To Skip, to Swap or to not Swap? Identifying Step Transition Types in Instructional Manuals

Hsiu-Yu Yang[1], **Michael Roth**[2], **Andreas Bulling**[1], **Carina Silberer**[1]
[1]University of Stuttgart [2]University of Technology Nuremberg
hsiuyu.yang.akd@gmail.com michael.roth@utn.de
andreas.bulling@vis.uni-stuttgart.de carina.silberer@ims.uni-stuttgart.de

Abstract

Large language models (LLMs) are increasingly used as procedural planners that provide guidance across applications. However, in human-assistive scenarios where the environment and users' knowledge constantly change, their ability to detect various step types for generating alternative plans is underexplored. To address this gap, we introduce a novel evaluation task and dataset to assess if models can identify steps that are *sequential*, *interchangeable*, and *optional* in textual instructions across five domains in a step-by-step manner. We compare seven LLM families from both open-source and proprietary spaces across varying sizes to a visually-informed baseline based on procedural knowledge graphs (PKG). Our results suggest that LLMs encode procedural knowledge, enabling them to identify step types with increasing effectiveness as training parameters and data size grow. However, all LLMs exhibit inconsistencies in reasoning on the mutual exclusivity of interchangeable and sequential step pairs. In contrast, the symbolic PKG baseline demonstrates stronger consistency in this aspect. Comprehensive analyses furthermore uncover limitations in LLMs' procedural reasoning abilities.

Keywords: procedural knowledge, evaluation, knowledge graph construction, transformers

1. Introduction

Large language models (LLMs) have shown impressive performance in abstract planning in various scenarios, from classical planning problems (Valmeekam et al., 2023b,a; Guan et al., 2023) and embodied household tasks (Song et al., 2022; Lin et al., 2023) to real-world human-assistive setups (Patel et al., 2023). Yet, in situated assistive contexts, where users' environment and knowledge constantly evolve, it is essential that AI agents have the ability to flexibly adapt plans (Bao et al., 2023).

For example, consider the task "Growing Geraniums" (Fig. 1): In some climates, planting can be done outdoors year-round. In this context, picking the right time to plant the flowers (Step 2) can safely be skipped. This simple variation illustrates a broader challenge: without the ability to detect and reason about such flexible step relationships, even advanced AI agents risk offering plans that are rigid, inefficient, or contextually inappropriate.

Achieving such adaptability requires a nuanced understanding of how steps in a task relate to one another, including types of transitions such as strict **sequential** dependencies, **interchangeable** actions, or **optional** step skipping. Accurately identifying these relationships is critical: it enables flexibility, allowing the task order to adapt to the current context; it ensures safety, by preserving steps that must remain sequential; and it supports personalization, by omitting or reordering steps when appropriate.

A common resource for enabling such understanding are procedural knowledge graphs (PKGs),

Figure 1: We evaluate whether models can identify various step types in instructional manuals, namely **sequential** ordering, **interchangeable** execution, and **optional** step skipping.

which represent tasks as networks of interconnected steps linked by semantic and temporal relations. By making relations explicit, PKGs provide a structured foundation to reason about variability and hence support dynamic reordering, substitution, or omission of steps as conditions change.

While graphs typically do not generalize across different procedures, LLMs could provide a complementary, data-driven perspective based on the availability of procedural texts on a large scale. Yet, their potential for modeling procedural relationships beyond surface-level instruction following remains largely underexplored. Examining this ability, however, is vital for assessing models' usefulness as trustworthy, dynamic guides in procedural tasks. It also taps on their physical knowledge of

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 9301–9320
11–16 May 2026.

the pre/postconditions of actions (Wu et al., 2023a; Brahman et al., 2023), and thus on their ability to reason about object state changes (Tandon et al., 2020) and resolve references (Anthonio and Roth, 2021; Rim et al., 2023).

In response to this research gap, we propose a procedural **step type identification task** in a text-based setting. We create the **S**equential-**I**nterchangeable-**O**ptional (SIO) dataset that contains step type annotations of procedural tasks across five domains featuring physical activities from WikiHow (Koupaee and Wang, 2018), an online instructional resource for everyday tasks. We formulate each step type identification task—sequential, interchangeable, and optional—as a binary classification problem, and examine various LLM families as well as a visually-aware procedural knowledge graph on the task. In our evaluation, we furthermore assess key aspects of procedural knowledge and reasoning in real-world scenarios, namely domain coverage (knowledge breadth), expert knowledge (knowledge depth), instruction compliance, and awareness of entity states, which put constraints on the order of steps. Our contributions are as follows:

- We introduce an evaluation framework (SIO), comprising a task definition and a dataset,[1] to benchmark models on their ability to identify sequential, interchangeable and optional step transition types in procedural instructions.
- We evaluate 11 advanced LLMs of seven different families and different scales, as well as a visually-aware heuristic PKG-based system.
- We report comprehensive analyses of the LLMs in terms of their knowledge depth and breadth, their adherence to instructions, and examine their rationales of correct step type classifications with respect to their ability to track entities.

Our results show that the SIO task is challenging for all models, particularly in identifying interchangeable and optional types. Model scale and training data size prove to be key factors for *procedural knowledge*. The most effective LLMs even suggest an awareness of entity states. However, inconsistencies on the mutually exclusive interchangeable and sequential pairs, as well as failures in task-specific instruction compliance, reveal significant limitations in the LLMs' *procedural reasoning* abilities, where the PKG shows an advantage.

[1]The SIO dataset is available at https://github.com/Mallory24/SIO_dataset/tree/main.

2. Related Work

Procedural Knowledge Modeling and Evaluation. Procedural knowledge entails comprehending a goal along with its feasible steps for achieving it, encompassing various levels of granularity and forms of reasoning (Zhang, 2022). We focus our discussion on instructional text as it is a common form for studying procedural information. On the hierarchical dimension, some works study goal–subgoal relationships (Zhou et al., 2022), goal–step inference (Yang et al., 2021) and cross-task generalization (Zhou et al., 2023a). The horizontal axis, i.e., step–step relations, has received extensive research attention, especially in understanding temporal step ordering (Zhang et al., 2020; Wu et al., 2022; Lal et al., 2024) and causal dependencies (Jang et al., 2023). Various aspects of reasoning about individual steps have been widely studied, including action condition inference (Wu et al., 2023a), co-reference resolution (Anthonio and Roth, 2021; Rim et al., 2023), and entity and entity state tracking (Tandon et al., 2020; Wu et al., 2023b; Kim and Schuster, 2023; Zhang et al., 2024). Our contribution is to advance research on understanding step–step relationships, focusing on **sequential order, interchangeable execution, and optional steps** (Zhou et al., 2023b). These are crucial aspects in assessing models on their physical reasoning capabilities, and have practical applications in generating alternative step sequences.

Procedural Knowledge Graph (PKG) Construction. A PKG effectively captures metadata, entity information and goal–step hierarchies, making it useful for representing complex procedural processes (Zhang, 2022). Various techniques have been employed to create PKGs, each focusing on a distinct aspect. Sakaguchi et al. (2021) use a directed acyclic graph to encode partial ordering of events for script generation. Jang et al. (2023) annotated step conditions (e.g., "completed") to train a graph generation model for capturing causal dependencies. Zhang et al. (2022) model the temporal and cross-modal evolution of entities for machine reading comprehension. Various works have explored multimodal grounding between instructional texts and videos to create visually-aware PKGs that represent diverse task demonstrations and are useful for downstream tasks. Zhou et al. (2023b) extract multiple step sequences for non-sequentiality acquisition in a path generation model. Ashutosh et al. (2023) use a PKG as a probabilistic prior for key step recognition, while Zhou et al. (2023a) use it to generate pseudo-labels to enable cross-task step recognition. Our work differs from prior works by inducing step types through a heuristic algorithm applied to a visually-aware probablisitic PKG.

3. Task Definitions

We focus on three step types—sequential, interchangeable and optional—that can occur in a given procedural task t consisting of ordered steps $(s_0, s_1 \ldots, s_{|t|-1})$. We frame step type identification as a *binary classification task*, where each decision considers either the relationship between adjacent steps or individual steps within a localized context window. This approach supports dynamic, real-time assistance in procedural tasks by allowing models to make incremental decisions grounded in local information. At the same time, it still incorporates a global perspective by conditioning each decision on the overall task completion, as we explain below. Although this formulation is limited to step pairs, broader step relationships can be inferred through transitive reasoning, making it a practical and scalable evaluation strategy. For instance, if steps (s_2, s_3) and (s_3, s_4) are sequential, we can infer the sequentiality of (s_2, s_4).

