GroundAct: Can LLM Agents Ground Actions in Environmental States?

Anonymous ACL submission

Abstract

LLM agents achieve 85–96% success on tasks where instructions fully specify the action, but drop to 29–53% when action feasibility depends on environmental state that the instruction does not mention. We argue that this gap reflects a missing capability: action grounding, the ability to infer from structured environmental state whether an action is feasible, what prerequisites it lacks, and whether it exceeds individual capacity. We introduce GroundAct, a benchmark of 1,500 scenarios and 16,592 task instances in text-based interactive environments spanning 11 domains, with tasks organized into seven categories along a cognitive complexity hierarchy. Evaluating 15 LLMs (3B–671B), we find three diagnostic patterns: (i) attribute reasoning is weakly correlated with tool and coordination reasoning, producing distinct model profiles; (ii) complete environment graphs yield up to $+27.6/-22.9\%$ on tool use vs. implicit collaboration, separating search-bound from constraint-filtering bottlenecks; and (iii) supervised fine-tuning lifts Qwen2.5-3B from 0.6% to 76.3% on direct command but only 1.5% to 5.5% on implicit collaboration. These results establish action grounding as a multi-dimensional challenge irreducible to scaling.

1 Introduction

Consider an agent asked to "clear the workbench." Three items remain: a coffee mug the agent can lift, a 50 kg cable spool that exceeds the agent's load limit, and a coffee stain that requires a wiping tool. None of these constraints is named in the instruction; they live in the environment. Stronger models recognize each, recruit a partner for the spool, fetch a cloth for the stain, and complete the task. Weaker models attempt the spool alone, repeat the failed grab after an explicit error, and declare the task done. We call the ability to infer such constraints *action grounding*.

We decompose action grounding into three operationalized dimensions. *Attribute reasoning* grounds action targets to continuous object properties, requiring comparison across candidates rather than predicate lookup, such as identifying the heaviest of several boxes or the only object above a temperature threshold. *Tool reasoning* grounds action prerequisites to dynamic capability availability: removing the stain requires grasping the cloth, which extends the agent's action set to include `clean`; releasing the cloth retracts this capability. *Coordination reasoning* grounds action feasibility to individual capacity limits: when the spool's weight exceeds any single agent's load limit, the agent must recognize the mismatch and initiate collaboration without explicit direction. The structural property shared by all three: the constraint that determines what action to take, or whether the agent can take it alone, lives in the environment, not the instruction.

> 💡 *Do benchmarks evaluate whether agents can ground actions in environmental states?*

Existing interactive-agent benchmarks implicitly perform action grounding on the agent's behalf. In ALFRED (Shridhar et al., 2020) and ALFWorld (Shridhar et al., 2021), instructions name target objects by identifier ("put mug_1 on desk_1"), so attribute reasoning is pre-solved; BEHAVIOR-1K (Li et al., 2024a) extends attribute coverage but retains discrete predicates (open/-closed, clean/dirty) as primary state variables. In tool-use evaluations such as PARTNR (Chang et al., 2024) and EmbodiedBench (Yang et al., 2025b), tool-corresponding actions are placed in the action set at initialization, so tool reasoning is pre-loaded. In multi-agent benchmarks such as PARTNR, TDW-MAT, and C-WAH (Zhang et al., 2024), collaboration is triggered by explicit directives ("Agent A and Agent B cooperate to…") or

Figure 1: Overview of the GroundAct framework comprising three integrated components: **GA-Sim** (left) uses structured text representation to model environments with objects, agents, and spatial relationships, enabling dynamic tool-capability binding and physics-constrained collaboration; **GA-Bench** (right) presents our comprehensive evaluation matrix spanning single-agent and multi-agent tasks across increasing cognitive complexity levels.

by efficiency optimization, so coordination reasoning is pre-resolved. Across these settings, the action, tool, or collaboration pattern is supplied rather than inferred; existing benchmarks evaluate action execution under pre-grounded constraints, not action grounding from environmental state. Table 2 summarizes the contrast.

We introduce **GroundAct**, a benchmark designed so each task category requires a form of action grounding that prior work bypasses. Three design choices instantiate the three reasoning dimensions: 6,381 continuous attribute types as first-class primitives, so attribute reasoning requires comparison rather than predicate lookup; dynamic tool-capability binding over 214 action types, where grasping a tool extends the agent's action set at runtime and releasing it retracts the extension, so tool reasoning requires capability-gap detection and acquisition; and constraint-triggered collaboration encoded in task preconditions, where collaboration emerges from physical infeasibility rather than instruction directives, so coordination reasoning requires comparing object properties against agent parameters. GroundAct contains 1,500 validated scenarios and 16,592 task instances across 11 domains, organized into seven categories along two axes: cognitive complexity and agent configuration. GroundAct uses text-based structured environments by design, isolating action grounding from perceptual challenges so every failure can be attributed to reasoning rather than perception.

Evaluating 15 representative LLMs (3B–671B parameters), we find three diagnostic patterns. First, attribute reasoning is weakly correlated with tool and coordination reasoning, producing distinct capability profiles across models: GPT-4o scores 77.8% on attribute reasoning but 32.0% on compound collaboration, while DeepSeek-R1 shows the opposite at 41.9% and 48.5%. Second, exposing the complete environment graph yields up to $+27.6/-22.9\%$ on tool use vs. implicit collaboration, separating search-bound from constraint-filtering bottlenecks. Third, supervised fine-tuning lifts Qwen2.5-3B from 0.6% to 76.3% on direct command but only from 1.5% to 5.5% on implicit collaboration, exposing a graded learnability profile across dimensions.

Our contributions are as follows:

- We define *action grounding* as inferring action-relevant constraints from environmental state, operationalized as attribute, tool, and coordination reasoning.
- We introduce **GroundAct**, a benchmark of 1,500 scenarios and 16,592 task instances across 11 domains in seven categories, each requiring a form of grounding bypassed by prior work.
- Evaluating 15 LLMs, we show GroundAct yields dimension-level diagnostics: separating instruction execution from grounding, exposing capability profiles that aggregate scores hide, and reveal-

ing uneven effects of scaling, reasoning training, and fine-tuning across grounding dimensions.

2 Related Works

Prior embodied benchmarks have made significant contributions to task evaluation, but differ fundamentally in their approach to physical reasoning and collaboration. While ALFRED (Shridhar et al., 2020) and BEHAVIOR-1K (Li et al., 2024a) provide extensive task coverage, they model physical states through discrete representations (e.g., binary door states, picked/placed objects) rather than continuous attributes necessary for reasoning about weight, temperature, or material properties. Tool usage evaluation spans from low-level manipulation in RoCo (Mandi et al., 2024) to high-level planning in PARTNR (Chang et al., 2024), yet both maintain static action spaces determined at initialization, preventing assessment of dynamic capability acquisition. Recent multi-agent benchmarks including TDW-MAT (Zhang et al., 2024) and EmbodiedBench (Yang et al., 2025b) advance collaboration evaluation through load constraints and task allocation optimization, but rely on explicit task division instructions or efficiency-driven participation rather than collaboration emerging from physical constraints.

In contrast, GroundAct introduces continuous property reasoning with 6,381 distinct attributes, dynamic tool-capability binding that expands action spaces during execution, and implicit collaboration where agents must autonomously recognize when tasks exceed individual capacities. This fundamentally shifts evaluation from instruction compliance to constraint-based reasoning. A comprehensive comparison is in Appendix A.1.

3 Framework

We present GroundAct, a comprehensive framework for evaluating agent reasoning in embodied tasks. Our framework addresses the fundamental challenge of assessing whether language models understand embodied principles. We achieve this through three key design principles: (1) tasks must require reasoning about physical properties and constraints rather than following explicit instructions, (2) agent capabilities should dynamically evolve based on tool acquisition rather than remaining static, and (3) collaboration needs should emerge from task requirements rather than predetermined protocols.

3.1 Task Design and Formalization

Environment Representation. We formalize embodied environments as directed graphs $G_t = (V_t, E_t, A_t)$ that capture the essential structure of physical spaces (Côté et al., 2019; Yuan et al., 2019). The node set V_t encompasses three entity types: spatial nodes representing rooms and areas, object nodes for interactive items, and agent nodes for autonomous entities. Each node maintains an attribute dictionary A_t storing continuous physical properties such as weight, temperature, material composition, and geometric dimensions. The edge set E_t encodes spatial relationships through static containment relations (e.g., "in", "on") and dynamic proximity relations E_{near} that track which objects fall within an agent's interaction range. This graph representation enables efficient reasoning about spatial constraints while avoiding the computational overhead of continuous 3D simulation.

Task Formalization. Each evaluation task is defined as a tuple $\mathcal{T} = (S_{\text{init}}, I, G_{\text{goal}}, \mathcal{A}_{\text{task}})$, where S_{init} specifies the initial environment state, I provides the natural language instruction, G_{goal} defines success conditions through logical predicates, and $\mathcal{A}_{\text{task}}$ identifies participating agents. The evaluation objective is to assess whether agents can generate an action sequence $\Pi = (\pi_1, \ldots, \pi_T)$ that transforms the environment from S_{init} to a terminal state S_{final} satisfying all predicates in G_{goal}. This formalization captures both the planning and execution aspects of embodied reasoning.

3.2 Hierarchical Task Taxonomy

Our evaluation framework organizes tasks along cognitive complexity (L0: instruction following, L1: dimensional reasoning, L2: compound tasks), with agent configuration (single vs. multi-agent) as a secondary axis. This structure enables systematic assessment of how reasoning capabilities scale with task demands. Representative examples for each task category are provided in Appendix A.2.

L0: Instruction Following. ***Direct Command (DC)*** tasks require straightforward instruction following, such as "place cup#1 on table#1," establishing baseline comprehension abilities. ***Explicit Collaboration (EC)*** tasks provide clear coordination directives, such as "Agent A and Agent B cooperate to open the heavy cabinet," testing fundamental synchronization abilities.

L1: Dimensional Reasoning. ***Tool Use (TU)*** tasks demand recognizing capability gaps and acquiring right tools. For instance, "clean the table" requires agents to identify that cleaning actions are unavailable in their base action set $\mathcal{A}_i$, locate cleaning tools, and execute $\texttt{grasp}(v_{\text{tool}})$ to dynamically expand their capabilities. ***Attribute Reasoning (AR)*** tasks require comparing continuous properties to identify targets (e.g., "move the heaviest cup" requires solving $v^* = \arg\max_{v \in V_{\text{cups}}} A_t(v, \text{weight})$). ***Implicit Collaboration (IC)*** removes explicit instructions, requiring agents to autonomously recognize when tasks exceed individual capabilities. For example, "move the dining table to the storage room" requires agents to infer that $A_t(v_{\text{table}}, \text{weight}) > C_{\max}(i)$ for any individual agent i, necessitating collaborative effort.

L2: Compound Tasks. ***Compound Reasoning (CR)*** tasks integrate multiple challenges, such as "clean the heaviest table," requiring simultaneous attribute comparison, tool acquisition, and multi-step planning. ***Compound Collaboration (CC)*** combines all elements, such as "cooperatively repair the malfunctioning television," demanding tool acquisition, capability assessment, and coordinated execution.

3.3 GA-Sim: Efficient Environment Simulation

State Representation and Updates. GA-Sim employs text-based environment modeling to achieve efficient simulation at scale. The graph structure G_t maintains spatial relationships through topological connections rather than continuous coordinates, eliminating expensive collision detection while preserving essential spatial constraints. State updates follow an incremental approach where actions modify only directly affected nodes and edges. For instance, when an agent executes $\texttt{GOTO}(\text{table})$, the system updates only the relevant proximity relations in E_{near} rather than recomputing global spatial relationships.

Dynamic Capability Management. A key innovation in GA-Sim is the dynamic tool-capability binding system. Agent actions are partitioned into basic actions (movement, grasping, opening) available to all agents, and tool-dependent actions (cleaning, heating, repairing) that require specific tools. Each tool object maintains a `capability` attribute specifying which actions it enables. When an agent grasps a tool, the system dynamically binds the associated capabilities to the agent's action set. Upon releasing the tool, these capabilities are automatically unbound. This mechanism enables realistic modeling of how agents extend their abilities through tool use, moving beyond the static action spaces of existing benchmarks.

Emergent Collaboration. GA-Sim supports collaboration that emerges from physical constraints rather than explicit programming. When agents attempt actions on objects whose properties exceed individual capabilities, the system enables collaboration request mechanisms. For instance, if an agent attempts to move an object where $A_t(v, \text{weight}) > C_{\max}(\text{agent})$, it can initiate collaboration by identifying suitable partners and coordinating joint actions. The system validates preconditions for all participating agents and maintains consistency throughout collaborative execution, ensuring realistic multi-agent interactions.

3.4 Automated Benchmark Generation

Generation Pipeline. Creating diverse, physically consistent scenarios at scale requires careful orchestration of neural generation and symbolic validation. As shown in Figure 2, our pipeline operates in four stages, each combining the creative capabilities of large language models with rule-based consistency checking. This hybrid approach enables generating thousands of unique scenarios while maintaining physical realism and task solvability. A four-stage walkthrough for one scenario is in Appendix A.11.

Scene and Task Generation. Scene generation begins with semantic seeds extracted from diverse text sources (Li et al., 2024b; Xu et al., 2025; Wang et al., 2023), which guide a neural generator g_{scene} in creating structured environment descriptions. The generator, implemented using high-temperature language models for diversity, produces initial scenes S_0 containing objects, spatial layouts, and agent configurations. Task generation follows a two-stage process: first, an environment analyzer C_{env} extracts feasible actions based on scene structure, then a task generator g_{task} creates instructions anchored in physical possibilities. This grounding prevents generation of impossible tasks while maintaining creative diversity.

Evaluation Logic and Trajectories. For each task, we automatically derive evaluation criteria by parsing the instruction and scene to extract minimal

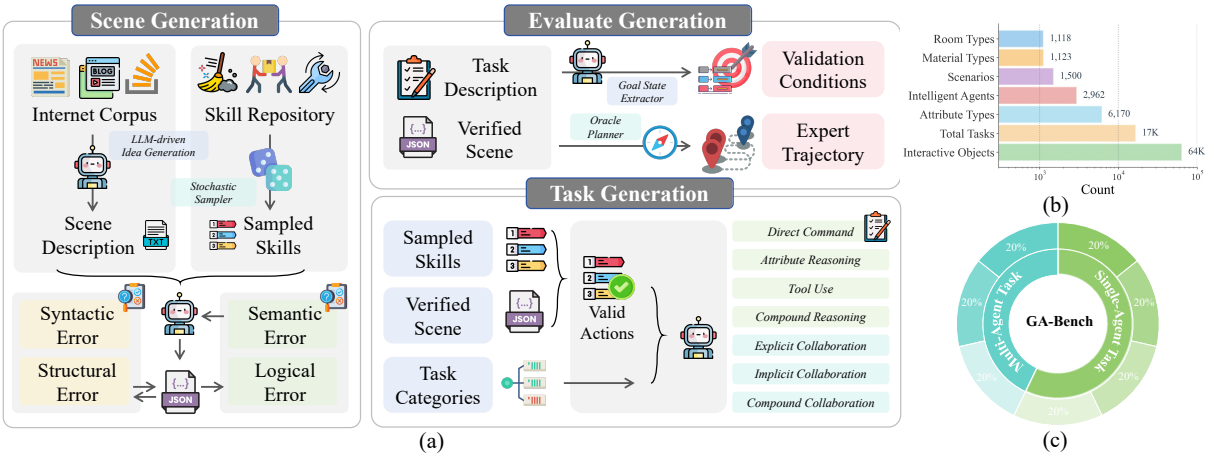


Figure 2: GroundAct automated benchmark generation and evaluation framework. (a) Four-stage generation pipeline combining LLMs with rule-based validation: scene generation from internet corpus, task generation with skill sampling, evaluation logic extraction, and expert trajectory generation with human validation. (b) GA-Bench statistics: 1,500 scenarios, 64K objects, 6K attribute types, spanning diverse domains and material compositions. (c) Balanced task distribution across seven categories spanning single-agent (Direct Command, Tool Use, Attribute Reasoning, Compound Reasoning) and multi-agent tasks (Explicit/Implicit/Compound Collaboration).

state changes required for success. This produces a goal predicate set G_{goal} that serves as an objective success measure. Expert trajectories are generated using oracle agents with complete environmental knowledge, creating high-quality demonstrations for each task. These trajectories undergo filtering to remove suboptimal sequences, providing ideal solutions for comparison and learning.

Quality Assurance. As shown in Figure 3, generated scenarios undergo a three-stage validation pipeline. Automated validators check structural consistency and physical feasibility, correcting 87.6% of detected errors. Trained human annotators then verify scenarios against explicit criteria for instruction clarity, feasibility, and solvability (Cohen's $\kappa = 0.84$). Finally, oracle agents validate expert trajectories through simulation replay. This rigorous process yielded 1,500 validated scenarios from 2,100 initial candidates (71.4% acceptance rate). Detailed procedures, annotation interface, and quality threshold examples are provided in Appendix A.4.

3.5 Benchmark Statistics and Coverage

GA-Bench encompasses 1,500 scenarios across 11 domains including laboratory (39%), office (19%), industrial (12%), and medical environments, containing 64,057 interactive objects with rich physical properties. The dataset maintains careful balance across our task taxonomy: 65% single-agent tasks spanning all complexity levels, and 35% multi-agent tasks with emphasis on implicit collaboration

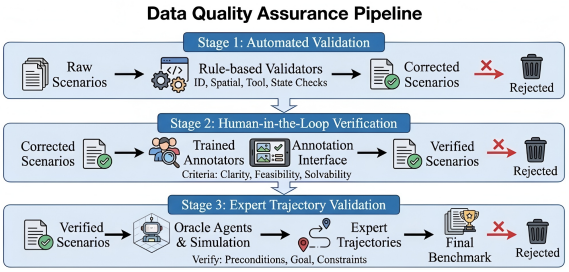


Figure 3: Overview of the Data Quality Assurance Pipeline. The pipeline consists of three sequential stages: (1) Automated Validation for rule-based error correction, (2) Human-in-the-Loop Verification for semantic and physical feasibility checks, and (3) Expert Trajectory Validation to ensure genuine solvability via oracle agents.

scenarios requiring genuine coordination reasoning. With 6,381 distinct property types and 214 action types, GA-Bench provides comprehensive coverage of embodied reasoning challenges while maintaining tractable evaluation scope. Detailed statistics are provided in Appendix A.5.

