CIRRA: DUAL-LEVEL CONTINUAL INSTRUCTION RECONCILIATION WITH ONGOING EXECUTION FOR EMBODIED ROBOT AGENTS IN INTERACTIVE HOUSEHOLD TASKS

Anonymous authors
Paper under double-blind review

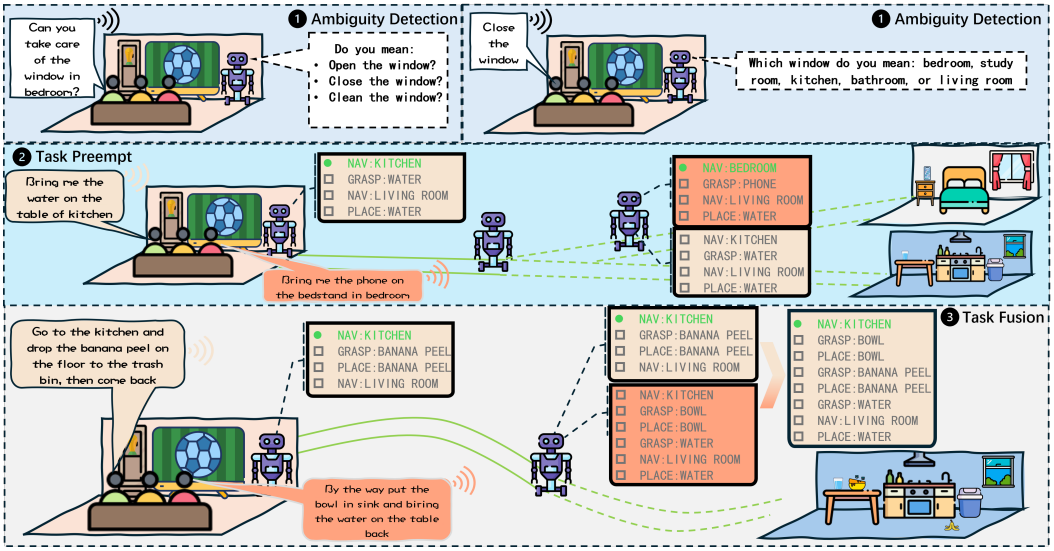

Figure 1: Demonstration of CIRRA's behaviors, including (1) different types of instruction ambiguity and their resolution, (2) task preemption, and (3) task fusion. The queue case, where an incoming instruction is directly scheduled for later execution, is omitted for brevity.

ABSTRACT

As the demand for assistance with household chores continues to grow, household robots have become a major focus of embodied intelligence. Real-world household deployment, however, inherently requires continuously replanning on human–robot interaction, as users may issue new instructions while previous requests are still under execution. Existing robot agents commonly accommodate such instructions by regenerating or extensively revising the remaining task sequence, which can introduce plan ambiguity, logical inconsistency, and redundant execution. To address these limitations, we formulate continual instruction reconciliation and propose CIRRA (Continual Instruction Reconciliation for Robot Agents), a dual-level framework combining LLM-based semantic reasoning with rule-constrained structural integration. CIRRA first uses semantic reasoning to determine whether an incoming instruction can be grounded to a unique executable skill and whether its required execution location is specified, resolving underspecified action and location information when necessary. It then preserves the ongoing subtask sequence as an execution backbone and generates fine-grained integration candidates by inserting the incoming subtasks into location-matched segments of the current task chain. Finally, the semantic reasoner evaluates only the modified segments to identify task dependencies and potential conflicts and select the most logically coherent local integration. By coupling structure-preserving local integration with semantic grounding and logical selection, CIRRA maintains alignment with the robot's current execution, mitigates plan ambiguity and logical in-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

consistency, and reuses shared subtasks to reduce redundant execution. We further introduce CHIRP (Continual Household Instruction Reconciliation and Planning), a text-based benchmark of 120 episodes spanning eight household environments and six categories of everyday activities. On CHIRP, CIRRA reaches 74.2% decision agreement, 30 points above the strongest replanning baseline, and every correct fusion decision it makes yields a correctly placed, conflict-free schedule. On a Unitree G1 humanoid, CIRRA interrupts ongoing skills at the correct moment in every trial and significantly outperforms all baselines on every metric.

1 INTRODUCTION

Recently, household robots have emerged as a major focus in embodied intelligence, driven by the growing demand for everyday assistance amid the accelerating pace of modern life, population aging, and persistent labor shortages in care and domestic services. Recent advances in robotic foundation models, particularly vision-language-action modelsBrohan et al. (2022); Zitkovich et al. (2023); Kim et al. (2024b); Intelligence et al. (2025); Bjorck et al. (2025), have made general-purpose household assistance increasingly feasible, enabling robots to understand natural-language instructions, perceive complex environments, and execute a diverse range of navigation and manipulation skills.

However, even capable action models alone remain insufficient for autonomous household operation Shi et al. (2025); Yang et al. (2025); Koo et al. (2025). To bridge this gap, embodied robot agents have extensively studied high-level task decomposition and long-horizon skill coordination Dalal et al. (2024); Yuan et al. (2025), task-state tracking and memory Wang et al. (2025); Koo et al. (2025), and adaptive replanning in response to execution failures or environmental changes Huang et al. (2022); Skreta et al. (2024). These advances enable an agent to translate a high-level user goal into a coherent sequence of executable skills, monitor task progress, and revise its plan when execution deviates from expectation.

Despite this progress, a critical challenge for household deployment remains comparatively under-explored: how to integrate a newly arriving task instruction with one that is already under execution. Although existing interactive agents can engage in dialogue and respond to user corrections, feedback, or task changes Wei et al. (2022); Kambhampati et al. (2024); Shinn et al. (2023); Li et al. (2023); Joublin et al. (2024); Borate et al. (2025); Lu et al. (2025), a common strategy for accommodating such input is to re-invoke a high-level model with the updated instruction and current execution context to regenerate or revise the remaining task sequence. Without explicit constraints on preserving unaffected portions of the ongoing plan, this generative replanning can introduce three key issues. First, it may create **plan ambiguity**, as a minor instruction update can cause substantial plan churn, unnecessarily reorder pending subtasks, and obscure how the revised plan aligns with the current execution state. Second, it may produce **logical inconsistency** by violating task dependencies or introducing conflicting action orderings. Third, it may lead to **execution redundancy** by failing to identify and reuse subtasks shared between the ongoing and incoming instructions. These limitations highlight the need for fine-grained, execution-aligned instruction integration rather than unconstrained plan regeneration.

To address these limitations, we propose CIRRA, a dual-level framework that combines LLM-based semantic reasoning with rule-constrained structural integration. Upon receiving a new instruction, the semantic reasoning level first determines whether its action description can be grounded to a unique executable skill and whether the location required for execution is explicitly specified (shown in Ambiguity Detection in Fig.1). It resolves underspecified action and location information to obtain a complete, location-aware representation of the incoming instruction. For a non-emergency instruction, the structural integration level then treats the robot's ongoing subtask sequence as an immutable execution backbone(shown in Task fusion in Fig.1): it retains every existing subtask and their relative ordering, and generates fine-grained integration candidates by inserting the incoming subtasks into segments of the sequence associated with the same execution location. The semantic reasoning level is subsequently invoked again, but only over the modified segments, to evaluate their task dependencies, identify potential conflicts, and select the most logically coherent local integration. In this way, CIRRA prevents the LLM from globally regenerating or reordering the ongoing plan, while still leveraging its semantic reasoning to accurately ground the incoming instruction

and determine its logically appropriate placement. The two levels therefore provide complementary guarantees: rule-constrained integration preserves the robot's current execution strategy, whereas LLM-based reasoning ensures instruction completeness, placement accuracy, and the logical coherence of each modified segment. Together, they enable fine-grained instruction reconciliation without disrupting ongoing execution or introducing redundant actions.

