arXiv:2607.17861v1 [cs.RO] 20 Jul 2026

# ConceptTree: Bringing Semantic Transparency to Black-Box Decision Making for Robotic Manipulation

**Yongyan Wen**[1], **Feifan Liu**[1], **Jinyi Chen**[1], **Bo An**[2], **Peng Liu**[1], **Siyaun Li**[1,†*]
[1]Harbin Institute of Technology, [2]Nanyang Technological University
siyuanli@hit.edu.cn

## Abstract

Establishing interpretable decision-making processes in long-horizon robotic manipulation is critical for enabling reliable human oversight and intervention. However, existing approaches to robotic manipulation largely treat skill selection as opaque mappings from observations to actions, offering limited transparency into how decisions are formed. In this work, we propose ConceptTree, a framework that reframes high-level manipulation skill selection as reasoning over human-interpretable concepts, representing high-level policies as a sequence of concept-level predicates over visual observations. Rather than relying on implicit latent representations, our method learns a normalized concept space grounded in visual inputs, over which a decision tree is trained to predict high-level skills. This formulation yields a transparent decision process that is both traceable and intervenable, enabling direct inspection and modification of policy behavior. We evaluate our approach on a set of real-world robotic manipulation tasks with increasing complexity. Experimental results show that ConceptTree consistently outperforms existing concept-based baselines, particularly in complex, long-horizon scenarios. Furthermore, we provide qualitative case studies showing that our model supports fine-grained intervention by modifying individual concepts, enabling targeted correction of decision errors without retraining.

## 1 Introduction

Recent advances in robotic manipulation have enabled robots to perform increasingly complex tasks in real-world environments [43, 16, 3], such as object rearrangement [8], assembly [24], and tool-use tasks [23]. Completing these tasks often involves long-horizon decision making and requires reasoning over continuously changing visual scenes [19], where success depends on selecting the appropriate high-level manipulation skill based on the current observation. To handle such complexity, a common paradigm is to decompose decision making into multiple levels of abstraction, analogous to hierarchical control in biological systems, where high-level planning and low-level execution are separated. In robotics, this is often instantiated as hierarchical policies [1], where a high-level module selects discrete skills based on observations, and low-level controllers execute them. Without an interpretable decision process, an incorrect skill choice can lead to failed trajectories, where the delayed failures are hard to diagnose.

Despite their strong performance, existing high-level decision approaches primarily rely on end-to-end neural policies or vision-language model-based policies, where decisions are produced by black-box systems [41, 34, 4, 9]. While these models are effective at capturing complex patterns, the reasoning behind their decisions remains implicit. As a result, it is often unclear why a particular skill is selected given an observation, making it difficult to interpret the process and debug the system. This lack of transparency is particularly problematic in real-world deployment, where failures must be diagnosed

*†Corresponding author.

Preprint.

and corrected in a reliable and expected manner. Without transparency in the decision-making process, when a robot selects an incorrect skill, it is challenging to determine whether the failure arises from perception errors, ambiguous scene understanding, or flawed decision logic, limiting the ability to correct the system systematically. Beyond debugging, transparent decision making also supports human trust in robotic systems by allowing users to inspect and understand the system's behavior [33].

While interpretability has been extensively studied in reinforcement learning and sequential decision making [13, 11, 27], interpretable high-level decision making for visual robotic manipulation remains relatively underexplored. Visual robotic manipulation introduces a distinct challenge: high-level skill choices are governed by physical preconditions and effects, such as whether an object is inside a container, or a door is open. An explanation must therefore reveal not only which visual features influence a decision, but how perceived scene states justify the selected skill in the evolving manipulation process. The closest relevant methods provide useful ingredients but do not fully address this need. Inherently interpretable methods, such as symbolic policies [10, 25] and decision trees [39], make decision structures explicit, but typically rely on low-dimensional states or structured inputs rather than raw visual observations. Post-hoc explanation methods [12, 31] can be applied to black-box policies, but they often identify influential image regions or features after training without exposing the decision logic that maps perceived scene properties to skill choices. Recent approaches that combine neural perception with structured decision models partially narrow this gap [36, 38, 26], yet their rules are often defined over latent features, making them difficult to inspect or intervene on in terms of task-relevant scene properties. These limitations point to a missing interface for robotic manipulation: high-level decisions should be grounded in visual observations through human-interpretable concepts that capture task-relevant object states and spatial relations, while the decision process over these concepts should remain explicit and traceable.

To address these limitations, we propose ConceptTree, a framework that combines semantically grounded concept learning with structured decision making for robotic manipulation. Its concept layer maps visual observations to manipulation-specific concepts, such as object states and spatial relations, forming a semantic interface for high-level skill reasoning. We construct concept-level annotations with vision-language models (VLMs) as verifiers of task-relevant object-state and spatial-relation predicates, enabling concept-layer training without manual concept labels. At inference time, the learned concept values are passed to a decision tree, which selects skills through inspectable decision paths. The resulting framework makes high-level decisions through a sequence of predicates over task-relevant concepts, so each prediction can be traced as an explicit decision path rather than an opaque output. We evaluate ConceptTree on four real long-horizon manipulation tasks with varying temporal dependencies and complexity. The results show that ConceptTree achieves the strongest overall completion rates among the compared methods, with particularly clear gains on more complex tasks where simple concept classifiers and black-box VLM policies degrade. Ablation studies further show that reliable concept supervision and temporal context improve decision quality while yielding compact trees with short concept-level decision paths. Finally, qualitative intervention results demonstrate that decision errors can be localized to specific concept values and corrected by modifying individual concepts without retraining.

## 2 Related Work

**Interpretability for Robotic Manipulation** Interpretability in robotic manipulation is closely related to broader efforts in interpretable machine learning [13] and sequential decision making [11, 27]. One line of work focuses on learning inherently interpretable policies, such as policy distillation [2, 18], symbolic policies [10, 25], linear models [37], or decision trees [36, 6, 29, 7, 39], where the decision process is explicitly represented. These approaches provide transparent reasoning and allow direct inspection of decision rules. However, they typically rely on low-dimensional state representations or structured inputs. Another line of work adopts post-hoc explanation methods to interpret black-box policies [12, 31]. These approaches aim to explain decisions after training, often by identifying input regions or features that influence action selection. While flexible and widely applicable, such methods may not faithfully reflect the internal decision process of the model and are limited in their ability to support reasoning, verification, or intervention. In complex manipulation tasks involving multiple objects and temporal dependencies, this gap becomes more pronounced, as explanations may not align with the underlying decision logic. Recent efforts that integrate neural perception

with structured decision models partially address this gap [38, 26], but typically construct decision rules over latent features rather than semantically meaningful variables. In contrast, our approach performs decision making directly over concept representations that are explicitly grounded in the visual scene. By combining reliable concept supervision with a structured decision mechanism, our method enables interpretable, verifiable, and intervenable reasoning in high-dimensional, sequential manipulation tasks.

**Interpretable Representation** Concept-based interpretability introduces human-interpretable intermediate variables (e.g., concepts) to expose the internal reasoning of neural networks [20]. A representative paradigm is the concept bottleneck model (CBM) [17], which decomposes prediction into concept inference followed by decision making over the concept space. This formulation provides a structured interface for inspection and intervention, and has been widely adopted as a foundation for interpretable learning. However, despite these advantages, existing CBM-based approaches share several fundamental limitations. First, concept representations are often either manually annotated or approximated through pretrained models, which can lead to a mismatch between the defined concepts and the actual visual properties in the environment. In particular, similarity-based supervision with pretrained models (e.g., CLIP [32] or GroundingDINO [21]) commonly used in label-free settings [28] does not guarantee that a concept is truly present in the scene, making it difficult to reliably capture task-relevant properties such as object states or spatial relations. Moreover, CBMs typically employ simple decision functions over concept representations, most commonly linear classifiers [42, 35, 15, 22]. In such formulations, interpretability is primarily reflected through concept weights, which provide limited insight into how multiple concepts interact to produce a decision. As a result, while individual concepts are interpretable, the overall decision process remains implicit and lacks a clear reasoning structure. Moreover, most CBM-based methods are developed and evaluated in classification settings, where concepts are relatively stable and well-aligned with semantic categories. In contrast, embodied manipulation tasks introduce additional challenges, including dynamic scenes, fine-grained object interactions, and strong temporal dependencies. These factors require concept representations that are not only semantically meaningful but also sufficiently reliable to support downstream decision making in structured environments.

## 3 ConceptTree

In this section, we present ConceptTree, a novel framework for interpretable high-level policy learning that unifies semantically grounded representation learning with structured decision making. Our approach decomposes the policy into two complementary components: a concept layer that maps visual observations to interpretable representations (concepts), and a decision module that performs reasoning over these concepts to select skills. This design allows decisions to be interpreted at both the representation and decision levels. Specifically, concept values provide a human-interpretable semantic description of the perceived scene, while the decision tree exposes the reasoning process that leads to the final skill. The overall pipeline is illustrated in Fig. 1.

We consider high-level decision making in long-horizon robotic manipulation under a hierarchical control framework. At each time step $t$, the agent selects a discrete skill $a_t \in \mathcal{A}$ from a predefined task-relevant skill set $\mathcal{A}$, which is then executed by a low-level policy. Our focus is on learning an interpretable high-level policy that maps visual observations to skill decisions. At time step $t$, the agent receives an RGB observation $o_t \in \mathbb{R}^{H\times W\times 3}$. Instead of directly mapping observations to skills, we introduce an intermediate concept representation that captures semantically meaningful properties of the scene. Specifically, each observation is mapped to a concept vector $\mathbf{c}_t \in \mathbb{R}^M$, where each dimension corresponds to a task-relevant concept (e.g., object states or spatial relations). To model short-term temporal dependencies, we optionally incorporate context from the previous step. Rather than feeding multiple frames into the perception module, we maintain a modular design in which each observation is processed independently into its concept representation. Temporal information is then incorporated at the decision level by combining concept vectors from consecutive steps. The effective input to the policy is therefore defined as

$$\tilde{\mathbf{c}}_t = \begin{cases} \mathbf{c}_t, & \text{w/o history} \\ [\mathbf{c}_{t-1}, \mathbf{c}_t], & \text{w/ history} \end{cases} \tag{1}$$

The high-level policy $\pi_\theta(a_t|\tilde{\mathbf{c}}_t)$ is thus formulated as a mapping from concept representations to actions. This formulation separates perception and decision making into two interpretable stages. The

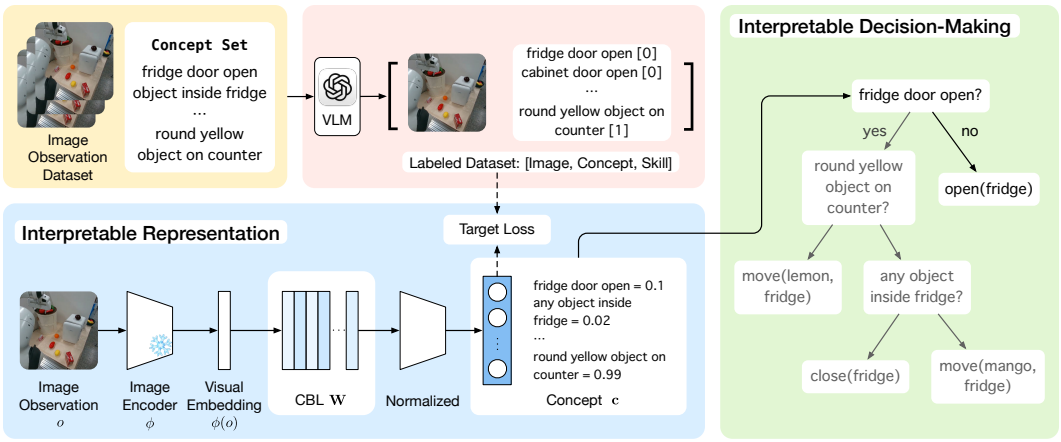


Figure 1: **Overview of the ConceptTree framework.** The model consists of three main components: (i) a visual encoder that extracts features from observations, (ii) a concept projection layer that maps features to a structured concept representation, and (iii) a decision tree that predicts high-level skills based on the concept vector. During training, a VLM provides binary concept judgments for supervision, while at inference time, the system operates without access to the VLM, relying solely on the learned concept representation and decision tree.

concept representation $\mathbf{c}_t$ provides a semantically grounded description of the scene, while the policy $\pi_\theta$ defines an explicit reasoning process over these concepts. This decomposition enables decisions to be analyzed, verified, and modified at the level of individual concepts, facilitating interpretable and controllable policy behavior.

