

Appendix

1 Questionnaire for Collecting Situation Dataset

To collect execution-time situations, a questionnaire was designed and published on Amazon Mechanical Turk. Fig. 1 shows the Mechanical Turk interface for one everyday task (i.e., drinking water). In the interface, each MTurker was provided with a task description, including steps for completing the task. The MTurkers were asked to respond to a questionnaire by identifying one step in the provided plan and describing a situation that might occur in that step within the blank. On the questionnaire, there are six everyday tasks (e.g., setting a dining table) associated with their steps, which were extracted from an existing dataset [1]. In the end, we have collected a dataset of 651 valid situations, where each instance of the dataset corresponds to a situation that prevents a service robot from completing a task in a dining domain. In the next section, we will discuss the statistics of the dataset.

2 Statistics of Situation Dataset

Figs. 2 and 3 show the statistics of situations for six everyday tasks used in our evaluation, where x -axis reflects the occurrence of each *distinguishable situations*, and y -axis represents each distinguishable situations, respectively. In the top left corner of each subfigure, (X) represents the number of distinguishable situations in each task. In the bottom right corner of each subfigure, Total = X represents the number of *situations* in each task. According to the two figures, we can see that there are at least 92 situations collected for each of the six tasks used in our evaluation, with 16 to 22 distinguishable situations.

3 Object Library for Simulation

For simulating dining tasks, we extracted 86 objects (e.g., cup, burger, folk, table, and chair) from an existing dataset [1]. Fig. 4 shows these objects, which is categorized into five groups: utensil, appliance, furniture, food, and beverage. From the figure, we can see that the category “utensil” contains the greatest number of objects (i.e., 29), while the category “beverage” contains the fewest ones (i.e., 8).

4 Closed-World Task Planners in PDDL

For each task in the evaluation, we developed a closed-world task planner in PDDL. PDDL, an action-centered language, is designed to formalize Artificial Intelligence (AI) planning problems, allowing for a more direct comparison of planning algorithms and implementations [2]. Fig. 5 shows a task planner for the task of “drinking water”, which consists of a domain file (**upper**) and a problem file (**lower**). In the upper subfigure, a set of predicates (e.g., `cup_at`) and a set of actions (e.g., `fill`) are predefined,

Task description → Task: Drinking water

Steps for completing the task → Plan:
Step 1: Walk to the dining room.
Step 2: Find a sink.
Step 3: Find a faucet.
Step 4: Turn on the faucet.
Step 5: Find a cup.
Step 6: Hold the cup.
Step 7: Fill the cup with water.
Step 8: Turn off the faucet.
Step 9: Drink water.

Please select a step where a situation happens.

Identify one step →

Describe a situation → Please describe the situation that prevents us from carrying out the plan.

Figure 1: The Mechanical Turk interface for the task of drinking water. Each MTurker was provided with a task description, including nine steps for completing the task. The MTurkers were asked to respond to a questionnaire by identifying one step (from Steps 1 to 9) in the provided plan and describing a situation that might occur in that step.