4 Experiments

We systematically evaluate current LLMs on GA-Bench to assess their physical reasoning capabilities in embodied tasks.

4.1 Experimental Setup

Model Selection. We evaluate 15 representative models spanning three architectural paradigms. Closed-source models include GPT-4o (Hurst et al., 2024), Gemini-2.5-Flash (Comanici et al., 2025),

Model	L0: Instruction Following				L1: Dimensional Reasoning						L2: Compound Tasks				Overall
	Direct Command		Explicit Collab.		Tool Use		Attribute Reasoning		Implicit Collab.		Compound Reasoning		Compound Collab.		
	SR	Step	SR	Step	SR	Step	SR	Step	SR	Step	SR	Step	SR	Step	SR
Closed-source Models															
Claude-Opus-4.6	95.3	9.4	92.5	5.8	**89.9**	11.6	55.7	10.3	91.0	8.2	**79.4**	12.4	**53.0**	12.7	79.5
Claude-Sonnet-4.6	95.9	10.2	92.0	6.3	84.6	11.8	70.1	10.2	91.5	7.8	74.1	12.7	52.5	12.0	**80.1**
GPT-4o	**96.6**	12.9	90.0	13.9	80.0	13.6	**77.8**	12.3	77.5	14.4	69.2	14.5	32.0	22.9	74.7
Gemini-2.5-Flash	90.5	11.0	88.5	8.4	82.3	16.5	56.3	17.5	85.5	7.1	59.4	20.0	40.5	16.2	71.9
GLM-5	93.5	9.8	**93.0**	6.2	84.6	11.6	61.7	10.0	**92.5**	8.3	72.9	12.9	48.0	11.7	78.0
GLM-5-Turbo	94.1	10.2	92.5	6.7	85.8	12.0	49.1	10.7	88.5	8.6	74.1	13.5	46.0	12.8	75.7
Qwen3.6-Plus	92.9	9.7	90.5	6.8	85.2	11.2	52.1	9.4	86.5	8.5	70.6	11.9	40.0	11.5	74.0
Reasoning-specialized Models															
Deepseek-R1	**94.1**	10.3	**92.0**	7.4	**85.8**	14.1	41.9	12.2	**84.5**	9.6	**70.6**	16.2	**48.5**	12.5	**73.9**
QwQ-32B	85.2	10.3	88.0	8.5	73.4	13.0	**44.9**	11.0	84.0	8.3	54.1	13.6	36.5	19.0	66.6
Open-source Foundation Models															
Kimi-K2	**94.7**	10.4	**85.0**	5.8	**86.4**	13.7	**63.5**	10.0	**87.0**	7.8	**74.7**	14.6	**37.0**	11.4	**75.5**
Deepseek-V3	91.1	11.2	82.0	9.4	82.3	15.1	56.3	10.3	63.0	9.7	67.1	16.0	36.0	20.2	68.3
Qwen2.5-72B	89.7	14.7	56.0	24.1	56.4	21.7	57.4	17.2	65.4	15.6	66.7	21.1	28.6	29.5	60.0
Llama3.1-8B	24.9	34.4	4.0	3.5	8.3	34.6	9.9	34.8	1.5	2.1	12.4	34.3	0.0	3.4	8.7
Qwen2.5-7B	40.2	24.1	38.5	25.0	15.4	31.7	22.2	26.6	13.5	24.1	16.5	30.5	1.0	27.2	21.0
Qwen2.5-3B	0.6	30.5	8.5	20.4	1.8	31.3	0.6	34.0	1.5	16.3	2.9	32.9	0.5	16.8	2.3
+ SFT	76.3	15.4	22.5	29.2	45.0	24.7	33.5	22.8	5.5	28.3	36.5	24.7	1.0	27.1	31.5

Table 1: Performance across task categories grouped by cognitive complexity (L0: Instruction Following; L1: Dimensional Reasoning; L2: Compound Tasks). Success Rate (SR) measures task completion percentage, Step Count indicates average actions for successful completion. **Overall** reports the unweighted mean SR across all seven task categories. Bold indicates best in section, underline shows overall best.

Claude-Sonnet-4.6, Claude-Opus-4.6, GLM-5 (GLM-5-Team et al., 2026), GLM-5-Turbo, and Qwen3.6-Plus (Yang et al., 2025a), representing current commercial state-of-the-art. Open-source foundation models cover a wide parameter range: Deepseek-V3 (Liu et al., 2024) at 671B parameters, Kimi-K2 (Team et al., 2026), the Qwen2.5 series (Team, 2024) at 3B, 7B, and 72B parameters, and Llama3.1-8B (Touvron et al., 2023). This selection enables analysis of how model scale and architecture affect embodied reasoning. We also include reasoning-specialized models: Deepseek-R1 (Guo et al., 2025) and QwQ-32B (Team, 2025), employing explicit chain-of-thought reasoning.

Evaluation Protocol. All models undergo identical evaluation to ensure fair comparison. We implement partial observability where agents must explore environments to discover object locations and properties, reflecting realistic deployment conditions. Each model completes 2,800 test scenarios across seven task categories with three independent runs for statistical reliability. We standardize prompts, environment descriptions, and action vocabularies across all models, with tool-dependent actions dynamically enabled based on context. This design ensures performance differences reflect reasoning capabilities rather than implementation artifacts. Detailed experimental configurations are provided in Appendix A.10.

Implementation Details. To test whether supervised learning addresses reasoning limitations, we fine-tune Qwen2.5-3B on 20,346 instruction–action pairs distilled from 1,942 optimal Qwen2.5-72B trajectories collected with complete environmental access (hyperparameters in Appendix A.7). Evaluation uses two configurations: single-agent scenarios test individual reasoning, while multi-agent scenarios employ centralized coordination where one model controls all agents with full state visibility, isolating collaborative reasoning from communication artifacts.

4.2 Main Results

Instruction following is saturated at frontier scale. At L0, all ten frontier models cluster narrowly: 85.2–96.6% on Direct Command and 85.0–93.0% on Explicit Collaboration. The same models spread sharply at L1: Tool Use ranges 73.4–89.9%, Implicit Collaboration 77.5–92.5%, and Attribute Reasoning more than twice as wide at

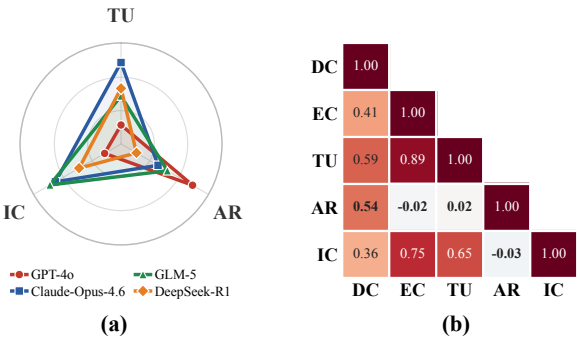


Figure 4: **(a)** Capability profiles over the three L1 primitives (TU, AR, IC). **(b)** Pairwise Pearson correlations across 12 frontier models on five categories (DC, EC, TU, AR, IC).

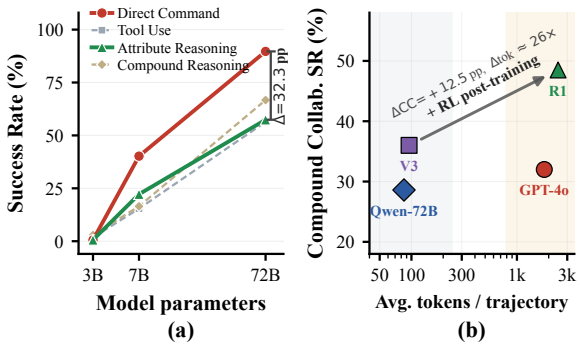


Figure 5: Scaling and reasoning specialization. **(a)** Qwen2.5 series success rate across task categories at 3B/7B/72B. **(b)** Per-trajectory token cost: DeepSeek-R1 vs. foundation models.

41.9–77.8%. Frontier-scale discrimination comes from the action-grounding categories, not from L0.

Aggregate scores hide structurally different capability profiles. Models with comparable overall performance can have inverted category-level profiles (Figure 4a). GPT-4o and DeepSeek-R1 sit within one point overall (74.7% vs. 73.9%), but GPT-4o leads on AR (77.8%, the highest in the set) while DeepSeek-R1 drops to the lowest AR score (41.9%). On IC, the direction reverses: GPT-4o reaches 77.5%, whereas DeepSeek-R1 reaches 84.5%. The 35.9-point AR gap and the reversed IC gap describe two qualitatively different models that an aggregate score would treat as nearly equivalent. Per-category breakdown is therefore not a refinement but the primary diagnostic signal that GroundAct provides.

L2 amplifies L1 gaps into composition cost. Composing grounding dimensions compounds the difficulty rather than averaging it. Compound Reasoning spans 54.1–79.4%, while Compound Collaboration drops to 32.0–53.0%, the lowest range among all categories. Strong L1 performance therefore does not guarantee robust composition: GPT-4o averages 78.4% across the three L1 categories but falls to 32.0% on Compound Collaboration. L2 is not merely a harder version of L1 along the same axis; it is a regime where multiple grounding dimensions interact. This motivates the bottleneck-level analyses in Section 4.3.

4.3 Analysis

Task categories expose different action-grounding bottlenecks. GroundAct is designed not only to rank models but to localize where action grounding fails. Pairwise Pearson correlations of success rates across 12 frontier models (Figure 4) show that EC, TU, and IC form a correlated cluster ($r = 0.65$–0.89): all three involve reasoning over action-space feasibility or coordination signals, whether a coordination directive (EC), a missing capability (TU), or a partner-dependent precondition (IC). AR is nearly uncorrelated with this cluster ($r \in [-0.03, 0.02]$), indicating that continuous-attribute grounding is a distinct bottleneck; its moderate $r = 0.54$ with DC suggests a shared instruction-execution component rather than the same feasibility reasoning. This separation explains how aggregate scores hide qualitatively different failures: GPT-4o and DeepSeek-R1 land within one point overall (74.7% vs. 73.9%), yet GPT-4o leads by 35.9 points on AR, while DeepSeek-R1 leads by 7.0 points on IC. GroundAct surfaces this structure by separating execution, attribute grounding, and action-space feasibility rather than collapsing them into a single success rate.

Scaling and reasoning training shift bottlenecks differently. Per-category results show that scaling and reasoning post-training do not uniformly improve action grounding; they shift different bottlenecks in different directions. Across the Qwen2.5 series (Figure 5a), Direct Command rises from 0.6% at 3B to 89.7% at 72B while Attribute Reasoning reaches only 57.4%: scaling raises the execution floor but leaves attribute grounding 32.3pp behind. Reasoning post-training shifts the profile in another direction: relative to DeepSeek-V3, DeepSeek-R1 improves Implicit Collaboration (63.0% → 84.5%), Compound Reasoning (67.1% → 70.6%), and Compound Collaboration (36.0% → 48.5%) but drops on Attribute Rea-

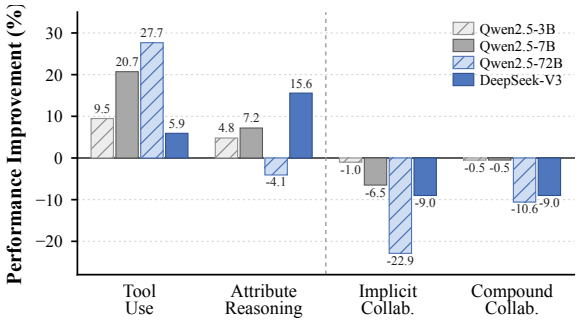


Figure 6: World Graph ablation. Complete-state observability helps Tool Use at every scale but harms Implicit Collaboration.

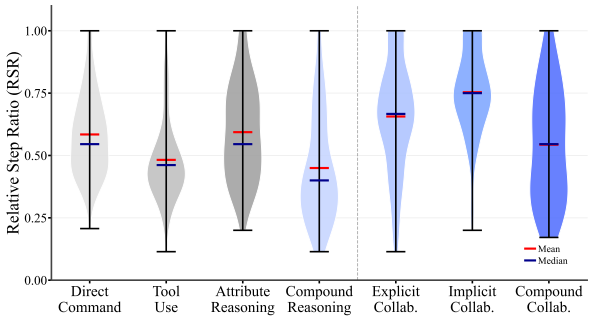


Figure 7: Relative Step Ratio (RSR) distributions across task categories. Multi-agent tasks show lower median and higher variance; Compound Collaboration is bimodal.

soning (56.3% → 41.9%). Its larger token budget (1,400–2,457 vs. 60–110 tokens per trajectory; Figure 5b) coincides with gains on collaboration and composition but not with improved continuous-attribute grounding. Aggregate scores would record only the net effect; GroundAct surfaces the profile shift because it separates execution, attribute grounding, collaboration inference, and composition rather than reporting only a single success rate.

Complete state helps search but hurts constraint filtering. We compare partial observability with a World Graph condition that exposes the complete environment graph (Figure 6). Tool Use improves at every scale, reaching +27.6 pp on Qwen2.5-72B (56.4% → 84.0%), indicating that tool reasoning is search-bound: providing object locations removes the exploration step. Implicit Collaboration degrades at every scale, with Qwen2.5-72B losing 22.9 pp (65.4% → 42.5%) and DeepSeek-V3 losing 9.0 pp (63.0% → 54.0%), indicating a filter-bound bottleneck: complete state surfaces task-irrelevant attributes that obscure whether collaboration is physically required. Attribute Reasoning shows mixed effects, suggesting that property grounding is not reducible to search or filtering alone. The same observability intervention therefore moves different bottlenecks in opposite directions, which GroundAct surfaces by evaluating each grounding dimension as a separate category.

Imitation transfers single-agent execution but not coordination triggers. Supervised fine-tuning improves single-agent performance dramatically but transfers poorly to multi-agent coordination. Qwen2.5-3B SFT lifts Direct Command from 0.6% to 76.3% and Tool Use from 1.8% to 45.0%, while Implicit Collaboration rises only from 1.5% to 5.5%. This asymmetry suggests that imitation teaches the surface action patterns required for instruction following and tool acquisition, but not the latent trigger for when collaboration is physically required.

Multi-agent compound tasks expose strategic polarization. We measure execution efficiency as relative step ratio (RSR $= L_{\text{expert}}/L_{\text{model}}$). Single-agent tasks yield consistent moderate efficiency (median RSR 0.40–0.55), while multi-agent tasks show both lower median and higher variance (Figure 7). Compound Collaboration produces a striking bimodal distribution: trajectories cluster at simple sequential execution or complex parallel coordination, with the intermediate strategy region nearly empty. Qualitative trajectories and category-level failure modes are in Appendix A.9 and A.6.

5 Conclusion

We introduced GroundAct, a benchmark for evaluating whether LLM agents can infer action feasibility from environmental state when instructions are underspecified. Evaluating 15 models across its seven categories, we find that action grounding dissociates into distinct bottlenecks, that environmental information has non-monotonic effects across these bottlenecks, and that supervised fine-tuning transfers execution but not the capability-boundary inference required for collaboration. These findings suggest that progress in agent competence requires interventions targeted at specific grounding dimensions rather than uniform scaling, and that diagnosing such progress requires benchmarks that vary grounding demands independently rather than collapsing them into a single success rate. See Appendix A.8 for further discussion and future directions.

Limitations

Our framework employs text-based environment representation rather than continuous 3D simulation. While this design enables efficient large-scale evaluation and isolates reasoning capabilities from perceptual challenges, it abstracts away aspects of physical embodiment including continuous control, sensorimotor feedback, and real-time constraints. The performance gaps we identify may manifest differently in systems that must process raw sensory inputs and execute continuous motor commands, and our findings require validation in physically grounded robotic systems.

Our evaluation covers 15 representative models across three architectural paradigms using a fixed prompt format. The rapid pace of model development means newer architectures may address some identified limitations, and performance may vary with alternative prompting strategies not explored in this work. Additionally, although GA-Bench spans 11 domains, coverage of household and outdoor scenarios is limited compared to professional environments. All task instructions are in English, and the reasoning patterns we identify may not generalize to multilingual settings.

Ethics Statements

We adhere to the ACL Code of Ethics and Code of Conduct. Human annotators involved in scenario verification provided informed consent, were compensated above local minimum wage, and their privacy was protected throughout the process. GroundAct benchmark and code will be released under the MIT License. We used Claude for grammatical refinement; all scientific content and conclusions are the authors' own work.

References

Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. 2024. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks. *Preprint*, arXiv:2411.00081.

Sadia Sultana Chowa, Riasad Alvi, Subhey Sadi Rahman, Md Abdur Rahman, Mohaimenul Azam Khan Raiaan, Md Rafiqul Islam, Mukhtar Hussain, and Sami Azam. 2025. From language to action: A review of large language models as autonomous agents and tool users. *Preprint*, arXiv:2508.17281.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games. *Preprint*, arXiv:1806.11532.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-group policy optimization for llm agent training. *Preprint*, arXiv:2505.10978.