### 3.1 Concept Supervision via VLM Annotation

To learn semantically grounded concept representations without manual annotation, we construct concept-level supervision using a VLM (e.g., GPT-5.1) as an external annotator. We first obtain a predefined task-relevant concept set $\mathcal{C} = \{c^{(1)}, \ldots, c^{(M)}\}$ from a large language model (LLM), which generates task-relevant concepts from high-level task specifications (see Appendix A for more details). Here, $c^{(k)}$ denotes the $k$-th semantic concept in the predefined concept set. Given a dataset of observations and ground-truth skill labels $\mathcal{D} = \{(o_t, a_t^*)\}_{t=1}^T$, where $a_t^*$ denotes the expert skill at time step $t$, we associate each observation $o_t$ with a predefined set of $M$ task-relevant concepts. For each concept $i \in \{1, \ldots, M\}$, we define a binary predicate over the observation, indicating whether the concept holds in the scene. We use a VLM to approximate these predicates. Specifically, for each observation $o_t$ and concept $\mathbf{c} \in \mathcal{C}$, the VLM produces a binary judgment:

$$y_t(c^{(k)}) \in \{0, 1\}, \tag{2}$$

where $y_t(c^{(k)}) = 1$ indicates that concept $c^{(k)}$ is present in $o_t$, and 0 otherwise. Collectively, this yields a concept label vector $\mathbf{y}_t = \big(y_t(c^{(1)}), \ldots, y_t(c^{(M)})\big)^\top \in \{0, 1\}^M$ for each observation. This process defines an augmented dataset $\mathcal{D}_c = \{(o_t, \mathbf{y}_t, a_t^*)\}_{t=1}^T$, which provides supervision for both concept learning and downstream decision making.

Importantly, the VLM is used only during dataset construction and is not required at inference time. This can be viewed as a form of supervision distillation, where concept-level reasoning from a large model is transferred into a lightweight and structured representation. Compared to similarity-based supervision, which produces continuous scores, the proposed verification scheme provides explicit binary signals aligned with task semantics, enabling more reliable supervision for concepts involving object states and spatial relations.

### 3.2 Concept Learning via Projection and Task-Aware Supervision

Given the concept supervision defined in Section 3.1, we learn a concept representation that maps visual observations to semantically grounded concept values.

**Concept Projection.** Each observation $o_t$ is first encoded into a visual feature vector: $h_t = \phi(o_t), h_t \in \mathbb{R}^d$, where $\phi$ denotes a fixed visual encoder. We then project the feature into the concept space $z_t = \mathbf{W}h_t, z_t \in \mathbb{R}^M$, where $\mathbf{W} \in \mathbb{R}^{M\times d}$ maps visual features to $M$ concept dimensions. To obtain normalized concept values, we apply a per-dimension calibration:

$$\mathbf{c}_t^{(k)} = \sigma(p_k z_t^{(k)} + q_k), \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function, $(p_k, q_k)$ are learnable parameters for each concept, and $\mathbf{c}_t^{(k)} \in (0, 1)$ denotes the learned concept representation predicted from observation $o_t$. Using the binary concept labels $y_t(c^{(k)})$ obtained from VLM verification, we train the concept representation with a binary cross-entropy loss:

$$\mathcal{L}_{\text{concept}} = \sum_{t=1}^{T} \text{BCE}(\mathbf{c}_t, \mathbf{y}_t), \tag{4}$$

where $\mathbf{c}_t = \left(\mathbf{c}_t^{(1)}, \ldots, \mathbf{c}_t^{(M)}\right)^\top$ is the learned concept representation.

**Skill-Aware Regularization.** In addition to concept supervision, we introduce a skill-aware objective to encourage the learned representation to retain information relevant for decision making. Specifically, we attach a linear classifier $f(\mathbf{c}_t)$ on top of the concept vector and optimize:

$$\mathcal{L}_{\text{skill}} = \sum_{t=1}^{T} \text{CE}(f(\mathbf{c}_t), a_t^*), \tag{5}$$

where $a_t^*$ is the ground-truth skill label. The overall objective for concept learning is:

$$\mathcal{L}_{\text{target}} = \mathcal{L}_{\text{concept}} + \lambda \mathcal{L}_{\text{skill}}, \tag{6}$$

where $\lambda$ controls the trade-off between semantic alignment and skill relevance. The resulting concept layer is not used as a complete predictor by itself. Rather, it serves as an intermediate semantic interface for the high-level policy, which is instantiated with a decision tree to obtain a transparent decision process.

### 3.3 Concept-based Policy Learning with Decision Tree

Given the learned concept representation $\mathbf{c}_t$, we perform high-level decision making by explicitly reasoning over the concept space. We model the high-level policy as a function $g : \mathbb{R}^{\tilde{M}} \to \mathcal{A}$ that maps concept representations to skill decisions $a_t = g(\tilde{\mathbf{c}}_t)$, where $\tilde{M} = M$ or $2M$ depending on whether temporal context (history) is used.

Specifically, we instantiate $g$ as a decision tree, where each internal node corresponds to a binary predicate over a single concept dimension:

$$d_n(\tilde{\mathbf{c}}_t) = \mathbb{I}\left[\tilde{c}_t^{(k)} \leq \tau_n\right], \tag{7}$$

where $k$ indexes a concept dimension and $\tau_n$ is a learned threshold. If $d_n(\tilde{\mathbf{c}}_t) = 1$, the sample follows the negative (left) branch of node $n$; otherwise, it follows the positive (right) branch. A decision corresponds to traversing a path from the root to a leaf node, resulting in a sequence of concept-level predicates:

$$\bigwedge_{n\in\mathcal{P}} d_n(\tilde{\mathbf{c}}_t), \tag{8}$$

where $\mathcal{P}$ denotes the set of nodes along the decision path. This formulation yields an explicit and interpretable reasoning process, where each decision is explained by a conjunction of concept-based conditions. The decision tree is then trained from the concept representation, allowing the reasoning structure to operate over fixed, semantically grounded inputs. Unlike a linear classifier [40] that explains predictions through global weights, the tree assigns each skill prediction to a local sequence of semantic predicates. This makes the selected skill directly traceable to the specific scene conditions tested along the path.

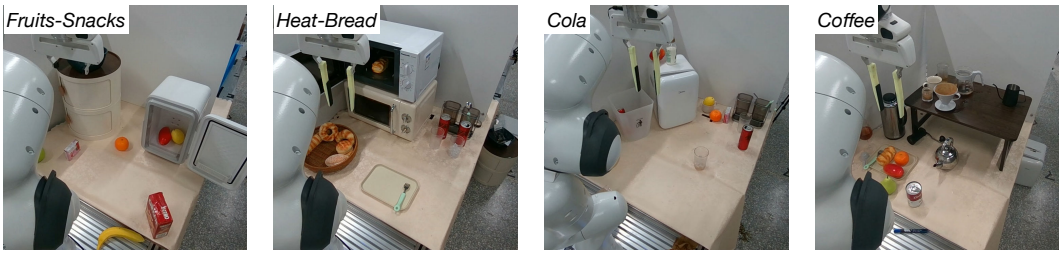


Figure 2: **Tasks evaluated in our study**, including storing fruits and snacks, heating and preparing the bread, preparing a cup of ice cola and storing the milk, manual coffee brewing, with example observations.

**Interpretability and Intervention.** In the formulation, each prediction can be traced to a sequence of concept-level predicates, providing a transparent explanation of the decision. Furthermore, the explicit structure supports targeted intervention. Given a modified concept vector $\tilde{\mathbf{c}}'_t$, the decision outcome can be directly recomputed as $a'_t = g(\tilde{\mathbf{c}}'_t)$, allowing one to analyze how changes in specific concepts affect the final decision. This enables fine-grained diagnosis and correction of errors without modifying the underlying model parameters. In contrast to CBMs, which typically use simple linear classifiers over concept representations, our approach performs structured reasoning through a tree-based decision process. This explicitly separates representation learning from decision making, enabling both semantically grounded perception and transparent, verifiable reasoning.

## 4 Experiment Results

In this section, we evaluate our approach on real long-horizon robotic manipulation tasks to assess both decision performance and interpretability, focusing on how semantically grounded representations and structured decision processes contribute to reliable and controllable behavior.

### 4.1 Experimental Setup

**Tasks.** We evaluate our approach on four long-horizon robotic manipulation tasks with progressively increasing complexity: *Fruits-Snacks*, *Heat-Bread*, *Cola*, and *Coffee*, as illustrated in Fig. 2. These tasks are designed to systematically vary along three key dimensions: temporal dependency, object interaction complexity, and sensitivity to visual details. Specifically,

- *Fruits-Snacks* serves as a baseline task with a fixed sequence of object placements and minimal temporal dependency, where decisions are largely determined by the current observation.
- *Heat-Bread* introduces temporal dependency through skill reuse, requiring the policy to reason about previously executed actions.
- *Cola* further increases difficulty by involving smaller objects and finer spatial interactions, making accurate perception more critical.
- *Coffee* represents the most challenging setting, combining temporal dependency with complex object interactions and sensitivity to subtle visual cues.

**Baselines.** We compare our approach with four representative baselines: (i) an end-to-end **VLM policy** that directly maps observations to skills according to the task description as a black-box model; (ii) **Label-free CBM** [28], a CLIP-based concept model trained with similarity-based supervision; (iii) **VLG-CBM** [35], a VLM-supervised concept model with a linear decision head. The concept-based baselines share the same concept sets as our method. We additionally evaluate (iv) **ConceptSLC**, which uses the same learned concept representation as ConceptTree but replaces the decision tree with the sparse linear classifier [40] used in [28]. This variant isolates the effect of the decision module and tests whether tree-structured reasoning provides benefits beyond sparse linear prediction over the same concepts.

Table 1: **Completion rate comparison on four tasks**. We run 5 seeds for each method (mean $\pm$ std in % and H means history). The maximum depth is set to 7 for ConceptTree across all tasks.

| Category | Method | Task | | | |
|---|---|---|---|---|---|
| | | *Fruits-Snacks* | *Heat-Bread* | *Cola* | *Coffee* |
| w/o H | VLM Policy | $42.50 \pm 1.24$ | $40.00 \pm 1.75$ | $21.33 \pm 0.93$ | $5.09 \pm 2.37$ |
| | Label-free CBM | $7.22 \pm 2.06$ | $6.29 \pm 1.28$ | $15.11 \pm 2.79$ | $9.82 \pm 3.77$ |
| | VLG-CBM | $63.89 \pm 2.60$ | $15.43 \pm 6.34$ | $13.56 \pm 1.45$ | $13.82 \pm 4.91$ |
| | ConceptSLC | $85.71 \pm 0.00$ | $29.71 \pm 1.40$ | $69.78 \pm 2.47$ | $14.55 \pm 6.50$ |
| | ConceptTree (Ours) | $\mathbf{95.56} \pm 2.54$ | $\mathbf{35.71} \pm 7.00$ | $\mathbf{78.44} \pm 5.60$ | $\mathbf{22.55} \pm 14.48$ |
| w/ H | VLM Policy | $43.06 \pm 2.78$ | $68.00 \pm 2.39$ | $19.56 \pm 0.99$ | $5.45 \pm 2.23$ |
| | Label-free CBM | $34.44 \pm 9.29$ | $11.71 \pm 3.56$ | $35.78 \pm 4.11$ | $13.45 \pm 3.04$ |
| | VLG-CBM | $56.39 \pm 17.22$ | $28.86 \pm 3.26$ | $24.67 \pm 5.63$ | $17.09 \pm 4.19$ |
| | ConceptSLC | $\mathbf{100.00} \pm 0.00$ | $79.71 \pm 3.88$ | $87.33 \pm 2.18$ | $18.18 \pm 6.08$ |
| | ConceptTree (Ours) | $94.92 \pm 6.22$ | $\mathbf{84.57} \pm 4.64$ | $\mathbf{95.11} \pm 4.48$ | $\mathbf{37.09} \pm 17.36$ |