Existing household benchmarks primarily focus on instruction following, dialogue, and task execution under predefined user goals, but are not designed to evaluate replanning when new task requests arrive during ongoing execution Shridhar et al. (2020); Padmakumar et al. (2022); Gao et al. (2022). Conversely, existing replanning benchmarks and evaluation settings largely study plan revision in response to execution failures or environmental changes, rather than continual user-driven task updates in interactive household settings. Consequently, there remains no dedicated benchmark at the intersection of household interaction and instruction-triggered replanning. To fill this gap, we introduce **CHIRP** (Continual Household Instruction Reconciliation and Planning), a text-based benchmark specifically designed to evaluate how embodied robot agents revise an ongoing household task plan in response to newly arriving user instructions. The benchmark spans eight representative household environments: bedroom, study room, kitchen, bathroom, living room, basement, garage, and yard. And covers six major categories of household activities: object fetching, cleaning, tidying, cooking, appliance operation, and waste disposal. Across all evaluated environments and task categories, CIRRA consistently outperforms existing replanning baselines, producing more accurate instruction integrations, more logically coherent task sequences, and less redundant execution.

2 Related Work

2.1 Robotic Action Foundation Models

Vision-language-action (VLA) foundation models Zitkovich et al. (2023); Team et al. (2024); Kim et al. (2024b); Li et al. (2024); Intelligence et al. (2025); Qu et al. (2025); Bjorck et al. (2025); Bu et al. (2025); Shukor et al. (2025) have shifted robot control from separately trained, task-specific policies toward reusable, language-conditioned action models that transfer across tasks, environments, and embodiments. By grounding the semantic knowledge of pretrained vision-language models in continuous action generation, these models increasingly serve as general-purpose skill interfaces that higher-level robotic systems can invoke. As a result, the bottleneck of household autonomy moves to the higher level: deciding which skills to execute and in what order, especially when new instructions arrive during execution.

2.2 Embodied Agents for Task Planning and Replanning

At this higher level, LLM-based embodied agents translate user goals into executable robot behaviors. SayCan grounds LLM-proposed actions in the affordances of pretrained skills Ahn et al. (2022), ProgPrompt and Code as Policies generate program-like plans over available actions and control APIs Singh et al. (2022); Liang et al. (2023), and Inner Monologue feeds success detection, scene descriptions, and human feedback back into planning Huang et al. (2022).

Subsequent agents add hierarchical skill orchestration Dalal et al. (2024); Yuan et al. (2025), persistent task memory Wang et al. (2025), and online user feedback Shi et al. (2025), and increasingly rely on replanning for robust long-horizon execution: CAPE corrects violated skill preconditions Raman et al. (2024), AdaPlanner refines plans from environmental feedback Sun et al. (2023), RePLan revises behavior after plan failures or unforeseen obstacles Skreta et al. (2024), and PRED preemptively revises upcoming actions when the environment changes Kim et al. (2024a). However, these methods predominantly treat replanning as recovery under a fixed user goal: the task itself stays unchanged, and the agent only adapts its behavior to execution-state mismatches.

This leaves an important case underexplored: in interactive households, users may issue additional task instructions while an earlier task chain is still being executed. Interactive agents incorporate corrections, feedback, or clarifications about the ongoing task Huang et al. (2022); Shi et al. (2025), but do not address how a new task should be reconciled with it. Simply prompting an LLM with the new instruction and the execution context gives no guarantee that unaffected subtasks and their order are preserved. A small update may thus cause **plan ambiguity**, where excessive plan churn obscures how the revised plan relates to the current execution state; **logical inconsistency**, where

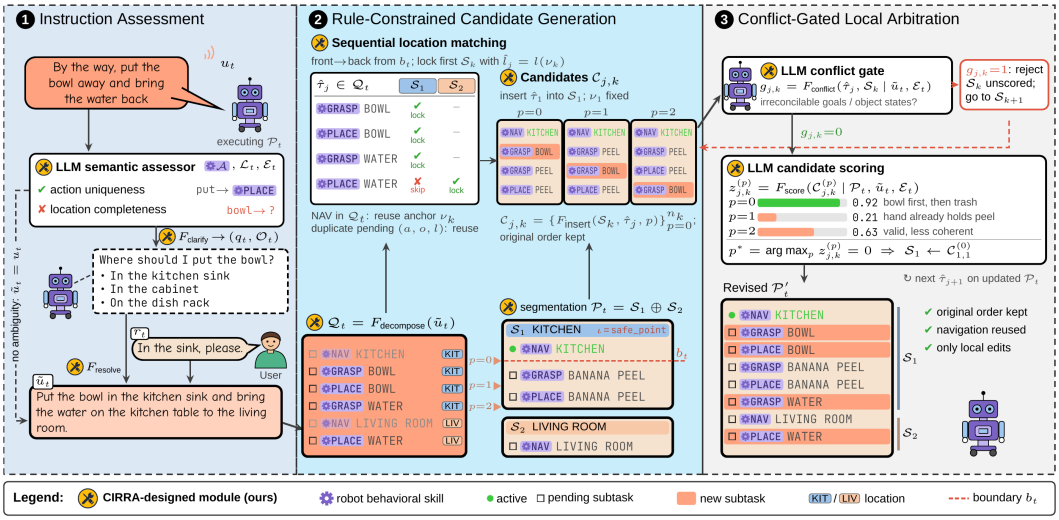

Figure 2: Overview of CIRRA: (1) instruction assessment and ambiguity resolution, (2) rule-constrained candidate generation within location-matched segments, and (3) conflict-gated LLM arbitration that yields the revised sequence $\mathcal{P}'_t$.

task dependencies are violated or actions are ordered in conflict; and **execution redundancy**, where navigation or manipulation subtasks shared by the ongoing and incoming instructions are not reused. To our knowledge, CIRRA is the first embodied-agent framework for continual, instruction-triggered replanning over an ongoing task chain. It combines execution-aligned structural integration with LLM-based semantic reasoning to preserve the unaffected part of the current plan while resolving all three issues.

3 Methodology

3.1 Problem Formulation

We consider an embodied robot agent that receives a new user instruction while executing an existing household task. At time step t, the agent maintains an ongoing task sequence

$$\mathcal{P}_t = [\tau_1, \tau_2, \ldots, \tau_N], \tag{1}$$

where each subtask is represented using a unified tuple

$$\tau_i = (a_i, o_i, l_i, s_i, \iota_i). \tag{2}$$

Here, a_i denotes an executable robot skill. We divide the robot skill set $\mathcal{A}$ into navigation skills $\mathcal{A}_{\text{nav}}$ and non-navigation behavioral skills $\mathcal{A}_{\text{beh}}$. The meanings of o_i and l_i depend on the corresponding skill type. If $a_i \in \mathcal{A}_{\text{nav}}$, the subtask does not operate on a target object, and we therefore set $o_i = \varnothing$. In this case, l_i denotes the destination of the navigation subtask. If $a_i \in \mathcal{A}_{\text{beh}}$, o_i denotes the object on which the behavior is performed, while l_i denotes the location at which the behavior should be executed. Thus,

$$o_i = \varnothing \quad \text{if} \quad a_i \in \mathcal{A}_{\text{nav}}. \tag{3}$$

The variable s_i denotes the execution status of the subtask:

$$s_i \in \{\texttt{pending}, \texttt{active}, \texttt{done}\}. \tag{4}$$

A `pending` subtask has not yet started, `active` denotes the subtask currently being executed, and `done` indicates that the subtask has already been completed.

The variable ι_i denotes the interruptibility mode of the subtask:

$$\iota_i \in \{\texttt{immediate}, \texttt{safe_point}, \texttt{uninterruptible}\}. \tag{5}$$

Specifically, `immediate` permits the subtask to be interrupted at any time. The `safe_point` mode permits interruption only after the subtask reaches its next predefined safe execution boundary. The `uninterruptible` mode requires the subtask to finish before an incoming instruction can affect ongoing execution. These modes apply to both navigation and behavioral subtasks and prevent plan integration from interrupting a robot skill at an invalid or unsafe state.

While executing $\mathcal{P}_t$, the robot receives an additional natural-language instruction u_t. Given the ongoing task sequence $\mathcal{P}_t$ and the incoming instruction u_t, our objective is to construct a revised task sequence $\mathcal{P}'_t$ that accommodates the new request while preserving the unaffected subtasks and execution structure of $\mathcal{P}_t$.