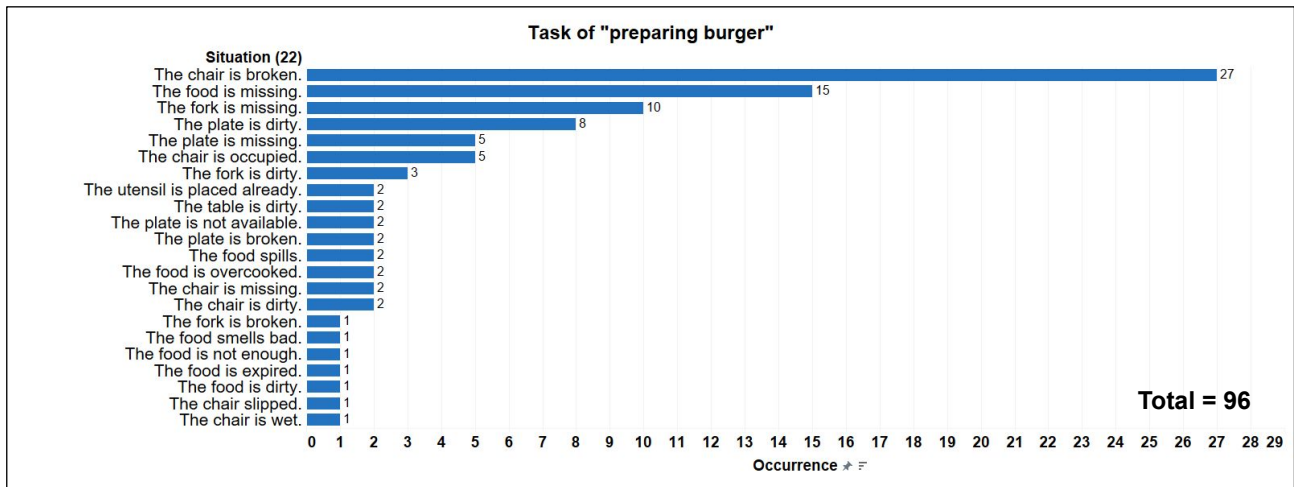
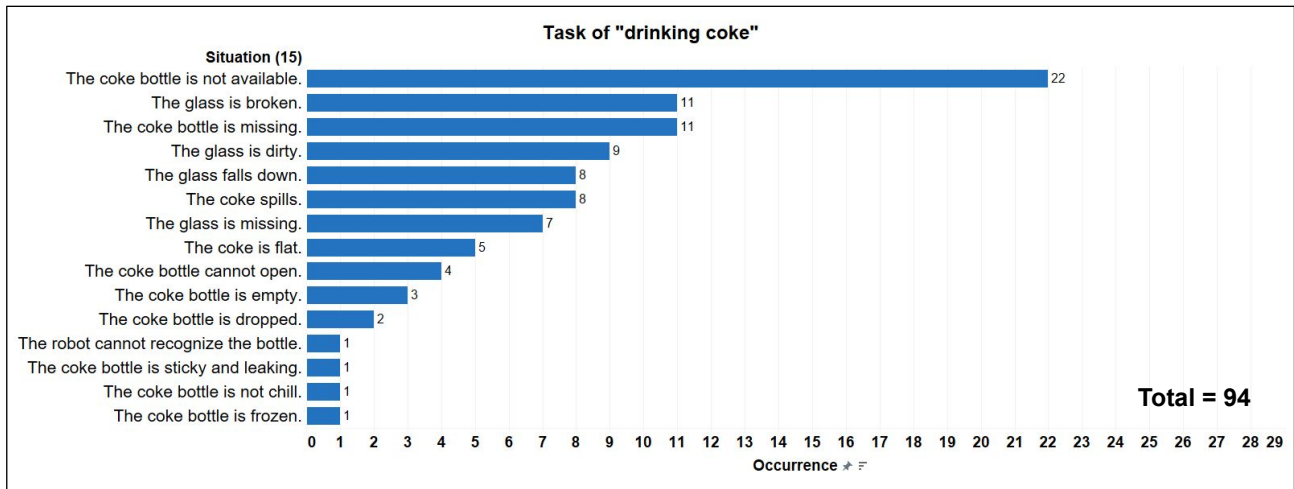
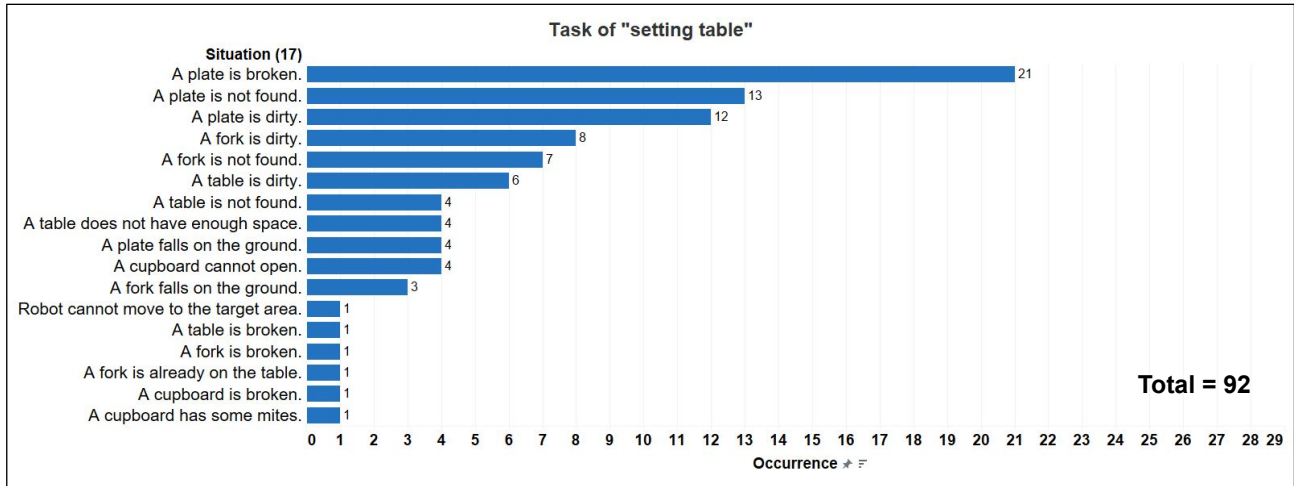