**Evaluation Metrics.** In long-horizon tasks, intuitively, an early wrong skill usually indicates a fundamental decision failure, whereas a late error may occur after most high-level decisions have already been made correctly. We evaluate high-level policies on real robot manipulation episodes using completion rate (CR), which measures how far a predicted skill sequence progresses before the first skill decision error. For an episode with length $T$, CR is defined as

$$\text{CR} = \frac{1}{T} \max_{m} \{a_t = a_t^*, \forall t \leq m\}. \tag{9}$$

### 4.2 Main Results

Unless otherwise specified, we use a fixed maximum tree depth of 7 for ConceptTree, which provides a good balance between decision performance and interpretability according to the ablation study. As shown in Table 1, ConceptTree achieves the strongest overall completion rates across the evaluated tasks, with particularly clear gains on the more challenging *Heat-Bread*, *Cola*, and *Coffee* tasks. Compared with the black-box VLM policy, ConceptTree is substantially more reliable, suggesting that directly querying a single VLM for high-level skill selection is insufficient for long-horizon manipulation. Among concept-based methods, Label-free CBM and VLG-CBM perform poorly as task complexity increases, indicating that concept bottlenecks alone do not guarantee robust high-level decision making. More importantly, ConceptTree consistently outperforms ConceptSLC on the temporally dependent and visually demanding tasks, although ConceptSLC slightly outperforms it on the easier *Fruits-Snacks* task with history. Since ConceptSLC uses the same learned concept representation as ConceptTree and differs only in the decision module, this comparison shows that the performance gain does not come merely from better concept learning. Instead, structured reasoning over concepts is critical for robust high-level skill selection. As further analyzed in Appendix F, ConceptTree also provides more compact explanations than the sparse linear model, using fewer concepts and shorter decision paths for test-time predictions. Overall, these results support our claim that ConceptTree benefits from making decisions explicitly in the concept space through an interpretable tree-structured process.

### 4.3 Ablation Studies

We analyze how different design components affect both decision performance and the structure of the learned decision tree. All ablation studies are conducted offline on a held-out validation set constructed from recorded robot observations. Results are summarized in Fig. 3 and Fig. 4. Across all settings, we observe that better performance consistently correlates with simpler decision structures, characterized by shallower trees, fewer concepts, and shorter decision paths.

**Effect of supervision quality (w/ CLIP target and w/o skill loss).** Concept supervision plays a dominant role in overall performance. Replacing VLM-based supervision with CLIP similarity

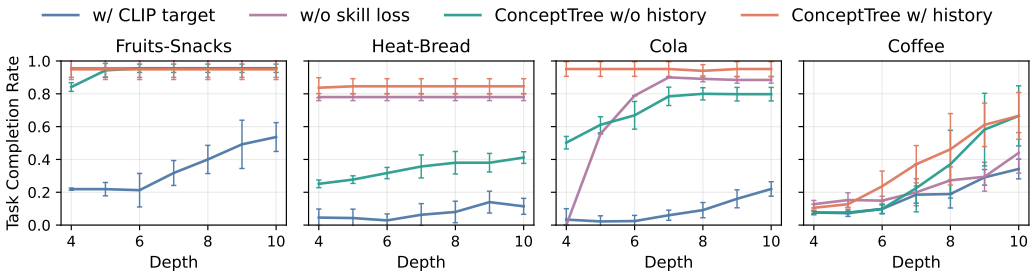


Figure 3: **Ablation study across four tasks.** Bars indicate standard deviation. Depth denotes the maximum allowed tree depth during training, not the actual depth.

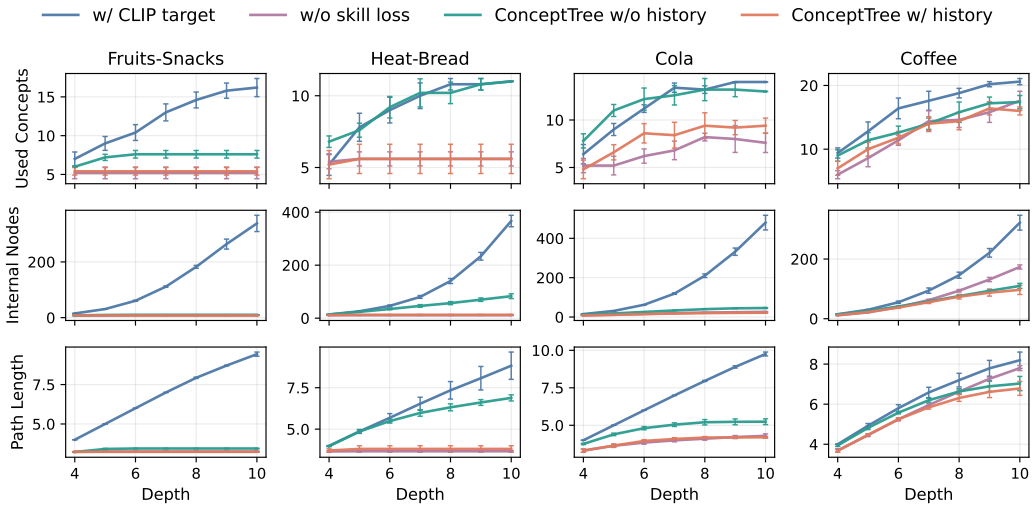


Figure 4: **Tree complexity under different ablation settings.** We report the number of used concepts and leaf nodes and bars indicate standard deviation.

leads to a substantial drop in performance across all tasks, accompanied by significantly larger trees with increased concept usage and longer decision paths. In contrast, removing the skill loss results in a more moderate performance degradation. These results indicate that the quality of concept supervision is critical for learning compact and effective decision structures, while task-aligned training provides additional but comparatively limited improvements.

**Effect of temporal context (w/o history).** Removing history input has little effect on simpler tasks such as *Fruits-Snacks*, but leads to clear performance drops in tasks with stronger temporal dependencies like *Heat-Bread* or *Coffee*. In contrast, the tree structure remains largely unchanged, suggesting that temporal information primarily improves decision accuracy under partial observability, rather than altering the structure of the learned policy. The impact of temporal context depends on task properties.

**Effect of tree depth.** Increasing the maximum tree depth improves performance up to a saturation point, reflecting increased expressive capacity. Notably, ConceptTree achieves strong performance even with very shallow trees (e.g., depth = 4) across most tasks. At this depth, the resulting decision paths are also short (with an average length below 4), indicating that accurate decisions can be made with only a few concept-level predicates. This suggests that the learned policies remain both effective and interpretable, without requiring deep or complex decision structures.

**Summary.** Across all settings, we observe a consistent relationship between concept quality, structural efficiency, and performance. Stronger concept supervision and task-aware training lead to more compact, selective, and shorter decision paths, whereas weaker supervision results in larger trees with longer and less efficient decision chains. These findings suggest that performance gains arise not only

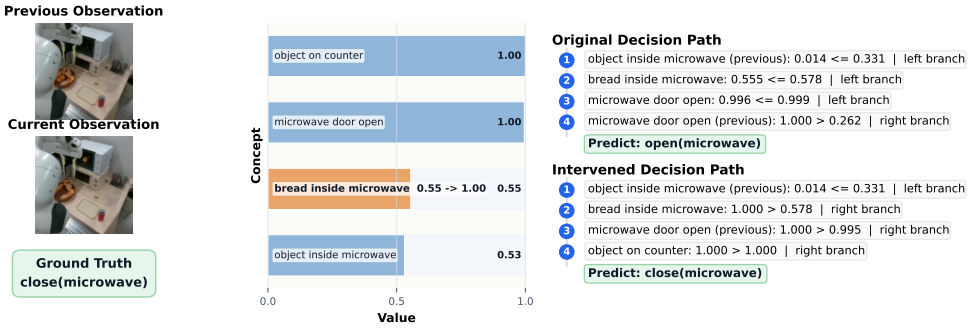


Figure 5: **Decision-path diagnosis and concept intervention.** In this *Heat-Bread* example, the original path predicts `open(microwave)` because the value of *bread inside microwave* falls below the split threshold. This is inconsistent with the observation, where the bread is already inside the microwave, so the concept is falsely treated as negative. Manually correcting this single concept value redirects the tree traversal and changes the prediction to the ground-truth skill `close(microwave)`.

from improved concept representations, but critically from how these concepts are structured into coherent and efficient decision processes.

### 4.4 Case Study

To further analyze our method, we present a qualitative case study illustrating the decision process and its response to concept-level intervention. As shown in Fig. 5, we show an example from the *Heat-Bread* task. By inspecting the concept values, we observe that the value of the concept *bread inside microwave* is underestimated ($0.555 < 0.578$), which leads to an incorrect decision.

The decision tree provides an explicit reasoning path for this prediction. As shown in the original decision path, the low activation value of the concept *bread inside microwave* causes the traversal to follow branches corresponding to the bread not being inside the microwave. This results in selecting the incorrect skill. We then perform an intervention by modifying the concept *bread inside microwave* to 1.0, reflecting the correct semantic state (see identification details in Appendix G). After the modification, the decision path changes accordingly, and the model predicts the correct skill `close(microwave)`. Notably, this intervention alters only a single concept value while keeping all other inputs unchanged. This example highlights two key properties of our approach. First, decisions are explicitly grounded in interpretable concept-level predicates, allowing errors to be traced to specific semantic variables. Second, the structured decision process enables fine-grained intervention, where modifying a concept can directly correct the model's behavior without retraining.

## 5 Conclusion

We presented ConceptTree, a framework for interpretable high-level decision making in robotic manipulation that unifies semantically grounded representation learning with structured decision processes. By introducing a concept layer trained with verification-based supervision and a decision tree that performs reasoning over concept representations, our approach enables decisions to be interpreted at both the representation and decision levels. Through experiments on long-horizon manipulation tasks, we showed that ConceptTree achieves strong decision performance while providing transparent and controllable behavior. In particular, our results demonstrate that decision errors can be traced to specific concept dimensions and corrected through targeted intervention, highlighting the practical value of concept-based reasoning for debugging and improving policy behavior. Overall, this work suggests that combining semantically grounded representations with explicit decision structures is a promising direction for building interpretable and reliable robotic systems. Despite these advantages, our approach has several limitations. The quality of concept representations depends on VLM-based supervision, which may be unreliable for subtle visual states or complex spatial relations. Moreover, the method relies on predefined concept and skill sets, which may limit scalability to more open-ended tasks.

## Acknowledgments and Disclosure of Funding

## References


[1] Shuanghao Bai, Wenxuan Song, Jiayi Chen, Yuheng Ji, Zhide Zhong, Jin Yang, Han Zhao, Wanqi Zhou, Wei Zhao, Zhe Li, et al. Towards a unified understanding of robot manipulation: A comprehensive survey. *arXiv preprint arXiv:2510.10903*, 2025.

[2] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in Neural Information Processing Systems*, 31, 2018.

[3] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *9th Annual Conference on Robot Learning*, 2025.

[4] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, pages 287–318. PMLR, 2023.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[6] Yashesh Dhebar and Kalyanmoy Deb. Interpretable rule discovery through bilevel optimization of split-rules of nonlinear decision trees for classification problems. *IEEE Transactions on Cybernetics*, 51(11):5573–5584, 2020.

[7] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. Toward interpretable-ai policies using evolutionary nonlinear decision trees for discrete-action systems. *IEEE Transactions on Cybernetics*, 54(1):50–62, 2022.

[8] Yan Ding, Xiaohan Zhang, Chris Paxton, and Shiqi Zhang. Task and motion planning with large language models for object rearrangement. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2086–2092. IEEE, 2023.

[9] Yilun Du, Sherry Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B Tenenbaum, Leslie Pack Kaelbling, et al. Video language planning. In *The Twelfth International Conference on Learning Representations*, 2024.

[10] Mark Edmonds, Feng Gao, Hangxin Liu, Xu Xie, Siyuan Qi, Brandon Rothrock, Yixin Zhu, Ying Nian Wu, Hongjing Lu, and Song-Chun Zhu. A tale of two explanations: Enhancing human trust by explaining robot behavior. *Science Robotics*, 4(37):eaay4663, 2019.