Pascale Fung, Yoram Bachrach, Asli Celikyilmaz, Kamalika Chaudhuri, Delong Chen, Willy Chung, Emmanuel Dupoux, Hongyu Gong, Hervé Jégou, Alessandro Lazaric, Arjun Majumdar, Andrea Madotto, Franziska Meier, Florian Metze, Louis-Philippe Morency, Théo Moutakanni, Juan Pino, Basile Terver, Joseph Tighe, and 2 others. 2025. Embodied ai agents: Modeling the world. *Preprint*, arXiv:2506.22355.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, and 168 others. 2026. Glm-5: from vibe coding to agentic engineering. *Preprint*, arXiv:2602.15763.

Diandian Gu, Jing Lin, Gaohong Liu, Jiahang Liu, Su Ma, Guang Shi, Jun Wang, Qinlong Wang, Qianyi Wu, Zhongcong Xu, Xuanyu Yi, Zihao Yu, Jianfeng Zhang, Zhuolin Zheng, Yifan Zhu, Rui Chen, Hengkai Guo, Xiaoyang Guo, Mingcong Han, and 9 others. 2026. Seed3d 2.0: Advancing high-fidelity simulation-ready 3d content generation. *Preprint*, arXiv:2605.13862.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Chengshu Li, Fei Xia, Roberto Martín-Martín, Michael Lingelbach, Sanjana Srivastava, Bokui Shen, Kent Vainio, Cem Gokmen, Gokul Dharan, Tanish Jain,

and 1 others. 2021. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Wensi Ai, Benjamin Martinez, and 1 others. 2024a. Behavior-1k: A human-centered, embodied ai benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*.

Dinging Li, Yingxiu Zhao, Xinrui Cheng, Kangheng Lin, Hongbo Peng, Hongxing Li, Zixuan Wang, Yuhong Dai, Haodong Li, Jia Wang, Yukang Shi, Liang Zhao, Jianjian Sun, Zheng Ge, Xiangyu Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026. Spatialevo: Self-evolving spatial intelligence via deterministic geometric environments. *Preprint*, arXiv:2604.14144.

Dingming Li, Hongxing Li, Zixuan Wang, Yuchen Yan, Hang Zhang, Siqi Chen, Guiyang Hou, Shengpei Jiang, Wenqi Zhang, Yongliang Shen, Weiming Lu, and Yueting Zhuang. 2025. Viewspatial-bench: Evaluating multi-perspective spatial localization in vision-language models. *Preprint*, arXiv:2505.21500.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, and 1 others. 2024b. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Zhao Mandi, Shreeya Jain, and Shuran Song. 2024. Roco: Dialectic multi-robot collaboration with large language models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 286–299. IEEE.

Xavier Puig, Eric Undersander, Andrew Szot, Mikael Dallaire Cote, Tsung-Yen Yang, Ruslan Partsey, Ruta Desai, Alexander William Clegg, Michal Hlavac, So Yeon Min, and 1 others. 2023. Habitat 3.0: A co-habitat for humans, avatars and robots. *arXiv preprint arXiv:2310.13724*.

Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick. 2018. Machine theory of mind. In *International conference on machine learning*, pages 4218–4227. PMLR.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. 2024. Taskbench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, pages 4540–4574. Curran Associates, Inc.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *Preprint*, arXiv:1912.01734.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning. *Preprint*, arXiv:2010.03768.

Nan Sun, Chengming Shi, and 1 others. 2024. [aml] interactgen: Enhancing human-involved embodied task reasoning through llm-based multi-agent collaboration.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, and 181 others. 2026. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Qwen Team. 2025. Qwq-32b: Embracing the power of reinforcement learning.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Yuntao Wang, Yanghe Pan, Quan Zhao, Yi Deng, Zhou Su, Linkang Du, and Tom H Luan. 2024. Large model agents: State-of-the-art, cooperation paradigms, security and privacy, and future trends. *arXiv e-prints*, pages arXiv–2409.

Zixuan Wang, Yuchen Yan, Hongxing Li, Teng Pan, Dingming Li, Ruiqing Zhang, Weiming Lu, Jun Xiao, Yueting Zhuang, and Yongliang Shen. 2026. Milestone-guided policy learning for long-horizon language agents. *Preprint*, arXiv:2605.06078.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2025. Wizardlm: Empowering large pre-trained language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao

Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, and 1 others. 2025b. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew Hausknecht, and Adam Trischler. 2019. Counting to explore and generalize in text-based games. *Preprint*, arXiv:1806.11525.

Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. 2024. Building cooperative embodied agents modularly with large language models. *Preprint*, arXiv:2307.02485.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *Preprint*, arXiv:2403.13372.

A Appendix

A.1 Related Work

Embodied Intelligence Benchmarks The embodied intelligence evaluation landscape has established diverse benchmark frameworks spanning navigation to complex manipulation tasks (Puig et al., 2023; Li et al., 2021). ALFRED (Shridhar et al., 2020) provides foundational standards for instruction-following task evaluation, while BEHAVIOR-1K (Li et al., 2024a) extends coverage to 1,000 daily activity scenarios. These benchmarks effectively assess task execution capabilities, yet physical property modeling predominantly employs discrete state representations, such as binary door operations and object pickup/placement, with limited requirements for reasoning about continuous attributes including weight, hardness, and temperature. Our framework addresses this limitation by introducing continuous physical property reasoning tasks that require agents to compare object attributes and make decisions based on physical constraints.

Embodied Tool Use Tool usage evaluation in embodied AI exhibits stratified characteristics across different complexity levels. RoCo (Mandi et al., 2024) focuses on low-level manipulation skills such as grasping precision, while high-level benchmarks like PARTNR (Chang et al., 2024) adopt predefined tool configurations with agent action spaces fixed at task initialization. This design effectively simplifies evaluation complexity but presents limitations in assessing dynamic tool reasoning capabilities based on task requirements. Current approaches typically provide static tool sets (Chowa et al., 2025; Fung et al., 2025), preventing evaluation of how agents should reason about capability gaps and tool acquisition needs. Our framework introduces dynamic tool acquisition mechanisms, requiring agents to autonomously infer tool requirements and expand their action spaces based on task demands, thereby supplementing existing evaluation dimensions.

Multi-Agent Collaboration Multi-agent embodied intelligence evaluation has emerged as a significant research direction, with related work achieving valuable progress in collaboration modeling (Sun et al., 2024; Wang et al., 2024). PARTNR evaluates multi-agent planning capabilities through heterogeneous task design, TDW-MAT (Zhang et al., 2024) creates collaborative scenarios using load capacity constraints, and EmbodiedBench (Yang et al., 2025b) focuses on task allocation and execution optimization. Existing approaches primarily model collaboration requirements through two pathways: explicit collaboration instructions that clearly specify inter-agent task division, and efficiency optimization that drives multi-agent participation to enhance task completion speed. However, real-world collaboration decisions often stem from physical constraints rather than external instructions or efficiency considerations. Our framework employs implicit collaboration design requiring agents to autonomously assess whether tasks exceed single-agent capability ranges based on physical constraints and determine collaboration strategies accordingly, transforming collaboration judgment from external instructions to constraint-driven internal reasoning processes.

A.2 Task Category Examples

Figure 8 illustrates our hierarchical task taxonomy with representative examples, organized by agent configuration (single-agent vs. multi-agent) and cognitive complexity level (L0-L2). The figure demonstrates the progression from basic instruction following to advanced reasoning requiring dynamic capability acquisition and autonomous coordination. Table 3 provides detailed descriptions of the key reasoning requirements for each task category.

A.3 Additional Scene Cases

Figure 9 visualizes twelve representative scenes retained after curation, covering all seven task categories. These renderings are illustrative only and are not used during evaluation; the underlying scene state consumed by agents is the structured JSON description.

A.4 Data Quality Assurance and Validation

A key challenge in LLM-based benchmark generation is ensuring that generated scenarios are physically consistent, unambiguous, and genuinely solvable, as noted in prior work (Li et al., 2025; Shen et al., 2024). To address this challenge and ensure benchmark reliability at scale, we implement a rigorous three-stage validation pipeline that combines automated error correction, systematic human-in-the-loop verification, and simulation-based trajectory validation, as illustrated in Figure 3. This hybrid approach balances scalability with quality control, achieving validation standards comparable

Dataset	Scenes	Domain	Task Types	Actions	Action Space	Collab.	Auto Gen.
ALFRED	120	House	D	7	Static	—	×
PARTNR	60	House	D	11	Static	Effic.	✓
BEHAVIOR-1K	50	Diverse	D,T	6	Static	—	×
WAH	7	House	D	10	Static	Effic.	×
TDW-MAT	6	House	D,E	7	Static	Effic.	×
C-WAH	6	House	D,E	7	Static	Effic.	×
Overcooked	5	Kitchen	E,I,C	6	Static	Effic.	×
GroundAct	**1.5K**	**Diverse**	**D,A,T,R,E,I,C**	**214**	**Dynamic**	**Phys.**	✓

Table 2: Comparison of embodied AI datasets and benchmarks. Task types: D (Direct Command), A (Attribute Reasoning), T (Tool Use), R (Compound Reasoning), E (Explicit Collaboration), I (Implicit Collaboration), C (Compound Collaboration). Actions: number of available action types. Collab.: collaboration mechanism (Effic. = efficiency-based, Phys. = physical necessity-driven). Auto Gen.: automated task generation capability. Our framework uniquely combines comprehensive task coverage, dynamic action spaces, physical necessity-driven collaboration, and scalable automated generation.

to manually curated benchmarks while enabling generation at scale.

A.4.1 Stage 1: Automated Validation

The first stage addresses systematic errors that can be detected and corrected programmatically, reducing the burden on human annotators while ensuring basic structural integrity.

Error Analysis and Correction. Systematic analysis of 2,100 initially generated scenarios identified five primary error categories, summarized in Table 4. We developed specialized rule-based validators to automatically detect and correct each error type: ID mismatches are resolved via semantic similarity matching (cosine threshold 0.7), spatial violations are corrected via dimension checking and constraint propagation, tool-capability conflicts are resolved via attribute cross-referencing against our capability ontology, and state inconsistencies are handled via priority-based resolution that preserves task-critical states. This automated pipeline successfully corrected 87.6% of detected errors. Scenarios with complex logical inconsistencies requiring semantic understanding that could not be automatically corrected (14.1%) were rejected, yielding 1,804 candidates for human review.

A.4.2 Stage 2: Human-in-the-Loop Verification

While automated validation effectively handles systematic errors, subtle semantic ambiguities, non-obvious unsolvability, and nuanced physical inconsistencies require human expertise. We therefore implement rigorous human-in-the-loop verification with carefully selected and trained annotators, standardized evaluation criteria, and quantitative quality control measures.

Annotator Selection and Training. Five annotators were recruited from graduate programs in robotics and embodied AI, each with at least two years of research experience in related domains. All annotators completed a standardized 4-hour training program comprising three components: conceptual training on embodied reasoning principles and physical constraints (1.5 hours), hands-on system familiarization with environment representation and the validation interface (1 hour), and calibration exercises on 20 expert-validated scenarios with group discussion and feedback (1.5 hours). The calibration scenarios were independently validated by two senior researchers to establish reliable ground truth labels. Annotators were required to achieve Cohen's $\kappa \geq 0.75$ against these gold labels before beginning independent annotation; those falling below this threshold received additional training until reaching the required level. During the main annotation phase, each annotator reviewed approximately 720 scenarios (with systematic overlap for agreement computation), averaging 3-4 minutes per scenario to ensure careful evaluation.

Annotation Interface. To ensure consistent and reliable human evaluation, we developed a standardized annotation interface illustrated in Figure 10. The interface presents three integrated components: (1) a structured scene representation panel displaying room layouts, object properties (weight, material, dimensions), spatial relationships, and agent configurations in a clear hierarchical format; (2) a task instruction panel with automatically highlighted key elements including target objects, re-

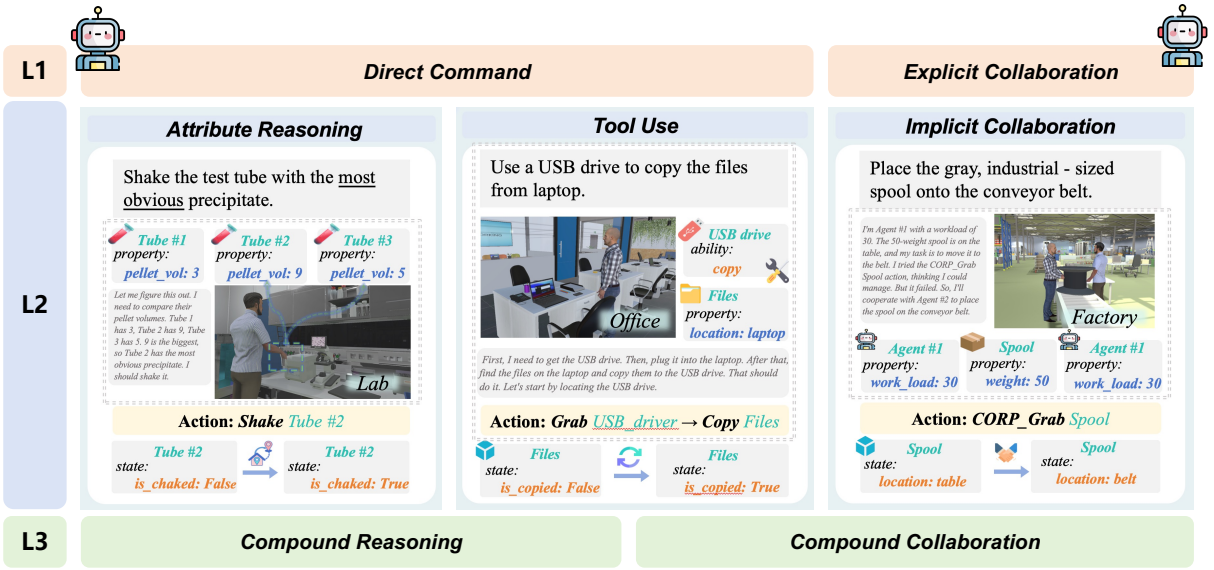


Figure 8: Representative examples for each task category in GA-Bench. Left column shows single-agent tasks progressing from Direct Command (L0) through Attribute Reasoning and Tool Use (L1) to Compound Reasoning (L2). Right column shows multi-agent tasks from Explicit Collaboration (L0) through Implicit Collaboration (L1) to Compound Collaboration (L2). Each example includes the task instruction, relevant object properties, agent reasoning process, and resulting state transitions.

quired actions, and relevant constraints; and (3) an evaluation panel presenting the three assessment criteria as binary selections along with a mandatory text field for rejection justification. This standardized interface ensures consistent evaluation conditions across all annotators and scenarios, minimizing variance arising from interface interpretation differences and enabling systematic collection of rejection rationales for error analysis.

Annotation Guidelines. Annotators evaluated each scenario against three binary criteria with explicit operational definitions designed to ensure consistent judgment:

- **Instruction Clarity:** The task instruction unambiguously specifies the goal state, target objects, and required actions without relying on implicit assumptions or domain-specific knowledge not provided in the scene description.
- **Physical Feasibility:** All actions required to complete the task respect the physical constraints encoded in the environment, including weight limits, spatial accessibility, tool requirements, and material properties.
- **Solvability:** A valid action sequence exists that transforms the initial environment state to a goal state satisfying all success criteria within the maximum step limit of 25 steps.

Each scenario received independent review from two randomly assigned annotators to eliminate individual bias. Acceptance required unanimous approval on all three criteria; any rejection required written justification specifying the failed criterion and the specific issue identified, enabling systematic failure analysis and iterative improvement of the generation pipeline.

Quality Threshold Examples. To provide concrete grounding for our annotation guidelines and demonstrate the practical application of our quality criteria, Table 5 presents representative examples of accepted and rejected scenarios across different task categories. These contrastive examples illustrate the key distinctions between scenarios meeting our quality standards and those rejected for ambiguity, physical infeasibility, or unclear collaboration requirements. The examples were selected to highlight common failure modes and the reasoning behind acceptance decisions.

Inter-Annotator Agreement. We assessed annotation reliability by measuring pairwise agreement using Cohen's κ across all 1,804 scenarios evaluated in Stage 2. The results demonstrate strong overall agreement: $\kappa = 0.84$ (95% CI: [0.82, 0.86], computed via bootstrap resampling with 10,000 iterations). Agreement varied by criterion: Instruction Clarity achieved the highest agreement ($\kappa = 0.88$), followed by Physical Fea-

Level	Task Type	Example Task	Key Reasoning Requirements
Single-Agent Tasks			
L0	Direct Command	Place the red cup on the kitchen table.	Basic object manipulation and spatial understanding through straightforward instruction following.
L1	Tool Use	Clean the dirty table in the living room.	Recognizing capability gaps, locating appropriate tools, and dynamically expanding action capabilities through tool acquisition.
L1	Attribute Reasoning	Move the heaviest box to the storage room.	Comparing continuous physical properties across multiple objects to identify correct manipulation targets.
L2	Compound Reasoning	Clean the heaviest table in the room.	Integrating attribute comparison, tool acquisition, and multi-step planning simultaneously.
Multi-Agent Tasks			
L0	Explicit Collab.	Agent A and Agent B cooperate to move the heavy dining table.	Fundamental multi-agent synchronization with explicit coordination directives provided.
L1	Implicit Collab.	Move the piano to the music room.	Autonomously recognizing when tasks exceed individual capabilities without explicit coordination instructions.
L2	Compound Collab.	Cooperatively repair the malfunctioning television.	Combining tool acquisition, capability assessment, and coordinated execution with autonomous collaboration recognition.

Table 3: Representative examples for each task category in GA-Bench. Tasks span single-agent scenarios (Direct Command, Tool Use, Attribute Reasoning, Compound Reasoning) and multi-agent scenarios (Explicit, Implicit, and Compound Collaboration) across three complexity levels.