Sequential. We define an adjacent step pair (s_i, s_{i+1}) as sequential if maintaining their order is necessary for successful task completion. The context window is limited to these two steps. The potential sequential step pairs form a sequence $[(s_0, s_1), ..., (s_{|t|-2}, s_{|t|-1})]$ of at most $|t| - 1$ pairs.

Interchangeable. At any given step s_i, a step pair $(s_{i+1},\ s_{i+2})$ is defined as interchangeable if s_{i+1} and s_{i+2} can be executed in any order without impacting task completion. In this case, s_i acts as the anchor step, and steps beyond its succeeding two-step context are not considered. Thus, starting from the first step s_0 as the anchor, a sequence of $[(s_1, s_2), ..., (s_{|t|-2}, s_{|t|-1})]$, totaling at most $|t| - 2$ pairs, may be interchangeable.

Optional. The optionality of a step is also determined within a two-step context. Specifically, for a step s_i, the next step s_{i+1} is deemed optional if one can skip it and directly move to s_{i+2} without affecting task completion. Beginning with anchor step s_0, optional steps may form a sequence $[s_1, s_2, ..., s_{|t|-2}]$ with at most $|t|-2$ optional steps.

4. Human Annotated SIO Dataset

We create a novel *Sequential, Interchangeable, Optional* (SIO) dataset with manual step type annotations for data in WikiHow, a large online instructional resource for everyday tasks[2], following the definitions in §3. We focus on five domains—*Food and Entertaining*, *Home and Garden*, *Hobbies and Crafts*, *Cars and Other Vehicles*, and *Computers and Electronics*—each featuring step transitions with strong logical dependencies. For example, "clear the soil of weeds" naturally precedes "add compost to the soil". Other domains like *Pets and Animals* involve mostly independent activities (e.g., "kill fleas" and "remove visible ticks"; Zhang et al. 2020) and are therefore less suitable for our study.

Specifically, we use the WikiHowClean dataset introduced by Zhang et al. (2020). It is a curated subset of WikiHow articles where a title is merged with each section name, forming what we call *task-parts*, e.g., *How to Grow and Care for Geraniums – Planting Your Geraniums*. WikiHowClean primarily comprises ordered task-parts with strong inter-step causal or temporal dependencies,[3] but it still contains flexible step transitions allowing step reordering or optional execution. The data hence represents a challenging yet valuable scenario for finding alternative executions for a given task.

Annotation Procedure. We sample 298 task-parts from WikiHowClean, 15 for the dev set and 283 for the test set. Two domains (*Cars* and *Computers*) are unseen in the dev set. We overall recruited 21 annotators, all students at our university, who were compensated with €13 per hour. Details to recruitment as well as the annotation guidelines are given in Appendix A.1.

For each procedural task, we ask three annotators to select the steps (for the optional type) or step tuples (for sequential or interchangeable step pairs) that satisfy the respective definitions described in §3. Moreover, for each selected step or step tuple as well as for the overall procedural task, annotators are instructed to indicate if expert knowledge is required, basing their judgment on general public perception rather than personal experience. If expert knowledge is not required, we mark the step (tuple) or task, respectively, as relying on commonsense knowledge, even when the task itself is unfamiliar. For example, to "Fix a Laundry Machine", no specialized expertise is required to know that "opening the door" must precede "unloading the laundry."

The SIO Dataset. For each step type, we assign a positive label for the majority-voted (2 out of 3) step pairs/steps and a negative label for the rest of the cases. In addition, we aggregate the corresponding majority-assigned expert-knowledge labels for the

[2] Available at www.wikihow.com

[3] Zhang et al. (2020) fine-tuned RoBERTa to automatically label the order of task-parts within WikiHow. The prediction file can be found here.

(A) How to Make Huevos Rancheros – Assembling the Dish (*Food & Entertaining*)		
S_0: Spread about 3 oz (85 g) of refried beans onto each of the tortillas	sequential:	[(1, 2)]
	sequ._expert:	[No]
S_1: Place 1 cooked egg onto each of the tortillas	interchangeable:	[(2, 3)]
	interch._expert:	[No]
S_2: Pour warm salsa over the eggs	optional:	[2]
S_3: Top the dish with avocado, lime juice, cilantro, cheese, or sour cream	optional_expert:	[No]
	task_expert:	No
(B) How to Clean Vintage Stereo Equipment – Cleaning the Equipment *(Computers & Electronics)*		
S_0: Unplug your stereo equipment	sequential:	[(0, 1), (1, 2), (5, 6), (6, 7)]
S_1: Remove the cover of the stereo component		
S_2: Spray compressed air throughout the unit's interior if needed	sequ._expert:	[No, No, No, No]
S_3: Spray contact cleaner on parts to be cleaned sparingly	interchangeable:	[(2, 3), (3, 4)]
S_4: Clean the potentiometers with contact cleaner	interch._expert:	[No, No]
S_5: Allow the unit to air out for several hours	optional:	[2]
S_6: Replace the component's case	optional_expert:	[No]
	task_expert:	No

Table 1: Two example task-parts of our SIO test set, consisting of a task's individual steps (left) and step (pair) annotations (right). The domain of each task is given in parentheses.

positive cases.[4] Table 1 shows examples of the SIO dataset, and Table 2 gives the dataset size. The label distribution across task domains is given in Table 5 in Appendix A.2. For the number of test instances per domain and step type, see Figure 3.

An analysis of the SIO dataset reveals that 91% of the step tuples labeled as SEQUENTIAL are never INTERCHANGEABLE, suggesting a mutual exclusivity between the these two step types. Moreover, nearly all INTERCHANGEABLE and OPTIONAL cases do not require expert knowledge, indicating that annotators mainly rely on commonsense knowledge to determine if a step pair/step belongs to these categories.

The average pairwise inter-annotator-agreements (IAA) for each step type range from slight to fair ($\kappa = .15 - .37$) on the dev set and fair to moderate on the test set ($\kappa = .26 - .45$). For the IAA across each domain, see Table 6 in Appendix A.3. Upon examination, we found that the IAA is lower for cases that involve domain-specific knowledge. For instance, on SEQUENTIAL, the *Car* domain has a lower IAA than the *Food* domain ($\kappa = .17$ vs. $\kappa = .29$), and has more cases that require expert-level knowledge (80 vs. 5) (cf. Fig. 5 in App. A.3).

5. Visual Step Type Inference

Online video demonstrations offer a valuable resource to explore alternative ways to realize a task. We build upon several works (Zhou et al., 2023a,b; Ashutosh et al., 2023) on constructing visually licensed procedural knowledge graphs (PKGs) by grounding tasks of the (text-based) procedural repository WikiHow (Koupaee and Wang, 2018) in (vision-based) instructional videos of HowTo100M (Miech et al., 2019b). In our case, we build topic-specific PKGs that represent the likelihood of individual step transitions for a task (§5.1). We then leverage this information to automatically infer step types for sequential order, interchangeable execution, and optional skipping (§5.2). Figure 2 shows the overall inference pipeline.

[4] If only two annotators mark a step tuple as true for a given type, and their judgments conflict at the expert-knowledge level, we label the expert knowledge requirement as `Unsure`.

	Dev (15 tasks)			Test (283 tasks)		
Step Type	pos.	neg.	all	pos.	neg.	all
SEQUENTIAL	44	20	64	980	78	1278
INTERCHANG.	11	38	49	149	846	995
OPTIONAL	7	42	49	98	897	995

Table 2: Instances of pos(itive) and neg(ative) step types across the dataset splits.

5.1. Video-mined PKG Construction

Data Resources. We use WikiHowClean as our underlying text-based procedural repository. As video resource, we use CAE (Yang and Silberer, 2023). It is a condensed set of HowTo100M which contains text–video-clip pairs that capture visually perceivable effect-causing actions. This promises a cleaner PKG construction than directly using HowTo100M.