3.2 Instruction Assessment and Ambiguity Resolution

Before integrating an incoming instruction with the ongoing task sequence, CIRRA first assesses whether the instruction can be unambiguously grounded into executable robot subtasks (Figure 2,1). Given the incoming instruction u_t, the LLM-based semantic assessor is provided with the robot's executable skill set $\mathcal{A}$, the available household locations $\mathcal{L}_t$, and the current environment context $\mathcal{E}_t$.

The assessor examines every action expression in u_t along two dimensions: action uniqueness and location completeness. An action expression is considered ambiguous if it cannot be uniquely mapped to one skill in $\mathcal{A}$. Its location information is considered incomplete if the location required for executing the corresponding action is omitted or cannot be uniquely determined from $\mathcal{L}_t$ and $\mathcal{E}_t$.

$$(q_t, \mathcal{O}_t) = F_{\text{clarify}}(u_t, \mathcal{A}, \mathcal{L}_t, \mathcal{E}_t), \tag{6}$$

When an action or execution location is underspecified, CIRRA does not autonomously infer the missing information. Instead, F_{clarify} generates a targeted clarification question q_t together with a compact set of contextually relevant candidate interpretations $\mathcal{O}_t$. These candidates are constrained by the robot's executable skill set $\mathcal{A}$, the available household locations $\mathcal{L}_t$, and the current environment context $\mathcal{E}_t$. The robot then presents q_t and $\mathcal{O}_t$ to the user and obtains a response r_t.

$$\tilde{u}_t = F_{\text{resolve}}(u_t, r_t, \mathcal{A}, \mathcal{L}_t, \mathcal{E}_t), \tag{7}$$

Based on this response r_t, F_{resolve} produces a normalized instruction $\tilde{u}_t$ in which every action is associated with a unique executable skill and an explicit execution location.

If no action or location ambiguity is detected, the clarification interaction is skipped and $\tilde{u}_t = u_t$. The resulting normalized instruction $\tilde{u}_t$ is then passed to the task-decomposition and plan-integration stage described next.

3.3 Rule-Constrained Execution-Aligned Candidate Generation

After obtaining the normalized instruction $\tilde{u}_t$, CIRRA decomposes it into an ordered sequence of executable incoming subtasks (Figure 2,2):

$$\mathcal{Q}_t = F_{\text{decompose}}(\tilde{u}_t) = [\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_M]. \tag{8}$$

Each incoming subtask is represented as $\hat{\tau}_j = (\hat{a}_j, \hat{o}_j, \hat{l}_j, \hat{s}_j, \hat{\iota}_j)$, where $\hat{s}_j = \texttt{pending}$. The sequence $\mathcal{Q}_t$ contains only the subtasks derived from the incoming instruction and has not yet been integrated with $\mathcal{P}_t$.

CIRRA first determines the earliest point at which the incoming instruction may affect ongoing execution. Let τ_{k_t} be the currently active subtask. If $\iota_{k_t} = \texttt{immediate}$, it may be paused immediately and resumed after the inserted subtasks are completed. If $\iota_{k_t} = \texttt{safe_point}$, integration is delayed until the next safe execution boundary. If $\iota_{k_t} = \texttt{uninterruptible}$, the active subtask must finish before the revised sequence can take effect. We denote the resulting earliest admissible integration boundary by b_t; execution before b_t remains fixed.

Navigation-anchored segmentation. The ongoing sequence $\mathcal{P}_t$ is organized into navigation-anchored segments. Every navigation subtask begins a new segment, and all subsequent non-navigation subtasks are assigned to that segment until the next navigation subtask:

$$\mathcal{P}_t = \mathcal{S}_1 \oplus \cdots \oplus \mathcal{S}_K, \qquad \mathcal{S}_k = [\nu_k, \tau_{k,1}, \ldots, \tau_{k,n_k}]. \tag{9}$$

Here, ν_k is the leading navigation subtask, and $l(\mathcal{S}_k) = l(\nu_k)$ defines the execution location of the segment. If the robot visits the same location multiple times, each navigation subtask starts a distinct segment, preserving the temporal structure of the original plan.

Sequential matching and candidate enumeration. CIRRA processes every non-navigation subtask $\hat{\tau}_j \in \mathcal{Q}_t$ individually. Beginning from the earliest segment permitted by b_t, it traverses $\mathcal{P}_t$ from front to back and compares $\hat{l}_j$ with the destination $l(\nu_k)$ of each segment. A segment is skipped when their locations differ. Once $\hat{l}_j = l(\nu_k)$, $\mathcal{S}_k$ is temporarily locked as the current insertion segment.

CIRRA keeps ν_k fixed and enumerates every position at which $\hat{\tau}_j$ can be inserted among the non-navigation subtasks:

$$\mathcal{C}_{j,k} = \{F_{\text{insert}}(\mathcal{S}_k, \hat{\tau}_j, p) \mid p = 0, \ldots, n_k\}. \tag{10}$$

The case $p = 0$ places $\hat{\tau}_j$ immediately after ν_k, while $p = n_k$ places it after all existing behavioral subtasks. Every candidate preserves the leading navigation subtask and the relative ordering of all original subtasks. Thus, integrating a behavior into a location-matched segment reuses the segment's navigation instead of introducing another navigation action.

Before enumeration, CIRRA also checks whether the matched segment already contains an active or pending subtask with the same action, object, and location. If so, the existing subtask is reused and no duplicate is inserted. Otherwise, $\mathcal{C}_{j,k}$ is passed to the LLM judge. Consequently, all modifications remain local, while the original subtasks and their relative ordering are strictly preserved.

3.4 LLM-Based Conflict-Gated Local Arbitration

For each location-matched segment, CIRRA applies a conflict-first arbitration procedure (Figure 2,3). The LLM first determines whether the incoming action has an irreconcilable conflict with any behavioral action in $\mathcal{S}_k$:

$$g_{j,k} = F_{\text{conflict}}(\hat{\tau}_j, \mathcal{S}_k \mid \tilde{u}_t, \mathcal{E}_t), \qquad g_{j,k} \in \{0, 1\}. \tag{11}$$

Here, $g_{j,k} = 1$ indicates a conflict, while $g_{j,k} = 0$ indicates compatibility. The conflict gate focuses on incompatibilities that cannot be resolved by changing the insertion position, such as mutually exclusive goals or contradictory object-state requirements. Ordering dependencies that can be resolved through an appropriate placement are handled during candidate scoring.

If $g_{j,k} = 1$, CIRRA rejects the entire candidate segment without scoring its insertion positions. The traversal then continues from $\mathcal{S}_{k+1}$. This avoids evaluating candidate orders that cannot produce a valid integration regardless of where the incoming action is inserted.

If $g_{j,k} = 0$, the LLM scores every candidate in $\mathcal{C}_{j,k}$:

$$z_{j,k}^{(p)} = F_{\text{score}}(\mathcal{C}_{j,k}^{(p)} \mid \mathcal{P}_t, \tilde{u}_t, \mathcal{E}_t), \qquad p = 0, \ldots, n_k. \tag{12}$$

The score measures whether the insertion preserves causal dependencies, valid object-state transitions, and a coherent local action ordering. CIRRA then selects the highest-scoring position:

$$p^* = \underset{p \in \{0,\ldots,n_k\}}{\arg\max}\, z_{j,k}^{(p)}, \qquad \mathcal{S}_k \leftarrow \mathcal{C}_{j,k}^{(p^*)}. \tag{13}$$

Once an insertion is accepted, CIRRA terminates the traversal for $\hat{\tau}_j$ and does not examine later segments, even if they share the same location. If the current segment is rejected by the conflict gate, traversal continues until another location-matched segment is found. The next incoming behavioral subtask is processed using the updated task sequence.

In this way, deterministic rules control the admissible segment and insertion positions, while the LLM only rejects conflicts and selects among local candidates. CIRRA therefore preserves the unaffected execution structure, reuses existing navigation, and prevents the LLM from globally regenerating or reordering the ongoing task sequence.

