FROM HOURS TO TASKS: EVALUATING EVIDENCE-GROUNDED DECISIONS BY LONG-VIDEO AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-video agents must integrate observations across time, track changing objects and states, and use tools to act. Evaluation requires executable tasks with agent-accessible evidence for key choices, so missing evidence is not mistaken for limited capability. We introduce EvidBind-Bench: 3,000 tasks across 13 domains grounded in hour-scale videos, with a design–binding method that fixes goals, constraints, and required decisions before linking them to observations, objects, times, and operational conditions. The framework comprises **(1) Automated task generation**, which instantiates reusable decision structures, binds choices to evidence, and generates instructions and execution paths, revising unsupported choices under unchanged requirements; **(2) Task review and repair**, which combines automated checks and human review, reuses shared assessments, and targets affected decisions, with acceptance checking support, executability, and requirement preservation; and **(3) Reference-agent evaluation**, which supplies ReAct and multi-agent frameworks with common public interfaces, scoring, and diagnostic execution records. Independent human audits of 390 sampled tasks estimate collection validity. Under the evaluated conditions, the complete procedure lowers combined labor and computational cost per estimated valid task by 72.86% relative to per-task review and repair. ReAct and the multi-agent system achieve 11.23% and 17.73% success, versus 91.73% for humans, revealing persistent difficulties on tasks passing construction checks. EvidBind-Bench combines low-cost task construction and agent evaluation to study the gap between solvability and reliable execution.

1 INTRODUCTION

Long-video agents must integrate observations across time, track changing objects and states, and use tools for long-horizon tasks. LongVideoBench (Wu et al., 2024) and EgoLife (Yang et al., 2025) assess long-video understanding; Ego2Web (Yu et al., 2026), VideoGAIA (Zhang et al., 2026), and VideoWebArena (Jang et al., 2025) connect video perception with tool execution. Yet evaluating evidence-grounded actions over hour-scale videos remains difficult: relevant evidence spans distant segments, and earlier observations may no longer describe later states. Executable tasks therefore need agent-accessible support for key choices, so missing evidence is not mistaken for limited capability.

Table 1 highlights two construction challenges. First, relevant clips and executable calls do not establish action support. Recording the milk purchased requires distinguishing an initially selected carton from one ultimately bought: an earlier readable label may identify a returned carton, while checkout shows only its replacement's back. An accepted product identifier does not establish the right identity. Second, separate review of shared evidence duplicates effort, while identity corrections can affect quantities, amounts, and state updates. Construction must locate decision-specific gaps and control rework while preserving goals and constraints.

EvidBind-Bench comprises 3,000 tasks across 13 domains, four settings, and five composable decision structures (Figure 1; definitions in Appendix A). Construction-stage acceptance determines inclusion; independent sampled human audits assess collection quality. ReAct (Yao et al., 2023) and multi-agent reference frameworks share public interfaces and scoring of goal states and execution

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

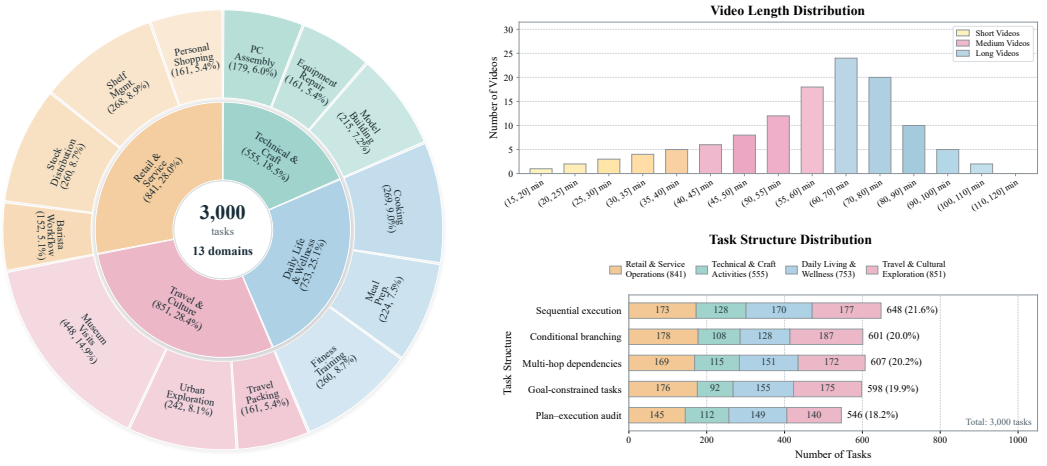

Figure 1: **Overview of EvidBind-Bench.** Left: 3,000 tasks across four settings (inner ring) and 13 domains (outer ring). Upper right: source-video durations, counting videos, not tasks or agent execution time. Lower right: task counts by decision structure and setting.

constraints, allowing alternative valid paths. Private bindings and solutions remain hidden. Their performance reveals a substantial gap between solvability and reliable execution.

Our design–binding method freezes goals, constraints, required decisions, and dependencies from reference designs and tool interfaces, then searches for bindings, public acquisition paths, and executable plans. Bindings link choices to observations, objects, times, and operational conditions. Public evidence must precede action; private records cannot justify unknown parameters. Prechecks, human decision review, and gap-directed repair preserve requirements. A replacement-carton mismatch, for example, triggers identity-cue search and rechecking of purchase entries and amounts. Shared fact findings are reusable, but decision support and acceptance remain task-specific. The repair control shares initial candidates, materials, repair resources, and acceptance criteria, with human feedback, per-task review, and whole-task rechecks. Independent audits assess repair and output quality alongside measured review and computation costs. Section 4.5 reports 72.86% lower combined cost per estimated valid task, including shared-check amortization at unequal output scales.

Our contributions are fourfold:

1. **An evidence-grounded benchmark for hour-scale videos.** Tasks assess temporal integration and tool use, with construction acceptance and independent audits estimating 96.4% overall and 92.9% complex-task validity.
2. **Requirement-preserving task generation.** Reusable structures bind required choices to public evidence and locate repair targets while preserving goals, constraints, and dependencies.
3. **Low-cost validation and repair.** Shared assessments and targeted rechecks reduce repetition. Our procedure and full per-task review achieve 87.6% versus 86.9% repair success and 96.4% versus 95.8% audited validity, contextualizing cost savings.
4. **Reference frameworks and capability diagnosis.** Under common interfaces and scoring, ReAct and multi-agent success reaches 11.23% and 17.73%, versus 91.73% for humans; execution records support failure analysis.

2 Related Work

Long-video understanding and planning. Long-video and egocentric benchmarks study how models retrieve and combine information across extended recordings, as exemplified by LongVideoBench and EgoLife (Wu et al., 2024; Yang et al., 2025). Planning-oriented benchmarks

Table 1: **Comparison with representative video-understanding, planning, and video-grounded agent benchmarks.** EvidBind-Bench combines hour-scale video, state-changing tool actions, entity–time evidence links to decisions, and checks of goals and execution constraints. Coverage units and QA/task counting conventions differ across benchmarks.

	Task coverage & video			Agent execution & interaction			Evidence & assessment	
Benchmark	**Coverage**	**Scale**	**Hour-scale video**	**Tool actions**	**Task-state updates**	**Simulated user turns**	**Evidence / reference**	**Evaluation target**
Video understanding and planning								
LongVideoBench (Wu et al., 2024)	10	6,678 QA	✓	×	×	×	Moment refs.	Answer
EgoPlan-Bench2 (Qiu et al., 2026)	4	1,321 QA	×	×	×	×	Action history	Next action
EgoCoT-Bench (Dai et al., 2026)	4	3,172 QA	×	×	×	×	STSG	Answer + reasoning
Multimodal and video-grounded agents								
Agent-X (Ashraf et al., 2026)	6	828 tasks	×	✓	×	×	Tool traces	Answer + trace
VideoWebArena (Jang et al., 2025)	6	2,021 tasks	×	✓	✓	×	Video QA	Web + QA
Ego2Web (Yu et al., 2026)	5	500 tasks	×	✓	✓	×	Visual cues	Web success
EgoBench (Liu et al., 2026)	4	1,045 tasks	×	✓	✓	✓	Cues + calls	DB + call coverage
EvidBind-Bench (Ours)	**13**	**3,000 tasks**	✓	✓	✓	✓	**Entity–time → decision**	**Goal + exec. constraints**

✓ Yes; × No. STSG: spatiotemporal scene graph; DB: database. User turns exclude tool/environment feedback. Coverage follows source-specific taxonomies: video categories, domains, task groups, environments, task categories, or scenarios. EgoPlan-Bench2 has 4 domains; EgoCoT-Bench has 4 task groups.

such as EgoPlan-Bench2 further connect observed task progress to action selection (Qiu et al., 2026). These settings assess answers or proposed actions; executing a state-changing task additionally requires selecting tool arguments and checking the resulting environment state.