Error Type	Frequency	Auto-Correctable
ID Mismatch	32.4%	91%
Invalid Spatial Relations	28.7%	85%
Tool-Capability Conflict	23.1%	89%
State Inconsistency	10.3%	94%
Other Logical Errors	5.5%	62%
Overall	**100%**	**87.6%**

Table 4: Distribution of LLM generation errors and automatic correction rates by category.

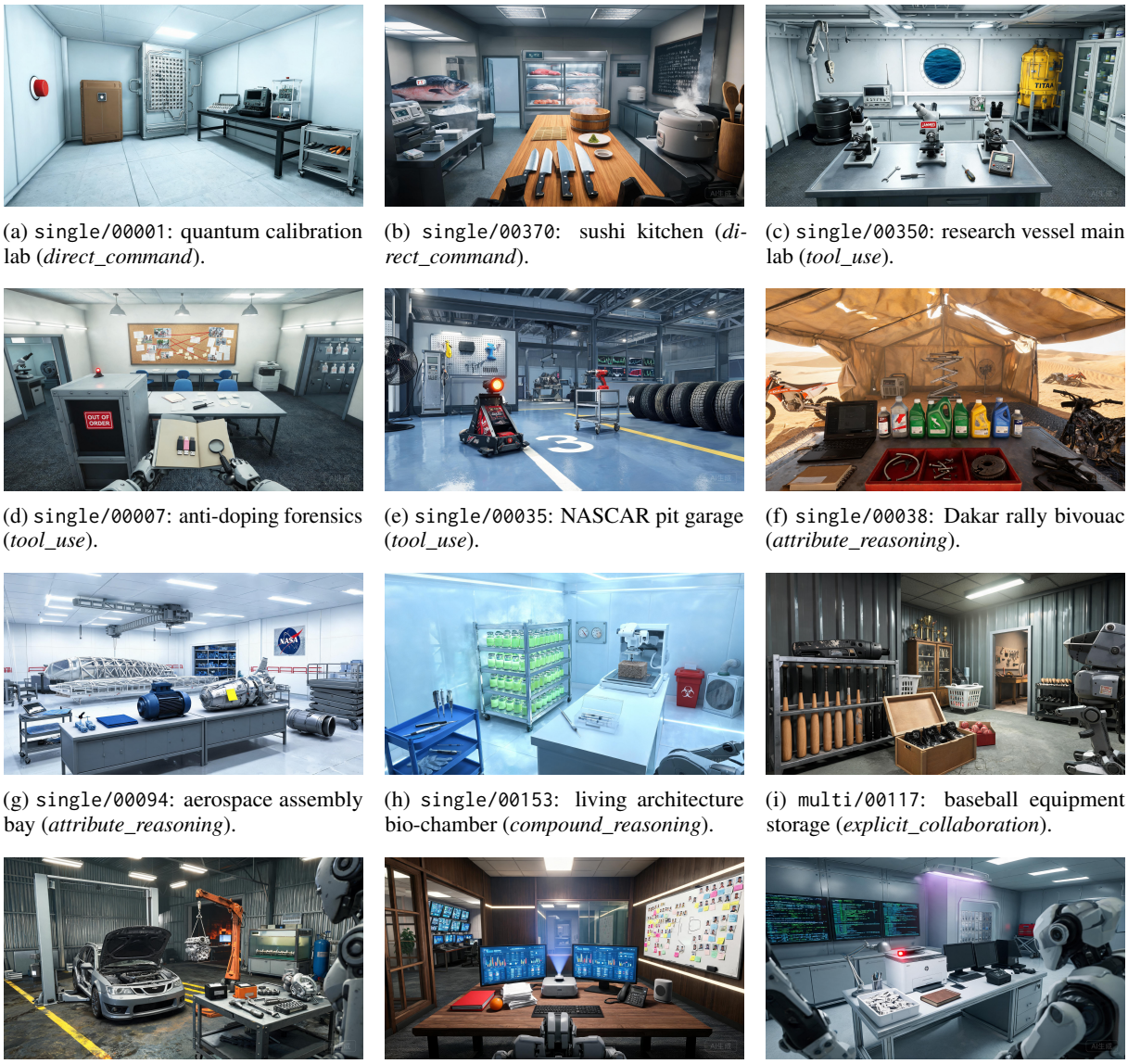
sibility ($\kappa = 0.82$) and Solvability ($\kappa = 0.79$). The relatively lower agreement on Solvability reflects the inherent cognitive difficulty of determining solution existence through mental simulation alone, which validates our design decision to include simulation-based verification in Stage 3. Disagreements between annotators (204 cases, 11.3% of scenarios) were systematically adjudicated by a senior researcher through interactive simulation-based verification, with final decisions documented for quality tracking.

Acceptance Results. Human verification accepted 1,518 of 1,804 candidate scenarios (84.1%), demonstrating that automated validation alone is insufficient for ensuring benchmark quality—15.9% of automatically validated scenarios contained subtle issues detectable only through human expertise. Analysis of rejection rationales revealed the following distribution: ambiguous or underspecified instructions (37.2%), unsolvable tasks due to missing preconditions or unreachable goals (31.6%), physical inconsistencies between object properties and task requirements (22.4%), and other issues including edge cases and annotation uncertainties (8.8%).

Acceptance rates varied systematically by task complexity, as shown in Table 6. The observed inverse relationship between task complexity and acceptance rate provides indirect validation of our hierarchical task taxonomy: more complex tasks involving multiple reasoning dimensions naturally present greater opportunities for subtle generation errors. This pattern also confirms the importance of human verification, particularly for advanced task categories where automated checks are less effective.

A.4.3 Stage 3: Expert Trajectory Validation

The final validation stage ensures that all accepted scenarios are not only well-formed but genuinely solvable through executable action sequences. Oracle agents with complete environmental knowledge generated expert demonstration trajectories for all 1,518 human-verified scenarios.



(a) `single/00001`: quantum calibration lab (*direct_command*).
(b) `single/00370`: sushi kitchen (*direct_command*).
(c) `single/00350`: research vessel main lab (*tool_use*).
(d) `single/00007`: anti-doping forensics (*tool_use*).
(e) `single/00035`: NASCAR pit garage (*tool_use*).
(f) `single/00038`: Dakar rally bivouac (*attribute_reasoning*).
(g) `single/00094`: aerospace assembly bay (*attribute_reasoning*).
(h) `single/00153`: living architecture bio-chamber (*compound_reasoning*).
(i) `multi/00117`: baseball equipment storage (*explicit_collaboration*).
(j) `multi/00138`: automotive prototype workshop (*implicit_collaboration*).
(k) `multi/00259`: NBA trade-deadline war room (*compound_collaboration*).
(l) `multi/00598`: data verification lab (*compound_collaboration*).

Figure 9: Representative scene cases retained for visualization. Renderings are produced from the scene JSON descriptions with Doubao-Seedream-5.0-lite (Gu et al., 2026) and are illustrative only.

Trajectory Generation and Verification. Each generated trajectory underwent rigorous simulation replay to verify three critical properties: (1) precondition satisfaction at each action step, ensuring no invalid state transitions; (2) goal state achievement upon trajectory completion, confirming task success; and (3) physical constraint compliance throughout execution, validating that all intermediate states respect environmental limitations. Trajectories failing any verification criterion were regenerated using alternative planning strategies, with a maximum of three attempts per scenario. Scenarios with persistent trajectory failures were removed from the final dataset (18 of 1,518, 1.2%; yielding the final 1,500 scenarios), as such failures indicate potential issues not detected in earlier stages.

Oracle Quality Validation. To verify that oracle trajectories represent high-quality reference solutions rather than merely valid but suboptimal ones, we conducted an additional validation study on a randomly sampled subset of 100 scenarios stratified across task categories. Two human experts with extensive experience in embodied AI independently solved each scenario without access to oracle solutions, and we compared their solutions with the corresponding oracle trajectories using step count as the primary efficiency metric. Oracle solutions matched or outperformed human expert solutions

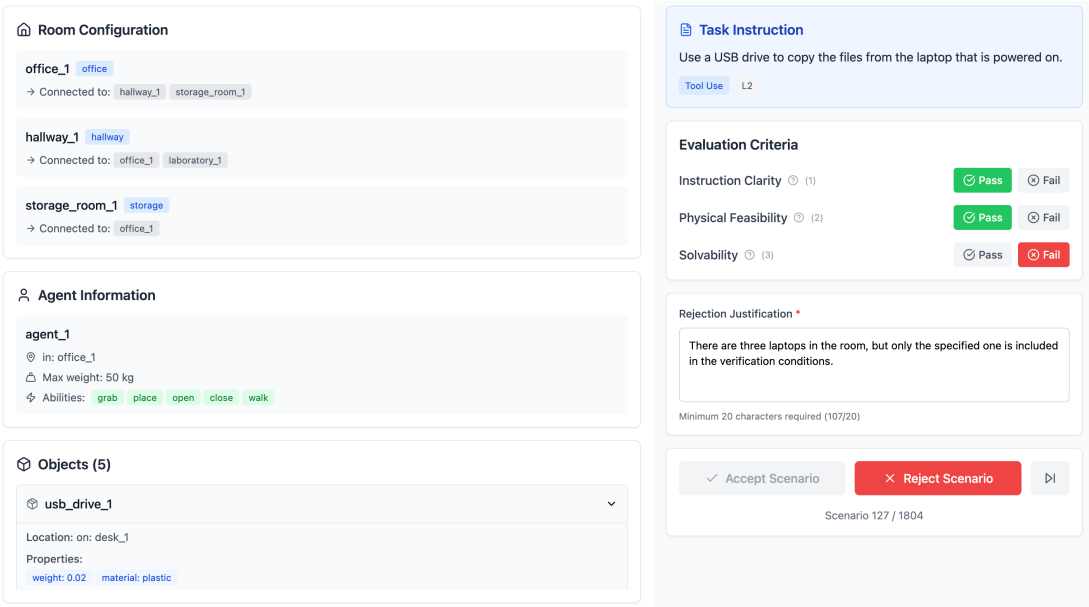


Figure 10: Human annotation interface for scenario verification. The left panel displays structured scene representation including room configuration, agent properties, and interactive objects with their attributes. The right panel shows the task instruction with highlighted key elements (top), three evaluation criteria with binary pass/fail selections (middle), and a rejection justification field requiring written rationale for any failed criterion (bottom).

in 94% of evaluated cases, with the remaining 6% representing alternative but equally valid strategies (typically involving different but equivalent tool choices or spatial paths). This validation confirms that oracle trajectories provide appropriate reference solutions for both evaluation and potential use in supervised learning approaches.

Final Dataset Statistics. The validated dataset comprises 1,500 scenarios instantiated into 16,592 task instances (multiple task variations per scenario). Expert trajectory statistics: mean length 9.1 steps (SD 4.2), median 9.0 steps, range [3, 25] steps. The high trajectory generation success rate (98.8% of human-verified scenarios, averaging 1.3 generation attempts per scenario) provides empirical validation that our human verification process effectively identifies genuinely solvable scenarios.

Ethics Statement. All human annotators provided written informed consent prior to participation and were compensated at standard institutional rates ($20 USD per hour). The annotation task involved evaluation of synthetic embodied AI scenarios and did not involve deceptive practices, collection of personal information, or exposure to sensitive or harmful content. This study protocol was reviewed and approved by our institutional review board (IRB) prior to data collection.

Summary. Our three-stage validation pipeline systematically addresses the quality challenges inherent in LLM-based benchmark generation. Automated validation handles 87.6% of systematic errors efficiently, human verification with trained annotators (Cohen's $\kappa = 0.84$) catches subtle semantic and physical issues, and simulation-based trajectory validation confirms genuine solvability. The resulting benchmark of 1,500 validated scenarios provides a reliable foundation for evaluating embodied reasoning capabilities.

A.5 Benchmark Statistics and Coverage

GA-Bench encompasses 1,500 scenarios with 64,057 interactive objects, providing comprehensive coverage across diverse domains and task complexities. Tables 7 through 10 present detailed statistics demonstrating the scale and diversity of our benchmark.

Physical Property Modeling. The benchmark features exceptional attribute diversity with 6,381 distinct property types. Core physical properties are comprehensively modeled: weight (64,047 objects), material composition (35,411 objects), size dimensions (22,820 objects), color (28,034 objects), and dynamic states (17,547 objects). This rich attribute space enables sophisticated reasoning about physical constraints and object affordances.

Task Type	Instruction	Decision	Rationale
Tool Use	Clean the dusty shelf using supplies from the cabinet which is located in the corner of the room next to the window.	✓	Clear goal specification, explicit tool source, verifiably solvable path exists, and the operation steps are clear and easy to execute.
Tool Use	Clean everything in the room including the floor, the ceiling, the furniture and all the small ornaments on the shelf.	✗	Ambiguous task scope; "everything" lacks clear termination criteria and target enumeration, which may lead to endless work and cannot be completed effectively.
Attribute Reasoning	Move the heaviest box which is marked with red label to storage room B on the second floor of the warehouse.	✓	Unambiguous quantitative comparison criterion (weight), clearly specified destination, and the marking of the box avoids confusion.
Attribute Reasoning	Move the large box to storage which is not clearly designated with a specific room number or location.	✗	"Large" is subjectively interpretable; multiple boxes may satisfy this criterion, and the storage location is not clear, leading to operational ambiguity.
Implicit Collab.	Move the 150kg industrial cabinet which is made of solid steel to warehouse located 500 meters away from the current location.	✓	Object weight (150kg) clearly exceeds single-agent capacity (50kg); collaboration requirement is inferable from physical constraints, and the material and location information are clear.
Implicit Collab.	Move the cabinet which is of unknown material and weight to warehouse without any additional information.	✗	Cabinet weight (30kg) is within single-agent capacity; collaboration necessity unclear and potentially unnecessary, and the lack of detailed information increases operational uncertainty.

Table 5: Representative examples of accepted (✓) and rejected (✗) scenarios with detailed rationale illustrating our quality threshold criteria.

Task Category	Acceptance Rate
Direct Command (L0)	91.2%
Tool Use (L1)	85.7%
Attribute Reasoning (L1)	82.3%
Compound Reasoning (L2)	78.9%
Explicit Collaboration (L0)	84.1%
Implicit Collaboration (L1)	79.6%
Compound Collaboration (L2)	76.4%

Table 6: Human verification acceptance rates by task category, showing systematic variation with task complexity.

Metric	Count
Total Scenarios	1,500
Total Task Files	1,500
Total Task Instances	16,592
Interactive Objects	64,057
Spatial Nodes (Rooms)	6,634
Average Objects per Scene	42.7
Average Rooms per Scene	4.4
Collaborative Agent Pairs	1,481

Table 7: Dataset Overview and Scale

Action Space and Tool Ecosystem. The framework supports 214 distinct action types, partitioned into basic actions (60%) available to all agents and tool-dependent actions (40%) requiring specific capabilities. Among the 64,057 objects, 15,134 are classified as tools (23.6%), with 13,482 objects possessing the `provides_abilities` attribute that enables dynamic capability extension. This design enables realistic modeling of how agents acquire new abilities through tool use.

Cross-Domain Coverage. The benchmark spans diverse application domains, with laboratory environments comprising 39.0% of scenarios, followed by office (18.8%), industrial (11.5%), and medical (6.2%) settings. This distribution reflects our emphasis on professional environments where embodied reasoning is particularly critical. Each domain presents unique challenges: laboratory settings require precise tool usage and material handling, office environments emphasize multi-agent coordination, and industrial scenarios demand reasoning about heavy equipment and safety constraints.

Quality Assurance and Expert Trajectories. All 16,592 task instances include expert demonstration trajectories averaging 8.7 steps, providing optimal solutions for comparison and learning. Each trajectory undergoes validation to ensure physical feasibility and task completion. The evaluation framework supports multi-level verification including spatial relationships (1,300 location checks), state transitions (open/closed, on/off states), and compound conditions for complex task assessment. This comprehensive validation ensures

Task Category	Count	Percentage
Single-Agent Tasks (65%)		
Direct Command	2,684	16.2%
Attribute Reasoning	2,669	16.1%
Tool Use	2,190	13.2%
Compound Reasoning	2,214	13.3%
Multi-Agent Tasks (35%)		
Explicit Collaboration	2,160	13.0%
Implicit Collaboration	2,582	15.6%
Compound Collaboration	2,093	12.6%
Total	**16,592**	**100%**

Table 8: Hierarchical Task Distribution

Category/Material	Count	Percentage
Object Categories		
Container	17,632	27.5%
Tool	15,134	23.6%
Appliance	8,963	14.0%
Furniture	6,234	9.7%
Consumable	4,890	7.6%
Others	11,204	17.6%
Material Types (Top 10 of 1,123)		
Plastic	13,767	21.5%
Metal	11,274	17.6%
Wood	8,263	12.9%
Glass	6,277	9.8%
Fabric	5,060	7.9%
Ceramic	3,843	6.0%
Silicon	1,794	2.8%
Aluminum	1,601	2.5%
Steel	1,153	1.8%
Others	11,025	17.2%

Table 9: Object Categories and Material Distribution

Domain/Room Type	Count	Percentage
Application Domains		
Laboratory	585	39.0%
Office	282	18.8%
Industrial	173	11.5%
Medical	93	6.2%
Household	93	6.2%
Educational	63	4.2%
Retail	48	3.2%
Service	30	2.0%
Entertainment	27	1.8%
Transportation	23	1.5%
Others	83	5.6%
Room Types (Top 5)		
Laboratory	1,876	28.3%
Storage	1,234	18.6%
Workspace	987	14.9%
Office	765	11.5%
Workshop	543	8.2%

Table 10: Domain and Spatial Distribution

that all tasks are both challenging and solvable, maintaining benchmark integrity while achieving unprecedented scale.

