ELVA: VERIFICATION BEGINS BEFORE THE VERIFIER

Anonymous authors
Paper under double-blind review

ABSTRACT

Web agents can report success even when every observation they rely on is correct. This can occur when observation processing preserves individual facts but loses which entities and actions they refer to, allowing unrelated facts to be combined into an unsupported completion proof. We formalize this failure as *inadmissible conjunction*. Our key idea is that verification must begin when evidence is collected, before these relationships are lost. We introduce ELVA, a runtime that preserves these relationships and checks whether earlier evidence remains valid as execution proceeds. These checks determine whether observations can jointly support completion or whether the agent needs further observation or repair. For the supported task dependencies, we prove a conditional guarantee excluding completion proofs that reuse stale action references or evidence whose supporting links have broken. Across five WebArena-Verified sites, ELVA achieves 47.51% overall task success, exceeding the strongest reproduced baseline by 6.15 percentage points. A controlled diagnostic also shows lower false-completion and missed-valid-evidence rates than a local verifier without lineage keys at the same evidence horizon. The largest ablation effects come from preserving evidence before verification and using verification results to guide subsequent actions.

1 INTRODUCTION

Large language model (LLM)-based web agents complete requests through long sequences of interactions under partial observability. Relevant information is distributed across pages, actions change the environment, and their consequences must be established through subsequent observations (Zhou et al., 2024). Reliable execution therefore requires both selecting actions and determining whether the available evidence justifies completion, further observation, or recovery.

Recent work improves observation–action alignment (Yang et al., 2025) and process-level assessment (Chae et al., 2025). These advances motivate a complementary question: **what information must survive execution for a completion decision to be justified?** A representation useful for selecting the next action may omit information needed to verify the task. Retaining an earlier observation also does not establish that it remains applicable after navigation or a modification.

We study a failure in which individually correct observations do not form a valid joint proof. Selection constraints must apply to the entity subsequently inspected, and evidence of a modification must concern the resource acted upon. If observation processing discards these bindings, a verifier can combine unrelated facts into an unsupported completion proof. We call this failure *inadmissible conjunction*. It requires no hallucinated observation: the missing information establishes whether the facts belong together. More history cannot reconstruct a binding absent from the retained evidence, while using only the current page can discard necessary support. The challenge is to preserve cross-step evidence while checking its attribution and continued validity.

Our key idea is to make the dependencies required for a completion proof constrain how evidence is produced and reused. ELVA (Evidence-Lineage Verification at the Action Boundary) connects observation construction, action-bound evidence, verification, and control through a shared contract. It preserves verifier-facing identifiers and live state, admits cross-step evidence through explicit lineage rules, and invalidates evidence when its supporting relationship breaks. Unresolved requirements direct observation, exact-resource readback, or bounded repair rather than being filled by unsupported inference.

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.

We establish a conditional structural guarantee excluding stale action references, broken collection-to-detail inheritance, and readback across later modifications. On five WebArena-Verified sites, ELVA achieves 47.51% overall success, 6.15 percentage points above the strongest reproduced baseline. A fixed-trajectory diagnostic tests lineage-aware evidence assessment, while matched ablations locate the largest task-level contrasts in observation construction and evidence-gated control.

Our contributions are: (i) a formulation separating individual evidence correctness from jointly valid completion proofs; (ii) an evidence-contract runtime with executable reuse and invalidation rules and a conditional guarantee; and (iii) complementary system, diagnostic, and stage-level evaluations. Together, they investigate verification as an interface spanning evidence collection and subsequent execution, rather than a terminal judge alone.

2 Related Work

2.1 Web Agents and Observation Representations

WebArena provides interactive web tasks, Mind2Web studies generalization across websites, and WorkArena and VisualWebArena extend evaluation to enterprise and visually grounded tasks (Zhou et al., 2024; Deng et al., 2023; Drouin et al., 2024; Koh et al., 2024). AgentOccam aligns observation and action representations, Agent-E distills DOM observations, and SeeAct separates planning from grounding actions in webpage elements (Yang et al., 2025; Abuelsaad et al., 2024; Zheng et al., 2024). These methods improve the information available for action selection; ELVA additionally requires retaining the bindings needed to verify later outcomes.