Interactive and video-grounded agent benchmarks. Moving from understanding to execution, τ-bench and Toolathlon assess interactive tool use and verifiable task completion (Li et al., 2026; Yao et al., 2024). VideoWebArena, Ego2Web, EgoBench, and VideoGAIA bring video context into web interaction or tool-assisted problem solving (Jang et al., 2025; Liu et al., 2026; Yu et al., 2026; Zhang et al., 2026). Their protocols include checks of outcomes, execution processes, or consistency with video evidence. EvidBind-Bench emphasizes hour-scale video tasks with explicit support for required decisions, while separating success under goal and execution constraints from trajectory-based analysis of evidence use.

Automated task construction and repair. Reliable evaluation also depends on how tasks are generated and repaired. Structured generation methods use tool graphs, environment models, or compositions of tool-grounded subtasks to guide synthesis (Hu et al., 2025; Shen et al., 2024; Shi et al., 2026). Incremental verification in TaskCraft and scene-graph-guided generation in EgoCoT-Bench establish precedents for efficient validation and structured evidence (Dai et al., 2026; Shi et al., 2026). EvidBind-Bench fixes goals, constraints, and required decisions before binding them to scene evidence. Links among observations, entities, time, and action conditions identify unsupported choices and guide additional observation or repair. To limit repeated review, the procedure reuses shared fact assessments and rechecks affected decision links, while retaining full execution replay and the original task requirements.

3 Benchmark Design and Construction

We search for executable, publicly supported decisions under fixed requirements. Generation produces candidates; screening, review, repair, and acceptance determine inclusion. Independent audits and agent evaluation follow. Appendices A–C provide specifications and operational details.

3.1 Automated Task Generation: From Scene Evidence to Decision Tasks

We **specify required choices before seeking their evidence**, connecting scenes and tools from 13 domains to five reusable decision topologies in four steps.

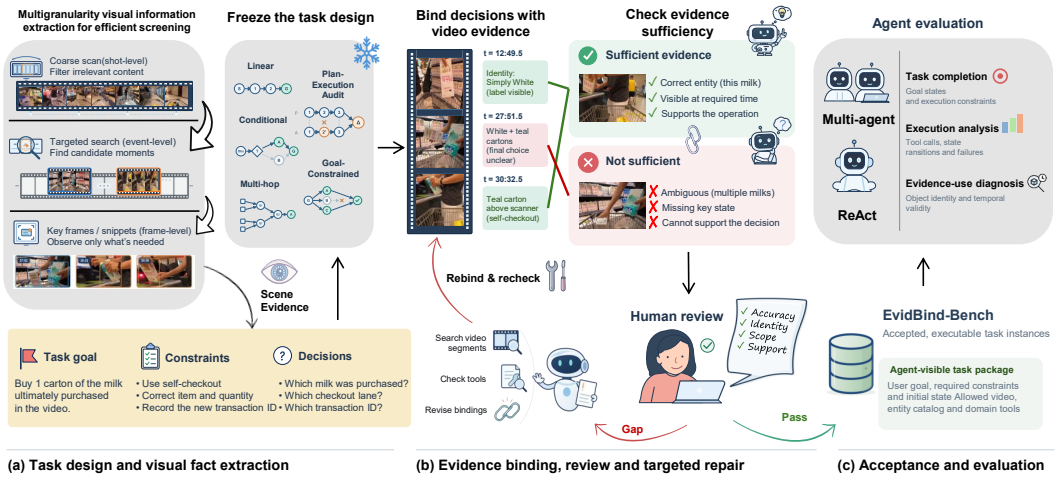

Figure 2: **Method overview.** Frozen requirements guide evidence binding and targeted repair. Shared findings reduce review effort; decision support, full replay, and specification checks determine acceptance. The milk-purchase example is expanded in Appendix A.4; independent task audits follow construction.

(1) Scene preparation. Coarse observations locate entities and events; targeted inspection resolves decision-changing ambiguity in identities, attributes, branches, and states. Records retain sources, candidates, event times, and state claims; unclear observations remain leads. Binding gaps trigger budgeted acquisition, charging failed attempts as well. Uninspected video remains unknown and unresolved gaps cannot pass.

Reviewed scene inputs comprise facts, a queryable catalog, tools, and an isolated state. Catalog candidates may be absent from the video; private provenance cannot reveal video answers through queries. Cached observations guide construction but enter execution history only through permitted public acquisition (Appendix B.1).

(2) Freezing a reusable design. From exemplar construction records, goals, and tool interfaces, the generator records required state changes, choices, and dependencies without filling target-video answers or evidence locations: $d = (g, c, Q, A)$, where g is the target state, c the constraints, Q the required entity, attribute, branch, and parameter decisions, and A their informational, conditional, and state dependencies on actions. A required decision follows from the public requirements, rather than every reference-trajectory step.

The five composable, non-exhaustive topologies are **sequential progression, conditional branching, multi-hop dependencies, joint constraints, and plan–execution comparison**. Fixed precedence assigns a primary label before evaluation; secondary tags remain, and repair cannot change the primary label. Domains follow terminal outcomes (Appendix A.2).

Transfer reuses decision roles and dependencies, not identities, answers, or timestamps. Each instance fixes its video, target event, and candidate scope. Evidence gaps cannot justify an easier event, invented candidate, omitted decision, or weakened constraint.

(3) Binding evidence to required decisions. Unlike tool-dependency organization in TaskCraft (Shi et al., 2026) or scene-graph evidence in EgoCoT-Bench (Dai et al., 2026), our binding and repair unit is a decision fixed before scene instantiation. Each binding $b_q = (E_q, \mu_q, J_q, \eta_q)$ records public sources/access paths, entity correspondences and unresolved candidates, applicable times, and the supported action condition. Different valid paths may satisfy a decision. Access cannot presuppose private answers; conditions cannot be weakened to fit evidence.

Scope adequacy is separate from decision support. Scope must retain decision-changing alternatives from public restrictions, catalogs, and observations, including unmatched observed objects. Exclusion needs pre-action evidence or public rules. Relevant uninspected intervals leave scope

unresolved. An omitted alternative invalidates the instance; scope expansion requires versioned reconstruction under unchanged requirements, retaining the failure and its cost.

Let h_k contain public instructions, observations, and tool results actually acquired before action step k; $\mathcal{W}_k$ contains in-scope interpretations consistent with this history and public rules. If $\chi_q(a, w)$ indicates satisfaction of decision q, then

$$\text{Support}(q, a \mid h_k) \iff \mathcal{W}_k \neq \emptyset \ \wedge \ \forall w \in \mathcal{W}_k,\ \chi_q(a, w). \tag{1}$$

Support allows multiple interpretations and valid actions; empty interpretations indicate conflict, and undecidable support remains unresolved. Later evidence cannot justify earlier actions. This construction-screening criterion neither proves scope completeness nor requires online computation by evaluated agents. Scope review is a bounded check of task-relevant sources, not proof of completeness over the entire video.

Search jointly extends bindings and a public acquisition–action prefix, tracking history, tool state, checkpoints, and alternatives. It expands the first dependency-enabled unresolved decision using bounded, screened, ranked proposals. Support and tool preconditions must hold before action. Gaps trigger observation; conflicts roll back implicated decisions and dependent bindings. Discarded observations must be publicly reacquired and state-dependent queries rerun. Shared token, proposal, observation-time, and tool-call caps charge every branch, failure, and replay without refunds. Exhaustion leaves unresolved instances; a complete supported, goal-satisfying path advances to packaging. Appendices B.2, B.3, and A.4 expand search, scope checks, and the milk example.

(4) Compiling candidates. The frozen design, bindings, and executed search path produce public instructions, a private reference trajectory, and a goal checker. The reference serializes the executed path; instructions preserve its public access assumptions, and changed clues or operations return to binding search. Bindings, reference steps, and answers stay private. Checking final goals and execution constraints admits alternative valid paths. Candidates proceed to review and replay under unchanged requirements.

3.2 Binding-Guided Low-Cost Review and Repair

Bindings organize screening, decision-level review, reusable findings, and targeted rechecks to reduce review effort per included task. Construction acceptance combines evidence findings, full replay, and specification checks; independent audits of frozen outputs and repair outcomes are separate evaluation activities.

Screening and review. Prechecks validate fields, references, identifiers, tool parameters, and dependencies, then replay from isolated state. Replay tests execution, not evidence support. Review checks public accessibility, entity correspondence, temporal applicability, omitted alternatives, and decision-invalidating interpretations. Reports identify the affected decision, missing support, and operation at risk; conflicts and undecidable cases cannot pass. Appendix C.1 details personnel, independent ratings, and adjudication; generator verdicts are not evidence.