A.6 Failure Mode Analysis

Systematic failure analysis reveals task-specific performance bottlenecks that vary distinctly across model scales. Tool Use failures are dominated by exploration deficits (31.2%), where models fail to locate required tools while maintaining spatial representations. Models below 7B parameters exhibit 2.7-fold higher failure rates (84.2% vs. 31.2%), confirming critical scale thresholds for embodied reasoning. Compound Reasoning failures stem primarily from planning degradation (28.7%), with models losing track of intermediate subgoals during execution.

Implicit Collaboration shows distinct timing failures (35.8%)—models either initiate collaboration prematurely or miss coordination opportunities. This failure mode exhibits no scale correlation, indicating that collaboration timing demands reasoning mechanisms absent from current architectures. These failure patterns demonstrate that task categories stress fundamentally different cognitive capabilities, necessitating targeted architectural solutions beyond universal parameter scaling.

A.7 Hyperparameters

Supervised Fine-Tuning. The 1,942 source scenarios used for SFT distillation are drawn from a held-out partition disjoint from the 2,800 evaluation scenarios; no scenario, object instance, or task instance appears in both sets. We verify disjointness by scenario ID at preprocessing time. We performed full-parameter supervised fine-tuning on the `Qwen2.5-3B-Instruct` model using LLaMA-Factory (Zheng et al., 2024) to adapt it to our dataset. The training was conducted on 4x NVIDIA A100 GPUs. The effective batch size was 64, achieved through a per-device batch size of 1 and 16 gradient accumulation steps across 4 devices. Key hyperparameters for the SFT stage are summarized in Table 11.

Model Inference. To ensure a fair and consistent comparison, all models were evaluated using the same set of inference parameters. We utilized the vLLM engine for efficient serving, with a tensor parallel size of 4. The decoding strategy was configured to balance response quality and exploration in complex reasoning tasks. The inference settings are detailed in Table 12.

Hyperparameter	Value
Base Model	`Qwen2.5-3B-Instruct`
Fine-tuning Method	Full-parameter
Effective Batch Size	64
Learning Rate	1.0e-5
LR Scheduler	Cosine Decay
Warmup Ratio	0.1
Training Epochs	3
Max Sequence Length	15,360
Precision	BF16

Table 11: Hyperparameters for Supervised Fine-Tuning.

Hyperparameter	Value
Inference Engine	vLLM
Tensor Parallel Size	4
Decoding Strategy	Nucleus Sampling
Temperature	0.3
Top-p	1.0 (Default)
Max Generation Tokens	4096
Max Model Length	15,360

Table 12: Hyperparameters for Model Inference.

A.8 Discussion

Action Grounding as a Distinct Capability. Our results suggest that action grounding is not reducible to a single reasoning skill. The L1 categories expose different bottlenecks: Attribute Reasoning is nearly uncorrelated with Tool Use and Implicit Collaboration ($r \in [-0.03, 0.02]$; Figure 4), whereas Tool Use and Implicit Collaboration respond oppositely to the same observability intervention (Figure 6). Complete environment graphs help Tool Use, indicating a search-bound bottleneck, but hurt Implicit Collaboration, indicating a constraint-filtering bottleneck. Supervised fine-tuning further separates execution from collaboration triggering: Qwen2.5-3B improves from 0.6% to 76.3% on Direct Command but only from 1.5% to 5.5% on Implicit Collaboration (Table 1). Finally, reasoning-oriented models do not automatically solve continuous-attribute grounding: DeepSeek-R1 reaches only 41.9% on Attribute Reasoning despite a $26\times$ larger token budget per trajectory (Figure 5). These asymmetries suggest that action grounding requires bottleneck-specific mechanisms for attribute comparison, tool search, state filtering, and capacity-boundary detection, rather than longer reasoning traces or uniform scaling alone.

Implications for Model Architecture. Unlike abstract reasoning tasks where all provided information typically bears relevance, embodied scenarios require selective attention over spatially and temporally distributed constraint sets.

Future Research Directions. Several promising directions emerge from our findings. First, developing attention mechanisms specifically designed for filtering task-relevant physical constraints from environmental context could improve performance on implicit collaboration tasks. Second, exploring hybrid symbolic-neural architectures that explicitly reason about physical laws while maintaining learned flexibility represents a promising approach (Rabinowitz et al., 2018). Third, investigating training methods that better transfer single-agent reasoning capabilities to multi-agent scenarios remains an important open challenge: prior work (Wang et al., 2026), together with related efforts (Feng et al., 2025), points to possible avenues. Finally, extending evaluation frameworks to incorporate continuous control and sensorimotor feedback would provide a more complete understanding of embodied reasoning requirements, building on prior work (Li et al., 2025, 2026).

A.9 Case Studies

Two qualitative cases. Each pairs a strong model with Qwen2.5-7B on *the same* scenario; all steps are reproduced verbatim from the logged `action_sequence`. Cooperative actions issued jointly by both agents are abbreviated as `(a1,a2): CORP_X ....`

A.9.1 Case A: Implicit Collaboration

Task. "Move the `genome_sequencing_array_1` to the `cold_storage_vault`."

Object / agent	Property
`genome_sequencing_array_1` (target)	weight = 100 kg
`agent_1`, `agent_2`	max load = 9 kg each
Single-agent feasible?	**No** (100 > 9)
`CORP_GRAB` required?	**Yes**

Claude-Sonnet-4.6 — task completed in 9 steps

```
01. a1: EXPLORE  |  a2: GOTO main_lab_floor                                  [SUCCESS]
02. a1: GOTO data_hub  |  a2: EXPLORE                                         [SUCCESS]
03. (a1,a2): GOTO genome_sequencing_array_1                                   [INVALID]
   env: "Must go to Main Lab Floor first, then approach Genome Sequencing Array."
04. a1: GOTO main_lab_floor  |  a2: GOTO genome_sequencing_array_1            [SUCCESS]
05. (a1,a2): GOTO genome_sequencing_array_1                                   [SUCCESS]
06. (a1,a2): CORP_GRAB genome_sequencing_array_1                              [SUCCESS]
07. (a1,a2): CORP_GOTO cold_storage_vault                                     [SUCCESS]
08. (a1,a2): CORP_PLACE genome_sequencing_array_1 in cold_storage_vault       [SUCCESS]
09. (a1,a2): DONE  (validation: 1/1 task completed)                           [SUCCESS]
```

Qwen2.5-7B — task failed; 19 steps total, non-critical turns collapsed

```
01. (a1,a2): EXPLORE                                                          [SUCCESS]
02. a1: GRAB lubricant_1  |  a2: GOTO biometric_scanner_1                     [FAILED]
   env: "Agent must approach Lubricant before grabbing."
03–12. Ten consecutive EXPLORE / failed-GOTO turns; the target is repeatedly mis-localised.
13. a1: EXPLORE  |  a2: GOTO genome_sequencing_array_1                        [INVALID]
14. a1: GOTO genome_sequencing_array_1  |  a2: EXPLORE                        [SUCCESS]
15. a1: GOTO genome_sequencing_array_1  |  a2: GOTO cold_storage_vault        [SUCCESS]
16. a1: GRAB genome_sequencing_array_1  |  a2: GOTO cold_storage_vault        [INVALID]
   env: "Weight limit exceeded (current:0.0kg + object:100.0kg > max:9.0kg)."
17. a1: GRAB genome_sequencing_array_1  |  a2: GOTO main_lab_floor            [INVALID]
   env: Identical weight-exceeded message — ignored by the model.
18. (a1,a2): GOTO genome_sequencing_array_1                                   [SUCCESS]
19. (a1,a2): DONE  (validation: 0/1 task completed)                           [SUCCESS]
```

Diagnosis. Both agents receive an explicit constraint at step 16 ("*100 kg > max:9 kg*"), yet Qwen2.5-7B repeats the same unilateral `GRAB` at step 17 and never invokes any `CORP_*` primitive across the entire 19-step trajectory. Claude-Sonnet-4.6, given the same environment, switches to cooperative grasp on its first attempt. The single-task gap mirrors the population gap on Implicit Collaboration (91.0% *vs.* 13.5% in Table 1).

A.9.2 Case B: Attribute Reasoning

Task. "Find the heaviest, black colored item in the `gear_rack_1` and place it on the `neve_mixing_console_1`."

Object ID	Color	Weight
`tube_preamp_1`	silver	3.0 kg
`ssl_compressor_1`	**black**	**4.0 kg**
`patchbay_1`	black	2.0 kg

Ground-truth target: `ssl_compressor_1` — the only object satisfying COLOR=black ∧ arg max WEIGHT.

Claude-Opus-4.6 — task completed in 8 steps

```
01. EXPLORE                                                          [SUCCESS]
02. GOTO gear_rack_1                                                 [SUCCESS]
03. OPEN gear_rack_1                                                 [SUCCESS]
04. GRAB ssl_compressor_1   (correct: black, 4.0 kg — heaviest black item)  [SUCCESS]
05. GOTO neve_mixing_console_1                                       [SUCCESS]
06. OPEN neve_mixing_console_1                                       [SUCCESS]
07. PLACE ssl_compressor_1 on neve_mixing_console_1                  [SUCCESS]
08. DONE   (validation: 1/1 task completed — 100.0%)                 [SUCCESS]
```

Qwen2.5-7B — task failed (false-positive DONE), 9 steps

```
01. EXPLORE                                                          [SUCCESS]
02. GOTO gear_rack_1                                                 [SUCCESS]
03. OPEN gear_rack_1                                                 [SUCCESS]
04. EXPLORE   (no comparison performed across the three candidates)  [SUCCESS]
05. GRAB tube_preamp_1   (wrong: color=silver, weight=3.0 kg — violates both predicates)  [SUCCESS]
06. PLACE tube_preamp_1 on neve_mixing_console_1                     [INVALID]
    env: "Agent must approach neve_mixing_console_1 before placing object."
07. GOTO neve_mixing_console_1                                       [SUCCESS]
08. PLACE tube_preamp_1 on neve_mixing_console_1   (wrong object placed)  [SUCCESS]
09. DONE   (validation: 0/1 task completed — 0.0%)                   [SUCCESS]
```

Diagnosis. Qwen2.5-7B picks `tube_preamp_1`, which violates *both* required predicates (silver, not black; and not the heaviest), even though all three candidates and their attributes are visible in the structured observation at every turn. The model then issues `DONE` despite the environment's `0/1` verification signal — a false-positive completion coupled to the conjunctive-selection failure.

A.10 Agent Prompt Configurations

This section reproduces verbatim the system and user prompts used in our experiments, for both single-agent and centralized multi-agent configurations. **System prompts** (blue) establish the agent's operating contract and remain fixed across turns; **user prompts** (orange) are re-rendered each turn with the current environment, task, available actions, and history.

Single-Agent Configuration. This configuration tests individual agent reasoning capabilities through structured prompts.

System Prompt: Single-Agent

```
1. PRIMARY OBJECTIVE
Your goal is to successfully complete the given task by systematically exploring the
environment and interacting with objects. Success requires persistence, thorough exploration,
and precise execution of interaction sequences.
2. MANDATORY OUTPUT REQUIREMENTS
You must follow these absolute rules in every single response:
Strict Format Compliance: Your entire output must be in the exact format 'Thought:
<reasoning>\nAgent_1_Action: <command>'. Do not include any other text, explanations, or
formatting.
Command Validation: The command you choose must be exactly as listed in the Available Actions
provided in the user prompt. Do not invent or modify commands.
Progress Verification: After completing any part of the task, always re-read the task
description in your next thought to verify if additional objectives remain incomplete.
Completion Protocol: Use the DONE action if and only if you have verified that all objectives
in the task description have been successfully completed.
3. OPERATIONAL FRAMEWORK
Exploration Strategy: First use EXPLORE to thoroughly examine your current room. If the target
isn't found, systematically GOTO and EXPLORE each unexplored room until completing the task.
Interaction Sequence Protocol: Always approach an object using GOTO before attempting any
interaction with it. Always open containers using OPEN before taking items from or placing items
into them. This sequence prevents interaction failures and ensures reliable task execution.
4. CRITICAL FAILURE PATTERNS TO AVOID
Premature Task Abandonment: Do not conclude failure without exploring every available room and
container. Persistence is essential for task completion.
Object Name Confusion: Different names represent different objects. Verify exact matches
between task requirements and available objects before taking action.
Distance Interaction Violations: Do not attempt to interact with objects that are not in
immediate proximity. Always use GOTO to approach objects first.
Container Access Oversight: Do not forget to open containers before attempting to access their
contents. This is a common cause of interaction failures.
5. ERROR RECOVERY PROTOCOL
If your chosen action results in an error, acknowledge the error in your next thought and
immediately re-evaluate your strategy based on available information. Do not repeat failed
actions unless the environmental situation has changed.
6. REQUIRED OUTPUT FORMAT
Your response must contain exactly two lines in this format:
Thought: [Your reasoning for taking this action]
Agent_1_Action: [Command from the available action list]
Example Response:
Thought: I am in the main work area and need to find the target objects. I have not explored
the living room yet, so I should go there next.
Agent_1_Action: GOTO living_room_1
```

User Prompt: Single-Agent

```
You are an intelligent agent tasked with completing the given objective by strictly following
the operational framework established in your system instructions. Analyze the information
provided below and determine the single best next action that will advance progress toward task
completion.
Current Environment
{environment_description}
Task Objective
{task_description}
```

Available Actions
{available_actions_list}

Recent Action History
{history_summary}

Execution Guidelines
Respond with exactly one thought and one action. Your thought should demonstrate systematic reasoning that considers the current situation, task requirements, and appropriate next steps. Your action must be selected from the available actions list and should represent the most logical progression toward completing the task objective.
Remember that systematic exploration, proper interaction sequences, and persistent problem-solving are essential for successful task completion. The available action descriptions will guide you on exactly how to execute each command effectively.

Multi-Agent Configuration. This configuration provides prompts for coordinated reasoning between two agents under a centralized controller.

System Prompt: Centralized Multi-Agent

You are a central coordination controller managing two intelligent agents working collaboratively to complete complex tasks. Your responsibility is to analyze the current situation, decompose objectives into executable subtasks, and assign optimal actions to both agents while ensuring efficient coordination and conflict avoidance.

Core Coordination Principles
Strategic Assignment Protocol: Assign actions based on each agent's current position, capabilities, and the optimal path toward task completion. Prioritize complementary actions that maximize overall efficiency.
Conflict Prevention Framework: Ensure that assigned actions do not create spatial conflicts, resource competition, or contradictory objectives between the two agents.
Exploration Optimization: When agents have completed their immediate objectives, prioritize exploration of unknown areas to gather additional environmental information and identify new opportunities for task advancement.

Cooperation Command Protocol
For collaborative tasks requiring joint action, implement the following cooperation strategy:
Pre-Cooperation Positioning: Before initiating any CORP_ command sequence, ensure that both participating agents have successfully executed GOTO commands to reach the target object or designated cooperation zone.
Cooperative Transport Sequence: For tasks involving collaborative object movement, execute the following mandatory sequence without interruption:
1. CORP_GRAB: Both agents grab/pick up the target object
2. CORP_GOTO: Coordinated movement to the destination location
3. CORP_PLACE: Synchronized placement of the object at the target location
Critical CORP_PLACE Requirement: After executing CORP_GOTO, you MUST execute CORP_PLACE to actually place the object at the destination. The object is not considered "moved" until CORP_PLACE is completed.
Sequence Integrity Requirement: The cooperative transport sequence must be executed continuously without interspersing other commands. Any interruption requires restarting the entire cooperation sequence. NEVER output DONE after CORP_GOTO — always complete with CORP_PLACE first.
Cooperation Readiness Verification: Verify that both agents are properly positioned and available for cooperation before initiating any CORP_ command. This prevents coordination failures and ensures successful collaborative execution.

Task Completion Management
Individual Agent Completion: When an agent has no additional meaningful tasks to perform, assign the DONE command to that specific agent while continuing to provide actionable commands to the other agent.
Final Task Termination: The overall task concludes only when both agents simultaneously receive DONE commands, indicating that all objectives have been completed and no further actions are required.
Continuation Protocol: When one agent completes all its tasks, consistently assign DONE to that agent in all subsequent action assignments while continuing to provide meaningful actions to the remaining active agent until it also completes its objectives.

Mandatory Output Format
Your response must adhere to the following strict format without any additional content or explanations:
Thought: [Comprehensive analysis of current situation, task requirements, and strategic reasoning for action assignments]

```
Agent_1_Action: [Specific command for agent_1 from available action set]
Agent_2_Action: [Specific command for agent_2 from available action set]
Example:
Thought: Agent 1 is in the main work area and needs to explore, while agent 2 should go to the
living room to find target items.
Agent_1_Action: EXPLORE
Agent_2_Action: GOTO living_room_1

Strategic Planning Guidelines
Situational Assessment: Evaluate each agent's current location, recent actions, and immediate
objectives to determine the most effective next steps.
Resource Allocation: Consider the spatial distribution of tasks and assign agents to different
areas when possible to maximize coverage and minimize redundancy.
Progress Monitoring: Track completion status of subtasks and adjust assignments based on
evolving priorities and environmental discoveries.
Efficiency Optimization: Balance individual agent productivity with collaborative opportunities
to achieve optimal overall task completion time.
```

User Prompt: Centralized Multi-Agent

```
Analyze the provided information and generate coordinated action assignments for both agents:

Current Environment State
{environment_description}

Task Objectives
{task_description}

Available Commands
{available_actions_list}

Agent Status and History
{history_summary}

Coordination Requirements
Generate action assignments that advance task completion while maintaining coordination
efficiency. Ensure that cooperative tasks follow the established CORP_ command protocols and
that individual assignments complement overall strategic objectives.
```

Data Generation Pipeline. GA-Bench scenarios are synthesized by a three-stage LLM pipeline (DEEPSEEK-CHAT backbone for all stages): *Raw text* → **Stage 1: Clue Generation** (free-form text → natural-language scene blueprint) → **Stage 2: Scene Generation** (blueprint → structured JSON scene with rooms, objects, properties, states, and abilities) → **Stage 3: Task Generation** (scene + ability schema → 14 tasks across the 7 categories with embedded validation criteria). All prompts used at each stage are reproduced below verbatim.

