toothbrush_holder, cutting_board, home_office, dining_room, nail_polish, pillow, tape, nightstand, bathroom_cabinet, bench, conditioner, cat, bed, keyboard, mouse All object names must be chosen from the above object list

(Initial Observation) Currently, you are standing in the home_office, and holding nothing in your right hand and nothing in your left hand.

(In-Context Examples)
Task: Watch TV
[**Find**] <remote_control>(1)
[**Find**] <television>(1)
[**SwitchOn**] <television>(1)
[**Find**] <couch>(1)
[**Sit**] <couch>(1)
[**Touch**] <remote_control>(1)
[**TurnTo**] <television>(1)
[**LookAt**] <television>(1)

Task: Turn on light
[**Walk**] <dining_room>(1)
[**Walk**] <light>(1)
[**Find**] <light>(1)
[**SwitchOn**] <light>(1)
[**Find**] <light>(2)
[**SwitchOn**] <light>(2)

Task: Go to sleep

[**Walk**] <bedroom>(1)
[**Walk**] <bed>(1)
[**Lie**] <bed>(1)
[**Sleep**]

Task: Brush teeth
[**Walk**] <bathroom>(1)
[**Walk**] <toothbrush_holder>(1)
[**Find**] <toothbrush_holder>(1)
[**Find**] <toothbrush>(1)
[**Grab**] <toothbrush>(1)
[**Walk**] <tooth_paste>(1)
[**Find**] <tooth_paste>(1)
[**Grab**] <tooth_paste>(1)
[**Pour**] <tooth_paste>(1) <toothbrush>(1)
[**Find**] <teeth>(1)
[**Scrub**] <teeth>(1)

Task: Take nap

F.3 GROUNDED DECIDING

(Instruction) You need to act as a home robot. At each moment, I will provide you with observations of your current environment, as well as the high-level task I want you to do, and previous mid-level sub-tasks that have been executed. Then, you need to select the best sub-task from the options I provide to complete the designated home task based on the observation and your past experience. When one choosed sub-task causes an error in the environment, you will be provided with the error information and the corresponding sub-task, and you need to re-choose a corrective sub-task at the current time step. For example, The sub-tasks that have been executed in the environment are:
[GRAB] <plate>(1)
[WALK] <dining room>(1)
The choosed sub-task is: [PUTBACK] <plate>(1) <table>(1)
The prompt (error information) would be: The sub-task: "[PUTBACK] <plate>(1) <table>(1)" caused an error: Script is not executable, since <character>(1) is not close to <table>(1) when executing "[PUTBACK] <plate>(1) <table>(1) [1]" Among the following actions, which action would you take.
A. [Find] <table>(1)
B. [Find] <plate>(1)
A corrective choice of sub-task would be (You just need to provide the mark before the option you want to choose): A

(Observation) Currently, you are standing in the bedroom, and holding nothing in your right hand and nothing in your left hand. pillow is clean. napkin is clean. pillow is dirty. bed is clean. mat is dirty. pillow is close to drawing. tablelamp is close to bed. mat is facing drawing. pillow is inside bedroom. mat is close to table. bed is close to drawing. table is close to mat. pillow is on floor. floor is close to bed. tablelamp is close to pillow. mat is close to curtain. bed is facing computer. mat is close to floor. pillow is facing drawing. curtain is close to mat. bed is close to tablelamp. wall is close to mat. napkin is inside bedroom. window is close to mat. pillow is close to tablelamp. bed is close to floor. pillow is close to wall. mat is close to filing_cabinet. drawing is close to bed. pillow is close to floor. bed is close to wall. filing_cabinet is close to mat. bed is close to nightstand. mat is inside bedroom. pillow is close to pillow. pillow is close to nightstand. mat is close to wall. wall is close to pillow. nightstand is close to pillow. nightstand is close to bed. drawing is close to pillow. floor is close to pillow. bed is inside bedroom. floor is close to mat. wall is close to bed. mat is close to window. pillow,napkin,pillow,mat,bed,pillow,pillow is inside bedroom.

Your task is: Take nap.

(History)
Your previously executed sub-tasks are:
[**Walk**] <bedroom>(1)

Among the following sub-tasks, which one would you take.
A. [FIND] <bed>(1)
B. [WALK] <couch>(1)
C. [WALK] <bed>(1)
D. [FIND] <couch>(1)
E. [FIND] <pillow>(1)
The best choice of sub-task is:

G VISUALIZED ACTION TREE

We visualized the action trees for the tasks listed in Table 6 for interested readers.

Task	**Action Tree**
Take nap	See Figure 11
Put on your shoes	See Figure 12
Pick up spare change on dresser	See Figure 13
Clean toilet	See Figure 14
Put alarm clock in bedroom	See Figure 15

Table 6: Action trees for various tasks.