Reuse and repair. Only findings with identical sources, versions, and temporal scopes are reusable; every task still needs decision-level review, and inclusion verdicts never transfer. For changed records and relation endpoints Δ, the recheck set is $\mathcal{R}(\Delta) = \{u \mid \exists x \in \Delta,\ x \rightsquigarrow u\}$, where $x \rightsquigarrow u$ includes equality and direct or transitive dependency, traced through old and new relations. Affected findings are invalidated and tasks suspended. Retained findings require semantic reference checks; incomplete dependencies or uncertain effects trigger full review. The independent audit protocol measures missed propagation and revisions with omissions by change type.

Repair corrects observations, bindings, parameters, and calculations while preserving goals, constraints, decisions, action dependencies, and scope. Shared fixes propagate; full replay rebuilds pre-action histories and requires new evidence before action. Comparing instruction versions with the frozen design checks semantic preservation and leakage. Acceptance requires decision support, no unresolved gaps/conflicts, successful replay, and unchanged requirements/scope. Checker fixes need versioning and regression checks; changed acceptance semantics create a separate benchmark revision.

Accounting and controlled comparison. With a fixed original-target set $\mathcal{T}_0$, total construction-review hours H_{total}, and included targets N_{inc}, define $C_{\text{review}} = H_{\text{total}}/N_{\text{inc}}$ and $\text{Coverage} = N_{\text{inc}}/|\mathcal{T}_0|$. Effort includes failed/unfinished targets and all versions; shared checks count once. Reconstruction retains target identity and cumulative budgets, contributes at most one inclusion per original target, and cannot count as repair of its original scope. Failures remain in the coverage denominator. Model/replay costs and experimental audit time are recorded separately; Section 4.5 reports combined cost per estimated valid task.

The no-binding control matches frozen requirements, initial candidates, resources, access, budgets, and acceptance; reviewers may inspect sources, repair locally, and retain notes, but recheck whole tasks. Independent, blinded audits of frozen outputs and the full repair pool assess validity and repair success without modifying measured versions. Successful repair must resolve a confirmed defect, preserve requirements, and introduce no leakage or validity defect; failed/unfinished repairs remain in the denominator. Appendix C details personnel, versioning, propagation, and complete accounting rules.

3.3 Evaluation Protocol and Reference Agent

We compare ReAct (Yao et al., 2023) with a five-role multi-agent **reference implementation**, sharing public inputs, tools, and isolated states while hiding bindings, trajectories, and answers. **Master** draws on Symphony (Yan et al., 2026) to plan from receipts; **Vision** uses VideoSeek (Lin et al., 2026) for overview/skim/focus; **Tool** validates and executes calls; **Code** filters and computes without changing state; **Reflection** checks evidence and receipts before writes or after failure.

The external scorer accepts any trajectory satisfying goals and execution constraints. SR measures completion, not evidence use or failure causes. Per-backbone budgets are identical and pooled across roles, charging observations, model/code/tool calls, retries, and elapsed time; runs stop at limits and receive final-state scores. Executable references do not establish budget feasibility, nor do over-budget references rule out efficient solutions. Results concern tested limits without assuming every task has a verified budget-feasible path.

4 Experiments

4.1 Evaluation Setup

We evaluate a fixed 3,000-task test set spanning 13 domains and five decision structures. Fine-tuning uses a separately constructed auxiliary set from disjoint source videos, excluding test tasks and their derived trajectories.Agents start from isolated states with source videos, public instructions, and p er mitted tools; private bindings, reference trajectories, and target answers are withheld.This setup se p arates public evaluation inputs from private construction records.

We pair QWEN3.8-VL 27B, QWEN3.5 9B, LFM2.5-VL 3B, and DeepSeek-V4.1 with ReAct and the multi-agent framework (Section 3.3). Comparisons share tasks, public information, tool interfaces, and total token, tool-call, and time budgets, pooled across multi-agent roles. Humans attempt the same full test set with the original videos and public tools. Success requires the final goal and every execution constraint, accepting any valid trajectory. Micro SR aggregates all tasks; Macro SR averages domain rates equally; structure-level rates use tasks within each structure. Appendix records evaluation settings and missing configuration values; Appendices expand Sections 4.3–4.5.

4.2 Main Results

Compared with ReAct, multi-agent SR rises from 11.23% to 17.73% for 27B and from 5.50% to 7.80% for 9B, but falls from 2.60% to 2.20% for 3B. At 27B, Cooking and Museum Visits improve, whereas Fitness Training and Barista Workflow favor ReAct.

Table 2: **Task success rates (%) on 3,000 EvidBind-Bench tasks across 13 domains and four settings.** Ours denotes the multi-agent reference framework. Micro SR aggregates all tasks; Macro SR averages domain rates equally. Public inputs, tools, and budgets are matched as in Section 4.1.

Scenario / domain	Locally deployed models						API model		Human
	QWEN3.8 27B		QWEN3.5 9B		LFM2.5-VL 3B		DeepSeek V4.1		
	ReAct	Ours	ReAct	Ours	ReAct	Ours	ReAct	Ours	
Retail & Service									
Personal Shopping	26.71	16.77	16.15	18.01	6.83	4.35	38.51	36.02	94.41
Shelf Mgmt.	1.49	3.73	1.12	3.73	0.00	0.00	1.12	1.49	90.67
Stock Distribution	6.92	6.92	0.38	2.31	0.38	2.31	1.15	3.08	88.08
Barista Workflow	49.34	35.53	22.37	17.76	22.37	17.76	28.95	17.11	88.16
Technical & Craft									
PC Assembly	2.79	10.06	1.12	1.12	1.12	1.12	1.12	6.70	93.30
Equipment Repair	3.73	11.18	0.62	5.59	0.62	0.00	1.24	1.24	91.30
Model Building	3.26	8.37	0.47	4.65	0.00	0.00	10.23	17.67	89.77
Daily Life & Wellness									
Cooking	3.35	46.84	1.12	5.95	0.74	1.12	12.64	23.05	94.42
Meal Prep.	5.36	16.07	1.79	3.57	0.00	0.89	8.04	8.93	92.86
Fitness Training	33.46	17.31	16.54	20.38	5.38	1.92	10.38	19.62	89.23
Travel & Culture									
Travel Packing	20.50	22.36	19.88	23.60	3.73	5.59	36.65	40.99	95.03
Urban Exploration	3.72	7.44	1.65	3.31	1.65	1.65	0.83	1.24	89.26
Museum Visits	6.47	24.11	2.46	4.02	0.67	0.22	12.28	21.65	94.64
Avg. (Macro SR)	12.85	17.44	6.59	8.77	3.35	2.84	12.55	15.29	91.63
Overall (Micro SR)	11.23	17.73	5.50	7.80	2.60	2.20	11.10	14.90	91.73

Multi-agent benefits depend on the backbone and task (Table 2).

Across structures (Figure 3), 27B multi-agent gains are largest on conditional and sequential tasks: 14.14 and 10.80 percentage points. Multi-hop and plan–execution audit SR remains at 4.78% and 3.30%, respectively.

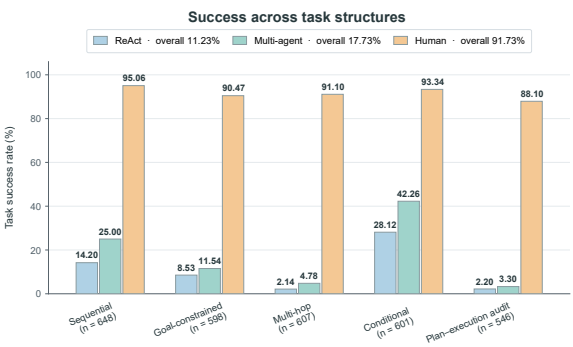

Figure 3: Task success by decision structure for the 27B backbone and humans.

Humans achieve 91.73% overall and 88.10–95.06% across structures, a 74.00-point overall lead over the 27B multi-agent configuration. To contextualize this gap, the appendix reports actual video access, tool calls, model tokens, elapsed time, budget exhaustion, and separate results for tasks with verified budget-feasible reference paths. Trajectory analysis distinguishes evidence-acquisition failures, entity or temporal mismatches, and execution or recovery failures at the first blocking error; budget exhaustion is recorded separately as a termination condition. These records identify cross-time evidence-integration failures in both frameworks. Section 4.3 examines training and evidence assistance.

4.3 Capability Analysis

Development-stage case inspection identifies recovery and dependency failures in ReAct, delegation and completion failures in the multi-agent framework, and repeated unproductive video searches in

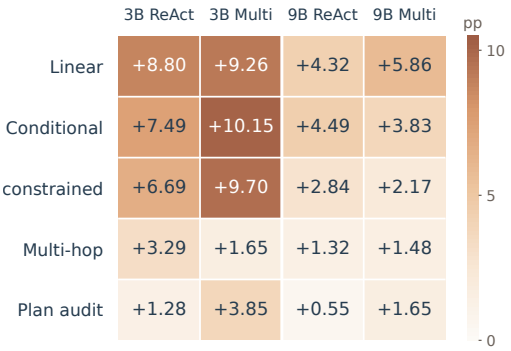

Figure 4: **Corrective fine-tuning gains.** Absolute success-rate changes (pp) by decision structure and model–framework pair.