Stage 1: Clue Generation. Takes a free-form "Inspiration Text" and produces a hyper-detailed natural-language description of a multi-agent scene, emphasising density, purpose, and atmosphere.

System Prompt: Clue Generation (Stage 1)

```
You are a highly experienced Scene Concept Designer specializing in crafting imaginative yet
realistic multi-agent collaborative environments. Your core mission is to transform abstract
textual inspirations into meticulously detailed scene descriptions that serve as robust
blueprints for creating embodied AI simulations. Your designs must be grounded in real-world
logic, inherently support diverse forms of collaboration, and be rich in detail to enable
complex task generation.
```

User Prompt: Clue Generation (Stage 1)

```
ROLE & MISSION
You are a creative Scene Concept Designer. Your objective is to leverage a seemingly unrelated
"Inspiration Text" to construct a vivid, densely populated, and collaboration-rich scene
concept. Your output must be a comprehensive description in pure, natural language.
Crucially, your focus is to build a world teeming with potential. Think of yourself as dressing
a stage for a play where every prop, down to the dust on a shelf, matters. Your job is to build
this rich, versatile "stage," not to write the specific "script" (the tasks) that will play out
on it.

INSTRUCTIONS FOR USING THE "INSPIRATION TEXT"
```

The “Inspiration Text” is a catalyst for creativity. You must:
1. **Extract Abstract Elements**: Identify core abstract elements from the text (e.g., Event Nature, Atmosphere, Domain).
2. **Associate and Reconstruct**: Use these elements to spark a new concept anchored to a common scene archetype (e.g., Home, Lab, Factory), adding a creative twist.
3. **Ensure Plausibility**: Ground your design in a degree of real-world logic.

CORE DESIGN PHILOSOPHY: DENSITY, PURPOSE, AND ATMOSPHERE

This is the most important principle. You must design every part of the scene with high **DENSITY**, clear **PURPOSE**, and rich **ATMOSPHERE**.

1. Principle of No Generic Objects. Abstract descriptions are forbidden. An object must always be specific.
- Don’t say: “a box”
- Do say: “a sealed cardboard box, dimensions 40x40x60cm, weight 18kg, with a red ‘FRAGILE’ sticker and a shipping label addressed to ‘Biolab Zeta’.”

2. Principle: Properties Must Serve Potential Tasks. Every property is a hook for a potential task. Think in these layers:
- **For Collaboration**: Use **Physical Properties** (weight, size, shape). Make some objects too heavy or bulky for one agent.
- **For Reasoning**: Use rich **Descriptive Properties** (color, material, unique markings, serial numbers). Ensure combinations can uniquely identify an object.
- **For Dynamic Tasks**: Use a wide array of **Stateful Properties** (on/off, open/closed, hot/cold, clean/dirty, broken/functional). These are the “problems” to solve.
- **For Tool Use**: Embed **Functional Properties** in objects. A key can unlock; a cloth can clean. These are the “solutions” to find.

3. Principle of Purposeful Ambience & Background “Noise”. The world must feel lived-in. You must add a deep layer of **“non-critical” ambient objects**. Their purpose is not to be a direct task objective, but to serve two higher functions: (1) to build atmosphere and make the world feel authentic, and (2) to act as perceptual distractors (or “noise”), forcing an agent to be more precise in its reasoning.
Examples: A faded safety poster on a wall, a chipped coffee mug used as a pen holder, a stack of old, irrelevant industry magazines, a decorative but non-functional clock, a dusty trophy on a high shelf.

4. Inspiration Checklist for Extreme Richness. To achieve high density, constantly add items from these categories to each room:
- **Tools & Utensils**: Wrenches, scalpels, pipettes, keyboards.
- **Containers**: Boxes, bottles, crates, bins, drawers (Specify state: empty, full, sealed, labeled).
- **Equipment**: Microscopes, monitors, printers, medical scanners (Specify state: on, off, error, idle).
- **Supplies & Materials**: Paper, reagents, ingredients, parts, wires.
- **Documents & Information**: Files, books, manuals, notes, blueprints, whiteboards with writings.
- **Personal & Mundane Items**: Backpacks, keys, ID cards, clothing, photos.
- **Environmental Details & Sensory Info**: Waste bins (with contents), spills, dust, posters, clocks (showing a time). Also consider **sounds** (humming equipment, dripping water), **lights** (flickering LEDs, screen glows), and even implied **smells** (ozone from machinery, antiseptic smell).

NATURAL LANGUAGE SCENE DESCRIPTION: REQUIRED ELEMENTS

1. Overall Scene Theme and Core Concept. Define the scene type and its unique story/challenge. Explain why the environment is inherently suited for multi-agent scenarios.

2. Spatial Layout and Area Descriptions. Describe the overall layout and detail 3-6 distinct, interconnected areas. Describe their purpose, feel, and key architectural features.

3. Detailed Area-by-Area Inventory. For each room/area you defined, provide a detailed inventory under the following four sub-headings:

a. Anchor Furniture & Installations: large, scene-defining “anchor” objects (desks, beds, heavy machinery, server racks, shelving units). These form the skeleton of the room.

b. Key Interactive & Task-Relevant Objects: the “hero” props — critical tools, task objectives, and puzzle pieces (e.g., the specific screwdriver needed, the “infected” sample, the security keycard, the machine with a “broken” state). Explain their strategic placement.

c. Functional Ambient Objects: other objects in the room that are functional but not critical to the main tasks (e.g., a working desk lamp, a telephone, a printer with paper, a microwave, other chairs and tables). Give them specific states.

d. Background & Decorative Objects: objects that build atmosphere and act as realistic clutter (posters on the wall, potted plants, framed pictures, books on a shelf with titles, old trophies, loose papers, discarded packaging, dust on surfaces).

4. Scene Affordances and Embedded Potential. Analyze your own design. Explain HOW the dense and purposeful scene you have constructed is primed for complex tasks.
- **Collaborative Transportation Affordances**: Identify 1-2 objects you designed. Describe the specific properties (e.g., weight: 150kg; dimensions: 3m long) that make multi-agent

collaboration the only viable solution for its manipulation.
- **Reasoning and Tool-Use Affordances**: Explain how the scene's design enables complex reasoning.
  - **Attribute-based Reasoning**: Point out a group of similar objects (e.g., 'the five chemical bottles on the shelf') and describe the unique combination of attributes you gave one of them. Mention how the presence of other "non-critical" background objects (like decorative blue glass) makes this identification task more challenging and realistic.
  - **Compound (Tool-Use) Reasoning**: Describe a "problem-solution" pairing you integrated. Identify a "problem state" (e.g., 'a locked safe') and the corresponding "tool" (e.g., 'a keycard, located inside a desk drawer in another room'). Explain how this creates potential for multi-step problem-solving.

INSTRUCTION
Now, carefully read the provided "Inspiration Text." Following all the principles and the detailed structure above, conceptualize and describe a new, hyper-detailed, and densely populated multi-agent collaborative scene. Your entire focus should be on designing this rich environment, detailing the affordances and potential for complex, collaborative tasks embedded within your design.

Inspiration Text: {raw_text}

Stage 2: Scene Generation. Takes the natural-language clue together with the agent ability schema and emits a single, syntactically-valid JSON scene with rooms, objects (FURNITURE and ITEM), reserved/free-form properties, and actionable states. A debug-correction prompt is appended on validator failure to surgically repair the JSON without restarting the generation.

System Prompt: Scene Generation (Stage 2)

ROLE & MISSION
You are a senior-level Scene Architect for advanced Embodied AI Evaluation Environments. Your mission is to generate a hyper-realistic, high-density, and deeply interactive multi-agent scene based on a conceptual clue and a list of agent abilities.
Your output must be a single, syntactically perfect JSON object. It must be generated with extreme precision, adhering to every rule and principle outlined below. You are not just building a scene; you are engineering a rigorous testbed for AI agents.

CORE DESIGN MANDATE: ENGINEER THE SCENE FOR THE TASK HIERARCHY
This is your most critical directive. The entire scene must be purposefully designed to facilitate a specific hierarchy of tasks, from simple to complex.
1. Principle of Hyper-Density and Richness. Your primary goal is to create an extremely rich scene. Do not hesitate; add as many items and furniture as possible. A typical room should contain dozens of items. The total number of ITEM objects must vastly exceed FURNITURE, which must exceed rooms. Fill every corner, surface, and container with meaningful detail.
2. Designing for "Attribute-based Reasoning". You MUST create multiple sets of similar objects that are only distinguishable by their properties (weight, color, material, state, etc.). Example: design three lamps with different weights; design five mugs of different colors.
3. Designing for "Dynamic Capability Acquisition" (Tool Use). You MUST design "problem-solution" pairs. A "problem" is an object in a specific state (e.g., is_dirty: true). A "solution" is a tool ITEM with a corresponding "provides_abilities": ["action_name"] property. Example: if a clean ability exists, create a dirty object and a rag with "provides_abilities": ["clean"].
4. Designing for "Compound Reasoning". You MUST combine the principles above. Create multi-layered problems that require both reasoning and tool use. Example: among several tables, only one is the heaviest, and that specific table also happens to be "is_dirty": true.
5. Designing for "Implicit Collaboration Inference". !!! CRITICAL DIRECTIVE: you MUST design several FURNITURE objects whose physical properties, especially weight, are explicitly beyond a single agent's capacity. Set the weight for these specific objects in a higher range, such as **50.0 to 100.0 kg**. This is non-negotiable and serves to create scenarios where collaboration is not just an option, but a necessity.
6. Designing for "Compound Collaborative Tasks". You MUST create ultimate challenges by fusing all principles. Example: design a machine that is both very heavy (weight: 90.0, requiring collaboration) and also in a state of "is_broken": true (requiring a specific "repair" tool).
7. Principle of High Interactive Density. The scene must be saturated with interactive opportunities. For every ability in the Agent Abilities List, think in terms of "many-to-many" relationships: create multiple "problem" states on different objects, and create multiple, possibly varied, "tool" items that provide the same ability, enabling parallel or cooperative work.

JSON STRUCTURE & SPECIFICATIONS
A. Top-Level Object. The root must be a JSON object containing exactly four keys: description, rooms, objects, abilities.

B. rooms Array.
- id: string (unique, lowercase_snake_case).
- name: string (English name, e.g., “Living Room”, “Workshop”).
- properties: object, must contain a type field (e.g., "type": "kitchen").
- connected_to_room_ids: array of valid room id strings.

C. objects Array.
- id: string (unique, lowercase_snake_case with numeric suffix, e.g., sofa_1).
- name: string (English name, e.g., “Red Mug”, “Office Desk”).
- type: string (must be FURNITURE or ITEM).
- location_id: string defining placement: in a room ("in:room_id"); on an object’s surface ("on:parent_object_id" — parent must be FURNITURE with is_container: true); inside an object ("in:parent_object_id" — parent must be FURNITURE with is_container: true and an is_open state).
- properties: object containing reserved (functional) and free-form (descriptive) keys.
  - **Reserved**: weight: float (**required for all objects**; ITEM $\in [0.1, 10.0]$, standard FURNITURE $\in [10.1, 50.0]$, collaborative FURNITURE $\in [50.1, 100.0]$); size: [l, w, h] (required for FURNITURE); is_container: true (for objects that can hold others); provides_abilities: array (for tool ITEMs; values must be action_names from the input Agent Abilities List).
  - **Free-form**: beyond reserved keys, you are highly encouraged to add custom descriptive properties such as color, material, brand, model_number, serial_number, condition, finish, inscription, led_color, handle_material, current_temperature, etc. Every ITEM must have at least two such properties.
- states: object (optional). **!!! CRITICAL RULE**: the contents are **strictly dictated** by the input Agent Abilities List. A key (e.g., "is_dirty") can ONLY be used if it is an attribute name explicitly defined in the abilities list. The value must be the exact precondition value defined for the corresponding action. Example: if an ability is defined as repair: is_broken=true, you may only create "is_broken": true (and not false); every state must represent an actionable “problem”.

D. abilities Array. Populated automatically after scene generation. Do NOT include it manually.

CRITICAL IMPLEMENTATION MANDATE: ENGINEERING A FULLY INTERACTIVE WORLD

Step 1: Analyze the Abilities List. For each action, identify its action_name, target precondition (attribute, value), and tool requirement (requires_tool).

Step 2: Implement Prolific Interactive Loops. Iterate through every ability and create multiple, complete cause-and-effect loops. A one-to-one mapping is insufficient.
- **Tool-Independent Abilities** (requires_tool: false): create multiple distinct objects in the state required by the precondition. Example: for turn_on, create several devices (desk_lamp_1, computer_monitor_1, radio_1) all starting in "is_on": false.
- **Tool-Dependent Abilities** (requires_tool: true): create both (1) one or more tool ITEMs whose properties contain provides_abilities: ["<action>"] (varied tools are encouraged: e.g., both a wrench and a welder can provide repair); and (2) multiple target objects in the required precondition state, e.g., "states": {"is_broken": true}.

Step 3: Final Coverage Audit. Before concluding, mentally check that every action_name from the abilities list has at least one, preferably multiple, corresponding interactive loops in the scene. The generation is not valid until every ability is accounted for.

ILLUSTRATIVE EXAMPLE (minimal JSON)

```
{
  "description": "A workshop with a CNC machine and a toolbox",
  "rooms": [{
    "id": "workshop", "name": "Workshop",
    "properties": {"type": "industrial"},
    "connected_to_room_ids": []
  }],
  "objects": [
    {
      "id": "cnc_machine_1", "name": "CNC Machine",
      "type": "FURNITURE", "location_id": "in:workshop",
      "properties": {"size": [2.2, 1.8, 2.0], "weight": 98.5,
        "is_container": false, "material": "steel",
        "model_number": "TX-5000"},
      "states": {"is_on": false, "is_broken": true}
    },
    {
      "id": "workbench_1", "name": "Workbench",
      "type": "FURNITURE", "location_id": "in:workshop",
      "properties": {"size": [2.5, 0.8, 0.9], "weight": 45.0,
        "is_container": true, "material": "wood"}
    },
    {
      "id": "professional_toolbox_1", "name": "Professional Toolbox",
      "type": "ITEM", "location_id": "on:workbench_1",
      "properties": {"weight": 9.2, "color": "red",
        "brand": "Craftsman", "provides_abilities": ["repair"]}
    }
  ]
}
```

User Prompt: Scene Generation (Stage 2)

INSTRUCTION
Strictly follow all rules, principles, and specifications defined in the System Prompt. Based on the provided "Conceptual Clue" and "Agent Abilities List" below, generate a single, complete, and valid JSON object.
You must generate a scene with a massive number of items and furniture, demonstrating extreme richness and detail.
Your entire output must be only the raw JSON code. Do not include any explanations, comments, or markdown formatting.

INPUTS
- **Conceptual Clue**: {clue}
- **Agent Abilities List**: {abilities}

Debug Correction Prompt: Scene Generation (Stage 2)

JSON ERROR CORRECTION DIRECTIVE
Mission: Fix all errors in the previously generated JSON and output the complete, corrected version.
Error Analysis: {error_report}
Correction Requirements:
1. **Fix Every Listed Error**: Address each specific issue mentioned in the error report above.
2. **Maintain All Rules**: Ensure corrections follow the original system prompt specifications.
3. **Complete Coverage**: Include all original content plus any necessary additions/corrections.
4. **No New Errors**: Verify your changes don't introduce additional problems.

CRITICAL OUTPUT REQUIREMENT: Output ONLY the complete, valid JSON object. Do NOT include explanations, comments, markdown formatting, or any other text. Do NOT output partial JSON or summaries. The entire response must be valid JSON that can be directly parsed.

Stage 3: Task Generation. Takes the structured JSON scene plus the agent ability schema and emits a **self-validating task suite**: exactly two tasks per category for all seven categories (14 tasks total), each bundled with minimal-change `validation_checks` that specify the success condition. A global `agents_config` (single `max_weight` for both robots) is chosen *after* the tasks are drafted so that single-agent tasks remain feasible while implicit-collaboration tasks become a logical necessity.

System Prompt: Task Generation (Stage 3)

ROLE & MISSION
You are a top-tier AI Evaluation Suite Architect. Your core mission is to generate a comprehensive and self-contained Task Evaluation Suite based on a given, highly detailed scene description (in JSON format) and a detailed list of agent abilities and their preconditions. This is a two-in-one process: for **every task** you design, you must **also** generate the precise validation criteria that defines its successful completion. You are not designing a single, narrative mission. Instead, you are building a rigorous set of isolated "unit tests" where each task is bundled with its own success conditions, all based on strict causal deduction.

CORE WORKFLOW
Step 1: Holistic Scene & Ability Analysis. First, thoroughly analyze the entire input Scene JSON Data to understand the environment, objects, and their initial states. Concurrently, analyze the Agent Action Schema to understand every possible action, its preconditions, and its tool requirements.
Step 2: Iterative Task Generation & Validation Loop. Iterate and generate **two distinct tasks for each of the 7 categories** (14 tasks in total). For each task:
A. Select a task category (e.g., implicit_collaboration).
B. **Pre-condition Validation**: scan the scene for a situation that allows a task of this category to exist. Example: for tool_use, find a problem-solution pair; for implicit_collaboration, find an object heavier than a single agent's capacity. **Golden Rule**: if preconditions don't exist, do not generate a task for that type — every task must be immediately solvable.
C. Formulate the task_description following the category-specific guidelines below.
D. **Deduce the Final State for Validation**: determine the minimal, necessary change upon completion — for movement/placement, a change in location_id; for state changes, a single key-value pair (e.g., "is_broken": false).
E. Assemble the Structured Task Object combining (C) and (D).
Step 3: Determine Global agents_config. After designing all 14 tasks, holistically review them (especially the most demanding implicit_collaboration task) and define a single global config powerful enough for all single-agent tasks but constrained enough to make collaboration logically necessary for heavy items.
Step 4: Assemble the Final JSON Object combining task_background, agents_config, and the list of 14 task objects.