4 Experiments

4.1 Task Representation and Benchmark

Benchmark. CHIRP contains 120 episodes set in a simulated home with eight rooms, covering six action categories of 20 episodes each (appliance control, cleaning, tidying, cooking, fetch and delivery, and waste handling). Each episode freezes the moment a new utterance arrives during an ongoing task and asks for the agent's reaction. Episodes are built as minimal pairs with nearly identical wording but different execution states, so a method that ignores execution state must miss one episode of each pair. The gold decisions comprise 40 fuse, 47 queue, 21 clarify and 12 preempt episodes, and each utterance comes in three surface forms. Every episode also provides a textual world snapshot (positions, object placements and held items), giving each baseline the environmental context it expects. Gold labels follow the decision doctrine of Section 3; the full schema, templates and annotation records are given in Appendix A.

Method	Appliance	Clean	Tidy	Cook	Fetch	Waste	Overall	Compile-correct
LLM-direct	65	80	55	35	45	75	59.2	62.8
Conjunctive replan	40	40	40	40	35	35	38.3	55.8
CoT replan Wei et al. (2022)	50	40	30	55	35	35	40.8	51.2
LLM-Modulo Kambhampati et al. (2024)	40	40	40	40	35	40	39.2	58.1
Reflexion Shinn et al. (2023)	45	35	40	45	50	50	44.2	58.1
ITP Li et al. (2023)	45	40	35	35	45	0	33.3	39.5
CoPAL Joublin et al. (2024)	25	20	20	20	40	40	27.5	32.6
CoMuRoS Borate et al. (2025)	40	40	40	30	10	35	32.5	44.2
DPPM Lu et al. (2025)†	35	50	50	50	40	5	38.3	51.2
SayCan Ahn et al. (2022)†	40	40	40	40	35	40	39.2	58.1
Memory-agent Glocker et al. (2025)†	50	40	35	45	40	45	42.5	58.1
Ours w/o judge	**85**	**85**	**70**	**65**	**75**	55	72.5	81.4
Ours (full)	**85**	**85**	**70**	**65**	45	**95**	74.2	93.0

Table 1: Decision agreement (DA, %) on the 120 CHIRP episodes. All methods share one Qwen3-8B compilation and differ only in reaction policy; *Compile-correct* covers the 43 correctly compiled episodes. †Operator-level adaptation (Appendix B).

Evaluation Metrics We report three metrics. **DA (decision agreement)** is the fraction of episodes whose response falls into the gold decision class (fuse, preempt, queue, or clarify). **TA (target agreement)** is the fraction of fuse-gold episodes in which a method fuses and places the new subtasks at the gold position without moving any old subtask. **EA (execution agreement)** is the strictest metric: the fraction of fuse-gold episodes in which, in addition, the whole schedule replays without conflict in the symbolic simulator and the new subtasks follow the gold order within their segment.

Evaluation Protocol We evaluate two conditions. In the primary **end-to-end** condition, methods receive the raw utterance and episode state, with all plan-consuming methods sharing a single cached Qwen3-8B compilation. The **oracle spec** condition replaces this compilation with the ground-truth specification to isolate scheduling from language understanding. All LLM calls use identical decoding settings and schema-constrained JSON output. Evaluation is deterministic and uses no LLM judges.

4.2 Baselines

We implement policy-level reproductions of ITP, CoPAL, CoMuRoS, DPPM, and SayCan in a shared harness, retaining original prompts where available. CoT, Reflexion, and LLM-Modulo follow their published strategies; LLM-Modulo uses our symbolic simulator as its verifier. SayCan and the memory agent serve as sequential-request baselines. LLM-direct predicts one of the four decisions from the state and utterance without generating a schedule. Conjunctive replan regenerates the full schedule from the state, existing plan, and new request. Ours w/o judge disables the LLM compatibility judge. Prompt sources and per-baseline adaptations are detailed in Appendix B.

4.3 Results and Analysis

Table 1 summarizes the results. CIRRA reaches 74.2% DA, 30 points above the strongest replanning baseline (Reflexion, 44.2%) and 15 points above LLM-direct (59.2%). Its remaining errors stem almost entirely from compilation: the shared Qwen3-8B compiles only 35.8% of utterances exactly, and on these 43 episodes CIRRA reaches 93.0%. Of its 31 errors, 28 originate in compilation and 3 are incorrect conflict rejections by the LLM judge; under oracle specifications CIRRA scores 100%. The rule-constrained stages also absorb compilation errors: 49 of the 77 imperfectly compiled episodes (64%) still yield the gold decision, and the majority of the 31 errors end in a clarification question rather than a wrong action. DA also understates the gap, because regeneration baselines break prior commitments, dropping or reordering pending steps in up to 84 of 120 episodes (CoMuRoS; CoPAL 51, DPPM 39; Appendix B), whereas CIRRA only inserts into the existing sequence and cannot violate a commitment by design.



(a) Deployment environment (b) Eight manipulation skills labeled with its interruptibility mode.

Figure 3: Real-world setup. (a) A Unitree G1 humanoid operates at table A (left) and table B (right);(b) The eight tasks forming the robot's behavioral skills.

Comparison on fusion episodes. Table 2 evaluates fusion, the core capability of CIRRA, on the 40 fuse-gold episodes. Five baselines never fuse. This includes LLM-Modulo: its verifier removes all commitment violations, yet the method never chooses to fuse, since a verifier can ensure that a plan is valid but not that it is the right reaction. The strongest baselines, ITP and DPPM, choose to fuse in 50.0% and 45.0% of the episodes, but only 15.0% and 12.5% of the episodes end in a fully correct schedule (EA). CIRRA fuses in 75.0% of the episodes and every schedule it fuses is correct, reaching 75.0% EA; without the judge, EA drops to 47.5%, showing that the judge removes conflicting or misordered insertions. With oracle specifications, CIRRA reaches 100% EA, whereas the best replanning baseline reaches only 40.0%, showing that the gap comes from how methods react rather than from language understanding.

Table 2: Fusion performance (%) on the 40 fuse-gold episodes; *oracle spec* feeds the ground-truth specification. *Conj. replan, LLM-Modulo, CoMuRoS, SayCan†, Memory-agent†.

Method	End-to-end			Oracle spec	
	DA	TA	EA	DA	EA
LLM-direct	27.5	22.5	22.5	100.0	100.0
Five baselines*	0.0	0.0	0.0	0.0	0.0
CoT replan	10.0	7.5	2.5	10.0	10.0
Reflexion	10.0	10.0	2.5	20.0	17.5
ITP	50.0	47.5	15.0	47.5	40.0
CoPAL	2.5	2.5	0.0	0.0	0.0
DPPM†	45.0	35.0	12.5	52.5	40.0
Ours w/o judge	72.5	72.5	47.5	100.0	100.0
Ours (full)	75.0	75.0	75.0	100.0	100.0

Effect of the judge. Comparing the two Ours rows isolates the LLM judge. Beyond the fusion gain above, its effect varies across categories (Table 1): it improves waste handling by 40 points by rejecting forbidden joint transports and falling back to queueing, but lowers fetch and delivery by 30 points because the 8B judge also rejects valid multi-item deliveries. Since all other components are rule-based, these errors reflect the capability of the judge model, which the scale ablation examines (Table 3).

Scale moves compilation, not the mechanism. Table 3 swaps the compiler and judge together from Qwen3-4B to 8B then to 32B under the identical contract. Compile fidelity rises from 31.7% to 35.8% and then to 68.3% and end-to-end DA tracks it from 65.8% to 74.2% and then to 91.6%, while the oracle condition holds at 100% at every scale: the bottleneck is language, the mechanism is scale-invariant, and gains from larger models accrue to our system directly.

Table 3: Compiler-scale ablation (%). Compiler and judge are swapped together; the symbolic arbitration is fixed. Compile fidelity: episodes compiled exactly as the oracle.

Compiler	Compile fidelity	Ours DA	Ours DA (comp.-corr.)	LLM-direct DA
Qwen3-4B	31.7	65.8	86.8	43.3
Qwen3-8B	35.8	74.2	93.0	59.2
Qwen3-32B	68.3	91.6	99.2	87.5

4.4 Real-World Deployment

On a physical robot, a wrong reconciliation decision has physical consequences: interrupting a skill at an invalid state can drop a held object, and appending a request that could have been fused costs an extra trip between locations. We therefore evaluate CIRRA on a real robot.