Table 3: **Independent benchmark validation.** Task audit, scorer calibration, and restricted-access search.

Check	Cases	Rate
Task validity		
All audited tasks	376/390	96.4%
Complex subset	145/156	92.9%
Scorer calibration		
False acceptance	4/300	1.3%
False rejection	8/300	2.7%
Restricted-access search		
Alternative found	19/200	9.5%

Note. Task validity requires public evidence and an executable solution. False acceptance and rejection use invalid and valid executions, respectively.

both. These qualitative observations lack matched pre/post error analysis.

Corrective supervised fine-tuning trains 3B and 9B under both frameworks, including the deployed VideoSeek scheduler, using only auxiliary rollouts (1,200 tasks, 60 videos). Formats, observations, failed calls, and interaction order match deployment; a stronger teacher sees only the student's current public information. Checks validate interfaces/fields, replay executions, and compare terminal tool states with recorded references, without independently verifying task semantics or pre-action evidence support. Appendix details framework-specific objectives and diagnostics.

With identical non-auxiliary test tasks and unchanged budgets, Figure 4 shows gains across all configurations: 4.32–9.26 percentage points for linear, 3.83–10.15 for conditional, 2.17–9.70 for goal-constrained, 1.32–3.29 for multi-hop, and 0.55–3.85 for plan–execution audit tasks. Joint fine-tuning does not isolate search, execution, recovery, or coordination effects. Smaller multi-hop and audit gains motivate recovery-focused preference pairs and state-grounded completion supervision; neither is evaluated here.

4.4 Benchmark Validity

Table 3 audits 390 of 3,000 frozen tasks, uniformly sampled without replacement or domain/complexity quotas. Two auditors uninvolved in construction review independently, with separate adjudication. Unweighted public-evidence and executability pass rates are 376/390 (96.4%) overall and 145/156 (92.9%) for the preclassified multi-hop/plan–execution audit subset. Sampling settings and domain/video counts are retained.

With labels based on task requirements and blinded to scorer verdicts, calibration accepts 292/300 valid executions, including legal alternatives, and rejects 296/300 invalid ones. Conditional false-rejection/acceptance rates are 2.7%/1.3%, not corrections to model SR. Low-SR comparisons additionally require independent review of all accepted executions and 20 random rejections per domain per configuration, with weighted aggregation.

Validity and scoring reliability do not establish cross-time dependence. Budgeted restricted-access searches on 200 prespecified cross-time tasks find alternatives for 17 using short windows (8.5%) and three without video (1.5%): 19 after deduplication (9.5%). For the remaining 181, no alternative is found within budget. These results characterize the tested access conditions and search procedure, without proving temporal necessity.

4.5 Construction and Repair Effectiveness

Table 4 compares the reported costs of per-task review and binding-guided repair, yielding 300 and 3,000 construction-accepted tasks, respectively. These counts describe accepted output, not original-goal-pool sizes, and do not determine coverage. Both procedures use common materials, repair models, and acceptance criteria. The baseline fully rechecks each revised task; our procedure

Table 4: Construction quality, repair effectiveness, and resource use.

Strategy	Scale	Quality & repair		Resource inputs	
		Repair (%)	Valid (%)	Labor (total h)	Model ($/task)
Per-task review	300	86.9	95.8	24.60	0.118
Binding-guided repair	3,000	87.6	96.4	52.50	0.130

Human time is valued at the common rate of $20/h in the combined-cost calculation.

reuses shared findings and targets affected decisions, retaining full replay and acceptance. Independent pre/post repair audits include failures, while separate output audits assess validity. Repair success is 86.9% versus 87.6%, and audited validity is 95.8% versus 96.4%, respectively. These close estimates contextualize resource use without establishing statistical equivalence. Because the correspondence between the runs' original-goal pools is undocumented, this comparison assesses unit costs at unequal output scales, rather than coverage gains under a verified common-pool, matched-budget design.

Following Section 3.2, we measure combined cost per estimated valid task. Let N_s denote construction-accepted output, H_s total human hours, M_s model/computation charges, and $\hat{q}_s$ independently audited whole-task validity for strategy s. At a common hourly labor rate w,

$$\widehat{C}_s^{\text{valid}} = \frac{wH_s + M_s}{N_s \hat{q}_s} = \frac{w\, C_{\text{review},s} + \bar{m}_s}{\hat{q}_s}, \quad C_{\text{review},s} = \frac{H_s}{N_s}, \quad \bar{m}_s = \frac{M_s}{N_s}. \tag{2}$$

The numerator includes all construction effort, including failed candidates and repeated revisions; experimental audits are accounted for separately. The denominator estimates valid output rather than independently certifying every accepted task. Repair success remains separate because repaired tasks already contribute to N_s.

At an hourly labor rate of $20, human time falls from 4.92 to 1.05 minutes per accepted task (78.66%), while model/computation cost rises from $0.118 to $0.130 (10.17%). Using the unrounded audit fractions $115/120$ and $376/390$, the combined cost per estimated valid task decreases from $1.8344 with full per-task review to $0.4979 with binding-guided review, representing a 72.86% reduction.

Reduced labor per accepted task outweighs higher model cost; total labor increases at the larger scale. The measured unit-cost reduction includes shared-check amortization at unequal scales, not an isolated equal-scale effect of binding. Appendix gives the reported records and unreported fields.

5 Discussion and Limitations

EvidBind-Bench separates task executability from public-evidence support for required decisions. Specialized roles can improve execution, but benefits vary across backbones and domains; success rates alone establish neither evidence use nor failure causes. Independent audits estimate residual task and scoring defects rather than certify every item. Short-window and no-video searches can reveal alternative solutions, but unsuccessful searches under fixed budgets do not prove temporal necessity. Our conclusions are limited to the evaluated videos, tools, models, and budgets; corrective fine-tuning results apply only to the tested 3B and 9B configurations under ReAct and the multi-agent framework.

6 Conclusion

We introduced EvidBind-Bench, comprising 3,000 tasks across 13 domains grounded in hour-scale videos. Its design–binding procedure fixes task requirements, links required decisions to scene evidence, and uses those links to guide review and repair. Evaluation with ReAct and multi-agent reference agents reveals a substantial gap in end-to-end task completion relative to human performance. The benchmark provides a basis for diagnosing this execution gap and improving the reliability of long-horizon assistants.

REFERENCES

Tajamul Ashraf, Amal Saqib, Hanan Gani, Muhra AlMahri, Yuhao Li, Noor Ahsan, Umair Nawaz, Jean Lahoud, Hisham Cholakkal, Mubarak Shah, Philip Torr, Fahad Shahbaz Khan, Rao Muhammad Anwer, and Salman Khan. Agent-X: Evaluating deep multimodal reasoning in vision-centric agentic tasks. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/51e8b7b635204ca57f3252b7450c6cec-Abstract-Conference.html.

Yang Dai, Dian Jiao, Tianwei Lin, and Wenqiao Zhang. EgoCoT-Bench: Benchmarking grounded and verifiable operation-centric chain of thought reasoning for MLLMs. *arXiv preprint arXiv:2605.19559*, 2026. doi: 10.48550/arXiv.2605.19559. URL https://arxiv.org/abs/2605.19559.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jian-Guang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. AgentGen: Enhancing planning abilities for large language model based agent via environment and task generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 496–507. Association for Computing Machinery, 2025. doi: 10.1145/3690624.3709321. URL https://doi.org/10.1145/3690624.3709321.

Lawrence Jang, Yinheng Li, Dan Zhao, Charles Ding, Justin Lin, Paul Pu Liang, Rogerio Bonatti, and Kazuhito Koishida. VideoWebArena: Evaluating long context multimodal agents with video understanding web tasks. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/5b555804d495321df2e3208cc27f4fbc-Abstract-Conference.html.

Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, Junteng Liu, Zhaochen Su, Yiyang Guo, Fan Zhou, Lueyang Zhang, Juan Michelini, Xingyao Wang, Xiang Yue, Shuyan Zhou, Graham Neubig, and Junxian He. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.25726.

Jingyang Lin, Jialian Wu, Jiang Liu, Ximeng Sun, Ze Wang, Xiaodong Yu, Jiebo Luo, Zicheng Liu, and Emad Barsoum. VideoSeek: Long-horizon video agent with tool-guided seeking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2603.20185.

