Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents

Wenlong Huang
UC Berkeley

Pieter Abbeel
UC Berkeley

Deepak Pathak*
Carnegie Mellon University

Igor Mordatch*
Google

Abstract

Can world knowledge learned by large language models (LLMs) be used to act in interactive environments? In this paper, we investigate the possibility of grounding high-level tasks, expressed in natural language (e.g. "make breakfast"), to a chosen set of actionable steps (e.g. "open fridge"). While prior work focused on learning from explicit step-by-step examples of how to act, we surprisingly find that if pre-trained LMs are large enough and prompted appropriately, they can effectively decompose high-level tasks into mid-level plans without any further training. However, the plans produced naively by LLMs often cannot map precisely to admissible actions. We propose a procedure that conditions on existing demonstrations and semantically translates the plans to admissible actions. Our evaluation in the recent VirtualHome environment shows that the resulting method substantially improves executability over the LLM baseline. The conducted human evaluation reveals a trade-off between executability and correctness but shows a promising sign towards extracting actionable knowledge from language models.

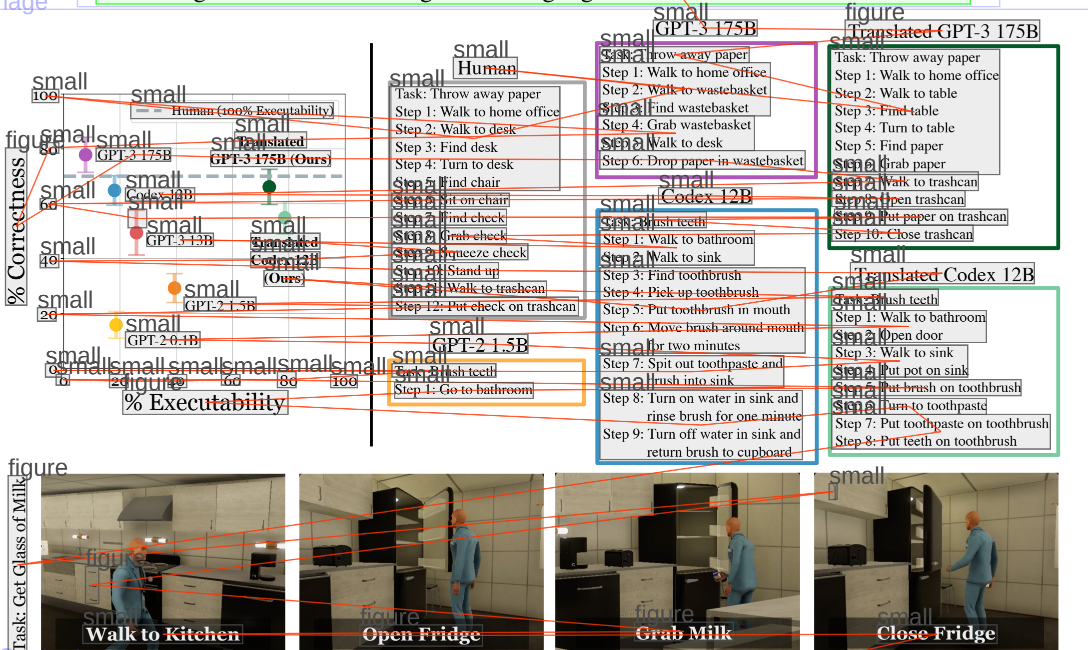

Figure 1: Executability v.s. semantic correctness of generated plans **(left)**, sample plans by different models **(right)**, and example environment execution (**bottom**). Large models can produce action plans indistinguishable from those by humans, but frequently are not executable in the environment. Using our techniques, we can significantly improve executability, albeit at the cost of correctness. More samples can be found in Appendix A.5.

*Equal advising. Correspondence to Wenlong Huang <wenlong.huang@berkeley.edu>.
Code and videos at https://huangwl18.github.io/language-planner

Contents

1 Introduction

Large language models (LLMs) have made impressive advances in language generation and understanding in recent years [10, 39, 40, 5]. See [4] for a recent summary of their capabilities and impacts. Being trained on large corpora of human-produced language, these models are thought to contain a lot of information about the world [42, 23, 3] - albeit in linguistic form.

We ask whether we can use such knowledge contained in LLMs not just for linguistic tasks, but to make goal-driven decisions that can be enacted in interactive, embodied environments. But we are not simply interested in whether we can train models on a dataset of demonstrations collected for some specific environment – we are instead interested in whether LLMs *already contain* information necessary to accomplish goals without any additional training.

More specifically, we ask whether world knowledge about how to perform high-level tasks (such as "make breakfast") can be expanded to a series of groundable actions (such as "open fridge", "grab milk", "close fridge", etc) that can be executed in the environment. For our investigation, we use the recently proposed VirtualHome environment [38]. It can simulate a large variety of realistic human activities in a household environment and supports the ability to perform them via embodied actions defined with a `verb-object` syntax. However, due to the open-ended nature of the tasks, it is difficult to autonomously evaluate their success. We rely on human evaluation (conducted on Mechanical Turk) to decide whether sequences of actions meaningfully accomplish posed tasks.

We find that large GPT-3 [5] and Codex [7] models, when prompted with a single fixed example of a task description and its associated sequence of actions, can produce very plausible action plans for the task we're interested in. Such completions reflect the information already stored in the model – no model fine-tuning is involved. Additionally, we only observe this effect in the larger models. Unfortunately, despite their semantic correctness, the produced action plans are often not executable in the environment. Produced actions may not map precisely to admissible actions, or may contain various linguistic ambiguities.

We propose several tools to improve executability of the model's outputs. First, we enumerate all admissible actions and map the model's output phrases to the most semantically-similar admissible action (we use similarity measure between sentence embeddings produced by a RoBERTa model [27] in this work, but other choices are possible). Second, we use the model to autoregressively generate actions in a plan by conditioning past actions that have been made admissible via the technique above. Such on-the-fly correction can keep generation anchored to admissible actions. Third, we provide weak supervision to the model by prompting the model with a known task example similar to the query task. This is somewhat reminiscent of prompt tuning approaches but does not require access to gradients or internals of the model.

Using the above tools to bias model generation, we find that we improve executability of action plans from 18% to 79% (see Figure 1) without any invasive modifications to model parameters or any extra gradient or internal information beyond what is returned from the model's forward pass. This is advantageous because it does not require any modifications to the model training procedure and can fit within existing model serving pipelines. However, we do find there to be some drop in correctness of the action sequences generated with the above tools (as judged by humans), indicating a promising step, but requiring more research on the topic.

To summarize, our paper's contributions are as follows:

- We show that without any training, large language models can be prompted to generate plausible goal-driven action plans, but such plans are frequently not executable in interactive environments.
- We propose several tools to improve executability of the model generation without invasive probing or modifications to the model.
- We conduct a human evaluation of multiple techniques and models and report on the trade-offs between executability and semantic correctness.

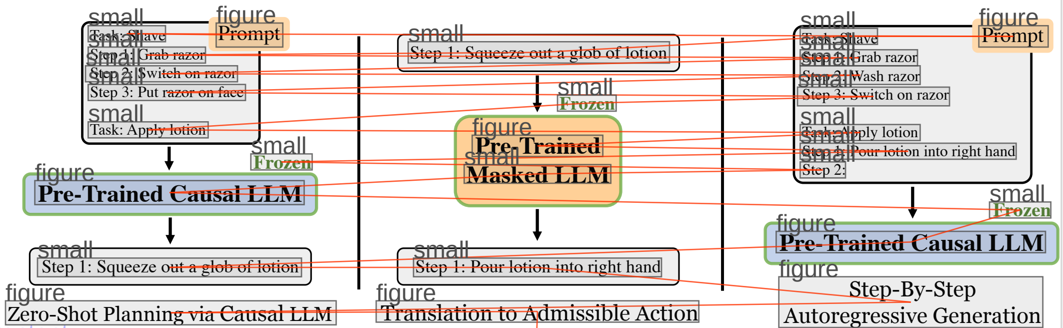

Figure 2: We investigate the possibility of extracting actionable knowledge from pre-trained large language models (LLMs). We first show surprising finding that **pre-trained causal LLMs** can decompose high-level tasks into sensible mid-level action plans (**left**). To make the plans executable, we propose to translate each step into admissible action via another **pre-trained masked LLM** (**middle**). The translated action is appended to the prompt used for generating the remaining steps (**right**). All models are kept **frozen** without additional training.

2 Evaluation Framework

Simulating open-ended tasks that resemble naturalistic human activities requires an environment to support a rich set of diverse interactions, rendering most existing embodied environments unsuitable for our investigation. One exception is VirtualHome [38], which we evaluate on as it models complex human activities, though only in a household setting. To measure correctness of the generated action plans, for which evaluating computationally is inherently difficult for these open-ended tasks, we conduct a human evaluation similar to Puig et al. [38]. We note that since no further training is involved throughout our investigations, the observations and findings presented in this paper should also translate to similar embodied environments, likely even beyond the household domain.

2.1 Evaluated Environment: VirtualHome

Preliminaries In VirtualHome, activities are expressed as programs. Each program consists of a sequence of textual action steps, where each step is written as: `[action] ⟨arg⟩(idx)`. Each `action` refers to one of the 42 atomic actions supported in VirtualHome, such as "walk" and "open". Full list of atomic actions can be found in Appendix A.4. Different actions take in different numbers of `arg`, such as "bedroom" and "fridge", that are necessary for specifying an interaction. Associated with each `arg` is a unique `id` specifying the corresponding node in the environment graph, in case of multiple instances of the same object class are present in the graph. For the sake of simplicity, we omit the `id` in the remaining discussions of this paper and allow automatic assignment by the environment. An example program is shown below for the task "Relax on sofa":

```
[WALK] ⟨living_room⟩(1)
[WALK] ⟨television⟩(1)
[FIND] ⟨television⟩(1)
[SWITCHON] ⟨television⟩(1)
[FIND] ⟨sofa⟩(1)
[SIT] ⟨sofa⟩(1)
[TURNTO] ⟨television⟩(1)
[WATCH] ⟨television⟩(1)
```

Evaluated Tasks We use the *ActivityPrograms* knowledge base collected by Puig et al. [38] for evaluation. It contains 2821 different entries annotated by Amazon Mechanical Turk (MTurk) workers. Each entry contains 1) a high-level task name (e.g. "Watch TV"), 2) detailed instructions expressed in natural language to complete the task (e.g. "Sit on my couch directly opposite my TV, switch on my TV with the remote control and watch"), and 3) an executable program containing all necessary steps for a robotic agent (example above). We omit the use of detailed instructions (2) as we desire direct extraction of executable programs (3) from only high-level task names (1). There are 292 distinct high-level tasks in the knowledge base, from which we randomly sample 88 held-out tasks for evaluation. The remaining 204 tasks are used as *demonstration set* from which we are allowed

Algorithm 1: Generating Action Plans from Pre-Trained Language Models

Notation Summary:
LM_P: text completion language model (also referred as **Planning LM**)
LM_T: text embedding language model (also referred as **Translation LM**)
$\{(T_i, E_i)\}_{i=1}^N$: demonstration set, where T is task name and E is example plan for T
C: cosine similarity function
P: mean token log probability under LM_P
Input: query task name Q, e.g. "make breakfast"
Output: action plan consisting of admissible env actions, e.g. "open fridge"

```
Extract most similar example (T*, E*) whose T* maximizes C(LM_T(T), LM_T(Q))
Initialize prompt with (T* + E* + Q)
while max step is not reached do
    Sample LM_P with current prompt to obtain k single-step action phrases
    for each sample â and each admissible env action a_e do
        Calculate ranking score by C(LM_T(â), LM_T(a_e)) + β · P(â)
    end for
    Append highest-scoring env action a_e* to prompt
    Append a_e* to output
    if > 50% samples are 0-length or highest score < ε then
        break
    end if
end while
```

to select as example(s) for prompting language models, or in the case of supervised fine-tuning baselines, they are used to fine-tune pre-trained language models.