[11] Claire Glanois, Paul Weng, Matthieu Zimmer, Dong Li, Tianpei Yang, Jianye Hao, and Wulong Liu. A survey on interpretable reinforcement learning. *Machine Learning*, 113(8):5847–5890, 2024.

[12] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International Conference on Machine Learning*, pages 1792–1801. PMLR, 2018.

[13] Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16 (1):45–74, 2024.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] Lijie Hu, Tianhao Huang, Huanyi Xie, Xilin Gong, Chenyang Ren, Zhengyu Hu, Lu Yu, Ping Ma, and Di Wang. Semi-supervised concept bottleneck models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2110–2119, 2025.

[16] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In *Conference on Robot Learning*, pages 2679–2713. PMLR, 2025.

[17] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

[18] Hector Kohler, Quentin Delfosse, Riad Akrour, Kristian Kersting, and Philippe Preux. Interpretable and editable programmatic tree policies for reinforcement learning. In *Workshop on Interpretable Policies in Reinforcement Learning@ RLC-2024*, 2024.

[19] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of Machine Learning Research*, 22 (30):1–82, 2021.

[20] Jae Hee Lee, Georgii Mikriukov, Gesina Schwalbe, Stefan Wermter, and Diedrich Wolter. Concept-based explanations in computer vision: Where are we and where could we go? In *European Conference on Computer Vision*, pages 266–287. Springer, 2024.

[21] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, pages 38–55. Springer, 2024.

[22] Yang Liu, Tianwei Zhang, and Shi Gu. Hybrid concept bottleneck models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 20179–20189, 2025.

[23] Zhenyu Lu, Ning Wang, and Chenguang Yang. A dynamic movement primitives-based tool use skill learning and transfer framework for robot manipulation. *IEEE Transactions on Automation Science and Engineering*, 22:1748–1763, 2024.

[24] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105):eads5033, 2025.

[25] Lirui Luo, Guoxi Zhang, Hongming Xu, Yaodong Yang, Cong Fang, and Qing Li. End-to-end neuro-symbolic reinforcement learning with textual explanations. In *International Conference on Machine Learning*, pages 33533–33557. PMLR, 2024.

[26] Sascha Marton, Tim Grams, Florian Vogt, Stefan Lüdtke, Christian Bartelt, and Heiner Stuckenschmidt. Mitigating information loss in tree-based reinforcement learning via direct optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[27] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. Explainable reinforcement learning: A survey and comparative review. *ACM Computing Surveys*, 56(7):1–36, 2024.

[28] Tuomas Oikarinen, Subhro Das, Lam M Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] Rohan Paleja, Yaru Niu, Andrew Silva, Chace Ritchie, Sugju Choi, and Matthew Gombolay. Learning interpretable, high-performing policies for autonomous driving. In *Robotics science and systems*, 2022.

[30] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[31] Nikaash Puri, Sukriti Verma, Piyush Gupta, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using specific and relevant feature attribution. In *International Conference on Learning Representations*, 2020.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[33] David Sobrín-Hidalgo, Ángel Manuel Guerrero-Higueras, and Vicente Matellán-Olivera. Generating explanations for autonomous robots: a systematic review. *IEEE Access*, 2025.

[34] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2998–3009, 2023.

[35] Divyansh Srivastava, Ge Yan, and Tsui-Wei Weng. Vlg-cbm: Training concept bottleneck models with vision-language guidance. *Advances in Neural Information Processing Systems*, 37:79057–79094, 2024.

[36] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6166–6175. PMLR, 09–15 Jun 2019.

[37] Maxime Wabartha and Joelle Pineau. Piecewise linear parametrization of policies: Towards interpretable deep reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[38] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Suzanne Petryk, Sarah Adel Bargal, and Joseph E Gonzalez. Nbdt: Neural-backed decision tree. In *International Conference on Learning Representations*, 2021.

[39] Yongyan Wen, Siyuan Li, Rongchang Zuo, Lei Yuan, Hangyu Mao, and Peng Liu. Skilltree: Explainable skill-based deep reinforcement learning for long-horizon control tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 21491–21500, 2025.

[40] Eric Wong, Shibani Santurkar, and Aleksander Madry. Leveraging sparse linear layers for debuggable deep networks. In *International Conference on Machine Learning*, pages 11205–11216. PMLR, 2021.

[41] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models. *arXiv preprint arXiv:2307.01848*, 2023.

[42] Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023.

[43] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

# A Concept Set Construction

To construct a task-specific concept set, we query an LLM (e.g., GPT-5.4) with the task description, skill library, and expert execution sequence. The goal is not to obtain generic visual categories, but to identify binary predicates that are observable from the robot's visual input and relevant to high-level skill selection. We focus on three types of concepts:

- **Object presence:** whether a task-relevant object is visible in the scene.
- **Object state:** whether an object has a relevant state, such as open, closed, filled, or empty.
- **Spatial relation:** whether a task-relevant relation holds between objects, such as inside, on, or near.

The generated concepts are then used as the predicate set for VLM-based concept annotation. We denote the finalized concept set as $\mathcal{C} = \{c^{(1)}, \ldots, c^{(M)}\}$, where $c^{(k)}$ is the $k$-th semantic concept. The prompt is shown below.

```
Prompt for concept set construction.

You are constructing a concept set for a robot manipulation task.

Your goal is to generate a fixed set of visual concepts based on given SKILL LIBRARY.

Important notes:
- Concepts must follow a strict naming convention
- Only use the allowed concept types listed below
- Do NOT invent new concept types
- Concepts must be visually observable from a single image

Skill library: [SKILL LIBRARY of the task]

Concept types:
1. Visibility
Format: <object>_visible

2. Spatial Relation (object-object or object-location)
Format: <object>_on_surface, <object>_on_counter, <object>_on_plate, <object>_inside_<
     container>, <object>_above_<object>
Examples: cup_on_surface, funnel_above_pot, milk_bottle_inside_fridge

3. Container State (binary content state)
Format: <object>_empty, <object>_contains_liquid, <object>_contains_dark_liquid
Examples: cup_empty, coffee_cup_contains_liquid, pot_contains_liquid

4. Object State / Configuration
Format: <object>_upright, <object>_tilted
Examples: gooseneck_kettle_upright, gooseneck_kettle_tilted

5. Interaction Proxy (only if visually obvious)
Format: <object>_pouring_to_<object>
Example: gooseneck_kettle_pouring_to_funnel

6. Appliance State
Format: <appliance>_door_open
Examples: fridge_door_open, cabinet_door_open, microwave_door_open

7. Generic Object Group (category abstraction)
Format: any_object_on_counter, any_object_inside_fridge, any_object_inside_cabinet,
     object_inside_<container>, object_on_<location>

8. Shape / Appearance Category (only predefined categories)
Format: round_<color>_object_on_counter, elongated_<color>_object_on_counter, small_<
     color>_box_on_counter
```

```
Examples: round_orange_object_on_counter, elongated_red_object_on_counter,
     small_light_box_on_counter

Output Requirements:
- Return ALL concepts following the formats above
- Do NOT include explanations
- Do NOT generate concepts outside these categories
- Avoid duplicates
```

### A.1 Manual Validation and Final Concept Sets

The LLM-generated concepts are treated as candidate concepts rather than the final concept set. We manually review the candidates using predefined criteria: concepts must be visually observable from the side-camera image, relevant to high-level skill selection, expressible as binary predicates, and non-redundant with other concepts. During this step, we remove ambiguous, non-visual, task-irrelevant, or duplicate concepts. The finalized concept sets are fixed before model training and **shared across ConceptTree and all concept-based baselines** to ensure a fair comparison.

To reduce annotation cost, we exploit the temporal structure of the collected skill demonstrations. For each skill execution, the raw dataset contains 50 side-camera observations captured after the skill starts (Appendix H.5). We request VLM annotations only for the first observation in each 50-frame window and assign the resulting concept labels to the remaining observations in the same window. This keeps the number of VLM queries small while providing concept supervision for all training frames. Although some later observations may occasionally deviate from the first frame in concept semantics, we found this approximation to work well in practice for training the concept layer. Table 2 lists the finalized concept sets used in our experiments.

Table 2: **Finalized task-specific concept sets.**

| Task | Concept Set |
|---|---|
| *Fruits-Snacks* | `round_orange_object_on_counter`; `round_yellow_object_on_counter`; `round_green_object_on_counter`; `elongated_red_object_on_counter`; `elongated_yellow_object_on_counter`; `small_red_box_on_counter`; `small_light_box_on_counter`; `round_object_inside_fridge`; `elongated_object_inside_fridge`; `round_object_inside_cabinet`; `elongated_object_inside_cabinet`; `box_object_inside_cabinet`; `any_object_on_counter`; `any_object_inside_fridge`; `any_object_inside_cabinet`; `fridge_door_open`; `cabinet_door_open` |
| *Heat-Bread* | `microwave_door_open`; `bread_visible`; `bread_on_surface`; `bread_inside_microwave`; `bread_on_plate`; `plate_visible`; `plate_on_surface`; `plate_contains_object`; `object_inside_microwave`; `object_on_plate`; `object_on_counter` |
| *Cola* | `fridge_open`; `cup_visible`; `cup_on_surface`; `cup_inside_fridge`; `cup_contains_liquid`; `can_visible`; `can_on_surface`; `bin_visible`; `object_inside_bin`; `milk_bottle_visible`; `milk_bottle_on_surface`; `milk_bottle_inside_fridge`; `object_inside_fridge`; `object_on_surface` |
| *Coffee* | `funnel_visible`; `pot_visible`; `kettle_visible`; `gooseneck_kettle_visible`; `cup_visible`; `coffee_cup_visible`; `coffee_powder_visible`; `funnel_above_pot`; `funnel_above_table_surface`; `cup_empty`; `cup_contains_liquid`; `coffee_cup_empty`; `cup_contains_dark_liquid`; `coffee_cup_contains_liquid`; `coffee_cup_contains_dark_liquid`; `funnel_empty`; `pot_contains_liquid`; `pot_empty`; `gooseneck_kettle_upright`; `gooseneck_kettle_tilted`; `gooseneck_kettle_pouring_to_funnel` |

### A.2 VLM-based Concept Annotation

Given the task-specific concept set, we construct binary concept labels for each visual observation using GPT-5.1 as an offline verifier. For each image, the VLM evaluates the full list of task-relevant concepts and returns a JSON object with one binary value per concept, where 1 indicates

that the predicate holds in the scene and 0 otherwise. We denote the binary value assigned to concept $c^{(k)}$ for observation $o_t$ as $y_t(c^{(k)})$. The complete VLM-derived concept target vector is $\mathbf{y}_t = \big(y_t(c^{(1)}), \ldots, y_t(c^{(M)})\big)^\top \in \{0,1\}^M$. This vector is used as supervision for training the concept layer.

```
You are a visual reasoning assistant.

You are given:
1) An input image.
2) A list of concepts.

Your task:
For EACH concept, determine whether it is TRUE (1) or FALSE (0) based ONLY on what is
     clearly visible in the image.

IMPORTANT RULES:
- Only use visual evidence from the image.
- Do NOT guess or assume anything not clearly visible.
- If uncertain, output 0.
- Be strict and conservative.
- Each concept must be evaluated independently.

Concept definitions:
- "on_surface":
  an object is clearly placed on a visible flat supporting surface.

- "inside_container":
  an object is clearly enclosed within a container-like structure, where the boundary of
      the container is visible.

- "container_open":
  a container has a visible opening, such that its interior is partially visible.

- "object_visible":
  at least one object is clearly visible in the scene.

- "object_near_boundary":
  an object is close to the edge of a surface or container opening.

Output format (STRICT):
Return ONLY a valid JSON object with NO explanation.

Format:
{{
  "concept_1": 0 or 1,
  "concept_2": 0 or 1,
  ...
}}

Now evaluate the following concepts:

{concept_list}
```

## B Training Details

All experiments are run on a workstation with an Intel Core i9-14900KF CPU with 128GB RAM and a single NVIDIA RTX 4090 GPU. In our setting, the training time is very short because the dataset and model are relatively small, with the concept layer and the decision tree training taking less than 2 minutes for each task.