Topic Clustering. To find sensible alterations in step orders, we first group similar tasks across the

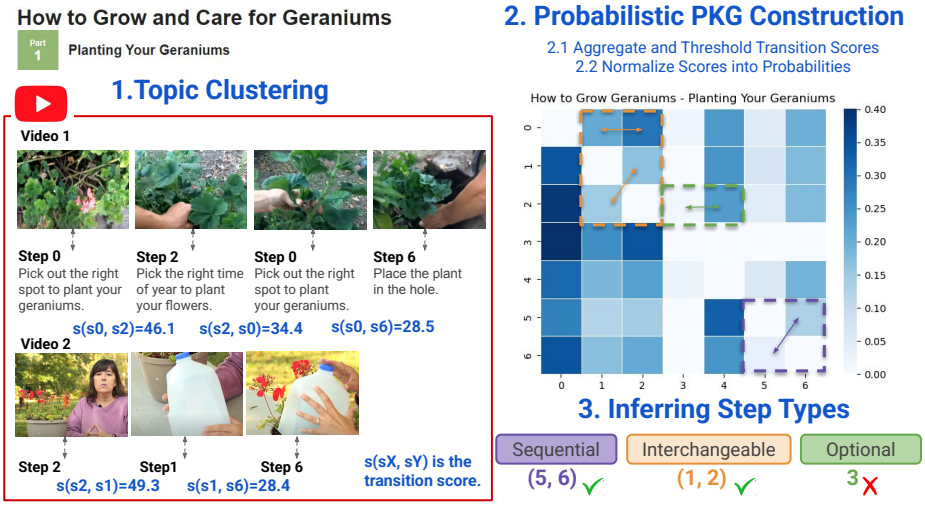

Figure 2: Overview of the PKG construction and step type inference, based on the example in Figure 1. It consists of three major steps: (i) topic clustering to group tasks, (ii) topic-wise PKG construction, and (iii) step-type inference based on the PKG. For simplicity, we only show the top aligned steps for each video segment and the step descriptions once.

two resources, e.g., *Grow Geraniums Indoors* and *Grow Geraniums in Pots*: We extract sentence representations of task titles in both WikiHowClean and CAE, and use agglomerative clustering to group them into topics.

Probabilistic PKG Construction. For each instructional topic X, we consider two inputs: (1) a key step library K^X sourced from a *single* task-parts t of WikiHowClean, which contains a sequence of step headlines $(s_0^t, ..., s_j^t, ..., s_{|t|}^t)$; (2) a video set V^X collected from CAE, where each $V_x = (v_1, ..., v_i, ..., v_{|V_x|}) \in V^X$ is a sequence of video clips on topic X.

Our goal then is to infer a topic-wise task graph $\mathcal{T}^X = (\mathcal{V}, \mathcal{E}, w)$ with the node set $\mathcal{V} = K^X$, the edge set $\mathcal{E}$ of one-hop step transitions that are grounded in the videos, and edge weights $w_{(j,k)}$ that denote the transition probabilities from $s_j^t \rightarrow s_k^t$. We will use "step" and "node" interchangeably in the following. We determine a valid node transition $s_j^t \rightarrow s_k^t$ through their respective cross-modal grounding scores, $f(v_i, s_j^t)$ and $f(v_{i+1}, s_k^t)$, to any consecutive video clips (v_i, v_{i+1}). To compute the cross-modal scoring f, we encode the video clips and texts with VideoCLIP (Xu et al., 2021), a language–video alignment model pretrained to be robust toward temporal misalignment, and compute the dot product between the pooled visual and the pooled textual encodings. We keep the `top-k` grounding scores as a quality threshold. In other words, if both $f(v_i, s_j^t)$ and $f(v_{i+1}, s_k^t)$ exceed a certain threshold, it is more likely that $s_j^t \rightarrow s_k^t$ is a valid step transition, licensed by observation (v_i, v_{i+1}). A transition score $s(s_j^t, s_k^t)$ is obtained by multiplying the respective grounding scores, i.e., $s(s_j^t, s_k^t) = f(v_i, s_j^t) \cdot f(v_{i+1}, s_k^t)$. The final transition scores of a pair $s(s_j^t, s_k^t)$ are aggregated and we prune the edge connections by keeping the ones whose scores lie above the `q-th` percentile among all. Finally, $w_{(j,k)}$ is the result of dividing $s(s_j^t, s_k^t)$ by the summed transition scores of all outgoing edges from s_j^t. The final topic-wise PKG represents probabilistically of how likely $s_j^t \rightarrow s_k^t$ is compared to $s_j^t \rightarrow s_l^t$ according to the aligned video clips. See Appendix C for details on preprocessing and hyperparameters.

5.2. Inferring Step Types

The probability difference between pairs of step transitions in PKG can be leveraged to infer step types as defined in §3. To reduce the complexity, we restrict the step transitions to occur within a context window that spans a maximum of two subsequent steps.

To extract a *sequential* pair (s_i^t, s_{i+1}^t), we compare the transition probability of $s_i^t \rightarrow s_{i+1}^t$ and $s_{i+1}^t \rightarrow s_i^t$ against a threshold `SEQ`:

$$(s_i^t, s_{i+1}^t) = \begin{cases} 1 & \text{if } w_{(i,i+1)} - w_{(i+1,i)} > \texttt{SEQ} \\ 0 & \text{otherwise} \end{cases}$$

The larger the threshold `SEQ`, the stricter the sequential order that must be held.

For an *interchangeable* step pair (s_{i+1}^t, s_{i+2}^t) to be valid, we require at any given step s_i^t, the absolute difference between the transition likelihood $s_i^t \rightarrow s_{i+1}^t$ and $s_i^t \rightarrow s_{i+2}^t$ to be smaller than a threshold `INT`, and $s_{i+1}^t \rightarrow s_{i+2}^t$ must not follow a strict

sequential order:

$$(s^t_{i+1}, s^t_{i+2}) = \begin{cases} 1 & \text{if} \quad |w_{(i,i+1)} - w_{(i,i+2)}| < \texttt{INT} \\ & \quad \wedge\, w_{(i+1,i+2)} - w_{(i+2,i+1)} < \texttt{SEQ} \\ 0 & \text{otherwise} \end{cases}$$

This indicates that there is no strict order of executing s^t_{i+1} or s^t_{i+2} after completing s^t_i. However, if the weight of $s^t_i \rightarrow s^t_{i+2}$ exceeds that of $s^t_i \rightarrow s^t_{i+1}$ by a threshold $\texttt{OPT}$, we infer that s^t_{i+1} is *optional*:

$$s^t_{i+1} = \begin{cases} 1 & \text{if } w_{(i,i+2)} - w_{(i,i+1)} > \texttt{OPT} \\ 0 & \text{otherwise} \end{cases}$$

We refer to Appendix C.2 for the algorithms.

6. Experimental Setup

Our experiments aim to **assess the ability of LLMs to recognize key aspects of procedural knowledge and reasoning, specifically, sequential, interchangeable, and optional steps**. In this context, we furthermore explore the following research questions (RQs):

(RQ1) Does an approach with intuitively acquired procedural knowledge offer benefits over LLMs?

(RQ2) What roles do model size and training data play in the procedural capabilities emergence?

(RQ3) To what extent do justifications of the LLMs reflect procedure-relevant reasoning?

6.1. Pipeline Baseline

To address RQ1, we use the modular inference method described in §5.2, termed as PKG BASELINE, and compare it against black-box LLMs. We use the SIO dev set to tune the thresholds $\texttt{SEQ}$ (.07), $\texttt{INT}$ (.25) and $\texttt{OPT}$ (.08) (see App. C.2 for details), and evaluate the step types inferred from the SIO test set.

6.2. Large Language Models

Model Selection. For a comprehensive exploration of our main research question as well as for RQ2, we assess open-source or non-proprietary instruction-tuned models[5] from various families and sizes, namely LLAMA-2-(7-70B), LLAMA-3.1-(8 and 70B), GEMMA-2-(2 and 9B), MISTRAL-7B, and MIXTRAL-8X7B. We also report upper-bound performance references using the proprietary reasoning models O3-MINI and GEMINI-FLASH-2.5.[6] Apart from model size, differences across families include context window size and training data. LLAMA-3.1-70B has the largest window size (128k tokens) and is trained on 15T tokens of multilingual text. In terms of architecture, MIXTRAL-8X7B uses a Sparse Mixture-of-Experts design (Shazeer et al., 2017), employing a gating mechanism to activate relevant experts based on the input, while the remaining open-sourced models are auto-regressive transformers. Appendix B gives the implementation details and full model comparisons.

Prompt Design. We use the instruction-following abilities of LLMs to identify step types through a two-shot prompt, consisting of one positive example and one negative example, where the evaluated step type is not identifiable within the task. To examine if the underlying LLMs employ procedural reasoning for their answer (RQ3), we furthermore prompt them zero-shot for rationales on their **correct** predictions.[7] For reasoning-based models, we directly extract their reasoning tokens from their two-shot prompt responses as the rationales. All the templates, example prompts, decoding strategies and the answer parser can be found in Appendix D.