Figure 2: **Top:** Details of situations in the task of “setting table”; **Middle:** Details of situations in the task of “drinking coke”; **Bottom:** Details of situations in the task of “preparing burger”; *x-axis* reflects the occurrence of each *distinguishable situations*, and *y-axis* represents each distinguishable situations, respectively. (X) in the top left corner of each subfigure represents the number of distinguishable situations in each task. Total = X indicates the number of situations in each task.

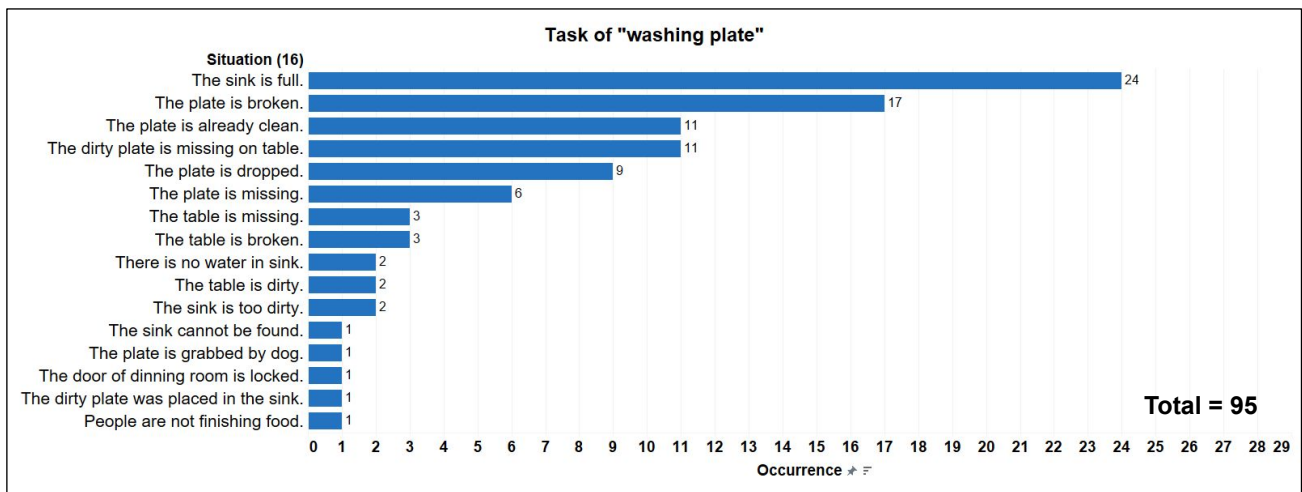
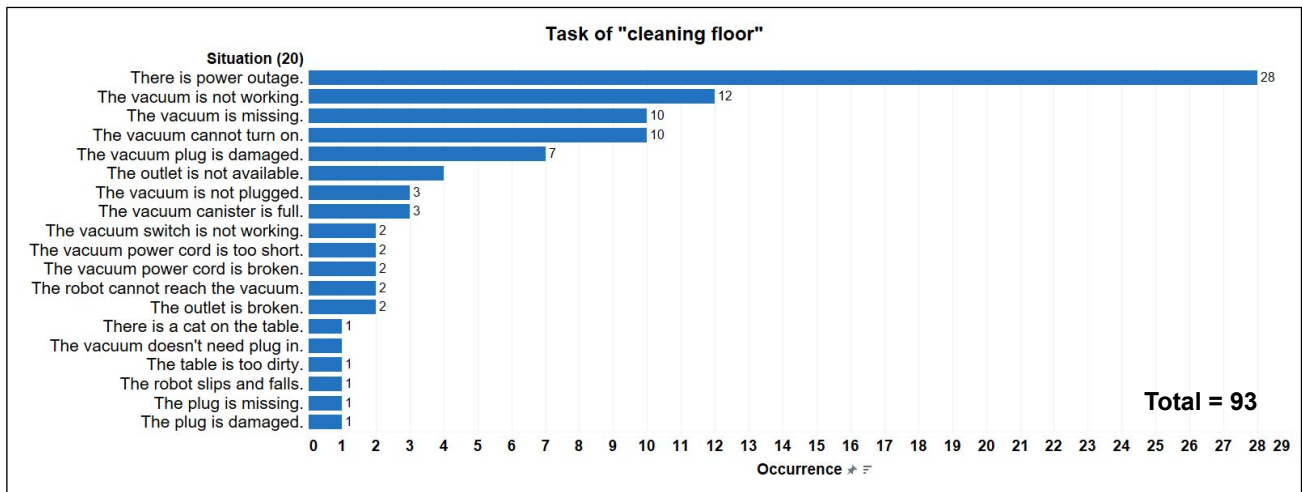