**Concept layer training.** For visual feature extraction, we use a ResNet-18 [14] backbone pretrained on ImageNet-1K [5] as a fixed encoder. Each preprocessed RGB observation is mapped to a 512-dimensional image feature. The training details are summarized in Table 3.

Table 3: **Training configuration for the ConceptTree concept layer.**

| Hyperparameter | Value |
|---|---|
| Visual encoder | ResNet-18 |
| Image feature dimension | 512 |
| Input resolution | $256 \times 256$ |
| Optimizer | Adam |
| Learning rate | 1e-3 |
| Batch size | 512 |
| Training steps | 1000 |

**Decision tree training.** We use a single axis-aligned decision tree implemented with `sklearn.tree.DecisionTreeClassifier` [30]. The decision tree is trained on the learned concept $\mathbf{c}_t$. For the no-history setting, the tree is trained directly on the concept values of the training observations and their corresponding skill labels. For the history setting, we construct an expanded training set by pairing history frames with current frames. Specifically, for each skill decision, we take the Cartesian product between the frames associated with the previous skill and the frames associated with the current skill, and concatenate their concept values as the tree input. Since the first skill has no previous skill context, we instead form the Cartesian product between its own frames and itself. At inference time, we use the same convention. For the first decision step in the history setting, the current concept vector is copied as the history input. Each paired sample is labeled with the current skill. The tree depth is controlled by a maximum depth parameter $d$, the split quality is measured by information gain using the entropy criterion, and class imbalance is handled with balanced class weights. Deeper trees are more expressive, but they reduce interpretability. We select a fixed depth of 7 for all tasks based on ablation analysis, which provides a good balance between performance and interpretability. In practice, we find that $d = 4$ works well for most tasks.

## C Ablation Study on Skill Loss Weight

We ablate the skill-aware loss weight $\lambda \in \{0, 0.05, 0.1, 0.2, 0.5, 1.0\}$ with the history input and evaluate each weight across tree depths $d \in [4, 10]$. For each task and weight, we report the best-performing depth in Table 4. Overall, introducing a nonzero skill-aware loss consistently improves over the w/o skill loss baseline. The task-level behavior is non-uniform. On *Coffee*, which exhibits the strongest visual ambiguity, larger skill-aware weights are most beneficial, the completion rate improves from $44.0\%$ at $\lambda = 0$ to $70.91\%$ at $\lambda = 0.2$ and further to $80.36\%$ at $\lambda = 1.0$. In contrast, *Cola* prefers much smaller weights, the best result is already obtained at $\lambda = 0.05$ ($95.33\%$), while larger weights substantially degrade performance (e.g., $73.78\%$ at $\lambda = 0.5$). *Heat-Bread* is best at $\lambda = 0.1$, whereas *Fruits-Snacks* peaks at $\lambda = 0.2$ with perfect test performance. These results suggest that the skill-aware loss primarily acts as a representation-shaping mechanism. Stronger skill supervision is especially useful in tasks where stage boundaries are visually ambiguous, as in *Coffee*, but it can over-distort the concept space in easier tasks where the representation is already sufficiently separable. We set $\lambda = 0.1$ for *Fruits-Snacks*, *Heat-Bread*, and *Cola* and $\lambda = 0.5$ for *Coffee* in the main results.

## D Concept Importance Analysis

We analyze the interpretability of the decision tree by tracing the concepts that appear along test-time decision paths. For each test step, we record the concepts used by the executed path, normalize their frequencies within that path, and then average the normalized contributions over the entire test set. Importantly, we perform the analysis at the level of individual episode steps (skills) rather than globally merging all occurrences of the same skill, since the same skill can be invoked multiple times under different visual contexts. We compare the resulting concept importance distributions for ConceptTree w/ history and ConceptTree w/o history, using a fixed depth per task.

Table 4: **Skill-aware loss weight ablation under the history input setting.** We report completion rate with mean±std in %, and the best depth $d$ for the corresponding task and weight.

| Parameter | *Fruits-Snacks* | *Heat-Bread* | *Cola* | *Coffee* |
|---|---|---|---|---|
| $\lambda = 0.0$ | $95.56 \pm 5.54$ ($d = 4$) | $78.00 \pm 2.84$ ($d = 4$) | $90.00 \pm 0.00$ ($d = 7$) | $44.00 \pm 12.35$ ($d = 10$) |
| $\lambda = 0.05$ | $97.14 \pm 2.54$ ($d = 4$) | $76.86 \pm 1.40$ ($d = 5$) | $\mathbf{95.33 \pm 3.17}$ ($d = 7$) | $58.91 \pm 10.45$ ($d = 10$) |
| $\lambda = 0.1$ | $94.92 \pm 6.22$ ($d = 4$) | $\mathbf{84.57 \pm 4.64}$ ($d = 5$) | $95.11 \pm 4.48$ ($d = 4$) | $58.55 \pm 6.34$ ($d = 10$) |
| $\lambda = 0.2$ | $\mathbf{100.00 \pm 0.00}$ ($d = 4$) | $80.57 \pm 10.29$ ($d = 5$) | $90.44 \pm 4.61$ ($d = 6$) | $70.91 \pm 11.33$ ($d = 10$) |
| $\lambda = 0.5$ | $96.83 \pm 4.38$ ($d = 4$) | $78.29 \pm 17.94$ ($d = 4$) | $73.78 \pm 5.10$ ($d = 8$) | $66.55 \pm 14.25$ ($d = 10$) |
| $\lambda = 1.0$ | $92.38 \pm 4.86$ ($d = 4$) | $74.00 \pm 14.04$ ($d = 4$) | $75.11 \pm 4.13$ ($d = 8$) | $\mathbf{80.36 \pm 5.06}$ ($d = 10$) |

The results in Fig. 6 show that the most influential concepts are highly consistent across the history and no-history variants, and they align well with human intuition about which visual cues should matter for each task. In other words, the tree tends to rely on semantically meaningful concepts, such as object state, container state, or spatial relations, rather than obscure or obviously spurious signals. This consistency suggests that adding history improves temporal context without changing the semantic basis of the decision process. Therefore, the decision tree is interpretable not only because it exposes an explicit rule path, but also because the concepts that dominate its predictions remain stable and human-understandable across settings.

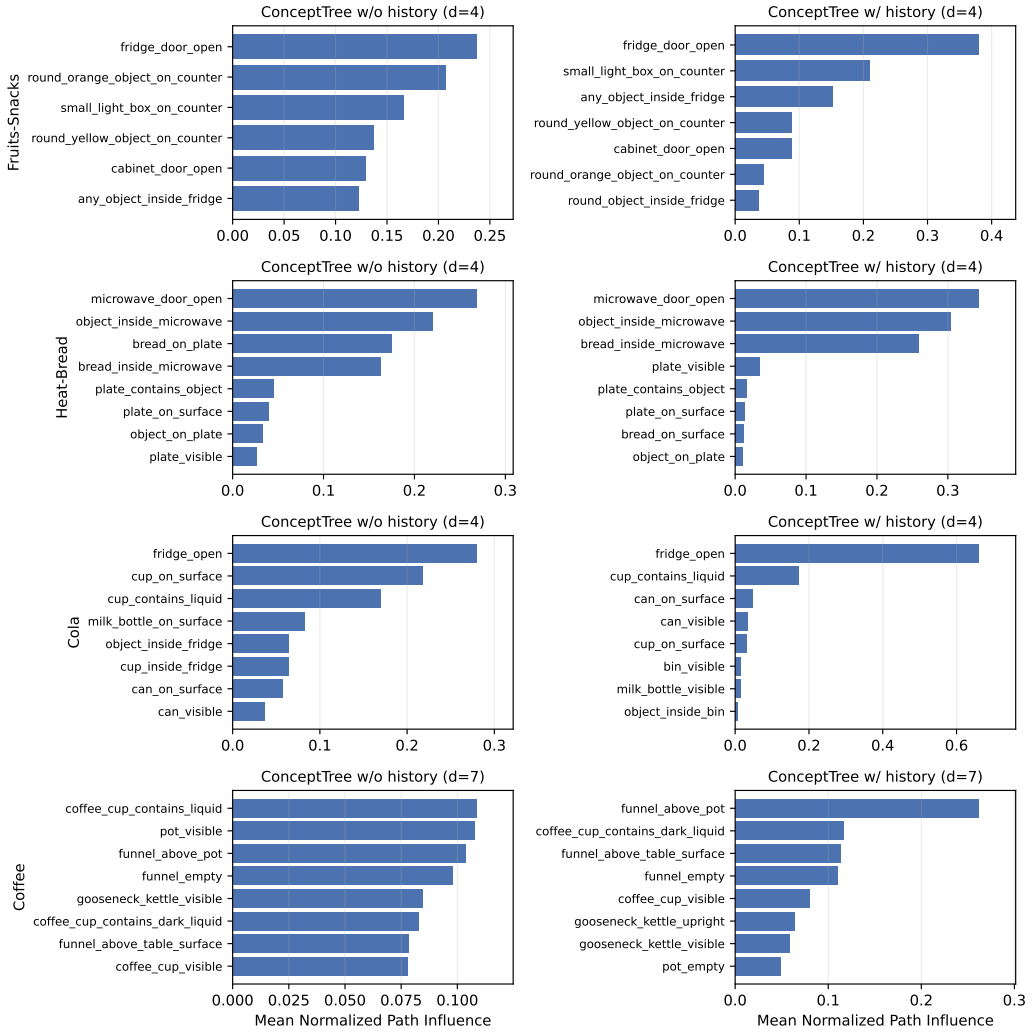


Figure 6: **Visualization of concept importance in decision paths across tasks.**

# E Decision Tree Visualization

We visualized the learned decision trees for each task in Fig. 7, 8, 9 and 10. These visualizations expose the actual sequence of concept predicates used by the high-level policy and make it possible to inspect how different branches correspond to different skill choices.

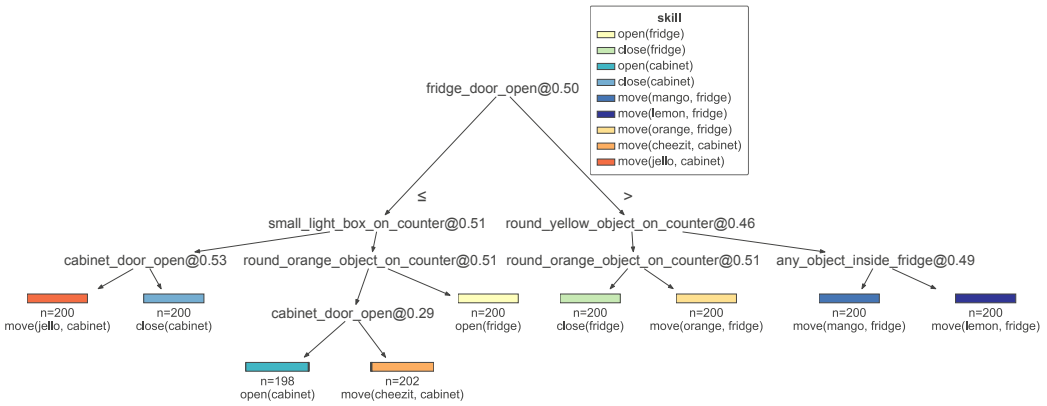


Figure 7: **Visualization of decision tree for the *Fruits-Snacks* task.**

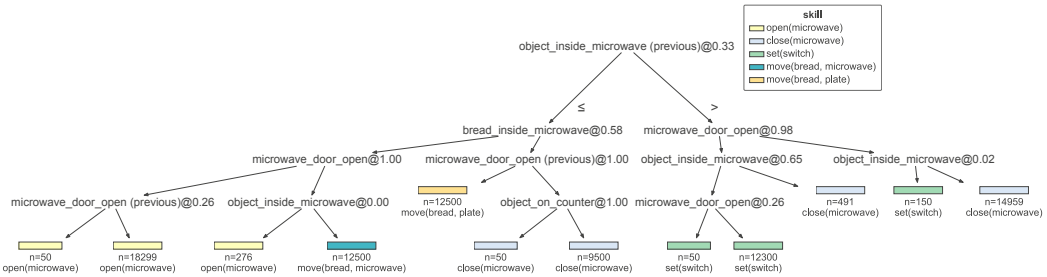


Figure 8: **Visualization of decision tree for the *Heat-Bread* task.**

# F Decision Tree vs. Sparse Linear Model

We further compare the interpretability of a single decision tree and a sparse linear classifier [40] when both operate on the **same task-specific concept representation** for a fair comparison. For the tree model, we use the corresponding ConceptTree model and report complexity statistics at maximum depth $d = 7$. For ConceptTree, we report local decision complexity, including UC (Used Concepts on test-time decisions), IN (Internal Nodes, standing for the number of parameters), and APL (Average Path Length). For the sparse linear model, we report global sparsity, including UC (Used Concepts), NZW (total NonZero Weights, indicating the effective parameter count), and CPS (Concepts Per Skill, i.e., the average number of nonzero concept coefficients for each skill class).