6.3. Evaluation Metrics

For each binary classification task of step types, i.e., SEQUENTIAL, INTERCHANGEABLE and OPTIONAL, we report precision (P), recall (R) and F_1 score averaged across step/step pair instances in the test set. We furthermore propose two error-quantifying metrics specific to this task: constraint violation (CV) and step range exceedance (SE). CV occurs when the answer does not adhere to the desired output constraint. For example, a non-adjacent pair like `(0,2)` is not a valid answer. SE is a subcase of CV and captures cases where the step index of a generated step/step pair exceeds the bounds of a provided step sequence. For example, in a task with a five-step sequence, an output of `(0,4)` is a CV error, while `(5,6)` is both, a CV and SE error.

7. Results

As shown in Table 3, among the open-sourced models, LLAMA-3.1-70B achieves the best performance averaged across all types (avg. F_1: 55.3%). The effectiveness on step type identification improves with model scale across families, with the GEMMA family showing the greatest gain (+15.3pp in F_1). Since GEMMA-2-9B was trained on four times more tokens than its smaller variant, the size of the training data is a likely factor for this large gain. The

[5] From the HuggingFace Library (version `4.39.2`; Wolf et al., 2020)

[6] Accessed through API calls.

[7] We found few-shot prompts led the models to syntactically generate rationales in a slot-filling manner following the example style, e.g., "the precondition of step x is to perform step y".

	Sequential			Interchangeable			Optional			All		
Model	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
PKG Baseline	80.2	30.1	43.8	14.6	59.1	23.5	8.3	25.5	12.5	34.4	38.2	26.6
Llama-2-7B	76.8	79.1	77.9	15.7	32.2	21.1	11.2	51.0	18.4	34.6	54.1	39.1
Llama-2-13B	78.0	63.8	70.2	14.3	67.8	23.6	12.6	60.2	20.9	35.0	63.9	38.2
Llama-2-70B	77.4	76.5	77.0	16.2	52.3	24.8	16.9	76.5	27.7	36.8	68.4	43.2
Llama-3.1-8B	77.0	**94.9**	85.0	15.6	**87.2**	26.5	18.3	52.0	27.1	37.0	**78.0**	46.2
Llama-3.1-70B	84.1	90.9	**87.4**	33.8	50.3	40.4	25.8	73.5	38.2	47.9	71.6	55.3
Gemma-2-2B	75.4	47.7	58.4	16.6	36.9	22.9	10.9	76.5	19.1	34.3	53.7	33.5
Gemma-2-9B	80.7	81.3	81.0	23.0	36.2	28.1	25.3	70.4	37.2	43.0	62.6	48.8
Mistral-7B	81.1	76.8	78.9	15.8	59.1	24.9	18.4	66.3	28.8	38.4	67.4	44.2
Mixtral-8x7B	79.7	88.3	83.8	20.8	40.3	27.4	23.9	60.2	34.2	41.5	62.9	48.5
O3-mini	89.2	78.4	83.4	**53.5**	55.7	**54.6**	27.9	**84.7**	42.0	56.9	72.9	60.0
gemini-flash-2.5	**93.2**	71.0	80.7	49.7	57.7	53.4	**36.8**	72.4	**48.8**	**59.9**	67.1	**61.0**

Table 3: Results (%): step type identification task. **Boldface** indicates the best result across models, best results within each release type (proprietary vs. non-proprietary) are underlined.

results thus indicate that both, model size and the amount of training data, play a crucial role for capturing procedural knowledge.

The LLMs are particularly strong on sequential. Although one could expect that this translates into the detection of interchangeable pairs due to mutual exclusivity, we find that this step type consistently proves to be the most challenging across open-source models (F_1 between 21.1% and 40.4%). For example, Llama-3.1-70B predicts the step pair "find an empty wine bottle" and "take the cap or cork off and set it aside" as both sequential and interchangeable. This controversy underscores the need for advancements in procedural reasoning. As for the proprietary models, they surpass Llama-3.1-70B by at most 5.7pp in F_1 score (column All). Moreover, gemini-flash-2.5 exhibits the least contradicting prediction patterns for sequential and interchangeable.

Turning to the PKG Baseline, we find that it is comparable to black-box LLMs in terms of precision on sequential and interchangeable and even outperforms Llama-2-13B. This suggests that grounding textual steps in a video resource enables some competitive procedural comprehension capabilities without expanding model complexity or expensive training resources. In addition, the performance gap between sequential and interchangeable is narrower than that of gemini-flash-2.5, which highlights a key advantage of PKG: its flexibility enables the manipulation of step transition probabilities, thereby injecting mutual exclusivity between sequential and interchangeable steps into the extraction heuristics (cf. §5.2). However, PKG Baseline falls short on optional, whereas the LLMs likely benefit from their ability to exploit linguistic cues for identifying optional steps, such as phrases like "if needed" (e.g., Step S_2 in Task (B) in Table 1).[8] To further examine the effect of such cues, we analyzed model performance on 22 optional steps containing them. All LLMs achieved F_1 scores above 50%, with performance improving as model size increases (Llama-3.1-70B: 95%).

Taken together, the evidence shows that LLMs can take advantage of linguistic cues for step type identification and encode more procedural knowledge with increasing size and likely with increasing training data (**RQ2**). However, **true procedural reasoning is not yet achieved** as evidenced by their weak understanding of the mutual exclusivity between sequential and interchangeable. PKG Baseline, in turn, offers the option to encode this concept, but likely suffers from its relatively small amount of demonstration videos (**RQ1**).

8. Analysis

We investigate key aspects of effective procedural task systems: the breadth and the depth of the covered procedural knowledge (RQ2), respectively, instruction adherence, and the ability to explain the impact of step changes (RQ3). Our analysis in the context of RQ2 focuses on open-source LLMs since many parameters of the proprietary models are undisclosed. However, we include their results as upper-bound reference.

Knowledge Breadth. As depicted in Figure 3, while all models perform equally well across domains on sequential, effectiveness improves with

[8] Other cues include "if desired", "if necessary", "if you'd like", "alternatively", "(optional)", "if you want".

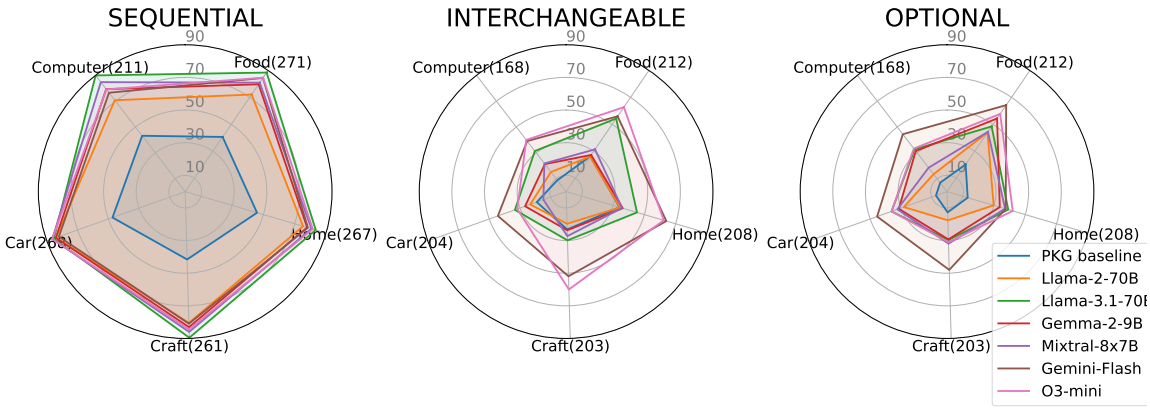

Figure 3: Model performance (in F_1) across various task domains. For a better visualization, we have selected the best-performing model of each family. (#) indicates number of instances for each domain.

Examples	Poor Rationales	Sound Rationales
(A) *How to Make an Avocado Shake*; Sequential (1, 2). Step 1: Blend the ingredients until the **mixture** is no longer chunky Step 2: Line the sides of your glasses with chocolate syrup, and pour the **puree** into the glasses	MIXTRAL-8X7B: The post-condition of Step 1 is a prerequisite for Step 2. However, it is possible to line the glass with chocolate syrup and pour the puree before blending, but this would result in a non-homogeneous shake.	LLAMA-3.1-70B: The precondition of pouring the puree into the glasses (Step 2) is to have a smooth puree (Step 1). O3-MINI: Steps 1 and 2 are also sequential since I use the puree from blending. GEMINI-FLASH-2.5: Although you can prepare the glasses beforehand, the *pouring* action relies on the blended state.

Table 4: Examples of explanations classified by their quality (poor vs. sound rationales). Refer to Table 7 in the Appendix A.4 for full step sequences.

bigger model size for the other two step types. However, model size alone is not the key factor. For instance, GEMMA-2-9B consistently outperforms LLAMA-2-70B. The former is eight times smaller, but has been been trained on four times more tokens. This finding indicates that a **broad coverage of domain knowledge requires a large amount of training data**.