Yunqi Liu, Tong Niu, Zitong Wang, Zhenlong Dai, Yuqi Qing, Weiqiang Wang, and Jian Liu. EgoBench: An interactive egocentric multimodal benchmark for tool-using agents. *arXiv preprint arXiv:2605.27820*, 2026. doi: 10.48550/arXiv.2605.27820. URL https://arxiv.org/abs/2605.27820.

Lu Qiu, Yi Chen, Yuying Ge, Yixiao Ge, Ying Shan, and Xihui Liu. EgoPlan-Bench2: A benchmark for multimodal large language model planning in real-world scenarios. *International Journal of Computer Vision*, 134, 2026. doi: 10.1007/s11263-026-02826-y. URL https://link.springer.com/article/10.1007/s11263-026-02826-y. Article 222.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. TaskBench: Benchmarking large language models for task automation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0148. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/085185ea97db31ae6dcac7497616fd3e-Abstract-Datasets_and_Benchmarks_Track.html.

Dingfeng Shi, Jingyi Cao, Qianben Chen, Weichen Sun, Weizhen Li, Hongxuan Lu, Fangchen Dong, Tianrui Qin, King Zhu, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Ge Zhang, Jiaheng Liu, Changwang Zhang, Jun Wang, and Wangchunshu Zhou. TaskCraft: Automated generation of agentic tasks. In *International Conference on Learning Representations*, pp.

43714–43734, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/48644509339cb3076f7b0407c7588af6-Abstract-Conference.html.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A benchmark for long-context interleaved video-language understanding. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/329ad516cf7a6ac306f29882e9c77558-Abstract-Datasets_and_Benchmarks_Track.html.

Haiyang Yan, Hongyun Zhou, Peng Xu, Xiaoxue Feng, and Mengyi Liu. Symphony: A cognitively-inspired multi-agent system for long-video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24031–24041, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Yan_Symphony_A_Cognitively-Inspired_Multi-Agent_System_for_Long-Video_Understanding_CVPR_2026_paper.html.

Jingkang Yang, Shuai Liu, Hongming Guo, Yuhao Dong, Xiamengwei Zhang, Sicheng Zhang, Pengyun Wang, Zitang Zhou, Binzhu Xie, Ziyue Wang, Bei Ouyang, Zhengyu Lin, Marco Cominelli, Zhongang Cai, Bo Li, Yuanhan Zhang, Peiyuan Zhang, Fangzhou Hong, Joerg Widmer, Francesco Gringoli, Lei Yang, and Ziwei Liu. EgoLife: Towards egocentric life assistant. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 28885–28900, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yang_EgoLife_Towards_Egocentric_Life_Assistant_CVPR_2025_paper.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024. doi: 10.48550/arXiv.2406.12045. URL https://arxiv.org/abs/2406.12045.

Shoubin Yu, Lei Shu, Antoine Yang, Yao Fu, Srinivas Sunkara, Maria Wang, Jindong Chen, Mohit Bansal, and Boqing Gong. Ego2Web: A web agent benchmark grounded in egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2603.22529.

Fan Zhang, Guangming Yao, Jinyang Wu, Hao Wu, Zheng Lian, Xinyu Geng, Jingdong Chen, Yi Yuan, and Pheng-Ann Heng. VideoGAIA: A benchmark for general AI assistants on agentic video understanding. *arXiv preprint arXiv:2608.14718*, 2026. doi: 10.48550/arXiv.2608.14718. URL https://arxiv.org/abs/2608.14718.

AI USE STATEMENT

In this work, we used generative AI tools solely for language polishing and grammar checking of the manuscript. All technical content, research ideas, methodology, experiments, analyses, and conclusions were developed and verified by the authors. We reviewed all AI-assisted edits and take full responsibility for the final content of this work.

ETHICS STATEMENT

This work does not involve the recruitment of human subjects or the collection of new personally identifiable or private data.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the proposed method, experimental settings, datasets, and evaluation protocols in the main paper and appendix. Additional implementation details and hyperparameters are included in the supplementary materials. Source code and configuration files will be released to support reproducibility.

A BENCHMARK DATA AND TASK SPECIFICATIONS

This appendix provides additional details on the construction and evaluation of EvidBind-Bench. We elaborate on task specifications, evidence bindings, and the review and repair procedure. We also describe the reference-agent configurations and experimental protocols. These details complement the main text and support reproducibility and further analysis.

A.1 VIDEO SOURCES, FILTERING, AND DATA SPLITS

The test collection contains 3,000 tasks across 13 domains and five decision structures, grounded in hour-scale videos. Auxiliary fine-tuning uses 1,200 tasks from 60 videos, with source videos disjoint from test videos; test tasks and their derived trajectories are excluded from fine-tuning. Test-run feedback nevertheless informed development, so this collection is not an untouched holdout. No separate development split is specified.

Table 5: **Dataset specification and split protocol.** Video-duration statistics count source videos, not tasks or execution time.

Field	Auxiliary training set	Test set
Task count	1,200	3,000
Source-video count	60	120 (from the overview histogram)
Domains / primary structures	13 / 5	13 / 5
Total source-video duration	Approximately 61 h (reference setting)	Approximately 122.6 h (histogram estimate)
Video counts by duration: $(15, 30]$ / $(30, 60]$ / $(60, 120]$ min	3 / 26 / 31 (reference distribution)	6 / 53 / 61
Source categories	Instructional and everyday-activity videos across the four settings	Instructional and everyday-activity videos across the four settings
Acquisition and release records	Record source URL, creator, access date, and applicable license or permission for each video. Release only materials permitted by the source-specific terms.	
Video screening: considered / included / excluded	80 / 60 / 20	160 / 120 / 40
Task screening: candidates / included / excluded or unresolved	1,500 / 1,200 / 300	3,600 / 3,000 / 600
Per-video task counts and instance IDs	Mean: 20.0; proposed IDs: `tr-vNNN-tNNN`	Mean: 25.0; proposed IDs: `te-vNNN-tNNN`
Split version and manifests	Proposed: `aux-v1`; task/video manifests and file hashes	Proposed: `test-v1`; task/video manifests and file hashes
Source overlap and near-duplicate control	Group identical sources, overlapping excerpts, and reuploads before splitting. Use source-ID matching, file hashes, visual similarity screening, and manual confirmation of suspected duplicates. Assign each source group to one split only.	

A.2 DOMAIN AND DECISION-STRUCTURE DEFINITIONS

Recording a purchase and adding a part to inventory may share the pattern "identify an entity, query its identifier, update a record." We represent the underlying task design as

$$d = (g, c, Q, A), \tag{3}$$

where g is the target state, c contains necessary constraints, Q contains entity, attribute, branch, and parameter decisions that require scene information, and A encodes their informational, conditional, and state dependencies on tool actions. Using the construction records of exemplar tasks, the goal, and the tool interface, the generator first identifies required state changes and choices, then records their dependencies. At this point it does not fill in answers or evidence locations from the target

video. With d frozen, search jointly extends scene bindings and a public acquisition–action prefix, tracking its history and tool state under A.

Our five representative topologies describe **sequential progression, conditional branching, multi-hop dependencies, joint constraints, and plan–execution comparison**. They are composable patterns, not an exhaustive taxonomy. Sequential tasks progress through required operations; conditional tasks select an action branch from observed conditions; multi-hop tasks resolve a choice through dependent information lookups; joint-constraint tasks select a candidate satisfying multiple independent requirements; and plan–execution tasks compare intended with observed outcomes. Routine tool preconditions alone do not establish joint-constraint selection, nor does passing a record identifier alone establish multi-hop reasoning.

Each design retains all applicable topology tags. For mutually exclusive counts and performance comparisons, we assign one primary label before evaluation using the first applicable pattern in a fixed order: plan–execution comparison, joint constraints, conditional branching, multi-hop dependencies, and sequential progression. Applicability is determined by the public goal and constraints, rather than incidental steps in the reference trajectory. Remaining patterns are retained as secondary tags, and evidence repair does not change the primary label. Domain assignment likewise follows the requested terminal outcome: *Cooking* concerns producing a dish through recipe execution, whereas *Meal Prep.* concerns preparing, portioning, packing, or storing ingredients or meals for later use. Mixed workflows are assigned according to that terminal outcome.

Transfer to a new scene reuses decision roles and dependencies, not entity identities, answers, or timestamps. Each instantiation also fixes its source video, target event, and queryable candidate set. Missing evidence cannot be addressed by switching to an easier event, inventing a new candidate, dropping a required decision, or weakening a constraint. A *required decision* follows from the public goal and constraints; it does not include every intermediate step in the constructor's reference trajectory.