ReAct interleaves reasoning with actions, Reflexion uses verbal feedback to improve subsequent attempts, and WebRL trains web policies through an online curriculum (Yao et al., 2023; Shinn et al., 2023; Qi et al., 2025). WebOperator and WebTactix explore action-aware search and parallel planning (Dihan et al., 2025; WebTactix, 2026). ELVA complements policy improvements by checking whether the evidence generated during execution can jointly authorize completion or should instead trigger further interaction.

2.2 Verification and Evidence Interfaces

WebVoyager and the Online-Mind2Web study evaluate automated judgments against human assessments (He et al., 2024; Xue et al., 2025). AgentRewardBench compares multiple judges; its NNetNav implementation judges summaries of observation changes (Lù et al., 2025). Web-Shepherd learns checklist-conditioned process rewards, while the Universal Verifier combines process and outcome assessment with criterion-specific evidence selection (Chae et al., 2025; Rosset et al., 2026). These works address judgment quality. Our formulation instead asks what relationships must remain available for a completion proof: full-trajectory access does not guarantee their preservation, and evidence selection alone does not demonstrate a binding error.

WebArena-Verified audits external task evaluation, and WebStep uses environment-level semantic state tracking for process evaluation (El hattami et al., 2025; Chung et al., 2026). ELVA's online checks use browser-visible evidence rather than privileged evaluator state. Its controlled diagnostic evaluates internal evidence policies, not superiority over independent judges on shared checkpoints.

2.3 Provenance and Online Admissibility

Data provenance distinguishes explanations of why a result holds from the origins of its values (Buneman et al., 2001). HANSEL extracts interactive evidence from web-agent trajectories for human verification (Zhang and Nam, 2026). ELVA makes a narrower operational distinction: recording an origin does not establish that a record remains applicable. Its task contract constrains which records may be joined, when their lineage expires, and which missing evidence requires further interaction.

3 PROBLEM FORMULATION

Task setting. We study online completion verification for web agents. Given an instruction U, an agent interacts with an environment whose state at step t is e_t. The browser exposes $s_t = B(e_t)$, and the agent executes action a_t, producing $\tau_B = (s_0, a_0, s_1, \ldots, a_{T-1}, s_T)$. We write $\tau_{B,0:t}$ for the prefix ending at s_t and $\mathcal{E}_t$ for the evidence retained after observation processing. Verification uses this evidence, not the benchmark evaluator or expected answers. The environment trace determines whether the instruction is satisfied; retained evidence determines what can be certified.

Completion requirements. The compiler returns $G = C(U) = (P, D, \Sigma)$, where P contains browser-verifiable predicates, D specifies their dependencies, and Σ defines scope, resource, entity, and value types. Dependencies connect requirements that cannot be assessed independently: selection constraints must apply to the entity subsequently inspected, and readback must concern the resource associated with a modification. We write $\tau_{B,0:t} \models_B G$ when a common type-consistent assignment satisfies the predicates and dependencies in browser-visible state. This differs from user-level satisfaction, $e_{0:t} \models U$: the contract may omit a material requirement, or the browser may not expose the relevant environment property.

Missing bindings. For a requirement $\exists x\,[p(x) \wedge q(x)]$, records establishing $p(x_1)$ and $q(x_2)$ with $x_1 \neq x_2$ do not supply a common witness. The verifier may not observe the mismatch. If π erases bindings,

$$\pi(\{p(x), q(x)\}) = \pi(\{p(x_1), q(x_2)\}) = \{p(?), q(?)\}, \quad x_1 \neq x_2. \tag{1}$$

Here, each question mark denotes an unavailable binding, not a shared variable. The retained representation is identical, although only the first record set supplies a common entity witness. Without other information resolving identity, further reasoning over this representation cannot distinguish the two cases. Treating the missing bindings as equal produces an *inadmissible conjunction*: individually correct facts are combined without evidence that they jointly satisfy the requirement. This concerns the available proof, not necessarily the underlying task outcome. Missing required keys must remain unresolved. ELVA's fail-closed rule rejects an unsupported certificate, not the possibility that the task succeeded; additional observation may recover the binding.