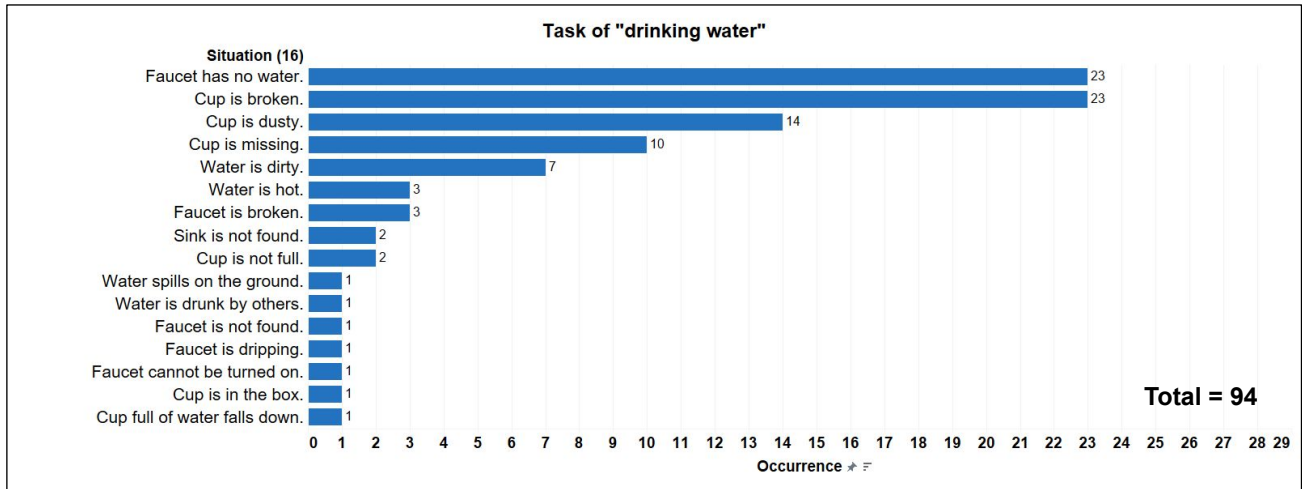


Figure 3: **Top:** Details of situations in the task of “drinking water”; **Middle:** Details of situations in the task of “cleaning floor”; **Bottom:** Details of situations in the task of “washing plate”; *x-axis* reflects the occurrence of each *distinguishable situations*, and *y-axis* represents each distinguishable situations, respectively. (X) in the top left corner of each subfigure represents the number of distinguishable situations in each task. Total = X indicates the number of situations in each task.