Table 6: **Decision structures and their boundaries.**

Structure / display variant	Defining criterion	Boundary / illustrative example
Sequential / Linear	Required operations progress in order.	Identify the purchased item, retrieve its catalog ID, and record the purchase.
Conditional	Observed conditions select a task-required action branch.	Replenish below the specified stock threshold; otherwise leave stock unchanged. An unknown stock level does not justify either branch.
Multi-hop	Dependent information lookups resolve a choice.	Passing a record identifier alone is insufficient; information obtained earlier must help resolve a later choice.
Joint constraints / Goal-constrained	Select a candidate meeting multiple independent requirements.	Routine tool preconditions alone are insufficient; selection must jointly satisfy task-level requirements.
Plan–execution comparison / Plan audit	Compare intended with observed outcomes.	Compare a planned shopping list with video-confirmed purchases and report missing or substituted items.

Structures may co-occur. Following Section 3.1, the primary label uses the fixed priority: plan audit, joint constraints, conditional, multi-hop, then sequential. Applicability follows the public task requirements, not incidental steps in a reference trajectory.

The reported domains are Personal Shopping, Shelf Mgmt., Stock Distribution, Barista Workflow, PC Assembly, Equipment Repair, Model Building, Cooking, Meal Prep., Fitness Training, Travel Packing, Urban Exploration, and Museum Visits (Table 2). Beyond the Cooking/Meal Prep. distinction above, complete domain-specific inclusion/exclusion rubrics and boundary examples are –. The

validity audit defines its complex subset as tasks labeled multi-hop or plan–execution audit before auditing; whether this selection uses only primary labels or all tags is –. The separate cross-time analysis uses 200 prespecified tasks; its exact selection rule and manifest are –. A multi-hop label therefore does not by itself establish membership in that analysis.

A.3 Public and Private Task Schemas

The scene input also includes a queryable catalog, tool interfaces, and an isolated initial task state. Video records events that occurred; the catalog supplies queryable attributes and may include candidates not observed in the video. Their provenance labels are retained for construction and review but are not returned by agent-facing queries, which do not directly identify the video answer. Tools query information or update task state. Facts, the catalog, and the tool environment undergo separate review; video facts reserved for construction are not exposed through agent-facing tools.

The generator packages the frozen design, bindings, and recorded public acquisition–action path into a **public task instruction, a reference execution trajectory, and a goal-checking specification**. The reference serializes the path executed during search rather than assembling it afterward from evidence alone. Instructions preserve the public access assumptions used in search; changes requiring different clues or operations return to Step (3). Bindings, reference steps, and answers remain private. The checker tests final goals and execution constraints, allowing alternative valid paths. Candidates enter Section 3.2 for review and replay; repairs preserve the original requirements.

A.4 Worked Task Examples

The manuscript provides a conceptual milk-purchase example with the observations and frames shown in Figure 2, but not a complete released task trace. Its public goal is to buy one carton of the milk product ultimately purchased in the video and record the transaction ID from the new purchase in the task environment. The displayed constraints require self-checkout and the correct item and quantity. The three displayed decisions ask which milk was purchased, which checkout lane to use, and which transaction ID to record. The example illustrates why readable early labels and checkout frames may be insufficient without the return-and-replacement link.

Consider a checkout task involving milk. An early observation may establish that a whole-milk carton was picked up, but not that it was ultimately bought. If it was returned and replaced, recording the purchased item requires a supported link among the replacement event, the item identifier, and the carton at checkout. When both cartons are represented within the declared scope, a missing identity or temporal link directs acquisition to the relevant clips. An omitted decision-changing carton instead invalidates the scope and requires reconstruction.

B Task Construction Algorithms and Implementation

This appendix specifies the construction protocol reported in Section 3. Configuration gaps are marked explicitly; pseudocode restates the described procedure and does not certify an unseen implementation.

B.1 Multigranularity Evidence Acquisition

Initial acquisition uses coarse observations to locate candidate entities and events, then examines relevant context and fine details. Each observation records its source, candidate entities, event time, and state claim. Unclear observations remain leads rather than established facts. In a purchase-recording task, for example, a readable milk label and a clear checkout frame may still be insufficient if the brief return-and-replacement event between them is missed.

After Step (2) freezes the design and Step (3) produces preliminary bindings, a decision q without adequate support triggers another acquisition round:

$$\mathcal{O}^{(t+1)} = \mathcal{O}^{(t)} \cup \mathrm{Collect}\Big(\delta_q, \mathcal{C}_q \mid \mathcal{O}^{(t)}, B_t\Big). \tag{4}$$

Here, $\mathcal{O}^{(t)}$ is the construction observation cache, δ_q identifies an identity, state, or temporal gap, $\mathcal{C}_q$ lists candidates to distinguish, and B_t is the remaining observation budget. $\mathrm{Collect}$ revisits records, then inspects identifiers, cross-clip links, state transitions, or relevant intervals. Cached observations guide binding proposals; cache membership neither verifies them nor places them in the current execution history h_k. A binding selects supporting sources E_q from observations and permitted catalog results, whose acquisition is checked in Step (3).

Acquisition need only resolve ambiguity that could make an action fail a fixed decision requirement, including its branch or parameter: two cartons with the same item identifier need not be distinguished if the required quantity is clear, whereas different identifiers require further evidence. Initial observations, follow-up acquisition, and failed attempts all count against the budget. An unresolved relation remains unresolved when the budget expires, and uninspected video remains unknown.

Table 7: Evidence-acquisition settings and reported rules.

Field	Value or rule
Initial observation order	Coarse localization, relevant context, fine details.
Initial sampling interval/count; frame resolution	24 sparse samples for coarse localization outside the initial event anchors; targeted inspection at 1 fps. Frames are resized to a maximum long edge of 768 pixels, preserving aspect ratio without upscaling.
Numerical observation-time / window limits	900 s wall-clock per scene-preparation batch, including indexing, extraction, validation, and review-record preparation. At most 2 supplemental windows outside the initial anchor scope, totaling at most 60 s of source video. Required anchors are retained; excessive scope is split into versioned batches.
No new accessible observation	Try the next alternative when the current queue contains no untried, permitted acquisition request within the remaining budget. Requests are deduplicated by source version, timestamps, sampling settings, resolution, and crop; an exhausted queue does not establish absence of evidence.

B.2 Search State, Expansion, and Rollback

Search control. Each node stores partial bindings, a public acquisition–action prefix, history h_k, isolated tool state, checkpoints, and alternatives. Select the first unresolved q in design order enabled by A under that prefix and state. Bounded, deduplicated proposals pair bindings with access/action extensions, including prerequisite and terminal operations; screen detectable access, scope, and tool violations. Rank by recorded gaps, then added access operations, breaking ties by proposal order; ranking does not establish $\mathrm{Support}$. Execute accesses justified by current public history; append actual observations and receipts to h_k. Check proposed support against h_k and tool preconditions against state before actions; record receipts and update state. Unresolved support triggers acquisition, not action execution; screening remains provisional pending Section 3.2. For a gap, $\mathrm{Collect}$ targets the first recorded gap and reranks updated alternatives; no new accessible observation triggers the next alternative. Scope omissions invalidate the instance and require reconstruction. On a binding or execution conflict, roll back to the latest implicated decision checkpoint: truncate the prefix, restore tool state, rebuild h_k from the retained prefix, invalidate bindings dependent on discarded history or state, then try its next alternative. Cached observations guide proposals; discarded observations must be reacquired and state-dependent queries rerun through publicly justified access before entering the new h_k. Empty lists trigger another bounded batch. All branches share caps on model tokens, proposals, observation time, and tool calls; generation, acquisition, execution, failures, and replay consume them without refunds. Search stops unresolved when alternatives and proposal allowance are exhausted, or the next operation exceeds its remaining budget. A complete, gap-free assignment whose executed path satisfies the goal and encoded constraints proceeds to Step (4) for packaging and review, not acceptance.

Protocol-level pseudocode. The following algorithm rewrites the search-control paragraph above without introducing numerical settings, candidate heuristics, or implementation claims not supplied by the manuscript.

1. Initialize a node with partial bindings, a public acquisition–action prefix, its actual history h_k, isolated tool state, checkpoints, and alternatives. Maintain the construction cache separately from this history.
2. Select the first unresolved decision in design order that dependencies A enable under the current prefix and state. Request a bounded proposal batch when its alternative list is empty and proposal allowance remains.
3. Pair bindings with access/action extensions, including prerequisite and terminal operations. Deduplicate and screen detectable access, scope, and tool violations. Rank by recorded gaps, then additional access operations, then proposal order; rank is not support.
4. Execute only accesses justified by the current public history and append actual observations and receipts. Before actions, check proposed support against h_k and tool preconditions against the isolated state. Record action receipts and state updates.
5. If support remains unresolved, do not execute the action. Apply $\mathrm{Collect}$ to the first recorded gap and rerank updated alternatives; if no new accessible observation results, try the next alternative.
6. If a decision-changing scope omission is found, invalidate the current instance and record reconstruction separately under its original target and remaining cumulative allowance.
7. For a binding or execution conflict, roll back to the latest implicated checkpoint: truncate the prefix, restore state, rebuild history from the retained prefix, invalidate dependent bindings, and try the next alternative. Reacquire discarded observations and rerun state-dependent queries through public access before adding them to the new history.
8. Charge all generation, observation, execution, failures, and replay to the shared caps without refunds. Stop unresolved if alternatives and proposal allowance are exhausted or the next operation exceeds a remaining cap.
9. A complete, gap-free assignment with an executed path satisfying the goal and encoded constraints proceeds to compilation and review; it is not yet construction-accepted.