2.2 Metrics

A program that commands the agent to wander around in a household environment is highly executable but is mostly not correct. On the other hand, a program composed of natural language instructions annotated by humans is likely correct but cannot be executed, because its format is ambiguous and may lack necessary common-sense actions (e.g. fridge must be opened before an agent can grab things from it). We thus consider two axes for evaluation: **executability** and **correctness**.

Executability Executability measures whether an action plan can be *correctly parsed* and *satisfies the common-sense constraints* of the environment. To be correctly parsed, an action plan must be syntactically correct and contain only allowed actions and recognizable objects. To satisfy the common-sense constraints, each action step must not violate the set of its pre-conditions (e.g. the agent cannot grab milk from the fridge before opening it) and post-conditions (e.g. the state of the fridge changes from "closed" to "open" after the agent opens it). We report the average executability across all 88 tasks and all 7 VirtualHome scenes.

Correctness Unlike most embodied environments where the completion of a task can be easily judged, the ambiguous and multimodal nature of natural language task specification makes it impractical to obtain a gold-standard measurement of correctness[1]. Therefore, we conduct human evaluations for the main methods. For the remaining analysis, we rely on a match-based metric that measures how similar a generated program is to human annotations. Specifically, we follow Puig et al. [38] and calculate the longest common subsequence (LCS) between two programs, normalized by the maximum length of the two. In the presence of multiple human-written programs for a single task, we take the maximum LCS across them. However, we note that the majority of the tasks only have one human annotation, but there are often many plausible ways to complete a certain task, making

[1] One approach could be measuring the similarity of the final environment state produced by executing predicted and human-written programs, but initial state must be kept fixed for each task, which are not appropriate for many tasks due to their open-ended nature.

this metric imperfect at evaluation program correctness[2]. Although correlation between the two is shown by Puig et al. [38], we consider it only as a proxy metric in replacement of unscalable human evaluation.

3 Method

In this section, we investigate the possibility of extracting actionable knowledge from pre-trained language models without further training. We first give an overview of the common approach to query large language models (LLMs) and how it may be used for embodied agents in Section 3.1. Then we describe an inference-time procedure that addresses several deficiencies of the LLM baseline and offers better executability in embodied environments. We break down the proposed procedure into three individual components, each discussed in Section 3.2, 3.3, 3.4. Pseudo-code is in Algorithm 1.

Since LMs excel at dealing with natural language text instead of the specific format required by VirtualHome as described in Section 2.1, we only expose natural language text to LMs. To do this, we define a bi-directional mapping for each atomic action that converts between the natural language format and the program format. For instance, "walk to living room" is mapped to `[WALK] ⟨living_room⟩(1)`. Full list of the mappings is in Appendix A.4.

3.1 Querying LLMs for Action Plans

Previous works have shown that large language models pre-trained on a colossal amount of data would internalize rich world knowledge that can be probed to perform various downstream tasks [39, 5]. Notably, autoregressive LLMs can even perform in-context learning, an ability to solve tasks using only contextual information without gradient updates [5]. Contextual information is given as part of the input prompt and LMs are asked to complete the remaining text. It often consists of natural language instructions and/or a number of examples containing the desired input/output pairs.

We adopt the same approach to query LLMs to generate action plans for high-level tasks. Specifically, we prepend one example high-level task and its annotated action plan from the *demonstration set* to the query task, as shown in Figure 2. To obtain text completion results, we sample from autoregressive LLM using temperature sampling and nucleus sampling [18]. We refer to this LM as **Planning LM** and the approach using this LM for plan generation as **Vanilla** ⟨**LM**⟩, where ⟨**LM**⟩ is replaced by specific language model such as GPT-3.

To improve the generation quality, we follow Chen et al. [7] to sample multiple outputs for each query. However, unlike Chen et al. [7] who investigate program synthesis and can choose the sample with highest unit test pass rate, we only consider the setting where one sample is allowed to be evaluated for each task. This is because repetitive trial-and-error is equivalent to probing the environment for privileged information, which should not be considered viable in our setting. For Vanilla ⟨LM⟩, to choose the best action plan X^* among k samples $(X_1, X_2, ..., X_k)$, each consisting of n_i tokens $X_i = (x_{i,1}, x_{i,2}, ..., x_{i,n_i})$, we select the sample with highest mean log probability as follows:

$$\underset{X_i}{\operatorname{argmax}} \left(P_\theta(X_i) := \frac{1}{n_i} \sum_{j=1}^{n_i} \log p_\theta(x_{i,j}|x_{i,<j}) \right) \quad \text{where } \theta \text{ parameterizes the Planning LM.} \tag{1}$$

3.2 Admissible Action Parsing by Semantic Translation

One issue arises when naively following the above approach to generate action plans: the plan expressed in free-form language often cannot be mapped to unambiguous actionable steps and thus is not executable by a robotic agent. Many reasons can cause such failures: 1) the output does not follow pre-defined mappings of any atomic action (e.g. "I first walk to the bedroom" is not of the format "walk to ⟨PLACE⟩"), 2) the output may refer to atomic action and objects using words unrecognizable by the environment (e.g. "microwave the chocolate milk" where "microwave" and "chocolate milk" cannot be mapped to precise action and object), or 3) the output contains lexically ambiguous words (e.g. "open TV" should instead be "switch on TV").

[2]Although LCS has a mathematical range of $[0, 1]$, we measure the LCS between different human-written programs for the same task and find an empirical maximum of 0.489.

Instead of developing a set of rules to transform the free-form text into admissible action steps, we propose to again leverage world knowledge learned by language models to semantically translate the action. For each admissible environment action a_e, we calculate its semantic distance to the predicted action phrase $\hat{a}$ by cosine similarity:

$$C(f(\hat{a}), f(a_e)) := \frac{f(\hat{a}) \cdot f(a_e)}{\|f(\hat{a})\| \|f(a_e)\|} \quad \text{where } f \text{ is an embedding function.} \tag{2}$$

To embed the output action phrase and environment actions, we use a BERT-style LM [10, 27] pre-trained with Sentence-BERT [41] objective, to which we refer as **Translation LM**[3]. The action embedding is obtained by mean-pooling the last layer hidden states across all tokens in that action phrase. While the set of admissible actions in our environment is discrete and possible to exhaustively enumerate, sampling or projection can be employed in larger discrete or continuous action spaces.

3.3 Autoregressive Trajectory Correction

Translating each step of the program after the entire program has been synthesized lacks consideration of achievability of individual steps and subjects to compounding errors. In practice, LLMs might output compounded instructions for a single step, even though it cannot be completed using one admissible action in the environment. To this end, we can instead interleave *plan generation* and *action translation* to allow for automatic trajectory correction. At each step, we first query Planning LM to generate k samples for a single action $(\hat{a}_1, \hat{a}_2, ..., \hat{a}_k)$. For each sample $\hat{a}$, we consider both its semantic soundness and its achievability in the environment. Specifically, we aim to find admissible environment action a_e by modifying the ranking scheme described in Equation 1 as follows:

$$\operatorname*{argmax}_{a_e} \left[\max_{\hat{a}} C(f(\hat{a}), f(a_e)) + \beta \cdot P_\theta(\hat{a}) \right] \quad \text{where } \beta \text{ is a weighting coefficient.} \tag{3}$$

Then we append the translated environment action a_e to the unfinished text completion. This way all subsequent steps will be conditioned on admissible actions instead of free-form action phrases generated by Planning LM. Furthermore, we can use Translation LM to detect out-of-distribution actions, those outside the capabilities of a robot, and terminate a program early instead of mapping to a faulty action. This can be achieved by setting a threshold ϵ such that if $\max_{\hat{a}, a_e} C(f(\hat{a}), f(a_e)) + \beta \cdot P_\theta(\hat{a}) < \epsilon$ at step t, the program is terminated early. Since we now sample Planning LM for individual steps instead of an entire sequence, another termination condition we consider is when $> 50\%$ of current-step samples are 0-length (excluding leading or trailing non-English text tokens).

3.4 Dynamic Example Selection for Improved Knowledge Extraction

So far in the text, we always give the same example in the prompt for all query tasks. However, consider the task of "ordering pizza". Prompting LLMs with this task may give the assumption that the agent is initialized in front of a computer, and the LLMs may guide the agent to search for a pizza store and click "checkout my cart". Although these are reasonable and feasible in the real world, such assumption cannot always be made as these interactions may not be supported in simulated environments. In fact, the closest series of actions that human experts give in VirtualHome may be "walking to a computer", "switching on the computer", and "typing the keyboard". Without being fine-tuned on these data, LLMs would often fail at these tasks.

To provide weak supervision at inference time, we propose to select the most similar task T and its example plan E from the *demonstration set* to be used as the example in the prompt. Specifically, we re-use the same Translation LM introduced in Section 3.2 and select (T^*, E^*) whose high-level task name T^* maximizes $C(f(T), f(Q))$, where Q is the query task. This approach bears resemblance to several recent works [37, 13, 26, 43]. An example is shown in Figure 2 where "Shave" is the most similar to the query task "Apply lotion".

Final Method Combining the various improvement discussed above, we refer to the final method as **Translated** ⟨**LM**⟩, where ⟨**LM**⟩ is replaced by specific language model used such as GPT-3.

[3] Note that this is a different LM than the GPT-style Planning LM. Using a single LM for both purposes could as well be possible and likely more efficient, but we leave such investigation to future works.

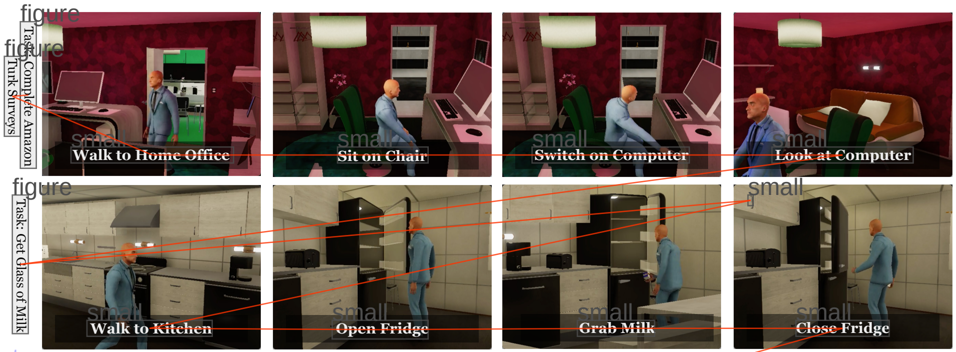

Figure 3: Visualization of VirtualHome programs generated by our approach. The top row shows the execution of the task "Complete Amazon Turk Surveys", and the bottom row shows the task "Get Glass of Milk". We show LLMs not only can generate sensible action plans given only high-level tasks but also contains the actionable knowledge that can be extracted for grounding in embodied environments.