The tree model uses a small number of concepts at test time, and each prediction corresponds to a short rule chain. As shown in Table 5, ConceptTree uses fewer concepts than the sparse linear model on *Fruits-Snacks*, *Heat-Bread*, and *Cola*, while keeping the average decision path within roughly three to four predicates. On the more difficult *Coffee* task, the number of active concepts becomes comparable, but the tree still produces a local path of only $5.84 \pm 0.09$ predicates on average. In contrast, the sparse linear classifier relies on distributed evidence, with 95-152 nonzero weights and roughly 15–17 concept coefficients per skill across tasks. This indicates that global sparsity in the parameter matrix does not necessarily translate into a concise explanation for an individual prediction.

We further perform a concept-budget analysis for the sparse linear classifier by retaining only the top-$K$ globally important concepts and masking the rest. The results show that sparse linear performance

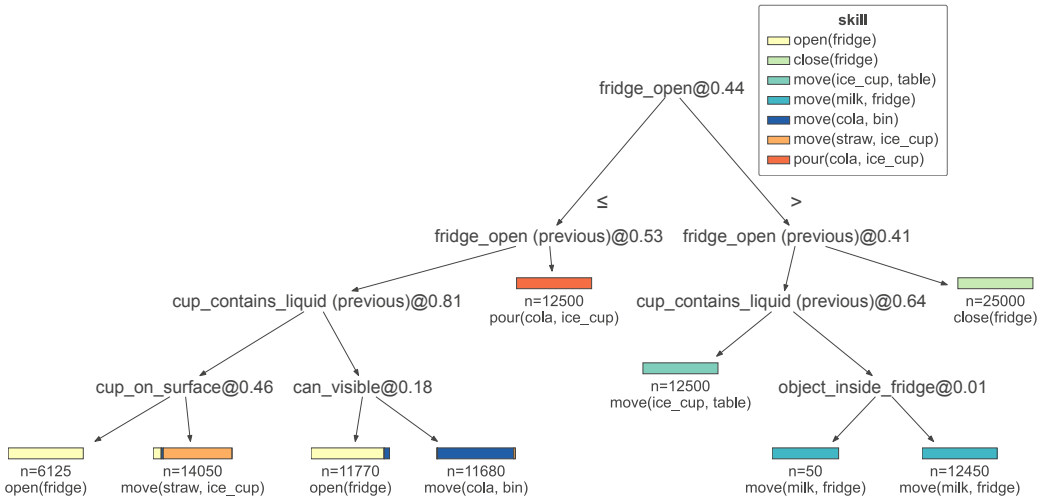


Figure 9: **Visualization of decision tree for the *Cola* task.**

Table 5: **Interpretability statistics for ConceptTree and the sparse linear classifier.** ConceptTree statistics measure concepts actually used along test-time decision paths, while sparse linear statistics measure global nonzero weights in the classifier.

| Task | ConceptTree | | | Sparse Linear Model | | |
|---|---|---|---|---|---|---|
| | UC↓ | IN↓ | APL↓ | UC↓ | NZW↓ | CPS↓ |
| *Fruits-Snacks* | 5.40±0.49 | 8.00±0.00 | 3.23±0.02 | 16.0±0.0 | 143.0±2.2 | 15.7±0.2 |
| *Heat-Bread* | 5.60±1.02 | 12.60±1.36 | 3.81±0.19 | 11.0±0.0 | 95.6±1.9 | 19.0±0.4 |
| *Cola* | 8.40±1.36 | 17.60±0.49 | 4.09±0.09 | 14.0±0.0 | 119.2±1.5 | 17.3±0.2 |
| *Coffee* | 14.00±1.10 | 55.40±4.50 | 5.84±0.09 | 13.4±0.8 | 152.2±7.2 | 18.7±0.9 |

usually requires a relatively large concept set. Completion rate is recovered only with about 12 concepts on *Cola*, nearly all 11 concepts on *Heat-Bread*, and 15 concepts on *Coffee*, while the easier *Fruits-Snacks* task is recovered with the top 7 concepts. This suggests that sparse linear models remain globally sparse but rely on broadly distributed concept evidence, especially as task complexity increases. For comparison, ConceptTree uses a much smaller set of concepts while achieving better performance.

**Concept influence comparison.** We analyze test-time concept influence for both model families. For ConceptTree, the influential concepts remain highly consistent between the history and no-history variants, as shown in Fig. 6, and align with intuitive task cues such as object state and spatial relations. The tree also exhibits sharper influence distributions, where a few dominant concepts explain most of the decision path. In contrast, the sparse linear weights in Fig. 11 are spread more evenly across multiple concepts, suggesting that its predictions are supported by distributed evidence rather than a compact set of decisive predicates. These results suggest that the tree bases its decisions on stable and semantically meaningful concepts while providing a more localized decision path than the linear model.

Overall, the two models provide different forms of interpretability. A sparse linear classifier can be sparse at the parameter level, but its predictions are supported by distributed weighted evidence and do not yield a concise local reasoning chain. This is limiting for robotic sequential decision making, where each skill choice affects later observations and failures must be traced to the specific scene condition that caused an incorrect branch in the task. In contrast, ConceptTree exposes a short concept-level path for each prediction, making the selected skill attributable to a small and stable chain of semantic predicates. Therefore, compared with a sparse linear model, the tree is better suited to interpretable high-level robotic decision making: it not only uses concepts selectively, but also provides stronger local interpretability for step-wise diagnosis and intervention.

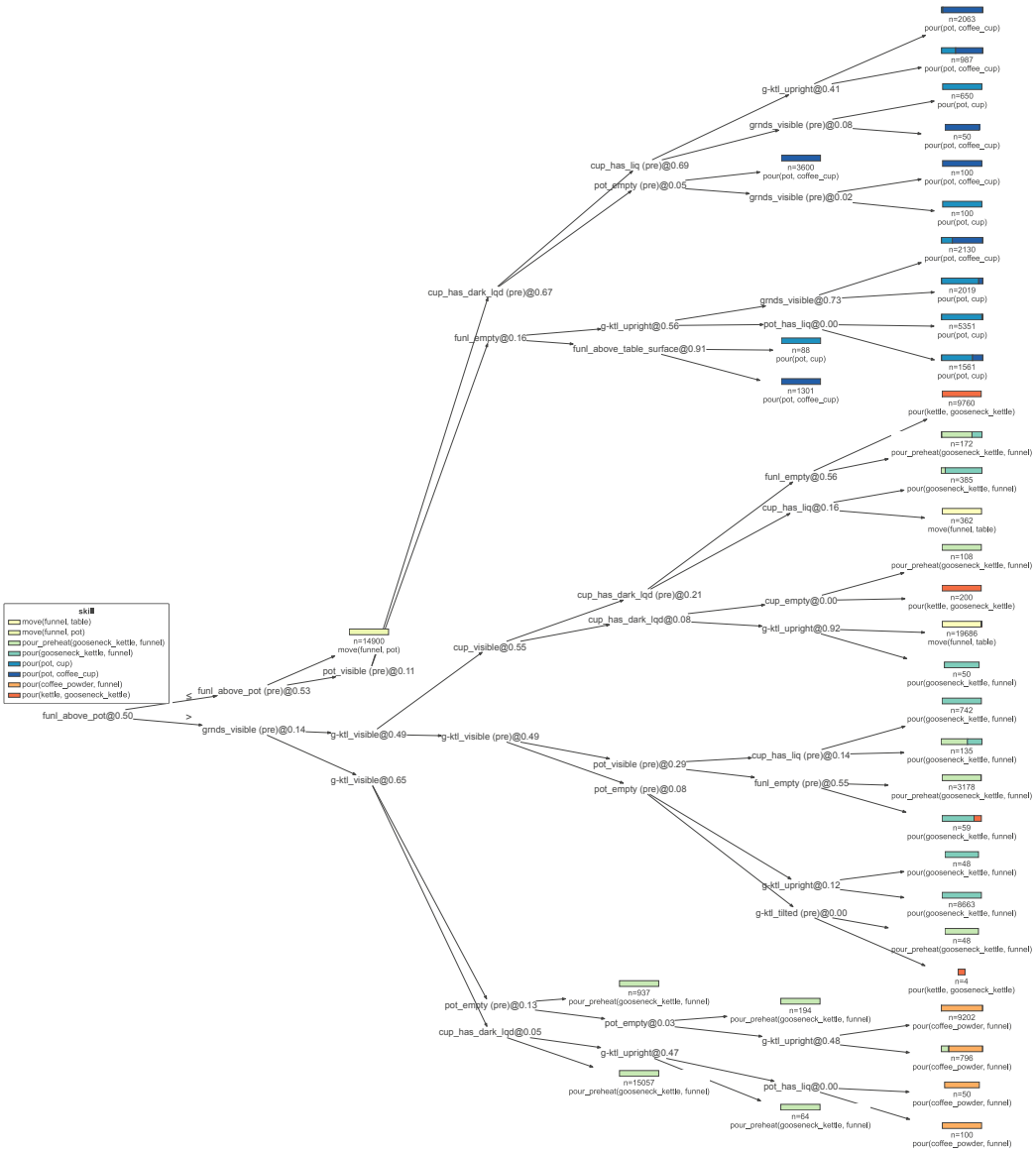


Figure 10: **Visualization of decision tree for the *Coffee* task.**

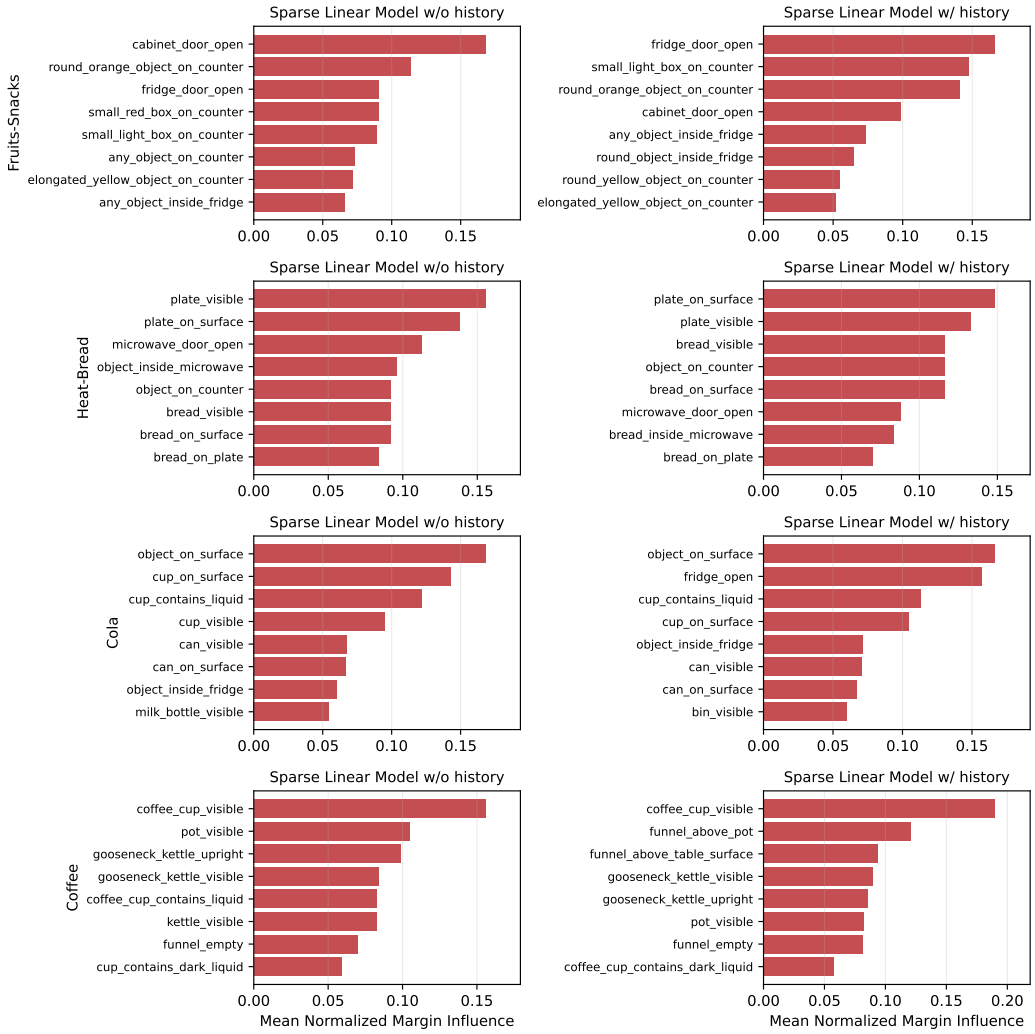


Figure 11: **Visualization of top-8 concept weights in sparse linear models across tasks.**

## G Additional Intervention Case Study

This section describes how we identify the concept to intervene on when ConceptTree predicts an incorrect skill. Rather than inspecting all concepts, we use the tree structure to find the first predicate where the erroneous decision path diverges from a nearby path that predicts the correct skill.