Table 8: Search-state distinctions required by the reported procedure. Concrete storage formats are not supplied.

State / outcome	Meaning and handling
Partial bindings / alternatives	Proposed support, unresolved gaps, and candidates; bounded deduplication and ranking.
Public prefix and h_k	Accesses/actions actually executed and the public results they obtained before the current action.
Observation cache $\mathcal{O}^{(t)}$	Proposal material; membership does not validate a claim or insert it into h_k.
Isolated state and checkpoints	Tool state updates; rollback restores an implicated checkpoint and invalidates dependent bindings.
Search or budget exhaustion	Unresolved; no action is justified by exhaustion.
Scope omission	Instance invalid; separately versioned reconstruction, not successful repair.
Complete executed candidate	Package and review; acceptance still requires evidence findings, full replay and specification checks.

B.3 Scope Adequacy and Conditional Applicability

An executable tool trajectory does not establish that its entity, time, and parameter choices are justified by publicly obtainable information. Whereas TaskCraft (Shi et al., 2026) organizes tasks around tool dependencies and EgoCoT-Bench (Dai et al., 2026) organizes reasoning evidence with scene graphs, our unit of binding and repair is the **required decision fixed before scene instantiation**. Each $q \in Q$ specifies what must be determined, when it applies, and which action branch or parameter it affects. It is not synonymous with a step in the reference trajectory, and different valid

Table 9: **Per-task resource use.** Resource columns represent averages over all task attempts; the final column reports the percentage of budget-terminated runs.

Model	Framework	Video (min)	Tool calls	Tokens (k)	Time (min)	Exhausted (%)
QWEN3.8-VL 27B	ReAct	6.8	18.4	14.8	8.4	18.0
	Multi-agent	8.3	26.7	23.6	12.8	27.0
QWEN3.5 9B	ReAct	5.9	16.8	12.4	5.8	14.0
	Multi-agent	7.6	25.2	21.7	9.1	24.0
LFM2.5-VL 3B	ReAct	4.6	12.3	9.6	3.7	8.0
	Multi-agent	6.1	19.8	17.8	6.4	16.0
DeepSeek-V4.1	ReAct	7.2	20.5	16.2	7.1	16.0
	Multi-agent	8.8	29.1	25.4	10.6	29.0
Human	–	10.4	14.6	–	16.2	2.0

Accounting. Video access sums the durations of accessed clips, including repeated accesses; it is distinct from source-video duration and wall-clock time. Tool calls include failed calls and retries. Tokens sum input and output across model calls, using each model's tokenizer; $1\text{k} = 1{,}000$ tokens. Multi-agent totals include all roles. Time measures elapsed wall-clock time. Exhaustion counts runs terminated by any applicable budget limit, with each run counted once.

execution paths may satisfy the same decision requirement. For each q, the generator proposes a binding

$$b_q = (E_q, \mu_q, J_q, \eta_q),$$

where E_q records public sources and permitted access paths; μ_q records entity correspondences and unresolved candidates; J_q records event times or intervals over which state claims hold; and η_q specifies the action condition that the evidence must support. In plain terms, a binding states **what was observed, which entity it concerns, when it applies, and which choice it justifies**. An evidence-access path cannot depend in advance on the answer or private annotations, and an action condition cannot be weakened merely to fit the available evidence.

Candidate scope is checked separately from decision support. Before binding search, the generator assembles a proposed scope from the public task restrictions, permitted catalog queries, and scene observations, retaining alternative identities, states, and event links that could change a required choice. This scope is fixed for the current instance. The catalog alone is not a complete inventory of video entities: a relevant observed object without a catalog match remains an unmatched alternative, and catalog absence cannot justify excluding it. In Section 3.2, reviewers inspect task-relevant source intervals and query results for omitted alternatives and unjustified exclusions as part of the existing decision-level review. Exclusions require cited evidence contained in the affected action's pre-action history, or an applicable public domain or tool rule; a proposed match or private annotation is insufficient. Uninspected intervals remain unknown; if they could contain a decision-changing alternative, scope adequacy remains unresolved. A discovered omission invalidates the current instance. Expanding the frozen scope requires a separately versioned reconstruction under the same goal and constraints, not a successful repair of the original instance. The failed instance and its incurred cost remain in the construction records.

We judge support using information available *before* an action. Let h_k contain the public task instruction, video observations, and tool results actually obtained before step k of a candidate execution. Let $\mathcal{W}_k$ be the interpretations within the declared candidate scope that are consistent with h_k and the domain and tool rules. Finally, let $\chi_q(a, w)$ indicate whether action a satisfies the fixed requirement for decision q under interpretation w. We define

$$\text{Support}(q, a \mid h_k) \iff \mathcal{W}_k \neq \emptyset \ \wedge \ \forall w \in \mathcal{W}_k, \ \chi_q(a, w). \tag{5}$$

The index k denotes an execution step, not video time. This definition establishes support *within the declared scope*; it does not establish that the scope is adequate. Construction acceptance requires both a separate scope-review finding and support for each required decision. Support need not recover a unique world state, and several actions may be valid. An empty $\mathcal{W}_k$ signals conflicting information; undecidable support remains unresolved, and later evidence cannot retroactively

justify an earlier action. This is a construction-screening criterion, not an online computation required of evaluated agents. Scope review is a bounded check of task-relevant sources, not a proof of completeness over the entire video.

Unspecified branch rules. The supplied method defines conditional dependencies and selects enabled decisions during search, but it does not specify an explicit N/A mechanism for inactive branch decisions. This appendix does not add an exemption to the requirement that every required decision have affirmative support. The following unresolved fields distinguish a missing operational rule from an actual N/A judgment.

B.4 Construction Models, Prompts, and Hyperparameters

These are construction settings, distinct from the evaluated backbones, fine-tuning teacher, and agent-execution budgets. The manuscript describes a generator, Collect, binding search, compilation, and repair, but does not provide the exact model configuration or prompt artifacts for these stages.

C Review, Repair, and Versioning Protocols

C.1 Review Rubrics and Personnel

The objective is to reduce human review effort **per included task** while preserving the original requirements and inclusion standard. Bindings organize procedural screening, decision-level review, shared findings, and targeted rechecks. *Construction acceptance* combines current evidence-review findings with full-task replay and specification checks; it does not routinely add an independent human re-review of every candidate. Independent audits of frozen outputs and repair outcomes belong to Section 4.

Prechecks validate fields, references, catalog identifiers, tool parameters, and dependencies, then replay the reference trajectory from an isolated state. Replay tests execution, not evidential support. Decision reviewers check public source accessibility, entity correspondence, temporal applicability, and whether remaining interpretations could invalidate a required choice. Conflicting or undecidable cases remain unresolved. Failure reports identify **the affected decision, missing support, and operation at risk**.

Shared-source checks, decision reviews, and repair rechecks use 2, 1, and 3 reviewers per claim, required decision, and repaired candidate, respectively, with personnel overlap recorded. Source reviewers assess reusable claims; decision reviewers assess task-specific links; repair reviewers compare changed links and requirements. Bindings and earlier source findings are inspectable, but generator verdicts are not evidence. Multiple reviewers record judgments independently before disagreements are adjudicated; unresolved cases cannot pass.

Table 10: Reviewer responsibilities reported by the supplied method. Auditor numbers follow the latest manuscript: two independent ratings and separate adjudication.

Review	Object / material	Judgment	Ratings
Shared source	Reusable source claim, source/version/time scope	Whether the claim is supported and reusable within that scope	2 per claim
Decision	Required decision; accessible sources, bindings and actual public history	Entity, time, operational support, remaining alternatives and scope adequacy	1 per decision
Repair recheck	Pre/post versions and affected links	Defect resolution, requirement preservation and absence of new leakage/validity defects	3 per candidate
Independent audit	Frozen outputs and repair samples; method/verdict hidden	Final validity or repair outcome; repair versions visible	2 per sample
Adjudication	Independently recorded disagreements	Additional reviewer resolves; unresolved cases cannot pass	Additional reviewer

Blank review-record fields. Claim/task/decision identifier; source/version/temporal scope; public access path; entity match; pre-action support finding; omitted-alternative check; affected operation; reviewer identifier and overlap; independent verdict; disagreement; adjudication; unresolved reason; and final status are the conceptual fields implied by the protocol. Exact released form, personnel identities/qualifications, agreement statistics, and completed anonymous examples are –. Generator verdicts cannot substitute for source evidence, and unchanged fields cannot establish semantic preservation.