4 Results

In this section, we first show that language models can generate sensible action plans for many high-level tasks, even without any additional training. Then we highlight its inadequacy when naively applied to embodied environments and demonstrate how this can be improved by again leveraging world knowledge learned by LLMs. Visualization of generated programs is shown in Figure 3.

Sampling from LMs Pre-trained LMs are sensitive to sampling parameters and the specific example given in the prompt. For all evaluated methods, we perform hyperparameter search over various sampling parameters, and for methods using a fixed prompt example, we report metrics averaged across three randomly chosen examples. To select the best run for each method, we rank the runs by the sum of LCS and executability, each normalized by human-expert scores. Further details are in Appendix A.1.

Model Choices For Planning LM, we evaluate a representative set of causal language models. For Translation LM, we mainly use Sentence-RoBERTa-355M and provide relevant ablations in Section 5.3. GPT-3 and Codex are accessed using OpenAI API, and the remaining models are accessed through open-source packages, Hugging Face Transformers [55] and SentenceTransformers [41], all without additional training (except for the fine-tuning baseline).

4.1 Do LLMs contain actionable knowledge for high-level tasks?

We first investigate whether LLMs can generate sensible action plans expressed in free-form language. We use the approach described in Section 3.1 to query pre-trained LLMs. To evaluate the correctness of generated action plans, we conduct human evaluations. For each model, we ask 10 human annotators to determine – by answering "Yes" or "No" – whether each task can be completed using provided action steps. To provide a reference of how humans might rate the action plans provided by other humans, we also ask annotators to rate the human-written action plans included in the VirtualHome dataset for the same set of tasks. In contrast to the free-form text output by LLMs, humans wrote the plans using a graphical programming interface that enforces strict syntax and a chosen set of atomic action vocabulary, which limit the expressivity and the completeness of their answers[4]. More details of our human evaluation procedure can be found in Appendix A.2.

We show the human evaluation results in Figure 1, where the y-axis shows correctness averaged across all tasks and all annotators. Surprisingly, when LLMs are large enough and without imposed syntactic constraints, they can generate highly realistic action plans whose correctness – as deemed by human annotators – even surpasses human-written action plans. We also observe some level of correctness for smaller models such as GPT-2. However, inspection of its produced output indicates

[4] Puig et al. [38] also conduct a human evaluation on 100 randomly sampled human-written programs and show that 64% of them are complete (i.e. contain all necessary steps). Readers are encouraged to refer to Puig et al. [38] for a more comprehensive analysis of the dataset.

Language Model	Executability	LCS	Correctness
Vanilla GPT-2 117M	18.66%	3.19%	15.81% (4.90%)
Vanilla GPT-2 1.5B	39.40%	7.78%	29.25% (5.28%)
Vanilla Codex 2.5B	17.62%	15.57%	63.08% (7.12%)
Vanilla GPT-Neo 2.7B	29.92%	11.52%	65.29% (9.08%)
Vanilla Codex 12B	18.07%	16.97%	64.87% (5.41%)
Vanilla GPT-3 13B	25.87%	13.40%	49.44% (8.14%)
Vanilla GPT-3 175B	7.79%	17.82%	77.86% (6.42%)
Human	100.00%	N/A	70.05% (5.44%)
Fine-tuned GPT-3 13B	66.07%	34.08%	64.92% (5.96%)
OUR FINAL METHODS			
Translated Codex 12B	78.57%	24.72%	54.88% (5.90%)
Translated GPT-3 175B	73.05%	24.09%	66.13% (8.38%)

Table 1: Human-evaluated correctness and evaluation results in VirtualHome. Although action plans generated by large language models can match or even surpass human-written plans in correctness measure, they are rarely executable. By translating the naive action plans, we show an important step towards grounding LLMs in embodied environments, but we observe room to achieve this without trading executability for correctness. We also observe a failure mode among smaller models that lead to high executability. For correctness measure, standard error of the mean across 10 human annotators is reported in the parenthesis.

that it often generates shorter plans by ignoring common-sense actions or by simply rephrasing the given task (e.g. the task “Go to sleep” produces only a single step “Go to bed”). These failure modes sometimes mislead human annotators to mark them correct as the annotators may ignore common-sense actions in their judgment as well, resulting in a higher correctness rate than the quality of the output shows.

4.2 How executable are the LLM action plans?

We analyze the executability of LLM plans by evaluating them in all 7 household scenes in VirtualHome. As shown in Table 1, we find action plans generated naively by LLMs are generally not very executable. Although smaller models seem to have higher executability, we find that the majority of these executable plans are produced by ignoring the queried task and repeating the given example of a different task. This is validated by the fact that smaller models have lower LCS than larger models despite having high executability, showing that this failure mode is prevalent among smaller models. In contrast, larger models do not suffer severely from this failure mode. Yet as a result of being more expressive, their generated programs are substantially less executable.

4.3 Can LLM action plans be made executable by proposed procedure?

We evaluate the effectiveness of our proposed procedure of action translation. We first create a bank of all allowed 47522 action steps in the environment, including all possible combinations of atomic actions and allowed arguments/objects. Then we use an off-the-shelf Sentence-RoBERTa [27, 41] as Translation LM to create embeddings for actions and output text. For better computational efficiency, we pre-compute the embeddings for all allowed actions, leaving minor computation overhead for our procedure over the baseline methods at inference time. As shown in Table 1, executability of generated programs is significantly improved. Furthermore, we also observe improved LCS because the translated action steps precisely follow the program syntax and thus are more similar to the plans produced by human experts. Sample output is shown in Figure 1 and a larger random subset of generated samples can be found in Appendix A.5.

To validate their correctness, we again perform human evaluations using the same procedure from Section 4.1. Results are shown in Table 1. We find that despite being more similar to human-written plans as they follow strict syntax, the programs are deemed less correct by humans compared to their vanilla counterparts. By examining the output, we observe two main sources of errors. First, we find Translation LM is poor at mapping compounded instructions to a succinct admissible action, e.g. “brush teeth with toothbrush and toothpaste”. Second, we find that the generated programs are sometimes terminated too early. This is partly due to the imperfect expressivity of the environment;

certain necessary actions or objects are not implemented to fully achieve some tasks, so Translation LM cannot map to a sufficiently similar action. This is also reflected by our human evaluation results of the programs written by other humans, as only 70% of the programs are considered complete.

5 Analysis and Discussions

5.1 Ablation of design decisions

We perform ablation studies for the three components of our proposed procedure, described in Section 3.2, 3.3, and 3.4 respectively. As shown in Table 2, leaving out any of the three components would all lead to decreased performance in both executability and LCS. An exception is Translated GPT-3 w/o Trajectory Correction, where we observe a slight improvement in LCS at the expense of a considerable drop in executability. Among the three proposed components, leaving out action translation leads to the most significant executability drop, showing the importance of action translation in extracting executable action plans from LLMs.

Methods	**Executability**	**LCS**
Translated Codex 12B	**78.57%**	**24.72%**
- **w/o** Action Translation	31.49%	22.53%
- **w/o** Dynamic Example	50.86%	22.84%
- **w/o** Trajectory Correction	55.19%	24.43%
Translated GPT-3 175B	**73.05%**	24.09%
- **w/o** Action Translation	36.04%	24.31%
- **w/o** Dynamic Example	60.82%	22.92%
- **w/o** Trajectory Correction	40.10%	**24.98%**

Table 2: Ablation of three proposed techniques.

5.2 Are the generated action plans grounded in the environment?

Since successful execution of correct action plans directly measures grounding, we calculate the percentage of generated action plans that are both *correct* and *executable*. We deem an action plan to be correct if 70% or more human annotators decide it is correct. Human-written plans are 100% executable, of which 65.91% are deemed correct. Results for LMs are shown in Figure 4.

Although smaller LMs such as GPT-2 can generate highly executable action plans as shown in Table 1, these executable plans mostly are not correct, as they often repeat the given example or do not contain all necessary steps. Increasing model parameters can lead to some improvement in generating plans that are both executable and correct, yet it scales poorly with the parameter count. In the meantime, action translation offers a promising way towards grounding actionable knowledge by producing executable and correct plans, though a large gap remains to be closed to reach human-level performance (65.91%).

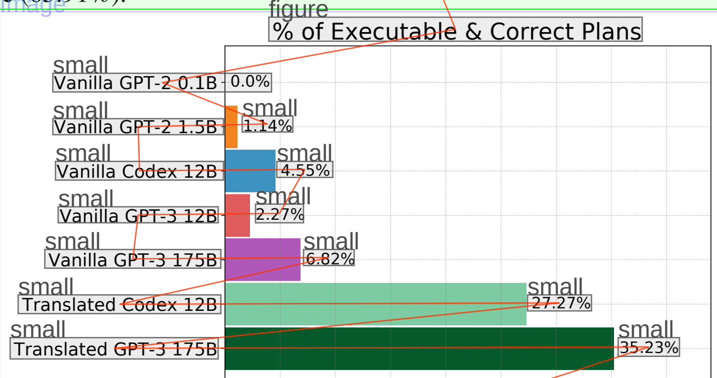

Figure 4: Percentage of both executable and correct action plans generated by LMs.

5.3 Effect of Different Translation LMs

In this section, we study the effect of using different Translation LM. We compare two size variants of Sentence BERT and Sentence RoBERTa [10, 27, 41] trained on the STS benchmark [6] and a baseline using averaged GloVe embeddings [35]. Results are shown in Table 3. Notably, we do not observe significant differences in executability and LCS across different variants of BERT and RoBERTa. We hypothesize that this is because any language models trained on reasonably large datasets should be capable of the single-step action phrase translation considered in this work. However, simply using average GloVe embeddings would lead to significantly reduced performance.

Translation LM	Parameter Count	Executability	LCS
CODEX 12B AS PLANNING LM			
Avg. GloVe embeddings	-	46.92%	9.71%
Sentence Bert (base)	110M	73.21%	24.10%
Sentence Bert (large)	340M	75.16%	20.79%
Sentence RoBERTa (base)	125M	74.35%	22.82%
Sentence RoBERTa (large)	325M	**78.57%**	**24.72%**
GPT-3 175B AS PLANNING LM			
Avg. GloVe embeddings	-	47.40%	12.16%
Sentence Bert (base)	110M	**77.60%**	**24.49%**
Sentence Bert (large)	340M	67.86%	21.24%
Sentence RoBERTa (base)	125M	72.73%	23.64%
Sentence RoBERTa (large)	325M	73.05%	24.09%

Table 3: Effect of different Translation LMs on executability and LCS.