Utensil (29)	Appliance (18)	Furniture (16)	Food (15)	Berevage (8)
dish cleaning bottle cooking pot frying pan trash can cloth napkin paper towel bucket coffee cup colander condiment bottle dish bowl drinking glass measuring cup mug rag wine glass chef knife condiment shaker cutlery fork cutlery knife cutting board dish rack oven tray mat wooden spoon sponge coffee filter wooden chopstick	blender dishwasher freezer microwave fridge oven stove washing machine kettle vacuum cleaner toaster air fryer dehumidifier water boiler ice cream maker juicer water filter coffee maker	bookshelf closet cpu table cupboard desk kitchen cabinet nightstand wooden chair piano bench table cloth coffee table couch dining table kitchen table kitchen counter pantry	bread cake ice cream noodles oatmeal peanut butter rice salt snack sugar oil pasta chips sauce steak	beer milk watermelon juice alcohol coffee juice tea wine

Figure 4: For simulating dining tasks, we extracted 86 objects from an existing dataset [1]. These objects are categorized into five groups: utensil, appliance, furniture, food, and beverage, with (X) representing the number of objects in each group.

where an action is defined by its preconditions and effects. For example, one of preconditions for action `fill` is $(\text{cup_is_held } ?c) \wedge (\text{cup_is_empty } ?c)$, and the action effect is $(\text{cup_is_filled } ?c)$. In the lower subfigure, a task problem is defined by an initial state and a goal state (i.e., a user is satisfied and the faucet is turned off.) A task plan for drinking water is generated after inputting these two files into a solver¹, as shown below:

```

_____ Task Plan for Drinking Water _____
S1: (walk rob dining kitchen)
S2: (find_faucet rob faucet_0 kitchen)
S3: (find_cup rob cup_1 kitchen)
S4: (hold rob cup_1 kitchen)
S5: (turnon rob faucet_0 kitchen)
S6: (fill rob cup_1 faucet_0 kitchen)
S7: (turnoff rob faucet_0 kitchen)
S8: (walk rob kitchen dining)
S9: (place rob cup_1 table_0 dining)
S10: (done cup_1 person_1)
_____