Verification objective. Let $\mathrm{Adm}_t(r, p)$ denote currently valid individual support for predicate p from record r. Required bindings must be grounded in evidence, not freely assigned. Each record induces a typed partial binding θ_r. Joint compatibility requires

$$\mathrm{Compat}_G(R) \iff \exists\theta : \ (\forall r \in R,\ \theta_r \subseteq \theta) \wedge (\forall d \in D,\ \mathrm{Edge}_d(R, \theta)), \tag{2}$$

where Edge_d checks the relationship required by dependency d. The common assignment reconciles grounded bindings, while the dependency checks establish how the supporting records are connected. Existence of an assignment alone cannot replace missing predicate support or make expired evidence valid. The specification-level acceptance condition is:

$$\begin{aligned}\mathrm{Accept}_t^{\mathrm{spec}}(G, \mathcal{E}_t) \iff \exists R \subseteq \mathcal{E}_t : [\forall p \in P,\ \exists r \in R : \mathrm{Adm}_t(r, p)] \\ \wedge\, \mathrm{Compat}_G(R).\end{aligned} \tag{3}$$

Acceptance implies $\tau_{B,0:t} \models_B G$ if extraction and dependency checks are correct, identities are valid, and no relevant external changes occur between paired observations. The converse is not required: contract satisfaction need not leave sufficient retained evidence. Extending acceptance to $e_{0:t} \models U$ additionally requires complete instruction coverage and faithful browser observability. Section 4 specifies the runtime checks and their conditional structural guarantee.

4 ELVA: EVIDENCE-LINEAGE VERIFICATION AT THE ACTION BOUNDARY

Figure 1 contrasts binding loss with ELVA's contract-guided workflow. A task contract defines predicates and dependencies shared by four components: APDC, action-bound witnesses, CIV, and evidence-gated control. Together, they preserve entity and action bindings, check whether evidence supports a common completion proof, and guide further interaction when support is missing.

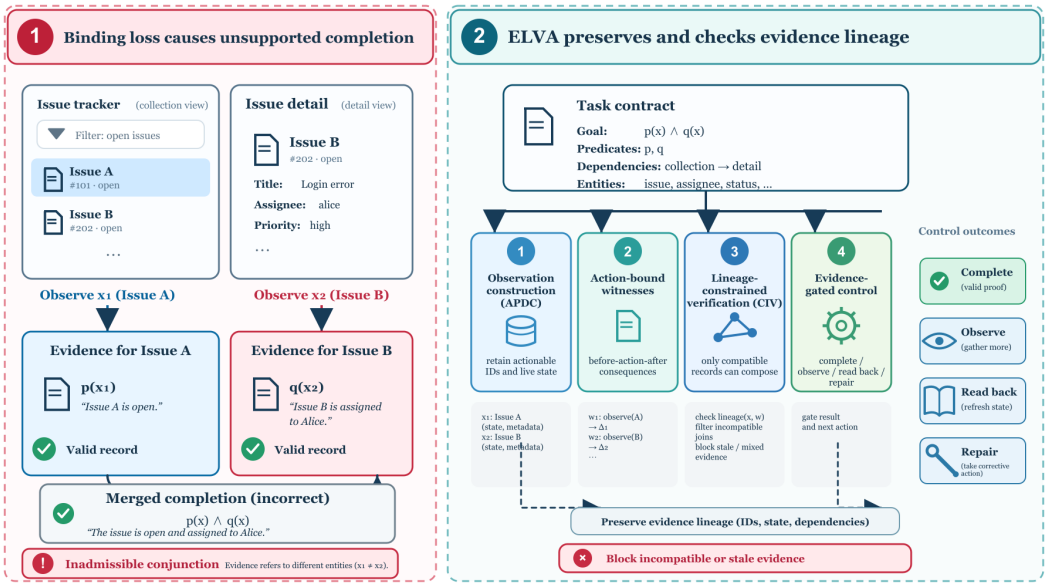

Figure 1: Binding loss and ELVA's evidence-lineage verification. Left: the selected records state that Issue A is open and Issue B is assigned to Alice. Removing their entity bindings allows an unsupported same-issue conjunction. Right: the task contract coordinates observation construction (APDC), action-bound witnesses, verification (CIV), and evidence-gated control. Preserved bindings constrain evidence reuse; the controller selects completion, observation, readback, or repair.

4.1 Preserving Evidence Before Verification

Task contracts. ELVA separates response-format requirements from website semantics and compiles scope, query, filter, ordering, entity, answer, and mutation requirements, their dependencies, and terminal modes. Hand-specified lexical and structural rules operate on the instruction and public start context, without reference answers or task-identifier lookup. Appendix D details their construction and coverage criteria. Recognized predicates start as UNSEEN. Missing evidence for a recognized predicate is distinct from an unrecognized material clause omitted during compilation.