5.4 Can LLMs generate actionable programs by following step-by-step instructions?

Prior works often focus on translating step-by-step instructions into executable programs. Specifically, instead of only providing a high-level task name, *how-to* instructions are also provided, as shown in Figure 5. Although this setting is easier as it does not require rich prior knowledge, *how-to* instructions can help resolve much ambiguity of exactly how to perform a high-level task when multiple solutions are possible. To investigate whether pre-trained LLMs are capable of doing this without additional training, we include these instructions in the prompt and evaluate LLMs with the proposed procedure. We compare to a supervised baseline from VirtualHome that trains an LSTM [17] from scratch on human-annotated data. Since the code to train the baseline is not publicly released and a different train/test split is likely used, we only show results reported in Puig et al. [38] as a crude reference. We also cannot compare executability as it is not reported. Results are shown in Table 4. Surprisingly, without being fine-tuned on any domain data, Translated Codex/GPT-3 can attain LCS close to supervised methods while generating highly executable programs.

```
Task: Read book
Description: Walk to home office,
turn on light, grab a book, sit in
chair, start to read the book.
Step 1: Walk to home office
Step 2: Walk to light
Step 3: Find light
Step 4: Switch on light
Step 5: Find novel
Step 6: Grab novel
Step 7: Find chair
Step 8: Sit on chair
Step 9: Read novel

Task: Find dictionary
Description: Move towards the
bookshelf, scan the bookshelf for
the dictionary, when the
dictionary is found, pick up the
dictionary.
```

Figure 5: An example prompt containing step-by-step instructions.

Methods	Executability	LCS
Translated Codex 12B	78.57%	32.87%
Translated GPT-3 175B	74.15%	31.05%
Supervised LSTM	-	34.00%

Table 4: Executability and LCS when conditioned on step-by-step instructions.

5.5 Analysis of program length

Shorter programs have a natural advantage of being more executable as they need to satisfy less pre/post-conditions, albeit being prone to incompleteness. To validate the proposed approach does not simply generate very short programs, we calculate the average program length across the 88 evaluated tasks. Results are shown in Table 5. Mirroring the observations made in Section 4.1 and Section 4.2, we find smaller LMs such as GPT-2 tend to generate shorter programs than larger models do while frequently repeating the given executable example. In contrast, larger models like Codex and GPT-3 can generate more expressive programs with high realism, yet consequently, they often suffer from executability. We show proposed procedure can find appropriate balance and is capable of generating programs that are highly executable while maintaining reasonable expressiveness as measured by program length.

Methods	**Executability**	**Average Length**
Vanilla GPT-2 1.5B	39.40%	4.24
Vanilla Codex 12B	18.07%	7.22
Vanilla GPT-3 175B	7.79%	9.716
Translated Codex 12B	78.57%	7.13
Translated GPT-3 175B	73.05%	7.36
Human	100.00%	9.66

Table 5: Average executability & program length of different methods.

6 Related Works

Large-scale natural language modeling has witnessed rapid advances since the inception of the Transformer architecture [53]. It has been shown by recent works that large language models (LLMs) pre-trained on large unstructured text corpus not only can perform strongly on various down-stream NLP tasks [10, 39, 40, 5] but the learned representations can also be used to model relations of entities [23], retrieve matching visual features [19], synthesize code from docstrings [15, 7], solve math problems [8, 46], and even as valuable priors when applied to diverse tasks from different modalities [28, 52]. Notably, by pre-training on large-scale data, these models can also internalize an implicit knowledge base containing rich information about the world from which factual answers (e.g. "Dante was born in ⟨PLACE⟩") can be extracted [36, 21, 9, 50, 42]. Compared to prior works in *single-step* knowledge extraction, we aim to extract *sequential* action plans to complete open-ended human activities while satisfying various constraints of an interactive environment.

Many prior works have looked into grounding natural language in embodied environments. A series of them parse language instructions into formal logic or rely mainly on lexical analysis to resolve various linguistic ambiguities for embodied agents [2, 33, 34, 51]. However, they often require many hand-designed rules or scale inadequately to more complex tasks and environments. Recently, many efforts have been put into creating more realistic environments with the goal to further advances in this area [38, 47, 48, 22, 44, 1]. At the same time, by leveraging the better representation power of neural architectures, a number of works have looked into creating instruction-following agents that can perform manipulation [29, 30], navigation [11, 54, 31], or both [49, 16, 12]. Recent works also use language as hierarchical abstractions to plan actions using imitation learning [45] and to guide exploration in reinforcement learning [32].

Notably, many prior works do not leverage full-blown pre-trained LLMs; most investigate smaller LMs that require considerable domain-specific data for fine-tuning to obtain reasonable performance. Perhaps more importantly, few works have evaluated LLMs in an embodiment setting that realizes the full potential of the actionable knowledge these models *already contain* by pre-training on large-scale unstructured text: the tasks evaluated are often generated from a handful of templates, which do not resemble the highly diverse activities that humans perform in daily lives [14, 20]. The development of VirtualHome environment [38] enables such possibility. However, relevant works [38, 25] rely on human-annotated data and perform supervised training from scratch. Due to the lack of rich world knowledge, these models can only generate action plans given detailed instructions of how to act or video demonstrations. Concurrent work by Li et al. [24] validates similar hypothesis that

LMs contain rich actionable knowledge. They fine-tune GPT-2 with demonstrations to incorporate environment context and to predict actions in VirtualHome, and evaluate on tasks that are generated from pre-defined predicates. In contrast, we investigate *existing* knowledge in LLMs without any additional training and evaluate on human activity tasks expressed in free-form language.

7 Conclusion, Limitations & Future Work

In this work, we investigate actionable knowledge *already contained* in pre-trained LLMs without any additional training. We present several techniques to extract this knowledge to perform common-sense grounding by planning actions for complex human activities.

Despite promising findings, there remain several limitations of this work which we discuss as follows:

Drop in Correctness Although our approach can significantly improve executability of the generated plans, we observe a considerable drop in correctness. In addition to the errors caused by the proposed action translation (discussed in Section 4.3), this is partially attributed to the limited expressivity of VirtualHome, as it may not support all necessary actions to fully complete all evaluated tasks (correctness is judged by humans). This is also reflected by that Vanilla LMs can even surpass human-written plans, which are restricted by environment expressivity.

Mid-Level Grounding Instead of grounding the LLM generation to low-level actions by using downstream data from a specific environment, we focus on high-level to mid-level grounding such that we evaluate raw knowledge of LLMs as closely and broadly as possible. Hence, we only consider the most prominent challenge in mid-level grounding that the generated plans must satisfy all common-sense constraints (characterized by executability metric). As a result, we assume there is a low-level controller that can execute these mid-level actions (such as "grab cup"), and we do not investigate the usefulness of LLMs for low-level sensorimotor behavior grounding. To perform sensorimotor grounding, such as navigation and interaction mask prediction, domain-specific data and fine-tuning are likely required.

Ignorant of Environment Context We do not incorporate observation context or feedback into our models. To some extent, we approach LLMs in the same way as how VirtualHome asks human annotators to write action plans for a given human activity by *imagination*, in which case humans similarly do not observe environment context. Similar to human-written plans, we assume the plans generated by LMs only refer to one instance of each object class. As a result, successful plan generation for tasks like "stack two plates on the right side of a cup" is not possible.

Evaluation Protocol We measure quality of plans by a combination of *executability* and *correctness* instead of one straightforward metric. To the best of our knowledge, there isn't a known way to computationally assess the semantic correctness of the plans due to the tasks' open-ended and multi-modal nature. Prior work also adopt similar combination of metrics [38]. We report two metrics individually to shine light on the deficiencies of existing LLMs which we hope could provide insights for future works. To provide a holistic view, we report results by combining two metrics in Section 5.2.

We believe addressing each of these shortcoming will lead to exciting future directions. We also hope these findings can inspire future investigations into using pre-trained LMs for goal-driven decision-making problems and grounding the learned knowledge in embodied environments.

Acknowledgment

We would like to thank OpenAI for providing academic access to the OpenAI API and Luke Metz for valuable feedback and discussions. This work was supported in part by Berkeley Deep Drive, NSF IIS-2024594, and GoodAI Research Award.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[2] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013.

[3] BIG-bench collaboration. Beyond the imitation game: Measuring and extrapolating the capabilities of language models. *In preparation*, 2021. URL https://github.com/google/BIG-bench/.

[4] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2021.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[8] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[9] Joe Davison, Joshua Feldman, and Alexander M Rush. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178, 2019.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *arXiv preprint arXiv:1806.02724*, 2018.

[12] Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

[13] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

[14] Brent Harrison and Mark O Riedl. Learning from stories: using crowdsourced narratives to train virtual agents. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[15] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

[16] Felix Hill, Sona Mokra, Nathaniel Wong, and Tim Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv preprint arXiv:2005.09382*, 2020.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[18] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[19] Gabriel Ilharco, Rowan Zellers, Ali Farhadi, and Hannaneh Hajishirzi. Probing text models for common ground with visual representations. *arXiv e-prints*, pages arXiv–2005, 2020.

[20] Peter A Jansen. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. *arXiv preprint arXiv:2009.14259*, 2020.

[21] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

[22] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

[23] Belinda Z Li, Maxwell Nye, and Jacob Andreas. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*, 2021.

[24] Shuang Li, Xavier Puig, Yilun Du, Clinton Wang, Ekin Akyurek, Antonio Torralba, Jacob Andreas, and Igor Mordatch. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

[25] Yuan-Hong Liao, Xavier Puig, Marko Boben, Antonio Torralba, and Sanja Fidler. Synthesizing environment-aware activities via activity sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6291–6299, 2019.

[26] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[28] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

[29] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[30] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *Proceedings of Robotics: Science and Systems. doi*, 10, 2021.

[31] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *European Conference on Computer Vision*, pages 259–274. Springer, 2020.

[32] Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. Ella: Exploration through learned language abstraction. *arXiv preprint arXiv:2103.05825*, 2021.

[33] Dipendra Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 992–1002, 2015.

[34] Dipendra K Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3):281–300, 2016.

[35] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[36] Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*, 2019.

[37] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227*, 2022.

[38] Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2018.

[39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[41] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[42] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

[43] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*, 2021.

[44] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347, 2019.

[45] Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

[46] Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. Generate & rank: A multi-task framework for math word problems. *arXiv preprint arXiv:2109.03034*, 2021.

[47] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749, 2020.

[48] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[49] Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv preprint arXiv:2108.04927*, 2021.

[50] Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. olmpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8: 743–758, 2020.

[51] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *2010 ieee international conference on robotics and automation*, pages 1486–1491. IEEE, 2010.