```

5 Prompt Design

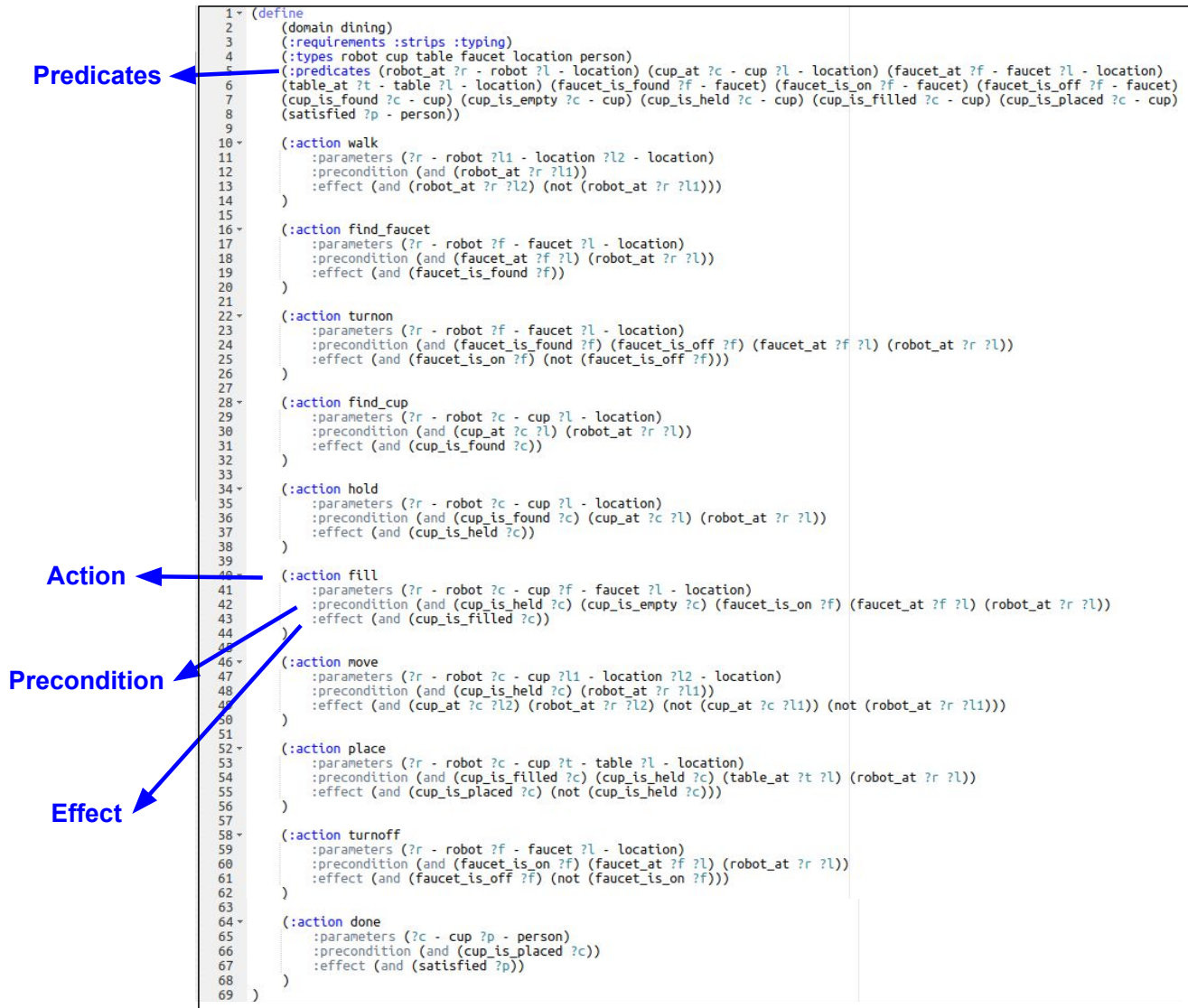
Fig. 6 shows three examples for prompt construction based on our Templates 1-3, respectively. In the figure, the interface, called Playground², is intended for testing GPT-3 online, where a user can text a prompt in the blank, and customize the hyperparameters of GPT-3 (e.g., model). In our case, we use the *text-davinci-002* model, which is the most capable engine. The prompt for Plan Monitor (PM) is constructed based on Template 1, where PM evaluates if the current task plan is feasible or not. In the **top** figure, we can know an action precondition that “one cannot fill a broken cup with water” according to the common sense from GPT-3. The prompts for Knowledge Acquirer (KA) are constructed based on Template 2 and Template 3, where KA extracts common sense to augment the classical task planner. In these two figures (**middle** and **bottom**), we can know that common sense that “one can use a bowl for drinking water” can be added into an action effect, according to the common sense from GPT-3.

References

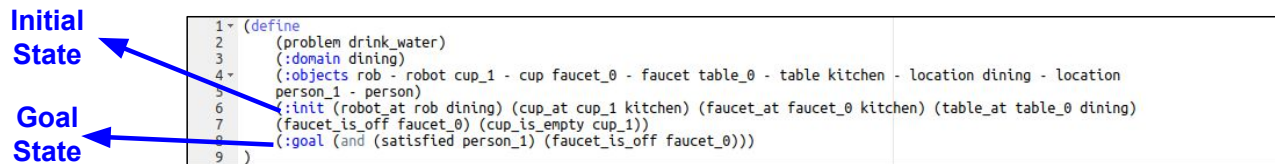
- [1] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, “Language models as zero-shot planners: Extracting actionable knowledge for embodied agents,” *Thirty-ninth International Conference on Machine Learning*, 2022.
- [2] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. SRI, A. Barrett, D. Christianson *et al.*, “Pddl— the planning domain definition language,” *Technical Report, Tech. Rep.*, 1998.

¹The solver is accessible at <http://editor.planning.domains/>

²The playground of GPT-3 is accessible at <https://beta.openai.com/playground>.