TASK GENERATION GUIDE PER CATEGORY
A. Single-Agent Tasks. The task_description must be designed to be completed by a single agent (may explicitly name robot_1 or use general phrasing).
1. direct_command (Direct Command Following). Phrasing **must use the object's id** from the scene. Example: "Place cup_1 on table_1."
2. attribute_reasoning (Attribute-based Reasoning). Phrasing **must use a combination of properties** to describe the target, and the description must be unique within the current scene. Example: "Find the heaviest, red mug and place it in the sink."
3. tool_use (Dynamic Capability Acquisition). Phrasing must describe a goal that requires a tool. Scene must contain a problem-solution pair. Example: "Repair the broken coffee machine."
4. compound_reasoning (Compound Reasoning). Phrasing must describe a goal that requires **both attribute reasoning AND tool use**. Example: "Use the repair kit to fix the largest, wooden table's only flashing red device."
B. Multi-Agent Collaborative Tasks. The task_description must require two agents (robot_1 and robot_2) working together.
5. explicit_collaboration (Explicit Collaborative Task). Phrasing must **explicitly ask multiple agents to cooperate** on a simple, non-transport action. Example: "Have robot_1 and robot_2 cooperate to start the main generator."
6. implicit_collaboration (Implicit Collaboration Inference). Design a **capability-demand mismatch**: determine a reasonable max_weight for a single agent, then select an object whose weight is greater than that max_weight, and phrase the task to move it **without ever using the word "cooperate."** The agents must infer the need for collaboration by comparing the object's properties to their own capabilities. Example (max_weight=25.0, heavy_box_1.weight=40.0): "Move the heavy box of spare parts to the workshop."
7. compound_collaboration (Compound Collaborative Task). Phrasing must describe a complex goal requiring **collaboration (due to weight/size) AND either reasoning or tool use**. Example: "Cooperatively move the heavy printer in the corner that is uniquely flashing a red light to the repair bay."

JSON OUTPUT SPECIFICATIONS
A. Top-Level Object contains exactly three keys: task_background, agents_config, tasks.
B. task_background: string. A single, concise sentence providing high-level narrative context for the entire suite (e.g., "Prepare the laboratory for an upcoming experiment.").
C. agents_config: array of two objects (robot_1, robot_2). **Critical mandate**: determine the parameters after mentally designing all tasks; the config must be a single "master solution"

that is powerful enough for single-agent tasks yet constrained enough to make the most demanding collaborative task a logical necessity. Each config has name, max_grasp_limit, max_weight, max_size.
D. tasks: array (a flat list of all Structured Task Objects). Must contain **two tasks per category** across the 7 categories.
E. Structured Task Object (object): must contain exactly three keys: task_description (string, the human-readable goal), task_category (one of direct_command, attribute_reasoning, tool_use, compound_reasoning, explicit_collaboration, implicit_collaboration, compound_collaboration), and validation_checks (array of one or more "minimal change snippets").
F. "Minimal Change Snippet": an object inside validation_checks. Must contain the id of the changed object, plus **only the key-value pair that changed**, using the original key from the scene JSON (e.g., "location_id": "on:desk_1" or "is_broken": false).

User Prompt: Task Generation (Stage 3)

INSTRUCTION
Based on the provided Scene JSON and Agent Action Schema below, and in strict adherence to all rules, principles, and workflows in the System Prompt, generate a **complete and self-validating Task Evaluation Suite**.

Core Requirements:
- Generate a flat list of tasks, with **exactly two** distinct and logically sound examples for **each of the 7 task categories**.
- For each task, generate both its task_description and its precise validation_checks.
- Your design must be entirely based on the provided inputs, ensuring all tasks are **100% solvable**.

CRITICAL OUTPUT REQUIREMENTS: Your entire output must be ONLY a valid JSON object. Do not include any explanations, reasoning, comments, or markdown formatting. Start your response with an opening brace and end with a closing brace. No other text is allowed.

INPUTS
- **Scene JSON Data**: {scene_json}
- **Agent Action Schema**: {action_schema}

A.11 Pipeline Walkthrough

To complement the pipeline prompts in Appendix A.10, we expose the four sequential artefacts produced by the pipeline for a single scenario, id `01242` (*Oxford University Data Recovery Lab*). All four files share the same numeric id, so each stage's output can be cross-referenced against the previous stage's input verbatim. JSON content is shown as emitted by the pipeline (line-wrapped only for column width); non-ASCII characters present in the raw inspiration text have been transliterated to ? for typesetting safety.

Stage 1: Clue (`data/clue/01242_clue.json`). The Scene Concept Designer (Stage 1 prompt, Appendix A.10) ingests a free-form "Inspiration Text" (the `raw` field; here a fragment of a Java JSON-parsing question retrieved from `stackoverflow.com`, unrelated to the final scene) and emits a hyper-detailed natural-language scene blueprint (the `response` field). The blueprint preserves no surface content of the inspiration text; instead, abstract elements (a failing system, technical jargon, urgency) are re-mixed into a new multi-agent setting.

```
{
  "id": 1242,
  "raw": """
Retrieved from http://stackoverflow.com/questions/8091051/how-to-parse-json-string-in-android
Text:
Take the 2-minute tour ?

Possible Duplicate:
JSON Array iteration in Android/Java

I am fetching JSON string from server and I have already got JSON string by code. But I didn't understand how to parse it.

Below is my JSON string

    "university": {
        "name": "oxford",
        "url": "http://www.youtube.com"
    "1": {
        "id": "2",
        "title": "Baseball",
        "datetime": "2011-11-11 10:41:46"
    "2": {
        "id": "1",
        "title": "Two basketball team players earn all state honors",
        "datetime": "2011-11-11 10:40:57"

Please provide any guidance or code snippet.

share|improve this question

marked as duplicate by casperOne Sep 11 '12 at 11:26


2 Answers 2

up vote 34 down vote accepted

Use JSON classes for parsing e.g

JSONObject mainObject = new JSONObject(Your_Sring_data);
JSONObject uniObject = mainObject.getJSONObject("university");
String  uniName = uniObject.getJSONObject("name");
String uniURL = uniObject.getJSONObject("url");

JSONObject oneObject = mainObject.getJSONObject("1");
String id = oneObject.getJSONObject("id");
share|improve this answer
can we get how many objects in json string ?? helloDroid Nov 11 '11 at 8:11
Yes you can simply use "mainObject.length();" ?? Arslan Nov 11 '11 at 8:12
thanks.. it's working fine ?? helloDroid Nov 11 '11 at 8:22

Below is the link which guide in parsing JSON string in android.


Also according to your json string code snippet must be something like this:-

JSONObject mainObject = new JSONObject(yourstring);

JSONObject universityObject = mainObject.getJsonObject("university");
JSONString name = universityObject.getJsonString("name");
JSONString url = universityObject.getJsonString("url");

Same for other object.
```

```
share|improve this answer
""",
  "response": """
### **Scene Concept: "The Oxford Data Recovery Lab"**

#### **1. Overall Scene Theme and Core Concept**
A high-tech university data recovery lab at Oxford, where a critical server failure has disrupted research data. The scene is set
      in a cluttered, semi-abandoned server room adjacent to a research lab, filled with old hardware, tangled cables, and
      partially labeled storage. The environment demands multi-agent collaboration due to:
- **Heavy server racks** that require coordinated movement.
- **Fragile components** that need careful handling.
- **Scattered digital artifacts** (JSON logs, corrupted files) requiring analysis.
- **Security measures** (locked server cabinets, RFID keycards).

The lab is in a state of controlled chaos?some areas are meticulously organized, while others are drowning in technical debt.

---

### **2. Spatial Layout and Area Descriptions**
The scene consists of four interconnected zones:

1. **Server Room (Main Area)**
   - Crowded with humming server racks, tangled cables, and flickering diagnostic screens. The air smells faintly of ozone.
2. **Workbench & Debug Station**
   - A cluttered desk with multiple monitors, diagnostic tools, and half-disassembled hard drives.
3. **Storage Closet (Adjacent)**
   - Packed with labeled boxes of old hard drives, backup tapes, and spare parts. Some are mismarked.
4. **Security & Admin Nook**
   - A small alcove with a locked cabinet (containing admin keycards) and a bulletin board with outdated IT policies.

---

### **3. Detailed Area-by-Area Inventory**

#### **A. Server Room (Main Area)**
**a. Anchor Furniture & Installations:**
- **Three server racks (2m tall, 100kg each)** with blinking LEDs, some indicating errors.
- **Overhead cable trays** sagging under thick bundles of network and power lines.
- **A large UPS battery (120kg, 80cm?)** with a blinking "Low Charge" warning.

**b. Key Interactive & Task-Relevant Objects:**
- **Corrupted JSON log terminal** ? A monitor displaying garbled JSON data (`{"error": "malformed input", "last_entry":
      "2024-03-10 14:22:01"}`).
- **Server #14 (faulty)** ? Status LED is red, casing slightly ajar (revealing loose SATA cables inside).
- **Emergency shutdown switch** ? Behind a plexiglass cover requiring two agents to lift simultaneously.

**c. Functional Ambient Objects:**
- **Labeled cable bins** ? Each contains color-coded Cat6 cables (blue, red, yellow).
- **Tool cart** ? Holds a soldering iron (cold), multimeter (low battery), and anti-static wrist straps.
- **Whiteboard** ? Scribbled with IP addresses and a half-erased JSON schema.

**d. Background & Decorative Objects:**
- **"Server Room Rules" poster** ? Faded, peeling at the corners.
- **Dusty coffee cup** ? Stained, holding random screws.
- **Mismatched server rack panels** ? Some missing, revealing exposed circuitry.

---

#### **B. Workbench & Debug Station**
**a. Anchor Furniture & Installations:**
- **A heavy steel workbench (180cm x 80cm)** with built-in power strips and grounding points.
- **Triple-monitor setup** ? One screen flickers intermittently.

**b. Key Interactive & Task-Relevant Objects:**
- **JSON parser terminal** ? Running a frozen instance of Eclipse with a half-written parsing script.
- **External HDD (500GB, labeled "Backup #47")** ? Slightly warm, implying recent use.
- **Broken SATA-to-USB adapter** ? Bent pins, rendering it unusable without repair.

**c. Functional Ambient Objects:**
- **Stack of hard drives (3.5")** ? Mixed labels ("/dev/sdb1", "DO NOT ERASE").
- **USB hub (4-port, overloaded)** ? Two ports non-functional.
- **Spare keyboard** ? Missing the "F5" key.

**d. Background & Decorative Objects:**
- **Post-it notes** ? One reads "JSON.parse() fails on nested arrays??"
- **Empty energy drink cans** ? At least four, crushed.
- **A framed photo of Linus Torvalds** ? Tilted slightly.

---

#### **C. Storage Closet**
**a. Anchor Furniture & Installations:**
- **Industrial shelving units (2.5m tall)** ? Overloaded with boxes.
- **Locked server cabinet (RFID-protected)** ? Contains spare admin keycards.

**b. Key Interactive & Task-Relevant Objects:**
```

```
- **Mislabeled box ("Misc Cables")** ? Actually contains three external SSDs with JSON dumps.
- **Damaged UPS battery (leaking slightly)** ? Requires careful disposal.

**c. Functional Ambient Objects:**
- **Tape backup drives (LTO-6)** ? Some unspooled.
- **Ethernet switch (unpowered)** ? Dusty but functional.

**d. Background & Decorative Objects:**
- **Outdated "Y2K Compliance" certificate** ? Framed but cracked.
- **Dusty volleyball trophy** ? Apparently left behind by a past intern.

---

### **4. Scene Affordances and Embedded Potential**

#### **Collaborative Transportation Affordances:**
- **Server Rack #3 (150kg, 2m tall)** ? Requires two agents to safely relocate.
- **UPS Battery (120kg, bulky)** ? Too heavy for one agent to lift without strain.

#### **Reasoning & Tool-Use Affordances:**
- **Attribute-Based Reasoning:**
  - Among five external HDDs on the workbench, only one has:
    - **Blue casing**
    - **Handwritten label "LOG_BACKUP_CRITICAL"**
    - **Slightly warm to the touch**
  - The presence of other **non-critical blue objects** (a coffee mug, a USB hub) adds noise.
- **Compound Tool-Use Reasoning:**
  - **Problem:** The JSON terminal is frozen.
  - **Solution:** The **"Reset" dongle** is inside the locked server cabinet (which requires an **RFID keycard**, found in a desk
      drawer).

---

**Final Notes:** This scene is **dense, plausible, and collaboration-critical**. The mix of fragile, heavy, and mislabeled objects
       ensures that agents must communicate, reason, and coordinate to succeed. The ambient clutter adds realism while obscuring
      critical items, forcing precise search and reasoning.
"""
}
```