[52] Maria Tsimpoukelli, Jacob Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *arXiv preprint arXiv:2106.13884*, 2021.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[54] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[55] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

A Appendix

A.1 Hyperparameter Search

For each evaluated method, we perform grid search over the following hyperparameters:

Name	Description	Search Values
epsilon (ϵ)	Out-of-distribution early termination threshold	$\{0, 0.4, 0.8\}$
temperature	sampling parameter adjusting relative token probabilities	$\{0.1, 0.3, 0.6\}$
k	number of samples generated by Planning LM	$\{1, 10\}$
beta (β)	weighting coefficient in action translation to trade off semantic and translation correctness	$\{0.3\}$
frequence_penalty	**OpenAI API only**; penalize new tokens based on their existing frequency in the text so far	$\{0.1, 0.3, 0.6, 0.9\}$
presence_penalty	**OpenAI API only**; penalize new tokens based on whether they appear in the text so far	$\{0.3, 0.5, 0.8\}$
repetition_penalty	**Hugging Face Transformers only**; penalize new tokens based on whether repeating existing text	$\{1.0, 1.2, 1.5, 1.8\}$

For methods that use fixed example across evaluated tasks, we search over the following three randomly chosen examples:

Example 1	**Example 2**	**Example 3**
Task: Use computer Step 1: Walk to home office Step 2: Walk to chair Step 3: Find chair Step 4: Sit on chair Step 5: Find computer Step 6: Switch on computer Step 7: Turn to computer Step 8: Look at computer Step 9: Find keyboard Step 10: Type on keyboard	Task: Relax on sofa Step 1: Walk to home office Step 2: Walk to couch Step 3: Find couch Step 4: Sit on couch Step 5: Find pillow Step 6: Lie on couch	Task: Read book Step 1: Walk to home office Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find chair Step 6: Sit on chair Step 7: Read novel

A.2 Details of Human Evaluations

Human evaluations are conducted on Amazon Mechanical Turk. For each method, we generate action plans for all 88 high-level tasks. To account for the expressivity of the VirtualHome environment [38], we include action plans written by human experts from the VirtualHome dataset as references in our human evaluations. The evaluations are conducted in the form of questionnaires containing all action plans whose order is randomly shuffled and whose corresponding methods are unknown to the annotators. Human annotators are required to answer all the questions in the questionnaire. For each question, the annotators need to answer either "Yes" or "No" indicating if they believe the action plan completes the task. For each method, we report *correctness* percentage averaged across 10 participated human annotators and all 88 tasks. We further report the standard error of the mean across human annotators. Screenshot can be found in Figure 6.

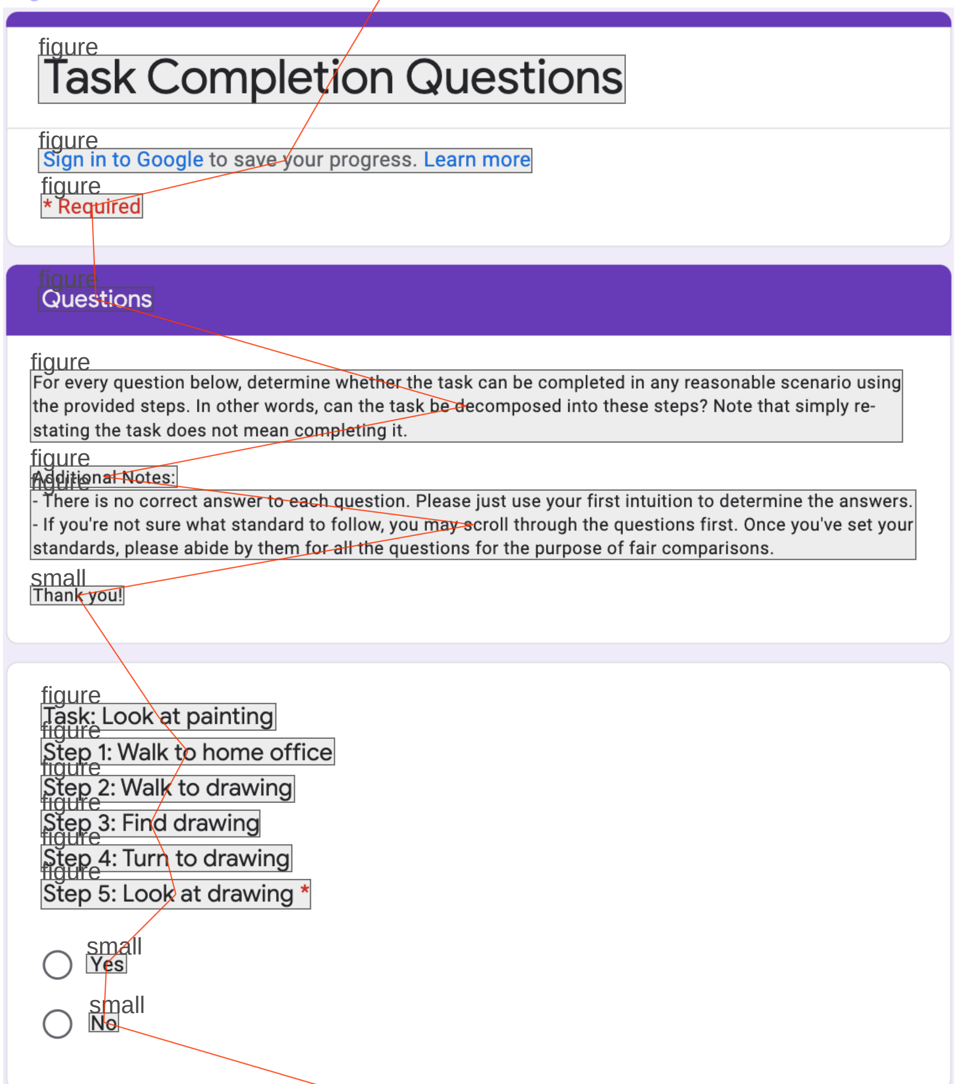

Figure 6: Screenshot of human evaluation interface, conducted as a Google Forms questionnaire.

A.3 All Evaluated Tasks

The evaluated tasks are part of the *ActivityPrograms* dataset collected by Puig et al. [38]. Some of the task names may contain misspelling(s).

1. Apply lotion
2. Arrange folders
3. Breakfast
4. Browse internet
5. Brush teeth
6. Change clothes
7. Change sheets and pillow cases
8. Collect napkin rings
9. Complete surveys on amazon turk
10. Compute
11. Decorate it
12. Do homework
13. Do work
14. Draft home
15. Draw picture
16. Dry soap bottles
17. Dust
18. Eat cereal
19. Eat cheese
20. Eat snacks and drink tea
21. Empty dishwasher and fill dishwasher
22. Entertain
23. Feed me
24. Find dictionary
25. Fix snack
26. Get glass of milk
27. Give milk to cat
28. Go to sleep
29. Grab things
30. Hand washing
31. Hang keys
32. Hang pictures
33. Iron shirt
34. Keep cats inside while door is open
35. Keep cats out of room
36. Leave home
37. Listen to music
38. Look at mirror
39. Look at painting
40. Make bed
41. Make popcorn
42. Organize closet
43. Organize pantry
44. Paint ceiling
45. Pay bills
46. Pick up toys
47. Play musical chairs
48. Prepare pot of boiling water
49. Push all chairs in
50. Push in desk chair
51. Put alarm clock in bedroom
52. Put away groceries
53. Put away toys
54. Put clothes away
55. Put mail in mail organizer
56. Put on your shoes
57. Put out flowers
58. Put up decoration
59. Read
60. Read newspaper
61. Read on sofa
62. Read to child
63. Read yourself to sleep
64. Receive credit card
65. Restock
66. Scrubbing living room tile floor is once week activity for me
67. Style hair
68. Switch on lamp
69. Take jacket off
70. Take shoes off
71. Tale off shoes
72. Throw away paper
73. Try yourself off
74. Turn off TV
75. Turn on TV with remote
76. Turn on radio
77. Type up document
78. Unload various items from pockets and place them in bowl on table
79. Use laptop
80. Vacuum
81. Walk to room
82. Wash dirty dishes
83. Wash face
84. Wash monitor
85. Wash teeth
86. Watch horror movie
87. Wipe down sink
88. Write book

A.4 Natural Language Templates for All Atomic Actions

VirtualHome requires action steps specified in a specific format, yet language models are trained to deal with mostly natural language. We thus define a natural language template for each atomic action and only expose the converted natural language text in all operations involving language models, i.e. autoregressive generation and action translation. After we obtain an entire generated program expressed in natural language, such as those in Figure 1 and Figure 2, we then convert each action step to the VirtualHome syntax. Full list of the atomic actions and their natural language templates can be found below.

Atomic Action in VirtualHome Syntax	**Natural Language Template**
`[CLOSE] ⟨arg1⟩(1)`	close ⟨arg1⟩
`[CUT] ⟨arg1⟩(1)`	cut ⟨arg1⟩
`[DRINK] ⟨arg1⟩(1)`	drink ⟨arg1⟩
`[DROP] ⟨arg1⟩(1)`	drop ⟨arg1⟩
`[EAT] ⟨arg1⟩(1)`	eat ⟨arg1⟩
`[FIND] ⟨arg1⟩(1)`	find ⟨arg1⟩
`[GRAB] ⟨arg1⟩(1)`	grab ⟨arg1⟩
`[GREET] ⟨arg1⟩(1)`	greet ⟨arg1⟩
`[LIE] ⟨arg1⟩(1)`	lie on ⟨arg1⟩
`[LOOKAT] ⟨arg1⟩(1)`	look at ⟨arg1⟩
`[MOVE] ⟨arg1⟩(1)`	move ⟨arg1⟩
`[OPEN] ⟨arg1⟩(1)`	open ⟨arg1⟩
`[PLUGIN] ⟨arg1⟩(1)`	plug in ⟨arg1⟩
`[PLUGOUT] ⟨arg1⟩(1)`	plug out ⟨arg1⟩
`[POINTAT] ⟨arg1⟩(1)`	point at ⟨arg1⟩
`[POUR] ⟨arg1⟩(1) ⟨arg2⟩(1)`	pour ⟨arg1⟩ into ⟨arg2⟩
`[PULL] ⟨arg1⟩(1)`	pull ⟨arg1⟩
`[PUSH] ⟨arg1⟩(1)`	push ⟨arg1⟩
`[PUTBACK] ⟨arg1⟩(1) ⟨arg2⟩(1)`	put ⟨arg1⟩ on ⟨arg2⟩
`[PUTIN] ⟨arg1⟩(1) ⟨arg2⟩(1)`	put ⟨arg1⟩ in ⟨arg2⟩
`[PUTOBJBACK] ⟨arg1⟩(1)`	put back ⟨arg1⟩
`[PUTOFF] ⟨arg1⟩(1)`	take off ⟨arg1⟩
`[PUTON] ⟨arg1⟩(1)`	put on ⟨arg1⟩
`[READ] ⟨arg1⟩(1)`	read ⟨arg1⟩
`[RELEASE]`	release
`[RINSE] ⟨arg1⟩(1)`	rinse ⟨arg1⟩
`[RUN] ⟨arg1⟩(1)`	run to ⟨arg1⟩
`[SCRUB] ⟨arg1⟩(1)`	scrub ⟨arg1⟩
`[SIT] ⟨arg1⟩(1)`	sit on ⟨arg1⟩
`[SLEEP]`	sleep
`[SQUEEZE] ⟨arg1⟩(1)`	squeeze ⟨arg1⟩
`[STANDUP]`	stand up
`[SWITCHOFF] ⟨arg1⟩(1)`	switch off ⟨arg1⟩
`[SWITCHON] ⟨arg1⟩(1)`	switch on ⟨arg1⟩
`[TOUCH] ⟨arg1⟩(1)`	touch ⟨arg1⟩
`[TURNTO] ⟨arg1⟩(1)`	turn to ⟨arg1⟩
`[TYPE] ⟨arg1⟩(1)`	type on ⟨arg1⟩
`[WAKEUP]`	wake up
`[WALK] ⟨arg1⟩(1)`	walk to ⟨arg1⟩
`[WASH] ⟨arg1⟩(1)`	wash ⟨arg1⟩
`[WATCH] ⟨arg1⟩(1)`	watch ⟨arg1⟩
`[WIPE] ⟨arg1⟩(1)`	wipe ⟨arg1⟩