Knowledge Depth. Across LLM families, a clear pattern emerges for SEQUENTIAL:[9] **Performance on expert-level knowledge cases improves with increasing model size**, while it declines on commonsense knowledge. The exception is the LLAMA-3.1 family, which performs consistently across knowledge types.

Instruction Compliance. We examine the instruction adherence ability of LLMs in the context of the step identification task with our two proposed error-quantifying metrics, constraint violation (CV) and step range exceedance (SE) (see §6.3). We found that most LLMs (except the LLAMA-2 family) are attentive to the boundary of input step length, and the SE errors shrink to 0.0, especially when the model size grows. As for the more general CV, it varies across families unlike the previous pattern. We remark that proprietary models generally excel at following instruction constraints. For full results, see Table 12 in Appendix E, and Table 13 for erroneous cases, which illustrate LLM outputs of non-sensical step sequences.

Procedural Reasoning in Rationales. Key aspects in reasoning about potential step order alterations in the procedural context include entity co-reference (Anthonio and Roth, 2021; Rim et al., 2023) and object state tracking (Tandon et al., 2020; Kim and Schuster, 2023; Zhang et al., 2024), which monitors the state changes that an entity undergoes throughout a procedural task. For example, sequential order must be preserved in S_1 and S_2 in Task (A) in Table 4, since they describe a state change of the entity referred to as "mixture" in S_1 and a derived new state that is lexicalized by the

[9]There are few instances requiring expert-knowledge for INTERCHANGEABLE and OPTIONAL, only 1 and 3 resp.

reference "puree" after blending in S_2.

We sampled 15 tasks (three tasks per domain) and examined the rationales of selected LLMs. Specifically, we compared the open-source LLM that performed overall best in our main experiment (LLAMA-3.1-70B; Table 3) as well as MIXTRAL-8X7B against the proprietary reasoning models O3-MINI and GEMINI-FLASH-2.5. An example of model explanations is given in Table 4. We found that LLAMA-3.1-70B can mostly resolve entity linking and object state tracking. In contrast, MIXTRAL-8X7B's output showcase a failed object linking by referring to the not blended mixture as "puree". However, LLAMA-3.1-70B produces less natural explanations compared to proprietary reasoning models, it adopts the phrasings of the prompt used for rationale generation ("pre/post-condition"). Table 14 in Appendix E gives further examples of rationales of the LLMs on their correctly predicted output.

9. Conclusions

We investigate the capabilities of LLMs in identifying various step types in instructional manuals, namely sequential order, interchangeable execution and optional step skipping, which is essential for adapting to individual users and situations in assistive settings. We compare them against a heuristic, visually-informed procedural knowledge graph (PKG). We find that the vast knowledge of LLMs makes them valuable sources of procedural knowledge, and that model size and data size are essential for both, procedural knowledge breadth (different domains) and depth (expert knowledge), where models of smaller size or data fall short. Notably, while the largest open-sourced LLM performs well on the task, it exhibits limited procedural reasoning, particularly in understanding the mutual exclusivity of the sequential versus the interchangeable step type. The PKG offers benefits in terms of these limitations, as heuristics can be imposed on the step transition probabilities to infer step types. However, uncovering its full potential for step type inference likely depends on constructing it from a vast amount of procedural data. Considering this complementary strength, a hybrid setup that augments LLMs' knowledge by cross-checking the visually-licensed PKG is worth investigating.

Acknowledgments

We thank the annotators for their contributions to the data annotation process. We also thank the anonymous reviewers for their valuable and constructive feedback. The research presented in this paper was funded by the DFG Emmy Noether program (RO 4848/2-1).

10. Bibliographical References

Talita Anthonio and Michael Roth. 2021. Resolving implicit references in instructional texts. pages 58–71, Punta Cana, Dominican Republic and Online.

Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. 2023. Video-mined task graphs for keystep recognition in instructional videos. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yuwei Bao, Keunwoo Yu, Yichi Zhang, Shane Storks, Itamar Bar-Yossef, Alex de la Iglesia, Megan Su, Xiao Zheng, and Joyce Chai. 2023. Can foundation models watch, talk and guide you step by step to make a cake? pages 12325–12341, Singapore.

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D Hwang, Xiang Lorraine Li, Hirona J Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. 2023. PlaSma: Making small language models better procedural knowledge models for (counterfactual) planning. *arXiv [cs.CL]*.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, volume 36, pages 79081–79094.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Yunseok Jang, Sungryull Sohn, Lajanugen Logeswaran, Tiange Luo, Moontae Lee, and

Honglak Lee. 2023. Multimodal subtask graph generation from instructional videos. *arXiv [cs.LG]*.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. pages 3835–3855, Toronto, Canada.

Mahnaz Koupaee and William Yang Wang. 2018. WikiHow: A large scale text summarization dataset. *arXiv [cs.CL]*.

Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Ray Mooney. 2024. CaT-bench: Benchmarking language model understanding of causal and temporal dependencies in plans. pages 19336–19354, Miami, Florida, USA.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. 2023. On grounded planning for embodied tasks with language models. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, volume 37, pages 13192–13200.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2019a. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019b. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640.

Dhruvesh Patel, Hamid Eghbalzadeh, Nitin Kamra, Michael Louis Iuzzolino, Unnat Jain, and Ruta Desai. 2023. Pretrained language models as visual planners for human assistance. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kyeongmin Rim, Jingxuan Tu, Bingyang Ye, Marc Verhagen, Eben Holderness, and James Pustejovsky. 2023. The coreference under transformation labeling dataset: Entity tracking in procedural texts using event models. pages 12448–12460, Toronto, Canada.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proscript: Partially ordered scripts generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2138–2149.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv [cs.LG]*.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022. LLM-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 2998-3009*, pages 2998–3009.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. pages 6408–6417, Online.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models : A critical investigation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu,

Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2022. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals. pages 4525–4542, Dublin, Ireland.

Te-Lin Wu, Caiqi Zhang, Qingyuan Hu, Alexander Spangher, and Nanyun Peng. 2023a. Learning action conditions from instructional manuals for instruction understanding. pages 3023–3043, Toronto, Canada.

Xueqing Wu, Sha Li, and Heng Ji. 2023b. OpenPI-C: A better benchmark and stronger baseline for open-vocabulary state tracking. pages 7213–7222, Toronto, Canada.

Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. 2021. VideoCLIP: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6787–6800, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hsiu-Yu Yang and Carina Silberer. 2023. Implicit affordance acquisition via causal Action–Effect modeling in the video domain. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 846–871, Nusa Dua, Bali. Association for Computational Linguistics.

Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. 2021. Visual goal-step inference using wikiHow. pages 2167–2179, Online and Punta Cana, Dominican Republic.

Huibin Zhang, Zhengkun Zhang, Yao Zhang, Jun Wang, Yufan Li, Ning Jiang, Xin Wei, and Zhenglu Yang. 2022. Modeling temporal-modal entity graph for procedural multimodal machine comprehension.

Li Zhang. 2022. Reasoning about procedures with natural language processing: A tutorial. *arXiv [cs.CL]*.

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. pages 4630–4639, Online.

Li Zhang, Hainiu Xu, Abhinav Kommula, Chris Callison-Burch, and Niket Tandon. 2024. OpenPI2.0: An improved dataset for entity tracking in texts. pages 166–178, St. Julian's, Malta.

Honglu Zhou, Roberto Martín-Martín, Mubbasir Kapadia, Silvio Savarese, and Juan Carlos Niebles. 2023a. Procedure-aware pretraining for instructional video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*.

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. 2022. Show me more details: Discovering hierarchies of procedures from semi-structured web data. pages 2998–3012, Dublin, Ireland.

Yu Zhou, Sha Li, Manling Li, Xudong Lin, Shih-Fu Chang, Mohit Bansal, and Heng Ji. 2023b. Non-sequential graph script induction via multimedia grounding. pages 5529–5545, Toronto, Canada.

A. The SIO Dataset

A.1. Annotation Procedure

Annotator Recruitment. We design a two stage annotation procedure. In the first stage, we recruit 9 volunteers experienced in NLP annotation, consisting of 6 doctoral students and 3 postgraduate students. The workload is distributed evenly, with each participant responsible for annotating 15 tasks. In the second stage, we recruit 12 more postgraduate students of a computational linguistics program. Each annotator is compensated at a rate of €13 per hour for their annotation work.

Annotation Task Description. In Figure 4, we present the task overview.