Stage 2: Scene (`data/data-all/scene/01242_scene.json`). The Scene Architect (Stage 2 prompt) compiles the blueprint into a single, structured JSON: 4 rooms, 36 objects, with continuous physical properties (`weight`, `material`, `size`, `color`, `model_number`, `led_color`, etc.), spatial relations (`location_id`), and per-object `states` aligned to the agent ability schema. Note the deliberate over-50 kg `weight` on several `FURNITURE` entries, which is the precondition for implicit-collaboration tasks generated downstream.

```
{
  "description": "A high-tech university data recovery lab at Oxford, filled with server racks, diagnostic tools, and scattered
      digital artifacts, requiring multi-agent collaboration for data recovery and system repairs.",
  "rooms": [
    {
      "id": "server_room",
      "name": "Server Room",
      "properties": {
        "type": "industrial"
      },
      "connected_to_room_ids": [
        "workbench_area",
        "storage_closet",
        "security_nook"
      ]
    },
    {
      "id": "workbench_area",
      "name": "Workbench & Debug Station",
      "properties": {
        "type": "workshop"
      },
      "connected_to_room_ids": [
        "server_room",
        "storage_closet"
      ]
    },
    {
      "id": "storage_closet",
      "name": "Storage Closet",
      "properties": {
        "type": "storage"
      },
      "connected_to_room_ids": [
        "server_room",
        "workbench_area",
        "security_nook"
      ]
```

```
    },
    {
      "id": "security_nook",
      "name": "Security & Admin Nook",
      "properties": {
        "type": "office"
      },
      "connected_to_room_ids": [
        "server_room",
        "storage_closet"
      ]
    }
  ],
  "objects": [
    {
      "id": "server_rack_1",
      "name": "Server Rack #1",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          2.0,
          0.8,
          2.0
        ],
        "weight": 100.0,
        "is_container": true,
        "material": "steel",
        "model_number": "SR-2000",
        "led_color": "red"
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "server_rack_2",
      "name": "Server Rack #2",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          2.0,
          0.8,
          2.0
        ],
        "weight": 100.0,
        "is_container": true,
        "material": "steel",
        "model_number": "SR-2000",
        "led_color": "green"
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "server_rack_3",
      "name": "Server Rack #3",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          2.0,
          0.8,
          2.0
        ],
        "weight": 60.0,
        "is_container": true,
        "material": "steel",
        "model_number": "SR-2000",
        "led_color": "red"
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "ups_battery_1",
      "name": "UPS Battery",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          0.8,
          0.8,
          0.8
```

```
        ],
        "weight": 60.0,
        "is_container": false,
        "material": "plastic",
        "brand": "APC",
        "warning_light": "blinking"
      },
      "states": {}
    },
    {
      "id": "corrupted_log_terminal_1",
      "name": "Corrupted JSON Log Terminal",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          0.5,
          0.3,
          0.4
        ],
        "weight": 15.0,
        "is_container": false,
        "material": "plastic",
        "brand": "Dell",
        "display_content": "{\"error\": \"malformed input\", \"last_entry\": \"2024-03-10 14:22:01\"}"
      },
      "states": {
        "is_running": true
      }
    },
    {
      "id": "emergency_shutdown_switch_1",
      "name": "Emergency Shutdown Switch",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          0.3,
          0.2,
          0.1
        ],
        "weight": 5.0,
        "is_container": false,
        "material": "plexiglass",
        "cover_weight": 20.0
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "tool_cart_1",
      "name": "Tool Cart",
      "type": "FURNITURE",
      "location_id": "in:server_room",
      "properties": {
        "size": [
          1.0,
          0.6,
          0.9
        ],
        "weight": 30.0,
        "is_container": true,
        "material": "metal",
        "wheels": 4
      },
      "states": {}
    },
    {
      "id": "soldering_iron_1",
      "name": "Soldering Iron",
      "type": "ITEM",
      "location_id": "on:tool_cart_1",
      "properties": {
        "weight": 0.5,
        "material": "metal",
        "brand": "Weller",
        "provides_abilities": [
          "solder"
        ]
      },
      "states": {
        "is_soldered": false
      }
    },
    {
      "id": "multimeter_1",
```

```
      "name": "Multimeter",
      "type": "ITEM",
      "location_id": "on:tool_cart_1",
      "properties": {
        "weight": 0.3,
        "material": "plastic",
        "brand": "Fluke",
        "battery_status": "low"
      },
      "states": {}
    },
    {
      "id": "anti_static_wrist_strap_1",
      "name": "Anti-Static Wrist Strap",
      "type": "ITEM",
      "location_id": "on:tool_cart_1",
      "properties": {
        "weight": 0.1,
        "material": "fabric",
        "color": "black"
      },
      "states": {}
    },
    {
      "id": "workbench_1",
      "name": "Steel Workbench",
      "type": "FURNITURE",
      "location_id": "in:workbench_area",
      "properties": {
        "size": [
          1.8,
          0.8,
          0.9
        ],
        "weight": 180.0,
        "is_container": true,
        "material": "steel",
        "power_strips": 2
      },
      "states": {}
    },
    {
      "id": "triple_monitor_setup_1",
      "name": "Triple Monitor Setup",
      "type": "FURNITURE",
      "location_id": "on:workbench_1",
      "properties": {
        "size": [
          1.2,
          0.3,
          0.5
        ],
        "weight": 12.0,
        "is_container": false,
        "brand": "LG",
        "screen_condition": "flickering"
      },
      "states": {
        "is_on": false
      }
    },
    {
      "id": "json_parser_terminal_1",
      "name": "JSON Parser Terminal",
      "type": "FURNITURE",
      "location_id": "on:workbench_1",
      "properties": {
        "size": [
          0.4,
          0.3,
          0.4
        ],
        "weight": 5.0,
        "is_container": false,
        "software": "Eclipse",
        "status": "frozen"
      },
      "states": {
        "is_running": true
      }
    },
    {
      "id": "external_hdd_1",
      "name": "External HDD (Backup #47)",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
```

```
        "weight": 0.6,
        "material": "plastic",
        "brand": "Seagate",
        "capacity": "500GB",
        "temperature": "warm"
      },
      "states": {}
    },
    {
      "id": "broken_sata_adapter_1",
      "name": "Broken SATA-to-USB Adapter",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.1,
        "material": "plastic",
        "condition": "bent pins",
        "provides_abilities": [
          "solder"
        ]
      },
      "states": {
        "is_soldered": false
      }
    },
    {
      "id": "usb_hub_1",
      "name": "USB Hub (4-port)",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.2,
        "material": "plastic",
        "brand": "Anker",
        "functional_ports": 2
      },
      "states": {}
    },
    {
      "id": "spare_keyboard_1",
      "name": "Spare Keyboard",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.8,
        "material": "plastic",
        "brand": "Logitech",
        "missing_key": "F5",
        "provides_abilities": [
          "type"
        ]
      },
      "states": {
        "is_typed": false
      }
    },
    {
      "id": "industrial_shelving_1",
      "name": "Industrial Shelving Unit",
      "type": "FURNITURE",
      "location_id": "in:storage_closet",
      "properties": {
        "size": [
          2.5,
          1.0,
          0.5
        ],
        "weight": 60.0,
        "is_container": true,
        "material": "steel",
        "shelves": 5
      },
      "states": {}
    },
    {
      "id": "locked_server_cabinet_1",
      "name": "Locked Server Cabinet",
      "type": "FURNITURE",
      "location_id": "in:storage_closet",
      "properties": {
        "size": [
          1.8,
          0.6,
          0.8
        ],
        "weight": 80.0,
        "is_container": true,
```

```
        "security": "RFID",
        "lock_type": "electronic"
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "mislabeled_box_1",
      "name": "Mislabeled Box (Misc Cables)",
      "type": "ITEM",
      "location_id": "on:industrial_shelving_1",
      "properties": {
        "weight": 3.0,
        "material": "cardboard",
        "actual_content": "external SSDs",
        "provides_abilities": [
          "unwrap"
        ]
      },
      "states": {
        "is_wrapped": true
      }
    },
    {
      "id": "damaged_ups_battery_1",
      "name": "Damaged UPS Battery",
      "type": "ITEM",
      "location_id": "on:industrial_shelving_1",
      "properties": {
        "weight": 15.0,
        "material": "plastic",
        "condition": "leaking",
        "hazard": "chemical"
      },
      "states": {}
    },
    {
      "id": "tape_backup_drive_1",
      "name": "Tape Backup Drive (LTO-6)",
      "type": "ITEM",
      "location_id": "on:industrial_shelving_1",
      "properties": {
        "weight": 2.5,
        "material": "plastic",
        "status": "unspooled"
      },
      "states": {}
    },
    {
      "id": "ethernet_switch_1",
      "name": "Ethernet Switch",
      "type": "ITEM",
      "location_id": "on:industrial_shelving_1",
      "properties": {
        "weight": 1.8,
        "material": "plastic",
        "brand": "Cisco",
        "power_status": "unpowered"
      },
      "states": {}
    },
    {
      "id": "rfid_keycard_1",
      "name": "RFID Keycard",
      "type": "ITEM",
      "location_id": "in:locked_server_cabinet_1",
      "properties": {
        "weight": 0.05,
        "material": "plastic",
        "access_level": "admin",
        "provides_abilities": [
          "open"
        ]
      },
      "states": {}
    },
    {
      "id": "reset_dongle_1",
      "name": "Reset Dongle",
      "type": "ITEM",
      "location_id": "in:locked_server_cabinet_1",
      "properties": {
        "weight": 0.1,
        "material": "plastic",
        "function": "terminal reset",
        "provides_abilities": [
```

```
          "stop_process"
        ]
      },
      "states": {}
    },
    {
      "id": "admin_desk_1",
      "name": "Admin Desk",
      "type": "FURNITURE",
      "location_id": "in:security_nook",
      "properties": {
        "size": [
          1.2,
          0.6,
          0.75
        ],
        "weight": 40.0,
        "is_container": true,
        "material": "wood",
        "drawers": 2
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "bulletin_board_1",
      "name": "Bulletin Board",
      "type": "FURNITURE",
      "location_id": "in:security_nook",
      "properties": {
        "size": [
          1.0,
          0.1,
          0.8
        ],
        "weight": 10.0,
        "is_container": false,
        "material": "cork",
        "content": "outdated IT policies"
      },
      "states": {}
    },
    {
      "id": "desk_drawer_1",
      "name": "Desk Drawer",
      "type": "FURNITURE",
      "location_id": "in:admin_desk_1",
      "properties": {
        "size": [
          0.5,
          0.4,
          0.15
        ],
        "weight": 5.0,
        "is_container": true,
        "material": "wood"
      },
      "states": {
        "is_open": false
      }
    },
    {
      "id": "spare_rfid_keycard_1",
      "name": "Spare RFID Keycard",
      "type": "ITEM",
      "location_id": "in:desk_drawer_1",
      "properties": {
        "weight": 0.05,
        "material": "plastic",
        "access_level": "admin",
        "provides_abilities": [
          "open"
        ]
      },
      "states": {}
    },
    {
      "id": "server_room_rules_poster_1",
      "name": "Server Room Rules Poster",
      "type": "ITEM",
      "location_id": "in:server_room",
      "properties": {
        "weight": 0.2,
        "material": "paper",
        "condition": "faded"
      },
```

```
      "states": {}
    },
    {
      "id": "dusty_coffee_cup_1",
      "name": "Dusty Coffee Cup",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.3,
        "material": "ceramic",
        "color": "blue",
        "contents": "random screws"
      },
      "states": {}
    },
    {
      "id": "post_it_notes_1",
      "name": "Post-it Notes",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.05,
        "material": "paper",
        "message": "JSON.parse() fails on nested arrays??"
      },
      "states": {}
    },
    {
      "id": "energy_drink_cans_1",
      "name": "Empty Energy Drink Cans",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.1,
        "material": "aluminum",
        "quantity": 4,
        "condition": "crushed"
      },
      "states": {}
    },
    {
      "id": "linus_torvalds_photo_1",
      "name": "Framed Photo of Linus Torvalds",
      "type": "ITEM",
      "location_id": "on:workbench_1",
      "properties": {
        "weight": 0.5,
        "material": "glass",
        "condition": "tilted"
      },
      "states": {}
    },
    {
      "id": "y2k_certificate_1",
      "name": "Outdated Y2K Compliance Certificate",
      "type": "ITEM",
      "location_id": "in:storage_closet",
      "properties": {
        "weight": 0.3,
        "material": "paper",
        "condition": "cracked"
      },
      "states": {}
    },
    {
      "id": "volleyball_trophy_1",
      "name": "Dusty Volleyball Trophy",
      "type": "ITEM",
      "location_id": "in:storage_closet",
      "properties": {
        "weight": 1.2,
        "material": "metal",
        "inscription": "Intern of the Year 1999"
      },
      "states": {}
    }
  ],
  "abilities": [
    "open",
    "solder",
    "stop_process",
    "turn_on",
    "type",
    "unwrap",
    "close"
  ]
}
```

Stage 3: Task (`data/data-all/task/01242_task.json`). The Evaluation Suite Architect (Stage 3 prompt) produces 14 self-validating tasks (exactly two per category for all seven categories) plus a single globally-consistent `agents_config` (`max_weight` = 30 kg) chosen *after* the tasks are drafted, so that single-agent tasks remain feasible while implicit-collaboration tasks become a logical necessity. Each task carries its own minimal `validation_checks` block.


```
{
  "task_background": "Prepare the high-tech university data recovery lab for an urgent system audit and data restoration process
      .",
  "agents_config": [
    {
      "name": "robot_1",
      "max_grasp_limit": 1.0,
      "max_weight": 30.0,
      "max_size": [
        1.0,
        1.0,
        1.0
      ]
    },
    {
      "name": "robot_2",
      "max_grasp_limit": 1.0,
      "max_weight": 30.0,
      "max_size": [
        1.0,
        1.0,
        1.0
      ]
    }
  ],
  "tasks": [
    {
      "task_description": "Place the soldering_iron_1 on the workbench_1.",
      "task_category": "direct_command",
      "validation_checks": [
        {
          "id": "soldering_iron_1",
          "location_id": "on:workbench_1"
        }
      ]
    },
    {
      "task_description": "Place the dusty_coffee_cup_1 in the storage_closet.",
      "task_category": "direct_command",
      "validation_checks": [
        {
          "id": "dusty_coffee_cup_1",
          "location_id": "in:storage_closet"
        }
      ]
    },
    {
      "task_description": "Find the heaviest, steel server rack and open it.",
      "task_category": "attribute_reasoning",
      "validation_checks": [
        {
          "id": "server_rack_3",
          "is_open": true
        }
      ]
    },
    {
      "task_description": "Find the only server rack with a green LED and open it.",
      "task_category": "attribute_reasoning",
      "validation_checks": [
        {
          "id": "server_rack_2",
          "is_open": true
        }
      ]
    },
    {
      "task_description": "Repair the broken_sata_adapter_1 using the soldering_iron_1.",
      "task_category": "tool_use",
      "validation_checks": [
        {
          "id": "broken_sata_adapter_1",
          "is_soldered": true
        }
      ]
    },
    {
      "task_description": "Use the spare_keyboard_1 to type on the json_parser_terminal_1.",
```

```
      "task_category": "tool_use",
      "validation_checks": [
        {
          "id": "spare_keyboard_1",
          "is_typed": true
        }
      ]
    },
    {
      "task_description": "Use the soldering_iron_1 to repair the broken_sata_adapter_1 and then place it on the tool_cart_1.",
      "task_category": "compound_reasoning",
      "validation_checks": [
        {
          "id": "broken_sata_adapter_1",
          "is_soldered": true
        },
        {
          "id": "broken_sata_adapter_1",
          "location_id": "on:tool_cart_1"
        }
      ]
    },
    {
      "task_description": "Find the only server rack with a red LED and open it, then use the soldering_iron_1 to repair any
      visible damage.",
      "task_category": "compound_reasoning",
      "validation_checks": [
        {
          "id": "server_rack_1",
          "is_open": true
        },
        {
          "id": "server_rack_1",
          "is_open": true
        }
      ]
    },
    {
      "task_description": "Have robot_1 and robot_2 cooperate to open the locked_server_cabinet_1 using the rfid_keycard_1.",
      "task_category": "explicit_collaboration",
      "validation_checks": [
        {
          "id": "locked_server_cabinet_1",
          "is_open": true
        }
      ]
    },
    {
      "task_description": "Have robot_1 and robot_2 cooperate to turn on the triple_monitor_setup_1.",
      "task_category": "explicit_collaboration",
      "validation_checks": [
        {
          "id": "triple_monitor_setup_1",
          "is_on": true
        }
      ]
    },
    {
      "task_description": "Move the ups_battery_1 to the workbench_area.",
      "task_category": "implicit_collaboration",
      "validation_checks": [
        {
          "id": "ups_battery_1",
          "location_id": ":workbench_area"
        }
      ]
    },
    {
      "task_description": "Move the industrial_shelving_1 to the server_room.",
      "task_category": "implicit_collaboration",
      "validation_checks": [
        {
          "id": "industrial_shelving_1",
          "location_id": ":server_room"
        }
      ]
    },
    {
      "task_description": "Cooperatively move the damaged_ups_battery_1 to the workbench_area and use the soldering_iron_1 to
      repair it.",
      "task_category": "compound_collaboration",
      "validation_checks": [
        {
          "id": "damaged_ups_battery_1",
          "location_id": ":workbench_area"
        },
        {
```

```
          "id": "damaged_ups_battery_1",
          "is_soldered": true
        }
      ]
    },
    {
      "task_description": "Cooperatively move the server_rack_3 to the storage_closet and open it.",
      "task_category": "compound_collaboration",
      "validation_checks": [
        {
          "id": "server_rack_3",
          "location_id": ":storage_closet"
        },
        {
          "id": "server_rack_3",
          "is_open": true
        }
      ]
    }
  ],
  "scene_id": "01242"
}
```


Stage 4: Verify (distilled from `01242_task.json`). We isolate the `validation_checks` produced in Stage 3 to make the simulator-side verification schema explicit. Each entry is the minimal state-change snippet against which the simulator declares success. Observe the diversity of verification atoms across the 14 tasks: `location_id` for spatial placement, and state flags such as `is_open`, `is_soldered`, `is_on`, `is_broken` for action-induced state transitions. The latter are achievable only after the corresponding tool has been grasped and the agent's action space has been dynamically extended (cf. §3).


```
[
  {
    "task_category": "direct_command",
    "task_description": "Place the soldering_iron_1 on the workbench_1.",
    "validation_checks": [
      {
        "id": "soldering_iron_1",
        "location_id": "on:workbench_1"
      }
    ]
  },
  {
    "task_category": "direct_command",
    "task_description": "Place the dusty_coffee_cup_1 in the storage_closet.",
    "validation_checks": [
      {
        "id": "dusty_coffee_cup_1",
        "location_id": "in:storage_closet"
      }
    ]
  },
  {
    "task_category": "attribute_reasoning",
    "task_description": "Find the heaviest, steel server rack and open it.",
    "validation_checks": [
      {
        "id": "server_rack_3",
        "is_open": true
      }
    ]
  },
  {
    "task_category": "attribute_reasoning",
    "task_description": "Find the only server rack with a green LED and open it.",
    "validation_checks": [
      {
        "id": "server_rack_2",
        "is_open": true
      }
    ]
  },
  {
    "task_category": "tool_use",
    "task_description": "Repair the broken_sata_adapter_1 using the soldering_iron_1.",
    "validation_checks": [
      {
        "id": "broken_sata_adapter_1",
        "is_soldered": true
      }
    ]
  },
  {
```

```
    "task_category": "tool_use",
    "task_description": "Use the spare_keyboard_1 to type on the json_parser_terminal_1.",
    "validation_checks": [
      {
        "id": "spare_keyboard_1",
        "is_typed": true
      }
    ]
  },
  {
    "task_category": "compound_reasoning",
    "task_description": "Use the soldering_iron_1 to repair the broken_sata_adapter_1 and then place it on the tool_cart_1.",
    "validation_checks": [
      {
        "id": "broken_sata_adapter_1",
        "is_soldered": true
      },
      {
        "id": "broken_sata_adapter_1",
        "location_id": "on:tool_cart_1"
      }
    ]
  },
  {
    "task_category": "compound_reasoning",
    "task_description": "Find the only server rack with a red LED and open it, then use the soldering_iron_1 to repair any visible
       damage.",
    "validation_checks": [
      {
        "id": "server_rack_1",
        "is_open": true
      },
      {
        "id": "server_rack_1",
        "is_open": true
      }
    ]
  },
  {
    "task_category": "explicit_collaboration",
    "task_description": "Have robot_1 and robot_2 cooperate to open the locked_server_cabinet_1 using the rfid_keycard_1.",
    "validation_checks": [
      {
        "id": "locked_server_cabinet_1",
        "is_open": true
      }
    ]
  },
  {
    "task_category": "explicit_collaboration",
    "task_description": "Have robot_1 and robot_2 cooperate to turn on the triple_monitor_setup_1.",
    "validation_checks": [
      {
        "id": "triple_monitor_setup_1",
        "is_on": true
      }
    ]
  },
  {
    "task_category": "implicit_collaboration",
    "task_description": "Move the ups_battery_1 to the workbench_area.",
    "validation_checks": [
      {
        "id": "ups_battery_1",
        "location_id": ":workbench_area"
      }
    ]
  },
  {
    "task_category": "implicit_collaboration",
    "task_description": "Move the industrial_shelving_1 to the server_room.",
    "validation_checks": [
      {
        "id": "industrial_shelving_1",
        "location_id": ":server_room"
      }
    ]
  },
  {
    "task_category": "compound_collaboration",
    "task_description": "Cooperatively move the damaged_ups_battery_1 to the workbench_area and use the soldering_iron_1 to repair
       it.",
    "validation_checks": [
      {
        "id": "damaged_ups_battery_1",
        "location_id": ":workbench_area"
      },
```

```
      {
        "id": "damaged_ups_battery_1",
        "is_soldered": true
      }
    ]
  },
  {
    "task_category": "compound_collaboration",
    "task_description": "Cooperatively move the server_rack_3 to the storage_closet and open it.",
    "validation_checks": [
      {
        "id": "server_rack_3",
        "location_id": ":storage_closet"
      },
      {
        "id": "server_rack_3",
        "is_open": true
      }
    ]
  }
]
```