A.5 Random Samples of Action Plans

Human	**Vanilla GPT3-175B**	**Translated GPT3-175B (Ours)**
Task: Go to sleep Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Find bed Step 4: Turn to bed Step 5: Look at bed Step 6: Sit on bed Step 7: Sleep	Task: Go to sleep Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Find pajamas Step 4: Put on pajamas Step 5: Find slippers Step 6: Put on slippers Step 7: Get in bed Step 8: Sleep	Task: Go to sleep Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Lie on bed Step 4: Sleep
Task: Browse internet Step 1: Walk to home office Step 2: Walk to chair Step 3: Find chair Step 4: Sit on chair Step 5: Find computer Step 6: Switch on computer Step 7: Turn to computer Step 8: Look at computer Step 9: Find keyboard Step 10: Type on keyboard	Task: Browse internet Step 1: Walk to home office Step 2: Walk to computer Step 3: Find computer Step 4: Grab computer Step 5: Find chair Step 6: Sit on chair Step 7: Browse internet	Task: Browse internet Step 1: Walk to home office Step 2: Walk to desk Step 3: Find chair Step 4: Sit on chair Step 5: Find computer Step 6: Switch on computer Step 7: Find mouse Step 8: Grab mouse Step 9: Turn to computer Step 10: Point at computer
Task: Wash teeth Step 1: Stand up Step 2: Walk to bathroom Step 3: Walk to sink Step 4: Find sink Step 5: Turn to sink Step 6: Look at sink Step 7: Find teeth Step 8: Wash teeth	Task: Wash teeth Step 1: Walk to bathroom Step 2: Walk to sink Step 3: Turn on faucet Step 4: Grab toothbrush Step 5: Put toothpaste on toothbrush Step 6: Put toothbrush in mouth Step 7: Brush teeth Step 8: Rinse mouth out with water Step 9: Turn off faucet Step 10: Walk back to bedroom	Task: Wash teeth Step 1: Walk to bathroom Step 2: Walk to sink Step 3: Find faucet Step 4: Switch on faucet Step 5: Walk to toothbrush Step 6: Grab toothbrush
Task: Brush teeth Step 1: Walk to bathroom Step 2: Walk to tooth paste Step 3: Find tooth paste Step 4: Grab tooth paste Step 5: Find toothbrush Step 6: Grab toothbrush	Task: Brush teeth Step 1: Walk to bathroom Step 2: Walk to sink Step 3: Turn on water Step 4: Grab toothbrush Step 5: Put toothpaste on toothbrush Step 6: Put toothbrush in mouth Step 7: Brush teeth	Task: Brush teeth Step 1: Find toothbrush Step 2: Turn to toothbrush Step 3: Point at toothbrush Step 4: Wash toothbrush
Task: Vacuum Step 1: Walk to home office Step 2: Walk to computer Step 3: Find vacuum cleaner Step 4: Switch on vacuum cleaner Step 5: Switch off vacuum cleaner	Task: Vacuum Step 1: Walk to home office Step 2: Walk to vacuum Step 3: Find vacuum Step 4: Grab vacuum Step 5: Find chair Step 6: Sit on chair Step 7: Vacuum	Task: Vacuum Step 1: Walk to home office Step 2: Walk to dresser Step 3: Open dresser Step 4: Find vacuum cleaner Step 5: Grab vacuum cleaner Step 6: Pull vacuum cleaner Step 7: Plug in vacuum cleaner Step 8: Switch on vacuum cleaner

Task: Change sheets and pillow cases Step 1: Find pillow Step 2: Turn to pillow Step 3: Look at pillow Step 4: Grab pillow	Task: Change sheets and pillow cases Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Change sheets and pillow cases Step 4: Walk to kitchen Step 5: Walk to sink Step 6: Wash sheets and pillow cases Step 7: Walk to bedroom Step 8: Put sheets and pillow cases on bed Step 9: Walk to living room Step 10: Sit on couch	Task: Change sheets and pillow cases Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Find sheets Step 4: Grab sheets Step 5: Find pillow Step 6: Grab pillow
Task: Wash dirty dishes Step 1: Walk to dining room Step 2: Walk to sink Step 3: Find sponge Step 4: Grab sponge Step 5: Find faucet Step 6: Switch on faucet Step 7: Put sponge on sink Step 8: Grab sponge Step 9: Find dish soap Step 10: Grab dish soap Step 11: Pour dish soap into sponge Step 12: Put back dish soap Step 13: Find plate Step 14: Grab plate Step 15: Scrub plate Step 16: Rinse plate Step 17: Find dishrack Step 18: Put plate on dishrack Step 19: Find plate Step 20: Grab plate Step 21: Scrub plate Step 22: Rinse plate Step 23: Put plate on dishrack Step 24: Rinse sponge Step 25: Put back sponge Step 26: Switch off faucet	Task: Wash dirty dishes Step 1: Walk to kitchen Step 2: Walk to sink Step 3: Wash dishes Step 4: Dry dishes Step 5: Put dishes away	Task: Wash dirty dishes Step 1: Walk to dining room Step 2: Walk to sink Step 3: Find faucet Step 4: Switch on faucet Step 5: Walk to plate Step 6: Grab plate

Task: Feed me
Step 1: Walk to dining room
Step 2: Walk to plate
Step 3: Find plate
Step 4: Grab plate
Step 5: Find food
Step 6: Grab food
Step 7: Put food on plate
Step 8: Find food
Step 9: Grab food
Step 10: Put food on plate
Step 11: Find fork
Step 12: Grab fork
Step 13: Eat plate
Step 14: Eat plate

Task: Feed me
Step 1: Walk to kitchen
Step 2: Walk to refrigerator
Step 3: Open refrigerator
Step 4: Grab food
Step 5: Find plate
Step 6: Put food on plate
Step 7: Find fork
Step 8: Put fork on plate
Step 9: Find spoon
Step 10: Put spoon on plate
Step 11: Find cup
Step 12: Pour water in cup
Step 13: Walk to table
Step 14: Sit at table
Step 15: Eat food

Task: Feed me
Step 1: Walk to dining room
Step 2: Walk to table
Step 3: Find table
Step 4: Turn to table
Step 5: Find chair
Step 6: Sit on chair
Step 7: Find food
Step 8: Grab food
Step 9: Find plate
Step 10: Put food on plate

Task: Breakfast
Step 1: Walk to dining room
Step 2: Walk to food
Step 3: Grab food
Step 4: Walk to bedroom
Step 5: Walk to television
Step 6: Switch on television
Step 7: Walk to bed
Step 8: Lie on bed
Step 9: Turn to television
Step 10: Watch television
Step 11: Eat food

Task: Breakfast
Step 1: Walk to kitchen
Step 2: Find cereal
Step 3: Pour cereal into bowl
Step 4: Pour milk into bowl
Step 5: Walk to living room
Step 6: Sit on couch
Step 7: Eat cereal

Task: Breakfast
Step 1: Walk to dining room
Step 2: Walk to freezer
Step 3: Find freezer
Step 4: Open freezer
Step 5: Find bread
Step 6: Grab bread
Step 7: Close freezer
Step 8: Find toaster
Step 9: Plug in toaster
Step 10: Put bread on plate
Step 11: Switch on toaster

Task: Read
Step 1: Find novel
Step 2: Grab novel
Step 3: Walk to bedroom
Step 4: Walk to chair
Step 5: Sit on chair
Step 6: Read novel

Task: Read
Step 1: Walk to home office
Step 2: Walk to novel
Step 3: Find novel
Step 4: Grab novel
Step 5: Find chair
Step 6: Sit on chair
Step 7: Read novel

Task: Read
Step 1: Walk to home office
Step 2: Walk to light
Step 3: Find light
Step 4: Switch on light
Step 5: Find novel
Step 6: Grab novel
Step 7: Find chair
Step 8: Sit on chair
Step 9: Read novel

Task: Style hair
Step 1: Find towel
Step 2: Turn to towel
Step 3: Point at towel
Step 4: Wash towel
Step 5: Turn to towel
Step 6: Look at towel
Step 7: Squeeze towel
Step 8: Walk to bathroom

Task: Style hair
Step 1: Walk to bathroom
Step 2: Walk to mirror
Step 3: Find hairbrush
Step 4: Grab hairbrush
Step 5: Find chair
Step 6: Sit on chair
Step 7: Style hair

Task: Style hair
Step 1: Walk to bathroom
Step 2: Walk to mirror
Step 3: Find hair
Step 4: Grab hair
Step 5: Find comb
Step 6: Grab comb
Step 7: Find brush
Step 8: Grab brush

Task: Pick up toys Step 1: Walk to bedroom Step 2: Walk to toy Step 3: Find toy Step 4: Grab toy Step 5: Walk to filing cabinet Step 6: Open filing cabinet Step 7: Put toy on filing cabinet Step 8: Find toy Step 9: Grab toy Step 10: Walk to filing cabinet Step 11: Put toy on filing cabinet Step 12: Find toy Step 13: Grab toy Step 14: Walk to filing cabinet Step 15: Put toy on filing cabinet Step 16: Close filing cabinet	Task: Pick up toys Step 1: Walk to living room Step 2: Walk to toy box Step 3: Find toy box Step 4: Grab toy box Step 5: Find chair Step 6: Sit on chair Step 7: Pick up toys	Task: Pick up toys Step 1: Find toy Step 2: Turn to toy Step 3: Point at toy Step 4: Wash toy
Task: Decorate it Step 1: Walk to home office Step 2: Walk to centerpiece Step 3: Find centerpiece Step 4: Grab centerpiece Step 5: Walk to table Step 6: Put centerpiece on table Step 7: Find drawing Step 8: Grab drawing Step 9: Find wall Step 10: Turn to wall Step 11: Put drawing on wall Step 12: Find drawing Step 13: Grab drawing Step 14: Turn to wall Step 15: Put drawing on wall	Task: Decorate it Step 1: Walk to home office Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find chair Step 6: Sit on chair Step 7: Read novel Step 8: Decorate it	Task: Decorate it Step 1: Walk to table Step 2: Walk to centerpiece Step 3: Look at centerpiece Step 4: Grab centerpiece Step 5: Point at centerpiece
Task: Put away groceries Step 1: Walk to dining room Step 2: Walk to freezer Step 3: Find freezer Step 4: Open freezer Step 5: Find food Step 6: Grab food Step 7: Put food in freezer Step 8: Close freezer	Task: Put away groceries Step 1: Walk to kitchen Step 2: Walk to refrigerator Step 3: Open refrigerator Step 4: Grab milk and eggs Step 5: Close refrigerator Step 6: Walk to counter Step 7: Put milk and eggs on counter Step 8: Walk to cabinet Step 9: Open cabinet Step 10: Put away milk and eggs Step 11: Close cabinet Step 12: Walk to refrigerator Step 13: Grab orange juice and bread Step 14: Close refrigerator	Task: Put away groceries Step 1: Walk to kitchen Step 2: Walk to fridge Step 3: Open fridge Step 4: Put food in fridge Step 5: Close fridge Step 6: Walk to dining room Step 7: Walk to livingroom Step 8: Walk to bedroom Step 9: Walk to bathroom