Setup. A Unitree G1 humanoid operates at two side-by-side tables, A and B (shown in (a) of Fig 3) , walking between them with its built-in locomotion controller (`NAV`). Its behavioral skills are eight policies, each fine-tuned from ImageWAM Zhang et al. (2026) on 50 demonstrations of one task. Each skill is assigned one of the three interruptibility modes of Section 3 (`immediate`,

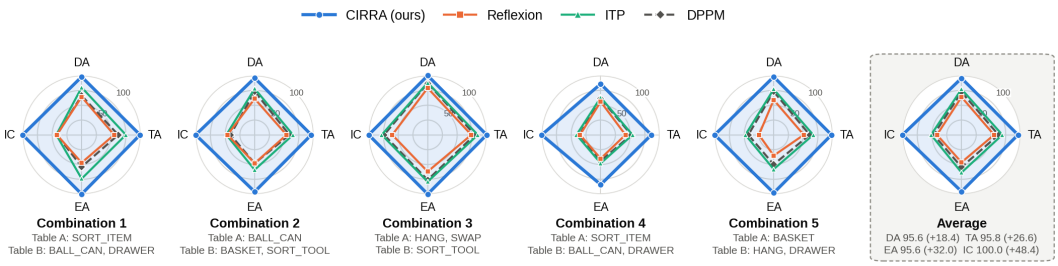

Figure 4: Real-world results (%). The first five radars show the five task combinations; the rightmost averages them and lists CIRRA's mean on each metric, with its gain over the best baseline in parentheses.

`safe_point`, or `uninterruptible`); in our setup, `NAV` is always `immediate`, and the mode of each manipulation skill is listed in Appendix C. Images from the onboard RGB-D camera are passed directly to the Qwen3-VL-8B as $\mathcal{E}_t$.

Task combinations. Each combination consists of three tasks placed on the two tables (listed below each radar in Figure 4) and targets one of three decisions: preemption (combinations 1 and 2), queueing (3), or fusion (4 and 5). Each combination is tested 25 times (125 trials in total), excluding trials in which a manipulation policy fails to grasp. In each trial, the new instruction arrives at a different point of the ongoing task sequence, so it may arrive while an `immediate`, `safe_point`, or `uninterruptible` subtask is active. Instructions and gold sequences are at Appendix C.

Baselines and metrics. We compare with the replanning baselines Reflexion, ITP and DPPM, none of which accounts for the interruptibility of the active subtask.. All methods are queried on identical execution-state snapshots and executed with the same skills. Besides DA, TA and EA, we report *interruption compliance* **(IC)**, the fraction of episodes in which the new instruction interrupts the active subtask at the correct moment given its interruptibility mode: immediately for an `immediate` subtask, at its next safe point for a `safe_point` subtask, and only after it finishes for an `uninterruptible` subtask.

Results. Averaged over the five combinations, CIRRA reaches 95.6% DA, 95.8% TA and 95.6% EA, 18.4, 26.6 and 32.0 points above the strongest baseline, ITP (Figure 4). The baselines' scores drop from DA to EA, as they often misplace the new subtasks even when they choose the right decision, whereas CIRRA keeps all three on par. The largest gap is in IC: CIRRA always interrupts at the correct moment (100%), since the boundary follows directly from the interruptibility mode, whereas the baselines reach at most 51.6% on average. Queueing (3) is the easiest combination and fusion beyond a navigation step (4) the hardest, where CIRRA's EA drops to 84% and the baselines reach 40–48%.

5 Limitations

CHIRP is text-based, and our real-robot study spans two tables and eight skills. Remaining errors stem almost entirely from the 8B compiler rather than the reconciliation mechanism, which is exact under oracle specifications, so CIRRA directly benefits from stronger compilers.

6 Conclusion

We presented CIRRA, a framework for reconciling newly arriving instructions with an ongoing household task. Instead of regenerating the plan, CIRRA inserts new subtasks into the current schedule after a boundary set by the interruptibility of the active subtask, avoiding plan ambiguity; an LLM conflict gate rejects insertions that break task dependencies, preventing logical inconsistency; and new subtasks are fused into segments at the same location, reusing pending navigation to avoid execution redundancy. On CHIRP, CIRRA reaches 74.2% decision agreement, 30 points above the strongest replanning baseline, and every correct fusion decision yields a correctly placed, conflict-free schedule. On a Unitree G1 humanoid, it interrupts ongoing skills at the correct moment in every trial and outperforms all baselines on every metric. These results show that constraining where an LLM may change a plan makes replanning both more accurate and safer to execute.

AI USE STATEMENT

Generative AI tools were used solely to improve the clarity and readability of the manuscript. All AI-assisted edits were reviewed and verified by the authors, who take full responsibility for the accuracy, originality, and integrity of the final manuscript.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Suraj Borate, Vipul Pardeshi, Madhu Vadali, et al. Llm-based generalizable hierarchical task planning and execution for heterogeneous robot teams with event-driven replanning. *arXiv preprint arXiv:2511.22354*, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025.

Murtaza Dalal, Tarun Chiruvolu, Devendra Chaplot, and Ruslan Salakhutdinov. Plan-seq-learn: Language model guided rl for solving long horizon robotics tasks. *arXiv preprint arXiv:2405.01534*, 2024.

Xiaofeng Gao, Qiaozi Gao, Ran Gong, Kaixiang Lin, Govind Thattai, and Gaurav S Sukhatme. Dialfred: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters*, 7(4):10049–10056, 2022.

Marc Glocker, Peter Hönig, Matthias Hirschmanner, and Markus Vincze. Llm-empowered embodied agent for memory-augmented task planning in household robotics. *arXiv preprint arXiv:2504.21716*, 2025.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al.

$$\backslash$$

pi_ $\{0.5\}$: $avision - language - actionmodelwithopen - worldgeneralization$. *arXiv preprint arXiv:2504.16054*, 2025.

Frank Joublin, Antonello Ceravola, Pavel Smirnov, Felix Ocker, Joerg Deigmoeller, Anna Belardinelli, Chao Wang, Stephan Hasler, Daniel Tanneberg, and Michael Gienger. Copal: corrective planning of robot actions with large language models. In *2024 ieee international conference on robotics and automation (ICRA)*, pp. 8664–8670. IEEE, 2024.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*, 2024.

Jinyeon Kim, Cheolhong Min, Byeonghwi Kim, and Jonghyun Choi. Pre-emptive action revision by environmental feedback for embodied instruction following agents. In *8th Annual Conference on Robot Learning*, 2024a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024b.

Myungkyu Koo, Daewon Choi, Taeyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Hamlet: Switch your vision-language-action model into a history-aware policy. *arXiv preprint arXiv:2510.00695*, 2025.

Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. Interactive task planning with language models. *arXiv preprint arXiv:2310.10645*, 2023.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Zhengdong Lu, Weikai Lu, Yiling Tao, Yun Dai, ZiXuan Chen, Huiping Zhuang, Cen Chen, Hao Peng, and Ziqian Zeng. Decompose, plan in parallel, and merge: A novel paradigm for large language models based planning with multiple constraints. *arXiv preprint arXiv:2506.02683*, 2025.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 2017–2025, 2022.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

Shreyas Sundara Raman, Vanya Cohen, Ifrah Idrees, Eric Rosen, Raymond Mooney, Stefanie Tellex, and David Paulius. Cape: Corrective actions from precondition errors using large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14070–14077. IEEE, 2024.

Lucy Xiaoyang Shi, Brian Ichter, Michael Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, James Tanner, Anna Walling, Haohuan Wang, Niccolo Fusai, et al. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. *arXiv preprint arXiv:2502.19417*, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

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

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *arXiv preprint arXiv:2209.11302*, 2022.

Marta Skreta, Zihan Zhou, Jia Lin Yuan, Kourosh Darvish, Alán Aspuru-Guzik, and Animesh Garg. Replan: Robotic replanning with perception and language models. *arXiv preprint arXiv:2401.04157*, 2024.