C.2 Dependency Propagation and Recheck Rules

Tasks may reuse a fact-review finding only for the same source, version, and temporal scope; each task still requires its own decision-level support check. A task's inclusion verdict is never reused. Let Δ contain changed records and endpoints of changed relations, and write $x \rightsquigarrow u$ when $u = x$ or u depends directly or indirectly on x. The dependency-based recheck set is

$$\mathcal{R}(\Delta) = \{u \mid \exists x \in \Delta,\ x \rightsquigarrow u\}. \tag{6}$$

Tracing uses the union of old and new relations. Findings in $\mathcal{R}(\Delta)$ are invalidated, and affected tasks are suspended until evidence review, replay, and specification checks pass again. Coverage depends on recorded dependencies and does not guarantee that all semantic effects are captured. Retaining other findings therefore requires checking source, entity, time, and parameter references against the revision, not merely unchanged graph edges. Incomplete dependencies or uncertain impact trigger full semantic review of potentially affected tasks.

The reported audit protocol separately audits sampled fact, entity, and temporal revisions: independent full-task reviewers identify decisions requiring re-review without seeing $\mathcal{R}(\Delta)$. Missed propagation is the fraction of these decisions absent from $\mathcal{R}(\Delta)$. The protocol also records revisions with any omission, by change type.

The manuscript notes that changed object correspondences can affect quantities, amounts, and state updates. A concrete propagation example would record the changed source claim, old/new entity relation, dependent calculation, affected decision IDs, suspended task IDs, invalidated findings, and repeated review/replay/specification checks. The corresponding instance-level identifiers and recorded outcomes are –; this is a reporting schema, not an additional observed case. The propagation protocol here is distinct from the empirical omission counts, which require actual independent audit records. These counts were not supplied and are marked – in Appendix **??**.

C.3 Repair, Reconstruction, and Checker Versioning

Failure reports guide additional observation, corrected entity or time bindings, and revised parameters or downstream calculations. Shared-source errors are corrected before propagating their effects. Repairs preserve the frozen goal, constraints, required decisions, action dependencies, and candidate scope. Affected links return to Step (1), and new evidence must be publicly obtainable *before* the supported action. Full replay from the isolated initial state rebuilds pre-action histories h_k and checks the resulting goal and constraints.

Repair rechecks compare both public instruction versions against the frozen design, including omitted restrictions, easier substitute goals, and answer hints; unchanged fields alone do not establish semantic preservation. Goal predicates and execution-constraint semantics remain fixed throughout the comparison. Checker implementation fixes require versioning and regression checks against those semantics; changed acceptance semantics constitute a separate benchmark revision.

Construction acceptance requires affirmative support findings for every required decision, no unresolved evidence gap or conflict, successful replay, and preserved public requirements and candidate scope. The no-binding control checks the same support conditions through full per-task review and whole-task rechecks after revision. It may retain notes; missing binding fields alone do not cause rejection. We retain each repair candidate's initial version, defects, revisions, human time, and final status, including failures.

All reconstructed versions retain the original target identifier, frozen design, source video, and target event. They share the original target-level cumulative construction allowance, and all incurred costs remain charged to that target; reconstruction does not reset or replenish the allowance. At most one included version per original target contributes to N_{inc}. Reconstruction outcomes are reported separately from repair success under an unchanged candidate scope.

Table 11: Edit boundaries stated by the method. Unspecified answer-value rules are not supplied by inference.

Change	Required treatment
Additional observation; corrected entity/time binding; revised parameter or downstream calculation	Repair is permitted while frozen requirements, action dependencies and candidate scope remain intact; re-review affected links and replay the complete task.
Shared-source error	Correct shared source first, propagate effects, suspend affected tasks until checks pass.
Expanded candidate scope	Invalidate the instance; separately versioned reconstruction under the original target, design, video, event and cumulative allowance.
Omitted public constraint / easier substitute goal / answer hint	Violates semantic preservation; cannot count as successful original-task repair.
Checker implementation fix	Version and regression-check against fixed goal and execution-constraint semantics.
Changed acceptance semantics	Separate benchmark revision.
Correction of a concrete target ID derived from an incorrect binding	Exact permitted rule and before/after example: –.

Goal predicates and answer values. The reported rule fixes goal predicates and execution-constraint semantics, and distinguishes implementation fixes from changed acceptance semantics. It does not fully specify whether every instantiated answer value is frozen, nor how a wrongly bound target identifier may be corrected without changing that predicate. Those operational rules, concrete checker versions, and paired examples are –. No additional permission to change answers or relax goals is introduced here.

C.4 Construction Records and Accounting Units

Let $\mathcal{T}_0$ be the set of original targets specified before construction begins, with $N_{\mathrm{tar}} = |\mathcal{T}_0| > 0$. Let H_{total} denote construction review time across all targets in $\mathcal{T}_0$, and let N_{inc} count original targets

Table 12: **Domain-level success rates after fine-tuning .**

Scenario / domain	N	LFM2.5-VL (3B)		QWEN3.5 (9B)	
		ReAct	Ours	ReAct	Ours
		Retail & Service			
Personal Shopping	161	18.01	16.77	21.74	22.98
Shelf Mgmt.	268	1.87	2.24	2.24	6.34
Stock Distribution	260	3.08	10.38	1.54	3.85
Barista Workflow	152	43.42	40.13	29.61	23.03
		Technical & Craft			
PC Assembly	179	5.59	7.26	2.23	2.23
Equipment Repair	161	4.35	1.86	1.86	8.07
Model Building	215	1.40	1.40	1.86	7.44
		Daily Life & Wellness			
Cooking	269	5.20	8.18	2.60	9.29
Meal Prep.	224	1.34	7.14	4.02	6.70
Fitness Training	260	15.77	10.77	22.69	25.38
		Travel & Culture			
Travel Packing	161	13.04	19.25	26.71	29.19
Urban Exploration	242	7.85	9.50	3.72	5.37
Museum Visits	448	4.69	3.57	4.46	6.25
Avg. (Macro SR)	—	**9.66**	**10.65**	**9.64**	**12.01**
Overall (Micro SR)	**3,000**	**8.23**	**9.20**	**8.27**	**10.87**

SR: success rate; N: number of tasks.

with a version meeting the inclusion conditions at the reporting cutoff. Mean human review cost is

$$C_{\text{review}} = \frac{H_{\text{total}}}{N_{\text{inc}}}, \qquad N_{\text{inc}} > 0. \tag{7}$$

Coverage is computed over the fixed original target set:

$$\text{Coverage} = \frac{N_{\text{inc}}}{N_{\text{tar}}}. \tag{8}$$

Failed and unfinished targets remain in $\mathcal{T}_0$ and in the denominator at the reporting cutoff. Repaired and reconstructed versions do not create new targets or alter this denominator.

Time includes source checks, binding preparation and maintenance, decision review, repair and reconstruction guidance, rechecks, adjudication, and acceptance-record maintenance for **all** candidates and versions, including failed and unfinished cases. Shared checks count once; additional versions are not counted as additional targets. Model and replay costs are recorded separately, and Section 4.5 additionally reports combined cost per estimated valid task. Experimental audit time is accounted for separately. With no included tasks, coverage is zero and total effort is reported; mean review cost is undefined.

For the controlled comparison, both conditions start from the same prespecified original-goal pool, frozen designs, initial candidates, scene materials, tool states, and candidate scopes. Unresolved goals remain in the coverage denominator. Both receive identical generation and review budget ceilings, reviewer qualifications, per-unit rating and adjudication rules, repair models, access rules, revision opportunities, and acceptance criteria. Resource consumption is measured separately. The control may inspect sources, revise locally, and reuse notes. New review findings and their costs remain condition-specific. Independent audits sample the frozen final collection and the **entire** repair pool, including failed and unfinished cases. Each sample receives 2 independent ratings from auditors uninvolved in its construction or repair; method labels and construction verdicts are hidden, while repair versions remain visible for comparison. An additional reviewer adjudicates disagreements. For multiply rated construction checks, reviewers record judgments independently before an additional reviewer resolves disagreements; unresolved cases cannot pass.

Repair success under an unchanged candidate scope uses independently confirmed defective entries as its denominator. Proceeding to reconstruction does not remove the original defective entry from

this denominator, and reconstruction alone is not counted as repair success. Success jointly requires **resolution of the original defect, preservation of the public requirements, and no new leakage or validity defect**; failed and unfinished repairs remain in the denominator. Each condition is recorded separately, retaining failure counts. Audits do not change inclusion or trigger repair of the measured version. Later revisions retain the original result and charge additional work as construction cost.