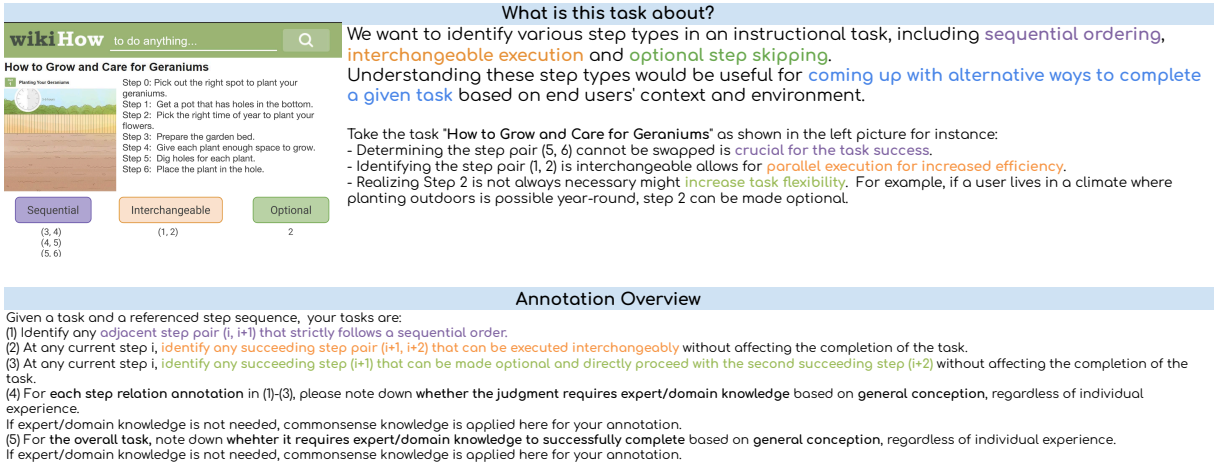

Figure 4: Overview of the annotation task.

Category Definitions. The annotation fields that the annotators need to annotate are defined as follows:

SEQUENTIAL `(i, i+1)`
At any given step `i` (start from step `0`), if you find the step pair `(i, i+1)` follows a strict sequential order (the task will fail if the order is swapped), please note down `(i, i+1)` in the column "Sequential `(i, i+1)`".

INTERCHANGEABLE `(i, (i+1, i+2))`
At any given step `i` (start from step `0` and iterate until the last second step), judge if step `i+1` & step `i+2` can be interchangeably executed without affecting the task completion. If yes, please note down `(i, (i+1, i+2))` in the column "Interchangeable `(i, (i+1, i+2))`". In this case, `(i+1, i+2)` can be executed interchangeably when one is at step `i`.

OPTIONAL `(i, (i+1), i+2)`
At any given step `i` (start from step `0` and iterate until the last second step), judge if one can omit step `i+1` and proceed to step `i+2` without affecting the task completion. If yes, please note down `(i, (i+1), i+2)` in the column "Optional `(i, (i+1), i+2)`". In this case, `(i+1)` can be skipped and one can go from step `i` directly to step `(i+2)` without affecting the task completion.

Step Level Expert Knowledge Required? For each of your selected step pair, please answer whether the decision requires expert/domain knowledge based on general conception, regardless of individual experience. If a step annotation does not require domain/expert knowledge, it could be solved by commonsense knowledge. For example, anyone who does not have prior experience in cutting a "durian" would know a step transition **Step 0:** use a knife to cut a "durian" → **Step 1:** take out the seed of a "durian") can not be swapped based on the commonsense knowledge that "Before taking out seeds from a fruit with a spiky hard surface, one has to cut it open".

Task Level Expert Knowledge Required? After annotating all the step relations, please also answer whether this task generally requires expert/domain knowledge in the column "Overall Task: Domain Knowledge Required?". Requiring expert/domain knowledge means one has to follow a step-by-step instruction to ensure the task can be completed even with similar prior experiences.

A.2. Label Distributions

The label distribution across procedural task domains is shown in Table 5.

	Dev						Test									
	Food (5)		Home (5)		Craft (5)		Food (59)		Home (59)		Craft (58)		Car (64)		Computer (43)	
Step Type	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.
SEQUENTIAL	13	12	19	3	12	5	193	78	184	83	210	51	213	55	180	31
INTERCHANGEABLE	6	14	2	15	3	9	34	178	54	154	26	177	23	181	12	156
OPTIONAL	4	16	1	16	2	10	34	178	26	182	13	190	17	187	8	160

Table 5: Label distribution across procedural task domains. (#) denotes number of tasks.

A.3. Inter-Annotator Agreements (IAAs)

Table 6 shows the IAAs across annotation fields in the dev and the test set, overall and broken down by procedural domains. The low IAA could be attributed to more instances of expert knowledge requirement as can be seen in Figure 5.

	Dev				Test					
	all	Food	Home	Craft	all	Food	Home	Craft	Car	Computer
SEQUENTIAL	0.15	0.24	-0.01	0.03	0.26	0.29	0.30	0.19	0.17	0.25
INTERCHANGEABLE	0.37	0.41	0.22	0.32	0.36	0.41	0.37	0.37	0.32	0.19
OPTIONAL	0.25	0.44	-0.02	0.11	0.45	0.58	0.40	0.42	0.38	0.32
Task-level expert	0.34	0.33	0.18	0.44	0.10	-0.02	0.10	0.14	0.00	0.11

Table 6: Average pairwise IAA overall (column all) and by procedural task domain.

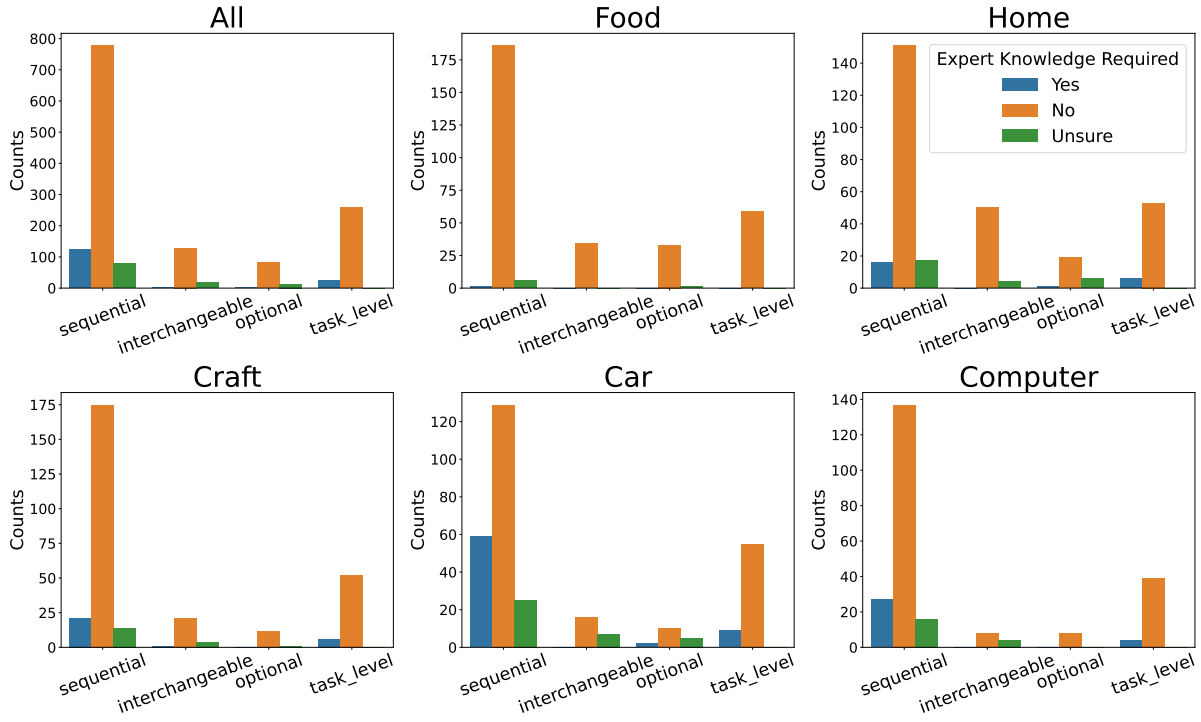

Figure 5: Test set: distribution of expert knowledge requirement across step types and across domains.

A.4. SIO Dataset Examples

Examples of the annotated SIO dataset are shown in Table 7.