Haotian Sun, Yuchen Zhuang, Lingkai Kong, Bo Dai, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. *Advances in neural information processing systems*, 36: 58202–58245, 2023.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Zixuan Wang, Bo Yu, Junzhe Zhao, Wenhao Sun, Sai Hou, Shuai Liang, Xing Hu, Yinhe Han, and Yiming Gan. Karma: Augmenting embodied ai agents with long-and-short term memory systems. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8. IEEE, 2025.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Yi Yang, Jiaxuan Sun, Siqi Kou, Yihan Wang, and Zhijie Deng. Lohovla: A unified vision-language-action model for long-horizon embodied tasks. *arXiv preprint arXiv:2506.00411*, 2025.

Haoqi Yuan, Yu Bai, Yuhui Fu, Bohan Zhou, Yicheng Feng, Xinrun Xu, Yi Zhan, Börje F Karlsson, and Zongqing Lu. Being-0: A humanoid robotic agent with vision-language models and modular skills. *arXiv preprint arXiv:2503.12533*, 2025.

Yuyang Zhang, Wenyao Zhang, Zekun Qi, He Zhang, Haitao Lin, Jingbo Zhang, Yao Mu, Xiaokang Yang, Wenjun Zeng, and Xin Jin. Imagewam: Do world action models really need video generation, or just image editing? *arXiv preprint arXiv:2606.19531*, 2026.

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*, pp. 2165–2183. PMLR, 2023.

A THE CHIRP BENCHMARK

A.1 CONSTRUCTION PIPELINE

The benchmark is built in three deterministic stages, all shipped with the code.

Stage 1: hand-authored catalog. Every episode originates from a hand-written catalog entry that fixes the scenario: the house, the robot's position and progress inside the ongoing task, the queued tasks, the injected utterance with two paraphrases, the oracle specification of the utterance, and the gold decision with its target. The catalog is organized as six categories of twenty episodes. The backbone of each category is a set of eight minimal pairs, one per room: the a-member injects a request whose location coincides with the scene the robot is currently working in and the b-member injects a nearly identically worded request anchored elsewhere, so a system that ignores execution state is guaranteed to miss one member of each pair. In four categories the pairs carry the plain gold split (a fuse, b queue); the two design decrees below reshape the remaining two, moving the waste a-members and one fetch b-member to clarify. Each category adds two preempt episodes (emergencies or explicit do-this-first orders) and two clarify episodes (underspecified action or location).

Stage 2: template generator. A template generator renders every catalog entry into a self-contained JSON episode: it expands the task templates into subtask sequences, stamps the world snapshot, renders the human-readable text vignette consumed by baselines, and copies the oracle specification and gold label. Subtasks follow the five-attribute tuple of the paper (skill, object, location, execution status, interruptibility mode); navigation subtasks carry no object attribute, and no subtask carries any duration attribute. All location attributes draw from one normalized room-level vocabulary (the eight rooms of the house), on both the ongoing tasks and the oracle specifications, so the method's exact-equality destination matching is well defined across the whole batch.

Stage 3: validation. A schema validator inside the generator checks every episode before it is written: all twelve axis fields are present and take legal values, gold decisions are one of the four classes, referenced rooms exist in the house, oracle subtasks are well-formed, and the no-duration rule holds. The 120 episodes are byte-reproducible from the catalog plus the placement stamper described below.

Design rules. Two decrees shape the label distribution. First, vague destination words ("take it out", "the usual place") never resolve to a room: a subtask location must be a concrete room, so such utterances are ambiguous by construction and gold clarify at injection. This moves eight waste-handling episodes and one fetch episode to clarify, which is why the waste category contains no fuse gold. Second, cooking is pinned to the kitchen: cooking-context episodes probe whether a system respects a pinned context rather than surface geometry. Labels were authored by the benchmark designers; every relabeling applied during construction is recorded in each episode's `provenance` field.

A.2 Episode Schema

Each episode JSON carries: `id`; `house` (the eight-room layout); `world` (robot position, user position, object placements, held items, standing facts); `active` (the ongoing task: subtask sequence, progress cursor, in-flight interruptibility mode); `queue` (zero or more queued tasks); `injection` (the utterance, two human-checked paraphrases, and the oracle specification); `gold` (decision, target with its placement anchors, rationale); `axes` (twelve generation axes: decision, phase, urgency, spatial relation, hands, gate, constraint, contract, category, clarify point, queue depth, stream length); `pair` (minimal-pair identifier); `provenance`; and `rendering` (the plain-text vignette consumed by baselines).

A.3 Category Composition

Table 4 gives the gold decision composition of the six categories.

Table 4: Gold decision composition of the six categories.

Category	Probe axis	Fuse	Queue	Clarify	Preempt
Appliance	same/diff	8	8	2	2
Clean	same/diff	8	8	2	2
Tidy	same/diff	8	8	2	2
Cook	pinned ctx.	8	8	2	2
Fetch	shared dest.	8	7	3	2
Waste	co-carry	0	8	10	2
Total		40	47	21	12

A.4 Placement Targets and Method-Structure Probes

Beyond the decision label, every fuse gold carries a `placements` target: the visit the new behavior must ride and its local position inside that visit (or a reuse marker when an identical pending subtask already exists). The decision labels are hand-authored; the placement layer follows one mechanical doctrine, the frontmost admissible position of the earliest matched visit, and is stamped by a judge-free symbolic pass and audited by hand, with the derivation recorded in each episode's `provenance`.

A twelve-episode probe supplement (released alongside the main batch, not counted in its tables) covers three mechanism structures the main batch does not isolate: duplicate reuse (PBR01–PBR04: the requested behavior is already pending in the matched visit and must be reused, not duplicated), insertion ahead of an immediate-interruptible in-flight behavior (PBI01–PBI04), and traversal past an inadmissible earliest visit to a later same-destination visit, with the undeliverable part of the request queued as remainder (PBV01–PBV04). The probes are generated and stamped by the same pipeline as the main batch.

ID	Utterance	Gold
HA01a	Switch off the bedside lamp too while you're in there.	fuse
HA01b	Turn on the exhaust fan in the bathroom too.	queue
HA02a	Power the printer down too while you're at the desk.	fuse
HA02b	Close the garage door too.	queue
HA03a	Switch the range hood on as well while you're there.	fuse
HA03b	Latch the living room window too.	queue
HA04a	Turn the towel warmer on too while you're in there.	fuse
HA04b	Switch off the desk lamp in the study too.	queue
HA05a	Turn the floor lamp on too while you're by the window.	fuse
HA05b	Turn the garden sprinkler on out in the yard too.	queue
HA06a	Check the breaker box too while you're down there.	fuse
HA06b	Start the dishwasher in the kitchen too.	queue
HA07a	Kill the garage lights too while you're there.	fuse
HA07b	Check the water heater in the basement too.	queue
HA08a	Turn the yard lights on too while you're out there.	fuse
HA08b	Close the bedroom window too.	queue
HA09	The smoke alarm is going off in the kitchen - go check right now!	preempt
HA10	The garage door is closing on the car - stop it now!	preempt
HA11	Turn off the lamp too.	clarify
HA12	Turn on the heater, it's getting cold.	clarify

Table 5: Episode listing (1 of 6): the Appliance category (20 episodes). The utterance is the original surface form.

ID	Utterance	Gold
HC01a	Wipe the windowsill in here too while you're at it.	fuse
HC01b	The garage floor needs sweeping too.	queue
HC02a	Give the desk a wipe too while you're dusting.	fuse
HC02b	The stove top in the kitchen needs a wipe too.	queue
HC03a	Do the stove top as well while you're wiping.	fuse
HC03b	The bathroom mirror needs wiping as well.	queue
HC04a	Wipe the mirror too while you're in there.	fuse
HC04b	The garden table outside needs a wipe too.	queue
HC05a	Do the TV stand too while you're wiping.	fuse
HC05b	The basement floor needs sweeping too.	queue
HC06a	Wipe the shelves down there too while you're at it.	fuse
HC06b	The desk in the study needs dusting too.	queue
HC07a	Wipe the workbench too while you're in there.	fuse
HC07b	The bedroom windowsill needs wiping too.	queue
HC08a	Wipe the garden table too while you're out there.	fuse
HC08b	The living room table needs a wipe too.	queue
HC09	The kids spilled juice all over the living room - clean it up now!	preempt
HC10	The bathtub is overflowing - turn off the tap now!	preempt
HC11	Wipe down the table too.	clarify
HC12	Clean up the mess when you can.	clarify

Table 6: Episode listing (2 of 6): the Clean category (20 episodes). The utterance is the original surface form.