Task: Draw picture Step 1: Walk to home office Step 2: Walk to chair Step 3: Find chair Step 4: Sit on chair Step 5: Find desk Step 6: Open desk Step 7: Find check Step 8: Grab check Step 9: Put check on desk Step 10: Find pencil Step 11: Grab pencil Step 12: Close desk	Task: Draw picture Step 1: Walk to home office Step 2: Walk to pencils Step 3: Find pencils Step 4: Grab pencils Step 5: Find paper Step 6: Sit on chair Step 7: Draw picture	Task: Draw picture Step 1: Walk to home office Step 2: Walk to wall Step 3: Walk to drawing Step 4: Find drawing Step 5: Turn to drawing Step 6: Look at drawing Step 7: Touch drawing Step 8: Pull drawing Step 9: Push drawing
Task: Entertain Step 1: Walk to home office Step 2: Walk to cheese Step 3: Find cheese Step 4: Find food Step 5: Find couch Step 6: Sit on couch Step 7: Find wine glass Step 8: Grab wine glass Step 9: Drink wine glass	Task: Entertain Step 1: Walk to living room Step 2: Walk to entertainment center Step 3: Find remote Step 4: Grab remote Step 5: Find chair Step 6: Sit on chair Step 7: Entertain self	Task: Entertain Step 1: Walk to livingroom Step 2: Walk to couch Step 3: Find couch Step 4: Sit on couch
Task: Type up document Step 1: Walk to home office Step 2: Walk to chair Step 3: Find chair Step 4: Sit on chair Step 5: Find computer Step 6: Switch on computer Step 7: Turn to computer Step 8: Look at computer Step 9: Find check Step 10: Grab check Step 11: Read check Step 12: Find coffee cup Step 13: Grab coffee cup Step 14: Drink coffee cup Step 15: Put back coffee cup Step 16: Find keyboard Step 17: Type on keyboard	Task: Type up document Step 1: Walk to home office Step 2: Walk to keyboard Step 3: Find keyboard Step 4: Grab keyboard Step 5: Find chair Step 6: Sit on chair Step 7: Type document	Task: Type up document Step 1: Walk to home office Step 2: Walk to computer Step 3: Find computer Step 4: Switch on computer Step 5: Find chair Step 6: Sit on chair Step 7: Turn to computer Step 8: Look at computer Step 9: Find keyboard Step 10: Grab keyboard Step 11: Type on keyboard Step 12: Stand up Step 13: Find printer Step 14: Switch on printer Step 15: Find printing paper
Task: Put on your shoes Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes Step 4: Grab shoes Step 5: Find chair Step 6: Sit on chair Step 7: Find feet both Step 8: Put shoes on feet both	Task: Put on your shoes Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes Step 4: Grab shoes Step 5: Find chair Step 6: Sit on chair Step 7: Put on shoes	Task: Put on your shoes Step 1: Find shoes Step 2: Turn to shoes Step 3: Point at shoes Step 4: Wash shoes

Task: Watch horror movie Step 1: Walk to home office Step 2: Walk to remote control Step 3: Find remote control Step 4: Grab remote control Step 5: Find couch Step 6: Sit on couch Step 7: Find television Step 8: Switch on television Step 9: Turn to television Step 10: Watch television	Task: Watch horror movie Step 1: Walk to home office Step 2: Walk to horror movie Step 3: Find horror movie Step 4: Grab horror movie Step 5: Find chair Step 6: Sit on chair Step 7: Watch horror movie	Task: Watch horror movie Step 1: Find television Step 2: Switch on television Step 3: Find cd player Step 4: Switch on cd player Step 5: Switch off cd player Step 6: Find remote control Step 7: Switch on remote control Step 8: Switch off remote control
Task: Turn on radio Step 1: Walk to home office Step 2: Walk to stereo Step 3: Find stereo Step 4: Find electrical outlet Step 5: Turn to electrical outlet Step 6: Plug in stereo Step 7: Switch on electrical outlet Step 8: Switch on stereo	Task: Turn on radio Step 1: Walk to home office Step 2: Walk to radio Step 3: Find radio Step 4: Grab radio Step 5: Find chair Step 6: Sit on chair Step 7: Turn on radio	Task: Turn on radio Step 1: Walk to home office Step 2: Walk to chair Step 3: Find chair Step 4: Sit on chair Step 5: Find remote control Step 6: Grab remote control Step 7: Find radio Step 8: Point at radio Step 9: Switch on radio Step 10: Put back remote control
Task: Look at mirror Step 1: Walk to bathroom Step 2: Walk to mirror Step 3: Find mirror Step 4: Turn to mirror Step 5: Look at mirror	Task: Look at mirror Step 1: Walk to bathroom Step 2: Walk to mirror Step 3: Look in mirror	Task: Look at mirror Step 1: Walk to bedroom Step 2: Walk to mirror Step 3: Find mirror Step 4: Turn to mirror Step 5: Look at mirror Step 6: Find face Step 7: Turn to face Step 8: Look at face Step 9: Find hair Step 10: Turn to hair Step 11: Look at hair Step 12: Find dress Step 13: Turn to dress Step 14: Look at dress
Task: Leave home Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes Step 4: Grab shoes Step 5: Put on shoes	Task: Leave home Step 1: Walk to front door Step 2: Open front door Step 3: Walk to car Step 4: Open car door Step 5: Get in car Step 6: Close car door Step 7: Drive to work	Task: Leave home Step 1: Find shoes Step 2: Grab shoes Step 3: Put on shoes
Task: Walk to room Step 1: Walk to home office Step 2: Walk to home office	Task: Walk to room Step 1: Walk to home office Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find chair Step 6: Sit on chair Step 7: Read novel	Task: Walk to room Step 1: Walk to dining room Step 2: Walk to livingroom Step 3: Walk to bedroom

Task: Hang pictures Step 1: Walk to bedroom Step 2: Walk to drawing Step 3: Find drawing Step 4: Grab drawing Step 5: Find wall Step 6: Put drawing on wall Step 7: Turn to drawing Step 8: Look at drawing Step 9: Find drawing Step 10: Grab drawing Step 11: Put drawing on wall Step 12: Turn to drawing Step 13: Look at drawing	Task: Hang pictures Step 1: Walk to home office Step 2: Walk to box of pictures Step 3: Find box of pictures Step 4: Grab box of pictures Step 5: Walk to wall Step 6: Hang pictures on wall	Task: Hang pictures Step 1: Walk to home office Step 2: Walk to table Step 3: Find keys Step 4: Grab keys Step 5: Walk to wall Step 6: Find hanger Step 7: Put keys on hanger
Task: Wipe down sink Step 1: Walk to dining room Step 2: Walk to rag Step 3: Find rag Step 4: Grab rag Step 5: Walk to sink Step 6: Wipe sink Step 7: Put back rag	Task: Wipe down sink Step 1: Walk to kitchen Step 2: Walk to sink Step 3: Grab sponge Step 4: Wipe down sink Step 5: Rinse sponge Step 6: Put sponge away Step 7: Wash hands	Task: Wipe down sink Step 1: Walk to bathroom Step 2: Walk to cleaning solution Step 3: Find cleaning solution Step 4: Grab cleaning solution Step 5: Find sink Step 6: Pour cleaning solution into sink Step 7: Find faucet Step 8: Switch on faucet Step 9: Scrub sink Step 10: Wash sink
Task: Take shoes off Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes Step 4: Find shoes Step 5: Take off shoes Step 6: Find dresser Step 7: Open dresser Step 8: Grab shoes Step 9: Put shoes in dresser Step 10: Close dresser	Task: Take shoes off Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes Step 4: Grab shoes Step 5: Find chair Step 6: Sit on chair Step 7: Take shoes off	Task: Take shoes off Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes
Task: Give milk to cat Step 1: Walk to home office Step 2: Walk to cat Step 3: Find cat Step 4: Turn to cat Step 5: Point at cat Step 6: Pull cat Step 7: Find milk Step 8: Grab milk Step 9: Find plate Step 10: Pour milk into plate Step 11: Touch cat Step 12: Scrub cat	Task: Give milk to cat Step 1: Walk to kitchen Step 2: Walk to fridge Step 3: Open fridge Step 4: Grab milk Step 5: Walk to cat Step 6: Pour milk into bowl Step 7: Put bowl on floor Step 8: Walk away from cat	Task: Give milk to cat Step 1: Find cat Step 2: Touch cat Step 3: Open fridge Step 4: Touch milk Step 5: Pour milk into bowl

Task: Grab things Step 1: Walk to novel Step 2: Turn to novel Step 3: Grab novel	Task: Grab things Step 1: Walk to home office Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find chair Step 6: Sit on chair Step 7: Read novel Step 8: Grab things	Task: Grab things Step 1: Find toy Step 2: Turn to toy Step 3: Point at toy Step 4: Wash toy
Task: Use laptop Step 1: Walk to home office Step 2: Walk to couch Step 3: Find couch Step 4: Sit on couch Step 5: Find laptop Step 6: Switch on laptop Step 7: Turn to laptop Step 8: Look at laptop Step 9: Find keyboard Step 10: Type on keyboard Step 11: Turn to keyboard Step 12: Look at keyboard Step 13: Type on keyboard Step 14: Turn to laptop Step 15: Look at laptop	Task: Use laptop Step 1: Walk to home office Step 2: Walk to laptop Step 3: Find laptop Step 4: Grab laptop Step 5: Find chair Step 6: Sit on chair Step 7: Use laptop	Task: Use laptop Step 1: Walk to home office Step 2: Walk to chair Step 3: Find chair Step 4: Sit on chair Step 5: Find laptop Step 6: Switch on laptop Step 7: Find mouse Step 8: Grab mouse Step 9: Find mousepad Step 10: Put mouse on mousepad Step 11: Turn to laptop Step 12: Point at laptop
Task: Organize pantry Step 1: Walk to dining room Step 2: Walk to pantry Step 3: Find food Step 4: Grab food Step 5: Turn to food Step 6: Look at food Step 7: Find garbage can Step 8: Put food on garbage can Step 9: Find vegetable Step 10: Grab vegetable Step 11: Turn to vegetable Step 12: Look at vegetable Step 13: Put vegetable on garbage can Step 14: Find dry pasta Step 15: Grab dry pasta Step 16: Turn to dry pasta Step 17: Look at dry pasta Step 18: Put back dry pasta Step 19: Find food Step 20: Turn to food Step 21: Look at food Step 22: Push food Step 23: Find noodles Step 24: Grab noodles Step 25: Turn to noodles Step 26: Look at noodles Step 27: Put back noodles	Task: Organize pantry Step 1: Walk to kitchen Step 2: Walk to pantry Step 3: Find pantry Step 4: Grab pantry Step 5: Find chair Step 6: Sit on chair Step 7: Organize pantry	Task: Organize pantry Step 1: Walk to pantry Step 2: Find pantry