Task Title (*Domain*)	Steps	Annotation
(A) *How to Make an Avocado Shake - Making the Shake* (*Food and Entertaining*)	Step 0: Mix the avocado, milk, ice, and sugar in a blender Step 1: Blend the ingredients until the mixture is no longer chunky Step 2: Line the sides of your glasses with chocolate syrup, and pour the puree into the glasses Step 3: Garnish the shake	sequential: [(0, 1), (1, 2), (2, 3)] sequential_expert: [No, No, No] interchangeable: [] interchangeable_expert: [] optional: [] optional_expert: [] task_expert: No
(B) *How to Iron on a Patch - Ironing on the Patch* (*Home and Garden*)	Step 0: Lay the base item on a flat, heat-resistant surface Step 1: Place the patch in the position you chose Step 2: Heat up an iron Step 3: Place a thin towel over the patch Step 4: Position the heated iron over the patch and press down Step 5: Remove the iron and allow the patch to to cool	sequential: [(0, 1), (3, 4), (4, 5)] sequential_expert: [No, No, No] interchangeable: [(1, 2), (2, 3)] interchangeable_expert: [No, No] optional: [] optional_expert: [] task_expert: No
(C) *How to Tattoo Leather - Applying the Design to Leather* *(Hobbies and Crafts)*	Step 0: Test an out of sight portion of the leather Step 1: Ink the main outline of your design Step 2: Wipe away excess ink as necessary Step 3: Add accents and details after the main body of the design Step 4: Fill in solid features of the design Step 5: Clean off any excess ink and show off your tattooed leather	sequential: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)] sequential_expert: [No, No, Unsure, No, No] interchangeable: [] interchangeable_expert: [] optional: [2] optional_expert: [No] task_expert: Yes
(D) *How to Replace Shocks - Installing New Shocks* *(Cars and Other Vehicles)*	Step 0: Fit the new shock back onto the suspension control arm Step 1: If needed, you may once again affix the anti-roll bar if you removed it earlier Step 2: Check your torque specs in the service manual Step 3: Repeat the steps to replace the other 3 shock absorbers, if necessary	sequential: [(0, 1)] sequential_expert: [Unsure] interchangeable: [] interchangeable_expert: [] optional: [1] optional_expert: [No]

Table 7: Examples of our SIO test set.

Model	Architecture	Context Window (Tokens)	Training Tokens
LLAMA-2 (7B, 13B) LLAMA-2 (70B)	Auto-regressive Transformer + Grouped-Query Attention to improve inference scalability	4k	2T
LLAMA-3.1 (multilingual, 8-70B)	Auto-regressive Transformer with Grouped-Query Attention	∼128k	∼15T+
Gemma-2 (2B) Gemma-2 (9B)	Auto-regressive Transformer Auto-regressive Transformer	8k 8k	2T 8T
Mistral-7B	Auto-regressive Transformer	∼32k	Unknown
Mixtral-8x7B	Sparse Mixture-of-Experts (SMoE)	∼32k	Unknown

Table 8: Comparison of open-source instruction-tuned LLMs across different architectures, context windows, and training tokens.

B. Implementation Details

All LLMs we evaluated here can be accessed through the HuggingFace Library (version: `4.39.2`) (Wolf et al., 2020):

`meta-llama/Llama-2-7b-chat-hf`,
`meta-llama/Llama-2-13b-chat-hf`,
`meta-llama/Llama-2-70b-chat-hf`,
`meta-llama/Meta-Llama-3.1-8B-Instruct`,
`meta-llama/Meta-Llama-3.1-70B-Instruct`,
`google/gemma-2-2b-it`,
`google/gemma-2-9b-it`,
`mistralai/Mistral-7B-Instruct-v0.3`,
`mistralai/Mixtral-8x7B-Instruct-v0.1`

The inference is run on a single NVIDIA RTX A6000 GPU within 2 hours through 4-bit quantization techniques. A detailed model comparison is given in Table 8.

C. PKG Construction and Inferred Step Types

C.1. PKG Construction

Preprocessing Details. For topic clustering (Step 1), we lower-cased and removed stop words from the task title before extracting the sentence features ($\mathbb{R}^1 \times 768$) using `all-mpnet-base-v2` model from the `sentence-transformer` library (Reimers and Gurevych, 2019). When performing cross-modal grounding w.r.t topic-wise PKG construction , we follow VideoClip's recipe to decode each video clip v_i at 30 fps and extracted features with S3D (Miech et al., 2019a) per second, resulting in $v_i = \{\mathbf{v}_x\}_{x=1}^{|v_i|}$ video tokens ($\mathbb{R}^{|v_i|} \times 512$). To calculate the cross-modal alignment, we mean pooled across the video tokens to represent a whole video clip representation ($\mathbb{R}^1 \times 512$). Since VideoCLIP expects an input of text–video clip pair, we duplicated a v_i to pair with each of the $s_j^t \in K^X$ and calculated the respective cross-modal grounding score $f(v_i, s_j^t)$ accordingly.

Hyperparameters. For the clustering, the following parameters are used: `task clustering linkage=Average`, `task clustering distance threshold=`0.5, `task clustering affinity=cosine`. Sanity check shows that the same task title (exact string match) across WikiHow and HowTo100M are clustered together. For the cross-modal grounding, `topk=`3 and `q-th=`25.

C.2. Inferring Step Types

Algorithm 1 show the functions on inferring step type from a topic-wise PKG. In Table 9, we provide the results on the SIO dev set when experimenting different combinations of step type identification hyperparameters. The final hyperparameter set we use is: `SEQ=` 0.07, `INT=` 0.25, `OPT=` 0.08.

Algorithm 1 Inferring Step Types

```
function FindSequential(PKG, SEQ_threshold)
    result ← empty list
    num_nodes, _ ← shape of PKG
    for i in range(num_nodes) do
        if i is the last node then
            continue
        end if
        SEQ_diff ← PKG[i, i+1] - PKG[i+1, i]
        if SEQ_diff > SEQ_threshold then
            item ← ((i, i+1))
            append item to result
        end if
    end for
    return result
end function
function FindInterchangeable(PKG, INT_threshold, SEQ_threshold)
    result ← empty list
    num_nodes, _ ← shape of PKG
    for i in range(num_nodes) do
        if i is the last or second-to-last node then
            continue
        end if
        INT_diff ← | PKG[i, i+1] - PKG[i, i+2] |
        SEQ_diff ← PKG[i+1, i+2] - PKG[i+2, i+1]
        if INT_diff < INT_threshold & SEQ_diff < SEQ_threshold then
            item ← ((i+1, i+2))
            append item to result
        end if
    end for
    return result
end function
function FindOptional(PKG, OPT_threshold)
    result ← empty list
    num_nodes, _ ← shape of PKG
    for i in range(num_nodes) do
        if i is the last or second-to-last node then
            continue
        end if
        OPT_diff ← PKG[i, i+2] - PKG[i, i+1]
        if OPT_diff > OPT_threshold then
            item ← i+1
            append item to result
        end if
    end for
    return result
end function
```

D. Prompt Design, Decoding Strategies and Answer Parsing

D.1. Templates and Prompts

The general template is shown in Table 10. To exemplify how we obtain model rationales, we give the template and an example prompt in Table 11.

D.2. Decoding Strategy

We employ greedy-search (`temperature`: 0.1) to ensure that the step type prediction is more deterministic and apply nucleus sampling (Holtzman et al., 2019) (`temperature`: 0.3, `top-p`: 0.9) for rationale generation to encourage more diverse and coherent text. For nucleaus sampling, we explore two sets of hyperparamers: [`temperature`: 0.1, `top-p`: 0.9] and [`temperature`: 0.3, `top-p`: 0.9], and found little

	Sequential			Interchangeable			Optional			All		
Hyperparameters	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
SEQ= 0.10, INT= 0.05, OPT= 0.08	45.0	11.0	18.0	17.0	9.0	12.0	9.0	14.0	11.0	24.0	11.0	14.0
SEQ= 0.07, INT= 0.25, OPT= 0.45	64.0	32.0	42.0	21.0	55.0	31.0	0.0	0.0	0.0	28.0	29.0	24.0
SEQ= 0.07, INT= 0.25, OPT= 0.10	64.0	32.0	42.0	21.0	55.0	31.0	0.0	0.0	0.0	28.0	29.0	24.0
SEQ= 0.07, INT= 0.25, OPT= 0.08	64.0	32.0	42.0	21.0	55.0	31.0	7.0	14.0	10.0	**31.0**	**34.0**	**28.0**

Table 9: Dev Results (%) of PKG Baseline: step type binary classification task. Boldface indicates best result.