A.5 Full Episode Listing

Tables 5 through 10 list every episode by category: identifier, the original surface form of the injected utterance, and the gold decision. The two paraphrases, the world snapshots, the ongoing and queued tasks, the oracle specifications, and the gold targets are in the released JSON files, one file per episode.

ID	Utterance	Gold
HT01a	The scarf on the bed goes into the wardrobe too.	fuse
HT01b	The book on the study desk needs to go back on the shelf too.	queue
HT02a	File the documents into the drawer too while you're there.	fuse
HT02b	The cushion in the living room goes back on the sofa too.	queue
HT03a	The mug on the counter goes in the cupboard too.	fuse
HT03b	The ball out in the yard goes back in the toy bin too.	queue
HT04a	Put the shampoo back on its shelf too while you're in there.	fuse
HT04b	The flashlight in the basement goes back in its drawer too.	queue
HT05a	The magazine goes back on the rack too while you're at the sofa.	fuse
HT05b	The extension cord in the garage needs coiling back onto its shelf too.	queue
HT06a	Put the flashlight back in its drawer too while you're down there.	fuse
HT06b	The scarf on the bed upstairs goes into the wardrobe too.	queue
HT07a	Coil the extension cord back onto its shelf too while you're there.	fuse
HT07b	The clean dishes on the counter go into the cupboard too.	queue
HT08a	The garden gloves go back in the shed box too while you're out there.	fuse
HT08b	The towel in the bathroom needs hanging back on the rack too.	queue
HT09	The stove is still on - turn it off first!	preempt
HT10	Grandpa slipped in the bathroom - go check on him now!	preempt
HT11	Put my cup away too.	clarify
HT12	Stash the spare keys somewhere safe.	clarify

Table 7: Episode listing (3 of 6): the Tidy category (20 episodes). The utterance is the original surface form.

ID	Utterance	Gold
HK01a	Put the kettle on too while you're at the stove.	fuse
HK01b	Also bring my glasses from the bedroom.	queue
HK02a	Toast a slice of bread as well while you're there.	fuse
HK02b	Also bring my laptop from the study.	queue
HK03a	Fill the water pitcher too while you're at the sink.	fuse
HK03b	The towel in the bathroom needs hanging back up too.	queue
HK04a	Warm up some milk as well while you're brewing.	fuse
HK04b	The living room table needs a wipe too.	queue
HK05a	Pour a glass of juice too while you're at it.	fuse
HK05b	Also bring the storage box up from the basement.	queue
HK06a	Rinse the salad greens too while the pot is going.	fuse
HK06b	The bike pump in the garage goes back on its hook too.	queue
HK07a	Rinse the berries for dessert too while you're mixing.	fuse
HK07b	The patio outside needs sweeping too.	queue
HK08a	Pour two glasses of water too while you're there.	fuse
HK08b	The cushion in the living room goes back on the sofa too.	queue
HK09	The baby is crying upstairs - go check on her right now!	preempt
HK10	Something's burning in the oven - pull it out now!	preempt
HK11	Add my usual sauce.	clarify
HK12	Grab the special seasoning my mom left.	clarify

Table 8: Episode listing (4 of 6): the Cook category (20 episodes). The utterance is the original surface form.

ID	Utterance	Gold
HF01a	While you're in the bedroom, grab my phone charger too.	fuse
HF01b	Also bring the bike pump from the garage.	queue
HF02a	Bring the documents from the study along as well.	fuse
HF02b	And bring me an apple from the kitchen.	queue
HF03a	Grab a snack bar too while you're in the kitchen.	fuse
HF03b	Also fetch my phone charger from the bedroom.	queue
HF04a	Bring the hair dryer too while you're in the bathroom.	fuse
HF04b	Also bring the watering can in from the yard.	queue
HF05a	Take the magazine on the couch up too.	fuse
HF05b	Also bring the toolbox up from the basement.	clarify
HF06a	Bring the spare bulbs up too while you're down there.	fuse
HF06b	Also bring my laptop from the study.	queue
HF07a	Grab the extension cord too while you're in the garage.	fuse
HF07b	Also bring a towel from the bathroom.	queue
HF08a	Bring the garden gloves in too while you're out there.	fuse
HF08b	Also bring the flashlight up from the basement.	queue
HF09	Quick, I need my medicine right now!	preempt
HF10	Water's leaking under the bathroom sink - go shut the valve now!	preempt
HF11	Bring me my mug.	clarify
HF12	Bring me my headphones.	clarify

Table 9: Episode listing (5 of 6): the Fetch category (20 episodes). The utterance is the original surface form.

ID	Utterance	Gold
HW01a	The recycling bag in there goes out too.	clarify
HW01b	Bring my pillow down to the living room while you're at it.	queue
HW02a	The study recycling goes out too.	clarify
HW02b	Also bring the signed documents to the living room for me.	queue
HW03a	The kitchen recycling goes out too.	clarify
HW03b	Grab me an apple on your way too.	queue
HW04a	The bathroom recycling goes out too.	clarify
HW04b	Bring the fresh towel to the living room for me too.	queue
HW05a	The recycling stack here goes out too.	clarify
HW05b	Take the cushion over to the bedroom for me while you're up.	queue
HW06a	The basement recycling goes out too.	clarify
HW06b	Bring the water bottles up to the kitchen too.	queue
HW07a	The garage recycling goes out to the curb too.	clarify
HW07b	Bring the soda crate into the kitchen too.	queue
HW08a	The yard recycling goes in too.	clarify
HW08b	Bring the picked tomatoes into the kitchen too.	queue
HW09	The delivery person is waiting at the gate - go now!	preempt
HW10	The trash truck is on our street - get the bins to the curb right now!	preempt
HW11	Toss the old box too.	clarify
HW12	Throw away the broken umbrella.	clarify

Table 10: Episode listing (6 of 6): the Waste category (20 episodes). The utterance is the original surface form.

B BASELINE REPRODUCTION DETAILS

All baselines run inside one harness under identical conditions: the same shared compilation feeds every plan-consuming method, every LLM call goes through one gateway with one decoding contract (temperature 0, fixed seed, thinking disabled, schema-constrained JSON), and every regeneration baseline emits reference-constrained schedules (`old:<id>` keeps a pending step, `new:<i>` schedules a new one), so the induced decision and the commitment violations are computed by one deterministic classifier with no LLM in the grading loop. For each baseline we record the first-hand source, what is reproduced faithfully, and every adaptation made to run it on our episodes. Raw source material (original prompt files where published) ships in `benchmark/baselines_ref/`.

ITP (Li et al., 2023). Faithful: the replanning context is the task guide plus completed steps plus the new request plus chat history, continued into the prior conversation, and the system plans one task thread at a time, as in the original drink-making deployment. Adaptations: the original emits GPT-4 function calls, ours emits leaf references so the induced decision is computable; the task guide's few-shot menu examples are replaced by our domain skill list; the backbone is the protocol-wide Qwen3-8B instead of GPT-4.

CoPAL (Joublin et al., 2024). Faithful, rebuilt from the original prompt files in the authors' repository: the Travi role frame (human, Travi, robot), the four numbered duties, the "do not replicate actions already done" instruction, the feedback re-prompt block (request, feedback, command history, JSON state), the output fields Goal / Objects needed / State summary / Remaining actions, and the Error, Reason, Suggestion triplet phrasing. Adaptations: the feedback triplet carries a user interjection instead of an execution error (the original replans on errors, not on new instructions, which is itself a point of comparison); the four-layer cognitive architecture collapses to the high-level replanning layer for a text benchmark; temperature follows the unified protocol instead of the original 0.8; the replan loop cap becomes one round.