A Domain File for Task “Drinking Water”

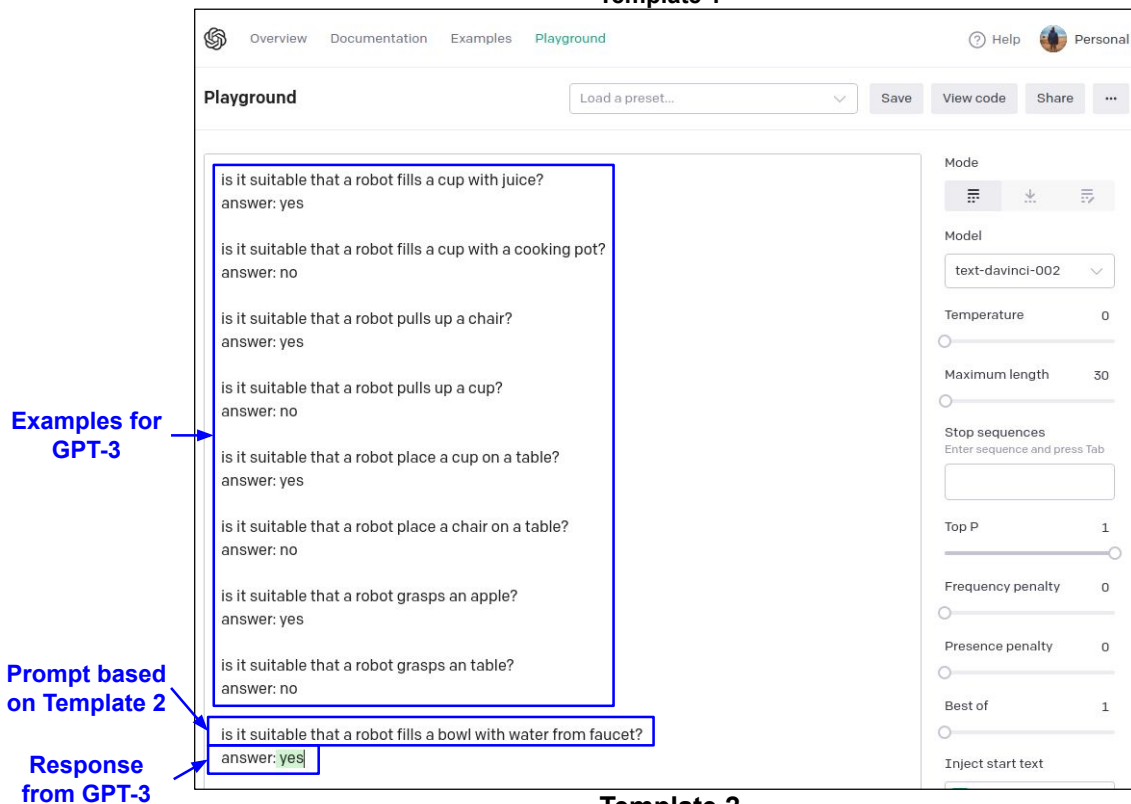


A Problem File for Task “Drinking Water”

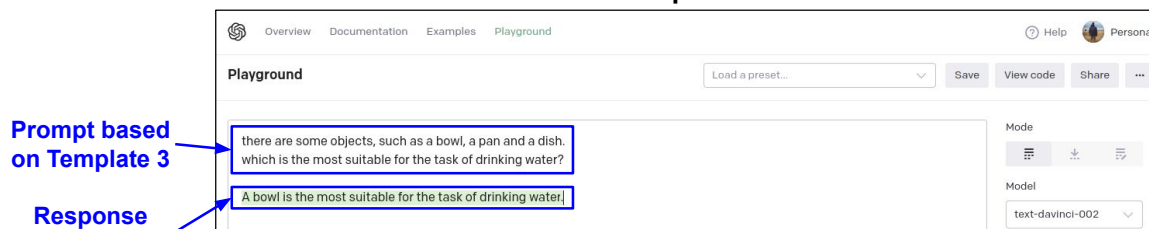
Figure 5: A closed-world task planner in PDDL for the task of “drinking water”, consisting of a domain file (upper) and a problem file (lower). In the upper subfigure, a set of predicates and a set of actions are predefined, where an action is defined by its preconditions and effects. In the lower subfigure, a task problem is defined by an initial state and a goal state.



Template 1



Template 2



Template 3

Figure 6: Three examples for prompt construction based on Templates 1-3.