Template
{"role": "user", "content": "We analyze step relations in a procedural task. **{Step Type Task Definition}** You will be given a task along with its corresponding step sequence wrapped in «». You will be prompted with a question wrapped in <>. **{(Applicable for interchangeable & optional) Constraint}**" } {"role": "assistant", "content":"Got it! Give me the first example." } {"role": "user", "content":"Great! Here is your first example: **«{Few-shot Example 1 (positive case)}»** **<{Step Type Question}>** **{Output Format Specification}**" } {"role": "assistant", "content":"**{Few-shot Example 1 Answer}**" } {"role": "user", "content":"Great! Here is another example: **«{Few-shot Example 2 (negative case)}»** **<{Step Type Question}>** **{Output Format Specification}**" } {"role": "assistant", "content":"**{Few-shot Example 2 Answer}**" } {"role": "user", "content":"Perfect! Here is another example: **«{Inference Time Input}»** **<{Step Type Question}>** **{Output Format Specification}**" }

Table 10: Few-shot (2 shots) template for answer generation.

change in the rationale quality (as a reference, the overall difference in F_1 on the step type identification tasks is $< 1\%$ across models).

D.3. Answer Parser

For sequential and interchangeable tasks, we parse the output as `{"step pairs": [(x, x+1), (y, y+1), ...]}`, while for optional we use `{"steps": [x, y, ...]}`. To extract valid outputs from a model's generated raw texts for automatic evaluation, we first remove all quotation marks and redundant white spaces and add quotation marks for the dictionary keys, e.g., `{"steps":}`. Then, we take the first occurrence of the desired output format as the model's prediction. If there is no desired output format, we assign an empty dictionary `{"step pairs": []}` as the model's prediction for sequential and interchangeable types, and `{"steps": []}` for the optional.

Template
<s>[INST] We analyze step relations in a procedural task. Your task is to provide a rationale for each of your previous selected **{sequential step pairs | interchangeable step pairs | optional steps}**. The rationale should be formulated based on the action preconditions and postconditions. You will be given a task along with its corresponding step sequence and your previous answers wrapped in «». You will be prompted with a question wrapped in <>. [/INST] Got it! Give me the first example. </s> <s>[INST] Great! Here is your first example: **«{Inference Time Input}»** <What is the rationale behind each of your selected **{sequential step pairs | interchangeable step pairs | optional steps}**> Provide the rationale for each step pair in a list of dictionary. Each dictionary contains keys **{step pair | step}** and 'reason'. No more details. [/INST]
Example Prompt - Sequential
<s>[INST] We analyze step relations in a procedural task. Your task is to provide a rationale for each of your previous selected sequential step pairs. The rationale should be formulated based on the action preconditions and postconditions. You will be given a task along with its corresponding step sequence and your previous answers wrapped in «». You will be prompted with a question wrapped in <>. [/INST] Got it! Give me the first example. </s> <s>[INST] Great! Here is your first example: «Task: How to Season a Grill - Cleaning the Grates, Step sequence: Step 0: Remove the grates from the grill -> Step 1: Brush the grates with a wire grill brush -> Step 2: Wash and dry the grates, Answers: 'step pairs': [(4, 5), (5, 6)]» <What is the rationale behind each of your selected sequential step pair?> Provide the rationale for each step pair in a list of dictionary. Each dictionary contains keys 'step pair' and 'reason'. No more details. [/INST]

Table 11: Zero-shot rationale generation template and an example prompt for the LLAMA-2 family.

E. Results: Figures and Tables

Table 12 details the instruction compilance error rate across LLMs that we examined. Table 13 displays such erroneous examples. Figure 6 shows the effectiveness by knowledge type. Rationales of the LLMs are provided in Table 14.

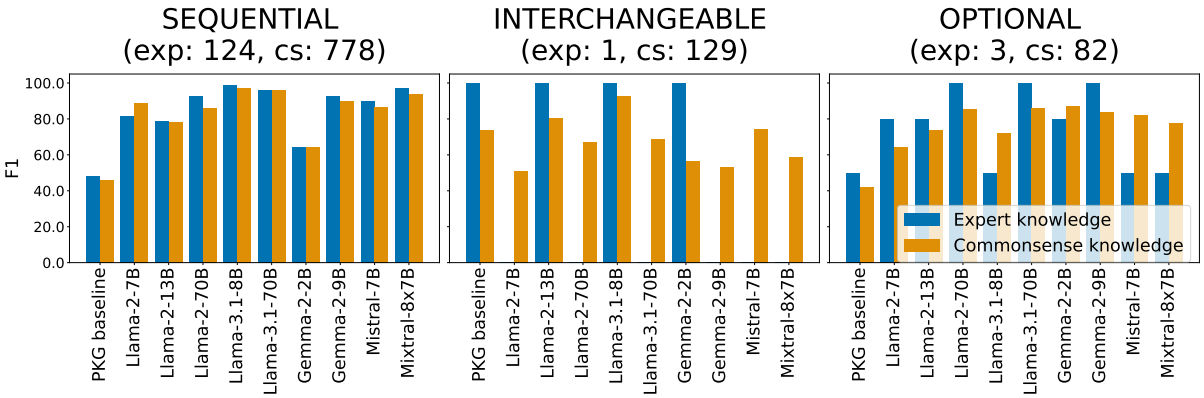

Figure 6: Knowledge type performance in F_1 score (%). (exp:#, cs:#) represents instances per knowledge type; note that only selected step types have expert knowledge annotations.

	Sequential ↓		Interchangeable ↓		Optional ↓	
Model	CV	SE	CV	SE	CV	SE
Llama-2-7B	6.71	0.35	74.91	0.0	26.5	0.0
Llama-2-13B	10.6	7.07	4.95	0.35	30.4	0.0
Llama-2-70B	4.59	4.59	40.64	1.77	24.73	0.0
Llama-3.1-8B	4.59	1.41	21.2	0.35	37.1	0.0
Llama-3.1-70B	10.25	0.0	34.98	0.0	30.74	0.0
Gemma-2-2B	0.35	0.0	81.63	0.0	82.33	0.0
Gemma-2-9B	6.36	0.0	10.95	0.0	30.04	0.0
Mistral-7B	3.89	0.0	11.66	1.06	29.33	0.0
Mixtral-8x7B	2.47	0.0	16.61	0.0	36.04	0.0

Table 12: Instruction compliance error rate (%): CV stands for “constraint violation”; SE stands for “step range exceedance”.

Examples	Correct Answer	Wrong Model Output
How to Grow Lavender - Planting Lavender Step 0: Water lavender in its nursery pot before planting Step 1: Dig a hole big enough for the spread roots Step 2: Prepare the soil Step 3: Prune the lavender Step 4: Prepare the roots Step 5: Plant the lavender	sequential: [[1, 2], [4, 5]]	LLAMA-3.1-70B: [[0, 4], [2, 5]] GEMMA-2-9B: [[5, 0]]
How to Build a Robot Car - Wiring the Car Step 0: Cut four lengths of wire Step 1: Solder two wires onto one motor Step 2: Solder the other ends to the motor driver Step 3: Repeat for the other motor Step 4: Connect the battery holder Step 5: Connect the motor driver circuit	interchangeable: None	LLAMA-3.1-70B: [[0, 1]] GEMMA-2-9B: [[1, 3]]
How to Remove Mold from Wood Furniture - Removing Stubborn Mold Step 0: Mix a bleach solution for stubborn mold Step 1: Apply the bleach solution to the furniture Step 2: Sand off solution resistant mold	optional: None	LLAMA-3.1-70B: [2] GEMMA-2-9B: [2]

Table 13: Examples of Constraint Violation Errors of LLMs across step types

Examples	LLMs' Rationales
How to Melt Silver Coins – Cleaning Silver Coins Sequential (3, 4) ... Step 3: Add salt and baking soda to the container Step 4: Drop the coins into the solution ...	LLAMA-3.1-70B: The precondition of dropping the coins into the solution (Step 4) is to have the solution of water, salt and baking soda ready (Step 3). GEMMA-2-9B: The precondition of dropping the coins is having the cleaning solution ready. MIXTRAL-8X7B: The precondition of dropping the coins into the solution (Step 4) is to have the salt and baking soda added to the container (Step 3).
How to Grow Cannas – Caring for Potted Cannas Interchangeable (1, 2) ... Step 1: Use well-drained, good quality soil Step 2: Purchase a pot with drainage holes ...	LLAMA-3.1-70B: Steps 1 and 2 can be performed in any order because they both involve preparation of the pot for planting and are independent of each other. Using good quality soil and having a pot with drainage holes are both necessary for the next steps, but the order does not matter. GEMMA-2-9B: Both steps are necessary for successful planting and growth. Well-drained soil ensures proper root health, and a pot with drainage holes prevents water-logging. MIXTRAL-8X7B: The postcondition of buying a pot (step 2) doesńt depend on the soil selection (step 1), and both steps can be performed independently.

Table 14: Examples of LLMs' rationales on their correctly predicted output.