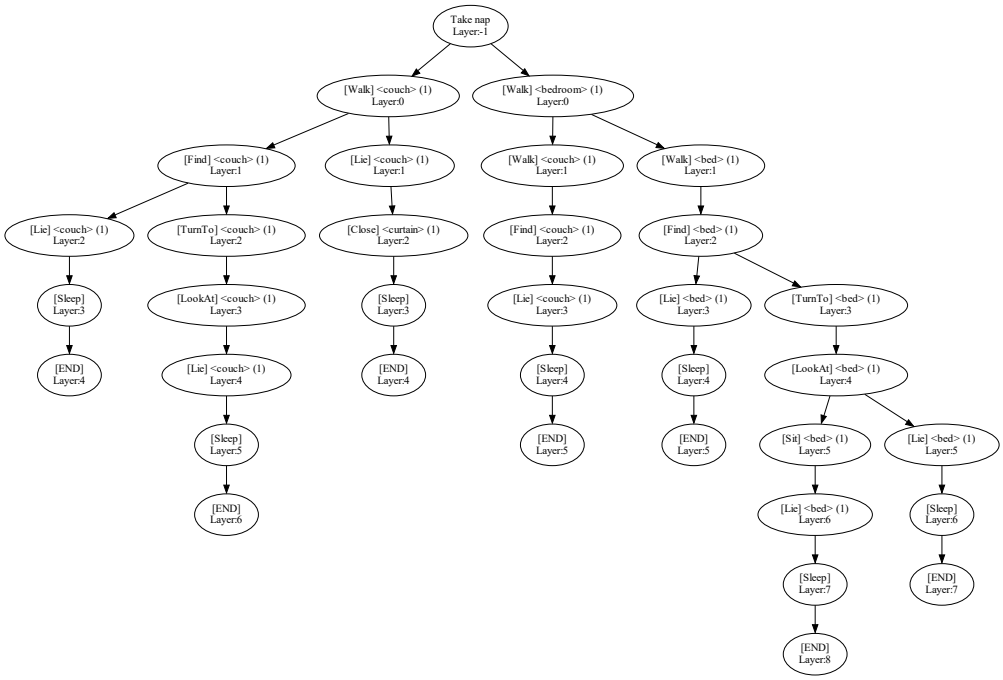

Figure 11: Visualization of the action tree for *Take nap*.

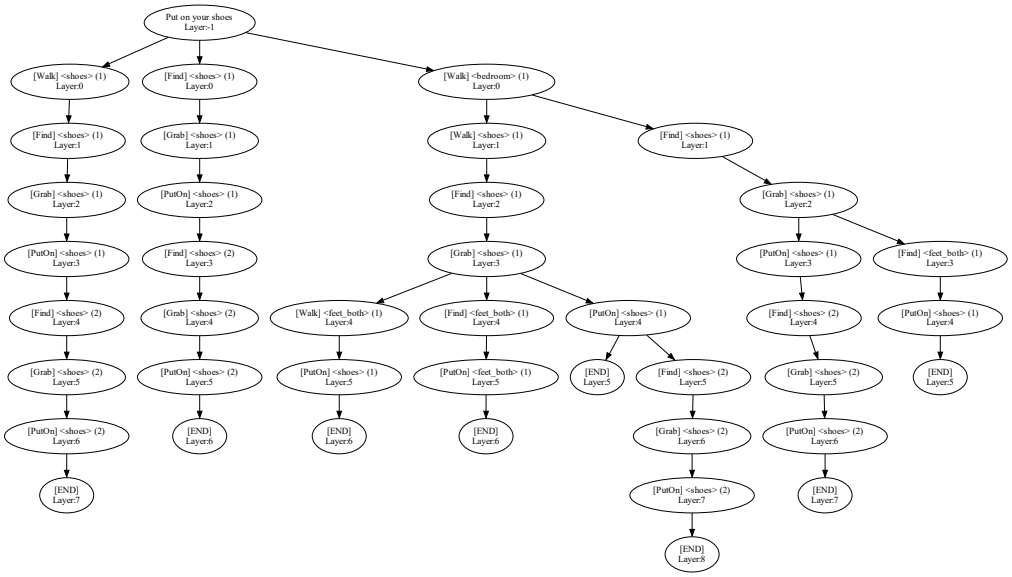

Figure 12: Visualization of the action tree for *Put on your shoes*.

Figure 13: Visualization of the action tree for *Pick up spare change on dresser*.

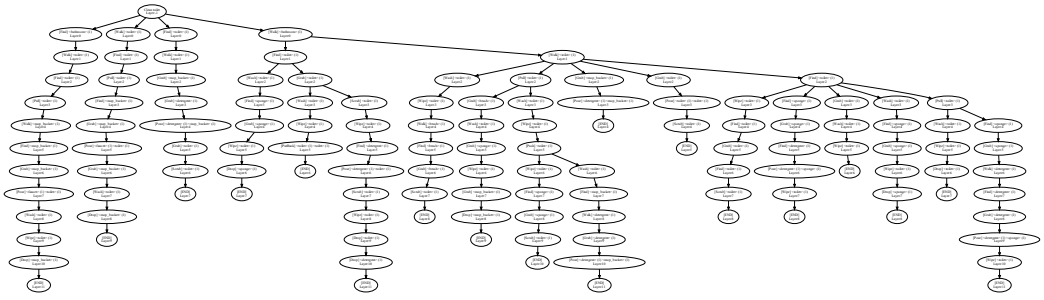

Figure 14: Visualization of the action tree for *Clean toilet*.

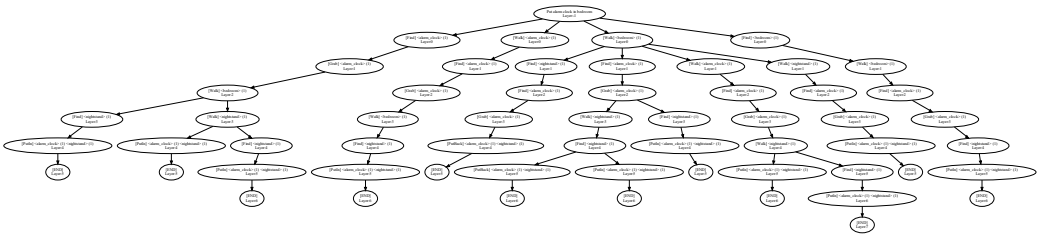

Figure 15: Visualization of the action tree for *Put alarm clock in bedroom*.