**Path-based diagnosis.** For an incorrect prediction $a_t = g(\tilde{\mathbf{c}}_t)$, let $\rho_{\text{err}} = (n_0, \ldots, n_m)$ be the traversed root-to-leaf path, where $n_0$ is the root node, $n_m$ is the reached leaf node, and each internal node $n_j$ tests a predicate $d_{n_j}(\tilde{\mathbf{c}}_t)$. Since the same skill may appear in multiple leaves, the ground-truth skill $a_t^*$ can correspond to multiple correct paths:

$$\mathcal{R}(a_t^*) = \{\rho : \text{leaf}(\rho) \text{ predicts } a_t^*\}. \tag{10}$$

We select the correct path that shares the longest prefix with the erroneous path:

$$\rho_{\text{near}} = \arg \max_{\rho \in \mathcal{R}(a_t^*)} \text{LCP}(\rho_{\text{err}}, \rho), \tag{11}$$

where LCP denotes the number of shared nodes from the root. Equivalently, this finds the correct-skill leaf with the deepest lowest common ancestor with the erroneous leaf. The last shared internal node is the nearest divergent parent node, whose split predicate identifies the concept dimension and threshold that send the two paths to different branches.

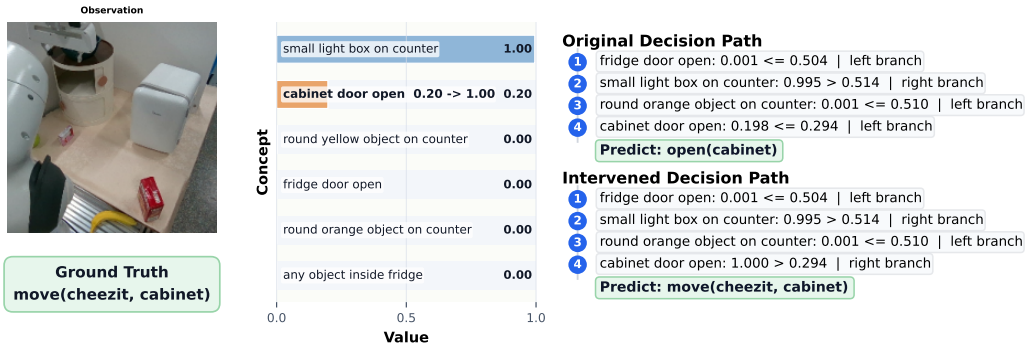


Figure 12: **Visualization of the intervention case study on *Fruits-Snacks*.**

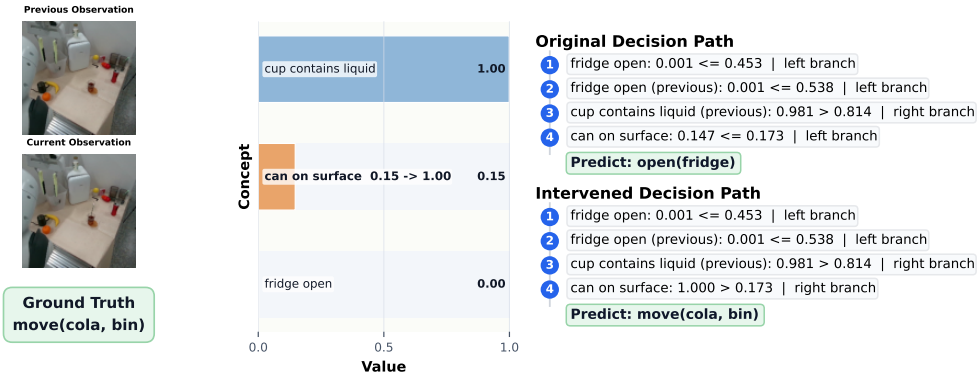


Figure 13: **Visualization of the intervention case study on *Cola*.**

**Intervention target.** Suppose the nearest divergent parent node tests dimension $k$ of $\tilde{\mathbf{c}}_t$ with threshold $\tau$. If $\rho_{\text{err}}$ and $\rho_{\text{near}}$ follow different branches at this node, then dimension $k$ is the first semantic variable that prevents the current prediction from reaching the nearest correct path. We use the corresponding concept as the intervention target. In the no-history setting, this dimension corresponds to a concept value in $\mathbf{c}_t$; in the history setting, it further indicates whether the concept belongs to $\mathbf{c}_{t-1}$ or $\mathbf{c}_t$. We then edit only this concept value, move it to the semantically correct side of the split, and recompute the tree prediction without changing model parameters. This provides a local counterfactual diagnosis: it identifies the first concept-level predicate that must change to redirect the decision toward the correct skill.

**Additional examples.** Figs. 12, 13, and 14 show additional intervention examples. In *Fruits-Snacks*, the model incorrectly predicts `open(cabinet)` instead of `move(cheezit, cabinet)` because the concept value of `cabinet_door_open` is 0.20, which does not match the actual observation and the threshold (0.29). Changing this single concept redirects the path to the correct skill. The *Cola* and *Coffee* examples follow the same procedure, illustrating how tree paths narrow failure analysis to a specific concept predicate rather than requiring a search over the full concept set.

## H Implementation Details

In this section, we provide details on the robotic task setup used in our experiments. The task suite, low-level skill library, and physical evaluation episodes are fixed across all compared methods, while the high-level decision module evaluated in this paper is ConceptTree rather than a black-box recurrent or transformer policy.

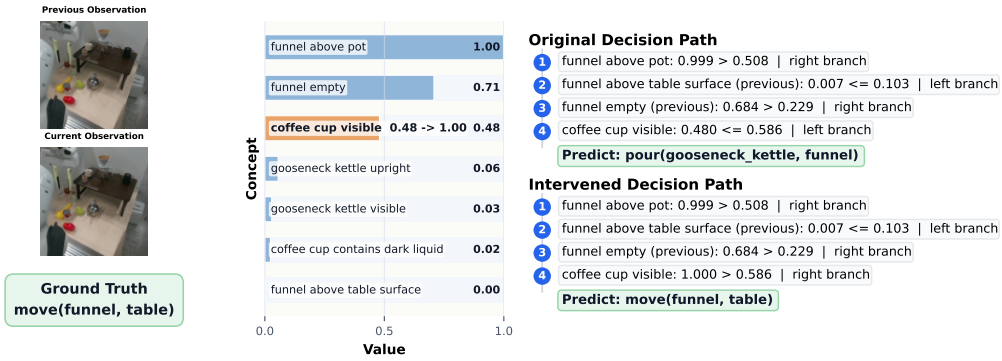


Figure 14: **Visualization of the intervention case study on *Coffee*.**

### H.1 Robot Platform and Low-level Skills

All experiments are conducted on a physical Franka Emika Panda 7-DOF robot arm with a stationary side camera that provides a global view of the workspace. The high-level policy selects from a predefined library of discrete skills, and each selected skill is executed by a pretrained low-level controller. These low-level controllers correspond to reusable motion primitives such as `open`, `close`, `move`, and `pour`. They are trained with the Human-in-the-Loop Sample Efficient Robotic Reinforcement Learning (HIL-SERL) framework [24], which incorporates human demonstrations and real-time corrective interventions during policy learning. In our experiments, the low-level skills are kept fixed across all methods so that evaluation focuses on high-level skill selection.

### H.2 Tasks and Skill Sequences

We evaluate on four long-horizon manipulation tasks: *Fruits-Snacks*, *Heat-Bread*, *Cola*, and *Coffee*. Each task is decomposed into a fixed task-relevant skill library and a corresponding expert execution sequence, as shown in Table 6. The tasks are designed to cover different sources of difficulty, including temporal dependency, repeated skill usage, fine-grained object interactions, and sensitivity to visual details. Fig. 15 illustrates the task scenes and skill sequences.

### H.3 Low-level Action Policy Details

The low-level action policies provide the executable primitives used by all high-level decision methods. Each skill policy is trained with HIL-SERL using human demonstrations and corrective interventions, and maps proprioceptive and visual observations to continuous end-effector commands. In our platform, robust vision-based skills can be obtained with a success rate exceeding 90% within approximately 1 hour of real-world training. To maintain consistency between consecutive skills, we use scripted resets to return the robot arm to a standard observation pose after each skill execution, preventing the arm or gripper from occluding the side-camera view used for high-level decision making.

### H.4 Preprocessing

RGB observations are captured from the side camera at 10 Hz with an original resolution of $1280 \times 720$ pixels. For high-level visual decision making, each raw side-camera image is center-cropped to remove peripheral workspace regions and resized to $256 \times 256$ pixels. Concept annotations are constructed over these preprocessed observations, and all compared high-level decision methods operate on the same observation stream and skill labels.

### H.5 Data Collection and Evaluation

During training data collection, the robot follows the expert skill sequence for each task, and each observation is associated with the current high-level skill label. Rather than using the entire trajectory of a low-level skill execution, we keep the first 50 side-camera frames after each skill starts. This

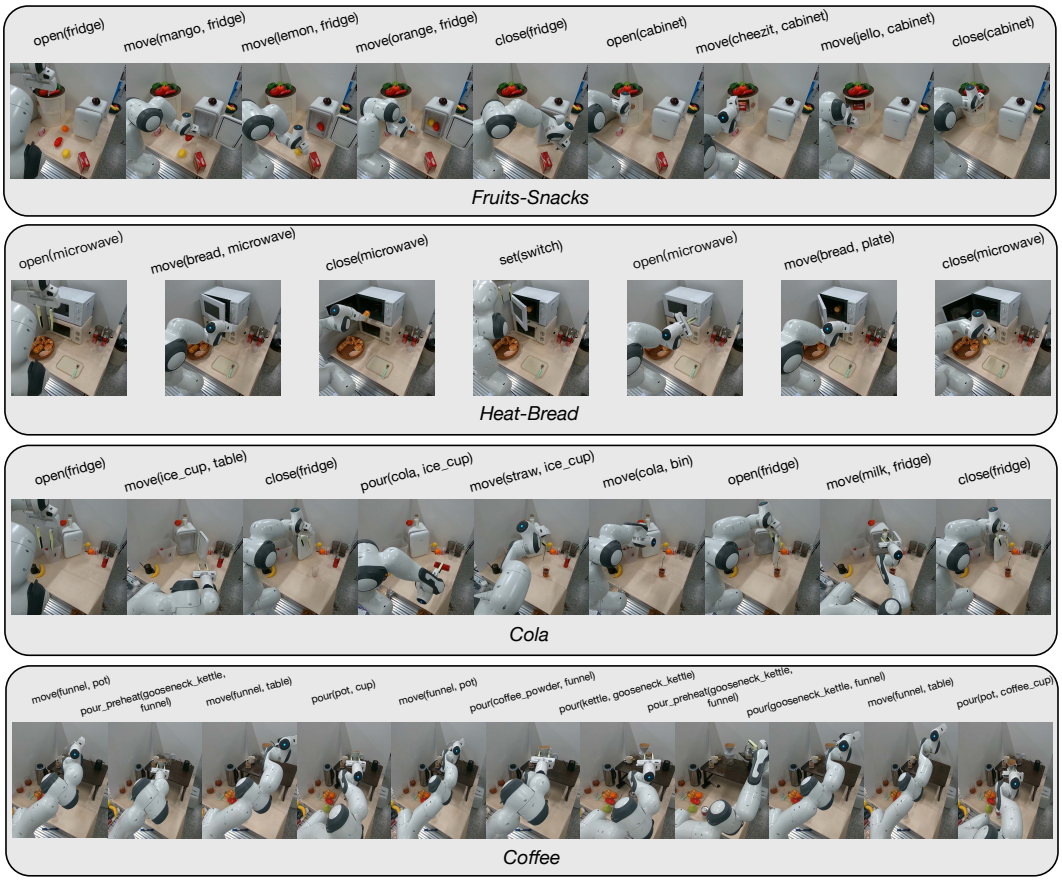


Figure 15: **Illustrations of the tasks in the experiments and skill sequences.**

window captures the visual conditions under which the high-level skill choice is made, while avoiding later frames dominated by low-level execution transients, intermediate object motions, and post-action states that can deviate from the original decision context. In practice, this selective sampling increases visual diversity across skill executions while reducing label ambiguity for high-level decision learning. We collect 5 episodes for each task, resulting in datasets containing 2250, 1750, 2250, and 2750 frames for *Fruits-Snacks*, *Heat-Bread*, *Cola*, and *Coffee*, respectively.