CoMuRoS (Borate et al., 2025). Faithful: the Task Manager prompt's static block (rules and format) and dynamic block (user command, chat history, robot state, task states, events); the three task states COMPLETED / IN_PROGRESS / INTERRUPTED; replanning reorders only non-completed tasks; the new instruction arrives as a detected event; whole-task granularity (tasks are never split), which is this family's native action space. Adaptations: the multi-robot allocation axis does not apply (single robot); the paper does not publish the verbatim template, so the block wording is rebuilt from its description; output is constrained to a task-order JSON.

DPPM (Lu et al., 2025)†. Faithful: the published merge instruction ("merge and revise two partial plans based on feedback from critics into one that meets the requirements"), the Partial Plan A / Feedback A / Partial Plan B / Feedback B field structure, the pure-JSON output requirement, and explicit conflict annotations (transport exclusivity in the original, hand capacity and forbidden co-carries here). Adaptations are larger, hence the dagger: the original is offline text travel planning, ours is an online embodied stream; Plan A comes from the live commitments rather than a parallel sub-planner; the verify-refine loop (up to ten rounds) becomes a single merge; the Cartesian candidate combination does not apply to a two-plan merge.

CoT replan (Wei et al., 2022). The chain-of-thought baseline of the DPPM comparison pool: identical task and input to Conjunctive replan, with the schema forcing an in-band `reasoning` field before the plan.

LLM-Modulo (Kambhampati et al., 2024). The generate-test baseline of the DPPM pool. The proposal LLM is checked by a sound verifier, and the verifier is our symbolic simulator (commitment accounting, location contracts, hand capacity, forbidden co-carries), with violations fed back for revision, up to three rounds. This is a deliberate concession: the strongest verifier available in this setting, isolating "verifier without enumeration" from "enumeration plus verifier."

Reflexion (Shinn et al., 2023). Proposal (sharing the Conjunctive replan prompt and cache), one self-reflection pass prompted to check for dropped steps, infeasible order, forbidden co-carries, and missed ride-alongs, then one revision. No external verifier: together with LLM-Modulo this forms the introspection-versus-external-check contrast.

SayCan (Ahn et al., 2022)†. A structural constant, stated openly: SayCan has no clarification, no emergency handling, and no weaving mechanism; each instruction is planned independently and executed in arrival order, so in this decision space it is queue-at-tail on every episode and involves no LLM call. The row quantifies the ceiling of the sequential one-request contract under the field's foundational system; its affordance-scoring internals are invisible to DA and TA and reproducing them would not change any reported number.

Memory-agent (Glocker et al., 2025)†. Faithful, using the original system message from the authors' repository: the object-list dependence, the full clarification rules (synonyms pass, near-matches ask, missing objects report, uncertainty asks), the single-request task-JSON output contract, and the chat-history mechanism (the active task enters through it). Its native clarification behavior makes this row meaningful on clarify episodes. Adaptations, hence the dagger: a third task type "action" is added (the original knows only placement and handover, without which sixty chore episodes are inexpressible at the vocabulary level and the comparison would measure vocabulary, not scheduling); its three native output forms merge into one schema for constrained decoding; the routing agent is bypassed (all inputs are task requests); the RAG memory idles because a frozen single moment has no long interaction history, so its memory axis is not assessed here; the backbone is the protocol-wide Qwen3-8B.

Considered and excluded. LLM+P (**?**) encodes problems into PDDL for a classical planner, but arbitration is not goal-reachability planning: commitment order, preemption semantics, and same-visit structure have no natural PDDL expression, so forcing the encoding would strawman the method. Code as Policies (Liang et al., 2023) and ProgPrompt (Singh et al., 2022) belong to the code-generation family (the LLM writes a program against the task tree), a different comparison family left as an extension. CoMuRoS's own paper reports no quantitative baseline to borrow.

Grading fairness. Classification and violation counting use the same deterministic classifier for every regeneration baseline. One known expressiveness boundary is disclosed: a flat reference sequence cannot express "ride the same visit" tree structure, so end-state fuse golds are systematically classified queue for flat-output systems. This is not a grading artifact but the direct cost of flattening away commitment structure, which is part of the paper's argument.

C Real-World Deployment Details

Skills. Table 11 lists the eight behavioral skills and the navigation skill exposed to CIRRA, together with their interruptibility modes. A skill is `immediate` if halting it at any time leaves no object in a hazardous state, `safe_point` if it passes through stable intermediate states, with a safe boundary after each placement or stage, and `uninterruptible` if it keeps an object in a precarious configuration throughout execution. Safe boundaries are detected online from the gripper state and a per-skill stage counter. A paused skill switches its policy to standby mode, identically for every method, and later resumes from the current observation.

Table 11: Real-world skill library. Each visuomotor policy is exposed to CIRRA as one behavioral skill with an interruptibility mode ι.

Skill	Task	ι	Safe boundary / rationale
BALL_CAN	Insert a ball into a hand-held can	uninterruptible	bimanual; releasing drops the can or ball
BASKET	Put a block into a basket	immediate	halting is harmless at any time
SWAP	Swap the items on two plates	safe_point	after each item is placed on a plate
DRAWER	Put a block into the bottom drawer	safe_point	after each stage (open, place, close)
SORT_ITEM	Sort items into baskets by color	immediate	halting is harmless at any time
HANG	Hang a tape roll on a rack	uninterruptible	the held tape is aligned with the peg
SORT_TOOL	Sort tools into a tool case	safe_point	after each tool is placed in its slot
STACK_CUBE	Stack cubes	safe_point	after each cube is released
NAV	Walk to table A or B	immediate	walking can be stopped at any time

Task combinations. Table A stands to the left of table B, and the tasks on each table are arranged from left to right in the order listed in Table 12. No ordering rule is given to the agent, which must infer the layout from $\mathcal{E}_t$. SORT_ITEM is a single skill whose item order is chosen by the policy. Each combination is run 25 times, and in each trial u_t is issued at a different phase of the ongoing sequence, e.g., during navigation, at the onset of a grasp, or mid-manipulation; the gold decision, boundary b_t and revised sequence are annotated for every trial before execution. Trials in which a manipulation policy fails to grasp are excluded. Ambiguity resolution is evaluated on CHIRP and not repeated here.

Table 12: Real-world task combinations. Tasks on each table are listed from left to right. In each of the 25 trials, u_t arrives while a different subtask of $\mathcal{P}_t$ is active, and the gold boundary b_t follows from the interruptibility mode of that subtask (Table 11). SORT_ITEM$(\cdot)$ lists the items to sort.

#	Layout	Ongoing $\mathcal{P}_t$	New instruction u_t	Gold decision and $\mathcal{P}'_t$
1	A: SORT_ITEM B: BALL_CAN, DRAWER	NAV(A), SORT_ITEM(red, blue, yellow)	"Now go to table B, put the tennis ball into the can, then put the block into the bottom drawer."	*preempt* at b_t: NAV(B), BALL_CAN, DRAWER, then NAV(A) and the unfinished part of $\mathcal{P}_t$
2	A: BALL_CAN B: BASKET, SORT_TOOL	NAV(A), BALL_CAN	"Now go and put the screwdriver on table B into the tool case."	*preempt* at b_t: NAV(B), SORT_TOOL, then NAV(A) and the unfinished part of $\mathcal{P}_t$, if any
3	A: HANG, SWAP B: SORT_TOOL	NAV(A), HANG, SWAP	"By the way, put the tools on table B into the tool case."	*queue*: NAV(A), HANG, SWAP, NAV(B), SORT_TOOL
4	A: SORT_ITEM B: BALL_CAN, DRAWER	NAV(B), BALL_CAN, NAV(A), SORT_ITEM(red), NAV(B), DRAWER	"By the way, put the yellow and blue items into the baskets of the same colors."	*fuse* into the A segment: NAV(B), BALL_CAN, NAV(A), SORT_ITEM(red), SORT_ITEM(yellow, blue), NAV(B), DRAWER
5	A: BASKET B: HANG, DRAWER	NAV(A), BASKET, NAV(B), DRAWER	"By the way, hang the tape on table B on the rack."	*fuse* into the B segment before DRAWER (left-to-right layout): NAV(A), BASKET, NAV(B), HANG, DRAWER