Task: Change clothes Step 1: Walk to bedroom Step 2: Walk to dresser Step 3: Open dresser Step 4: Find dress Step 5: Grab dress Step 6: Put on dress Step 7: Close dresser	Task: Change clothes Step 1: Walk to bedroom Step 2: Walk to closet Step 3: Find clothes Step 4: Grab clothes Step 5: Find chair Step 6: Sit on chair Step 7: Change clothes	Task: Change clothes Step 1: Walk to bedroom Step 2: Walk to dresser Step 3: Find dresser Step 4: Open dresser Step 5: Find pants Step 6: Grab pants Step 7: Put on pants Step 8: Find shirt Step 9: Grab shirt Step 10: Put on shirt Step 11: Close dresser
Task: Put up decoration Step 1: Walk to home office Step 2: Walk to drawing Step 3: Find drawing Step 4: Find centerpiece Step 5: Grab centerpiece Step 6: Walk to wall Step 7: Grab drawing Step 8: Put drawing on wall Step 9: Find centerpiece Step 10: Grab centerpiece Step 11: Walk to table Step 12: Put centerpiece on table	Task: Put up decoration Step 1: Walk to home office Step 2: Walk to decoration Step 3: Find decoration Step 4: Grab decoration Step 5: Find chair Step 6: Sit on chair Step 7: Put up decoration	Task: Put up decoration Step 1: Walk to livingroom Step 2: Walk to table
Task: Make bed Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Find sheets Step 4: Grab sheets Step 5: Pull sheets	Task: Make bed Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Find sheets Step 4: Find blanket Step 5: Find pillow Step 6: Put sheets on bed Step 7: Put blanket on bed Step 8: Put pillow on bed	Task: Make bed Step 1: Stand up Step 2: Find sheets Step 3: Find blanket Step 4: Find pillow Step 5: Find comforter Step 6: Find bed
Task: Scrubbing living room tile floor is once week activity for me Step 1: Find mop Step 2: Turn to mop Step 3: Point at mop Step 4: Wash mop Step 5: Turn to mop Step 6: Look at mop	Task: Scrubbing living room tile floor is once week activity for me Step 1: Walk to living room Step 2: Walk to tile floor Step 3: Find scrub brush Step 4: Grab scrub brush Step 5: Scrub tile floor Step 6: Put down scrub brush Step 7: Walk to kitchen	Task: Scrubbing living room tile floor is once week activity for me Step 1: Walk to livingroom Step 2: Walk to mop Step 3: Find mop Step 4: Grab mop Step 5: Walk to bathroom Step 6: Find bucket Step 7: Switch on faucet
Task: Read yourself to sleep Step 1: Walk to bedroom Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find bed Step 6: Sit on bed Step 7: Find bookmark Step 8: Grab bookmark Step 9: Read novel Step 10: Put bookmark on novel Step 11: Sleep	Task: Read yourself to sleep Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Find book Step 4: Grab book Step 5: Find chair Step 6: Sit on chair Step 7: Read book Step 8: Sleep	Task: Read yourself to sleep Step 1: Walk to bedroom Step 2: Walk to bed Step 3: Lie on bed Step 4: Sleep Step 5: Read book

Task: Hand washing Step 1: Walk to bathroom Step 2: Walk to sink Step 3: Find faucet Step 4: Switch on faucet Step 5: Find soap Step 6: Grab soap Step 7: Find hands both Step 8: Scrub hands both Step 9: Put back soap Step 10: Wash hands both Step 11: Switch off faucet Step 12: Find towel Step 13: Grab towel Step 14: Wipe hands both	Task: Hand washing Step 1: Walk to bathroom Step 2: Walk to sink Step 3: Turn on faucet Step 4: Grab soap Step 5: Wash hands Step 6: Rinse hands Step 7: Dry hands	Task: Hand washing Step 1: Walk to bathroom Step 2: Walk to sink Step 3: Find faucet Step 4: Turn to faucet Step 5: Find hands both Step 6: Wash hands both
Task: Throw away paper Step 1: Walk to home office Step 2: Walk to desk Step 3: Find desk Step 4: Turn to desk Step 5: Find chair Step 6: Sit on chair Step 7: Find check Step 8: Grab check Step 9: Squeeze check Step 10: Stand up Step 11: Walk to trashcan Step 12: Put check on trashcan	Task: Throw away paper Step 1: Walk to home office Step 2: Walk to wastebasket Step 3: Find wastebasket Step 4: Grab wastebasket Step 5: Walk to desk Step 6: Drop paper in wastebasket	Task: Throw away paper Step 1: Walk to home office Step 2: Walk to table Step 3: Find table Step 4: Turn to table Step 5: Find paper Step 6: Grab paper Step 7: Walk to trashcan Step 8: Open trashcan Step 9: Put paper on trashcan Step 10: Close trashcan
Task: Tale off shoes Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes Step 4: Find shoes Step 5: Take off shoes Step 6: Grab shoes Step 7: Find mat Step 8: Put shoes on mat	Task: Tale off shoes Step 1: Walk to home office Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find chair Step 6: Sit on chair Step 7: Read novel	Task: Tale off shoes Step 1: Walk to home office Step 2: Walk to shoes Step 3: Find shoes
Task: Hang keys Step 1: Walk to bedroom Step 2: Walk to dresser Step 3: Find purse Step 4: Grab purse Step 5: Open purse Step 6: Find keys Step 7: Grab keys Step 8: Find bowl Step 9: Put keys on bowl Step 10: Close purse Step 11: Put back purse	Task: Hang keys Step 1: Walk to home office Step 2: Walk to key hook Step 3: Find key hook Step 4: Grab key hook Step 5: Find keys Step 6: Hang keys on key hook Step 7: Walk to front door Step 8: Leave house	Task: Hang keys Step 1: Walk to home office Step 2: Walk to table Step 3: Find keys Step 4: Grab keys Step 5: Walk to wall Step 6: Find hanger Step 7: Put keys on hanger

Task: Make popcorn Step 1: Walk to dining room Step 2: Walk to food Step 3: Find food Step 4: Grab food Step 5: Walk to microwave Step 6: Turn to microwave Step 7: Open microwave Step 8: Put food in microwave Step 9: Close microwave Step 10: Switch on microwave	Task: Make popcorn Step 1: Walk to kitchen Step 2: Walk to microwave Step 3: Open microwave Step 4: Put bag of popcorn in microwave Step 5: Close microwave Step 6: Push buttons on microwave Step 7: Wait for popcorn to pop Step 8: Get bowl out of cupboard Step 9: Get butter out of fridge Step 10: Get salt out of cupboard Step 11: Pour popcorn into bowl Step 12: Add butter and salt to bowl Step 13:	Task: Make popcorn Step 1: Walk to kitchen Step 2: Walk to microwave Step 3: Find microwave Step 4: Open microwave
Task: Receive credit card Step 1: Walk to home office Step 2: Walk to envelope Step 3: Find envelope Step 4: Grab envelope Step 5: Walk to home office Step 6: Walk to scissors Step 7: Find scissors Step 8: Open envelope Step 9: Find creditcard Step 10: Grab creditcard	Task: Receive credit card Step 1: Walk to mailbox Step 2: Open mailbox Step 3: Grab credit card Step 4: Walk to home office Step 5: Walk to desk Step 6: Put credit card on desk Step 7: Sit at desk Step 8: Look at credit card	Task: Receive credit card Step 1: Walk to home office Step 2: Walk to cabinet Step 3: Find cabinet Step 4: Open cabinet
Task: Read to child Step 1: Walk to bedroom Step 2: Walk to child Step 3: Find child Step 4: Greet child Step 5: Walk to filing cabinet Step 6: Open filing cabinet Step 7: Find novel Step 8: Grab novel Step 9: Walk to chair Step 10: Pull chair Step 11: Sit on chair Step 12: Turn to novel Step 13: Look at novel Step 14: Turn to novel Step 15: Point at novel Step 16: Turn to novel Step 17: Point at novel Step 18: Read novel Step 19: Read novel Step 20: Read novel	Task: Read to child Step 1: Walk to home office Step 2: Walk to novel Step 3: Find novel Step 4: Grab novel Step 5: Find chair Step 6: Sit on chair Step 7: Read novel	Task: Read to child Step 1: Walk to child Step 2: Read book

Task: Eat cheese Step 1: Walk to dining room Step 2: Walk to freezer Step 3: Open freezer Step 4: Find cheese Step 5: Grab cheese Step 6: Eat cheese	Task: Eat cheese Step 1: Walk to kitchen Step 2: Walk to refrigerator Step 3: Open refrigerator Step 4: Grab cheese Step 5: Eat cheese	Task: Eat cheese Step 1: Walk to dining room Step 2: Walk to table Step 3: Find table Step 4: Turn to table Step 5: Find chair Step 6: Sit on chair Step 7: Find food Step 8: Grab food Step 9: Find plate Step 10: Put food on plate
Task: Unload various items from pockets and place them in bowl on table Step 1: Walk to home office Step 2: Walk to table Step 3: Find bowl Step 4: Grab bowl Step 5: Put bowl on table Step 6: Find pants Step 7: Find bills Step 8: Grab bills Step 9: Find phone Step 10: Grab phone	Task: Unload various items from pockets and place them in bowl on table Step 1: Walk to home office Step 2: Walk to bowl on table Step 3: Find bowl on table Step 4: Grab bowl on table Step 5: Unload various items from pockets and place them in bowl on table	Task: Unload various items from pockets and place them in bowl on table Step 1: Find bowl Step 2: Turn to bowl Step 3: Point at bowl Step 4: Wash bowl
Task: Switch on lamp Step 1: Walk to bedroom Step 2: Walk to floor lamp Step 3: Find floor lamp Step 4: Switch on floor lamp	Task: Switch on lamp Step 1: Walk to lamp Step 2: Switch on lamp Step 3: Walk to chair Step 4: Sit on chair Step 5: Read book	Task: Switch on lamp Step 1: Walk to bedroom Step 2: Walk to floor lamp Step 3: Find floor lamp Step 4: Turn to floor lamp Step 5: Switch on floor lamp Step 6: Find bed Step 7: Lie on bed
Task: Take jacket off Step 1: Walk to bedroom Step 2: Walk to jacket Step 3: Find jacket Step 4: Find jacket Step 5: Take off jacket Step 6: Find dresser Step 7: Open dresser Step 8: Close dresser	Task: Take jacket off Step 1: Walk to home office Step 2: Walk to jacket Step 3: Find jacket Step 4: Grab jacket Step 5: Find chair Step 6: Sit on chair Step 7: Take jacket off	Task: Take jacket off Step 1: Walk to home office Step 2: Walk to jacket Step 3: Find jacket