For evaluation, we deploy each trained high-level policy on the physical robot under controlled variations in task-irrelevant object categories, object locations, and lighting conditions. At each decision point, the policy predicts the next skill from the current observation; if the prediction is incorrect, we record the error but execute the ground-truth skill before continuing the episode. This protocol evaluates high-level decisions under real robot observations while allowing later decisions in the same long-horizon episode to be measured. For each method, we run 10, 10, 10, and 5 evaluation episodes on *Fruits-Snacks*, *Heat-Bread*, *Cola*, and *Coffee*, respectively.

# I Additional Visualization of Episodes and Decision Paths

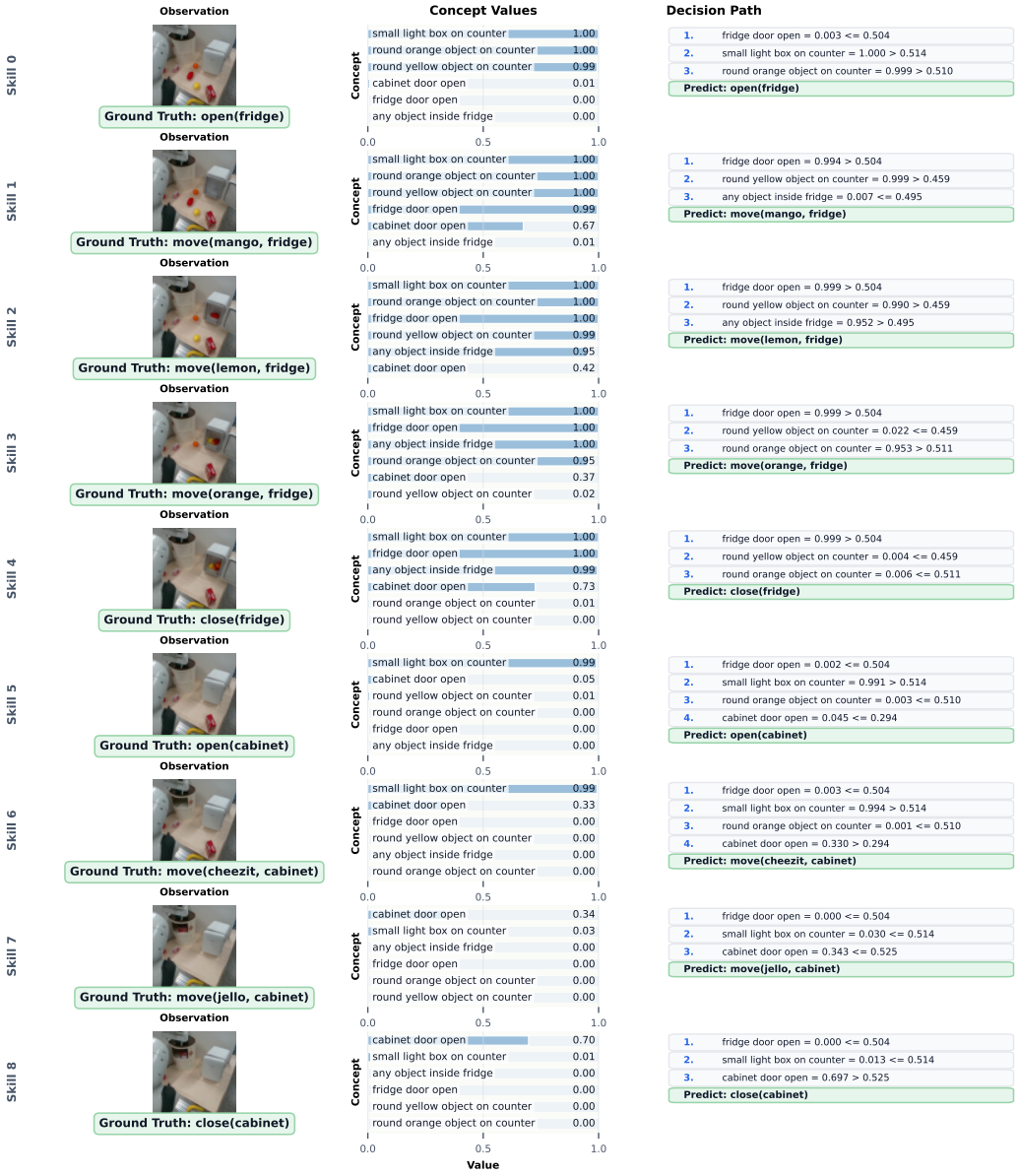


Figure 16: **Visualization of a test episode for the *Fruits-Snacks*.**

Table 6: **Task settings used in the experiments.** For each task, we list the predefined skill library and the expert execution sequence used for high-level decision evaluation.

| Task | Skill Library | Execution Sequence |
|---|---|---|
| *Fruits-Snacks* | `open(fridge)`<br>`close(fridge)`<br>`open(cabinet)`<br>`close(cabinet)`<br>`move(mango, fridge)`<br>`move(lemon, fridge)`<br>`move(orange, fridge)`<br>`move(cheezit, cabinet)`<br>`move(jello, cabinet)` | `open(fridge)`<br>`move(mango, fridge)`<br>`move(lemon, fridge)`<br>`move(orange, fridge)`<br>`close(fridge)`<br>`open(cabinet)`<br>`move(cheezit, cabinet)`<br>`move(jello, cabinet)`<br>`close(cabinet)` |
| *Heat-Bread* | `open(microwave)`<br>`close(microwave)`<br>`set(switch)`<br>`move(bread, microwave)`<br>`move(bread, plate)` | `open(microwave)`<br>`move(bread, microwave)`<br>`close(microwave)`<br>`set(switch)`<br>`open(microwave)`<br>`move(bread, plate)`<br>`close(microwave)` |
| *Cola* | `open(fridge)`<br>`close(fridge)`<br>`move(ice_cup, table)`<br>`move(milk, fridge)`<br>`move(cola, bin)`<br>`move(straw, ice_cup)`<br>`pour(cola, ice_cup)` | `open(fridge)`<br>`move(ice_cup, table)`<br>`close(fridge)`<br>`pour(cola, ice_cup)`<br>`move(straw, ice_cup)`<br>`move(cola, bin)`<br>`open(fridge)`<br>`move(milk, fridge)`<br>`close(fridge)` |
| *Coffee* | `move(funnel, table)`<br>`move(funnel, pot)`<br>`pour_preheat(gooseneck_kettle, funnel)`<br>`pour(gooseneck_kettle, funnel)`<br>`pour(pot, cup)`<br>`pour(pot, coffee_cup)`<br>`pour(coffee_powder, funnel)`<br>`pour(kettle, gooseneck_kettle)` | `move(funnel, pot)`<br>`pour_preheat(gooseneck_kettle, funnel)`<br>`move(funnel, table)`<br>`pour(pot, cup)`<br>`move(funnel, pot)`<br>`pour(coffee_powder, funnel)`<br>`pour(kettle, gooseneck_kettle)`<br>`pour_preheat(gooseneck_kettle, funnel)`<br>`pour(gooseneck_kettle, funnel)`<br>`move(funnel, table)`<br>`pour(pot, coffee_cup)` |

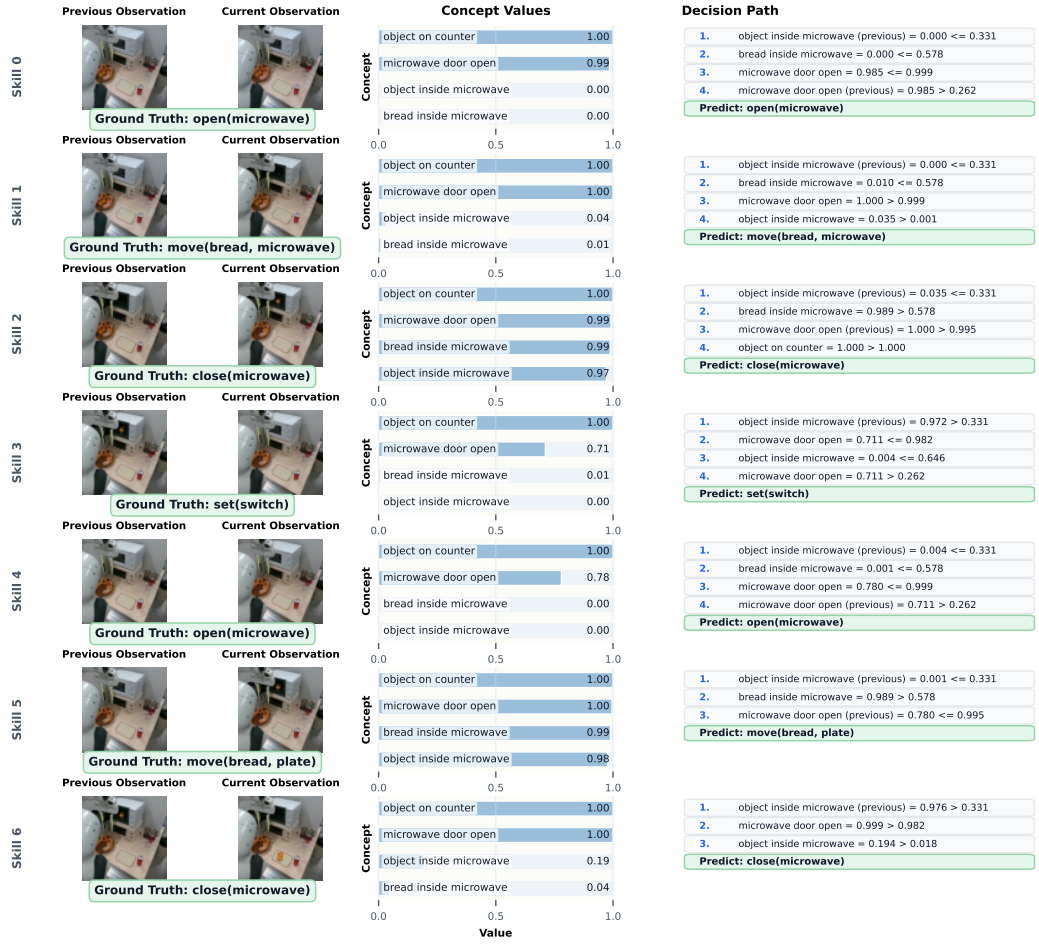


Figure 17: **Visualization of a test episode for the *Heat-Bread* with history input.**

Figure 18: **Visualization of a test episode for the *Cola* with history input.**