Evidence-preserving observations. Action-Preserving DOM Compilation (APDC) constructs compact semantic units retaining role, name, actionable identifier, state, parent, and candidate action. Expanded options and compact tables retain descendant identifiers. For the visible actionable-unit set $\mathcal{U}_{\text{action}}(s_t)$, it audits

$$\forall u \in \mathcal{U}_{\text{action}}(s_t), \qquad \text{id}(u) \in \text{IDs}(\text{APDC}(s_t)). \tag{4}$$

The preservation claim applies to observations passing this audit. Extraction and rendering take $O(n)$ time for n accessibility-tree nodes, while ranking takes $O(n \log n)$. The shared browser ingestion layer exposes live control values and otherwise unlabeled native options on both sides of the APDC ablation.

4.2 Binding Evidence to Action Transitions

The runtime maintains $X_t = (S_t, A_t, O_t, L_t)$: snapshots, actions, outcomes, and an append-only evidence ledger. Given dispatch status d_t, it applies

$$X_{t+1} = \text{Update}_G(X_t, s_t, a_t, s_{t+1}, d_t). \tag{5}$$

Each action is associated with its pre-action snapshot. Its transition witness records before/after fingerprints and URLs, target and value, semantic control-state change, dispatch failure, and visible

errors. Guards append assessments anchored to the observation or transition they inspect. Issuing an action is thus separate from establishing its intended consequence.

Snapshot fingerprints and references. A snapshot is fingerprinted from the final public URL and normalized accessibility observation; a consecutive identical fingerprint is not appended twice. Normalization retains identifier tokens present in the browser text. A stored action identifier requires a matching source fingerprint, label, and available destination. Redirects bind to the observed final URL. Ambiguous snapshot pairs, duplicate label destinations, missing keys, stale identifiers, and detected mismatches fail closed.

Scope and resource identity. Collection scope normalizes pagination, whereas mutation-resource equality retains the full query. Entity keys use a public destination when available, otherwise a label/identifier hash interpreted with source scope and snapshot. These checks establish guarded key use, not real-entity identity from string equality alone. Appendix A details canonicalization.

4.3 Lineage-Constrained Evidence Composition

Consequence-Indexed Verification (CIV) admits historical evidence through explicit lineage relationships. Its eligible anchor set is

$$W_t = \{\iota(s_t)\} \cup W_t^{\mathrm{open}} \cup W_t^{\mathrm{read}}, \tag{6}$$

where $\iota(s_t)$ is the current snapshot fingerprint. Eligibility depends on the validity of the supporting relationship rather than a fixed history length.

Collection-to-detail lineage. W_t^{open} includes a source collection and its opening transition only when an executed, nonfailed, error-free click or navigation resolves to a unique ordered before/after pair. Every subsequent snapshot must retain the selected detail route and heading identity. Breaking that lineage invalidates inherited collection evidence.

Commit-to-readback lineage. W_t^{read} includes a commit's immediate post-state and later explicit reload, revisit, or navigation probes returning to the exact resource before the next commit. CIV checks the requested payload against the observed consequence and rejects visible errors or unchanged post-states as evidence of a successful change. A later commit closes the earlier readback window.

For the newest active assessments selected by the certificate view in Section 4.4, individual admissibility is

$$\begin{aligned} \mathrm{Adm}_t(r,p) \equiv\ & \mathrm{Active}_t(r,p) \wedge \mathrm{Status}(r) = \mathrm{SUPPORTED} \\ & \wedge \mathrm{ScopeMatch}(r,p) \wedge \mathrm{EntityUnify}(r,p) \\ & \wedge \mathrm{ValueEntails}_B(r,p) \wedge \neg\, \mathrm{PublicError}(\Lambda(r)). \end{aligned} \tag{7}$$

Here, $\Lambda(r)$ resolves the record's snapshot or transition anchor. Scope matching uses typed URL fields, namespace paths, breadcrumbs, or active controls. Entity unification requires grounded keys; a missing key is not an arbitrary free variable. Value entailment uses deterministic family-specific checks, including normalized equality, numeric comparison, active-control state, ordering, semantic change, and payload/readback matching. Active records must resolve to witnesses in W_t.

Compatibility additionally requires Equation (2). The six supported dependency families connect scope to constraints, constraints to selection or coverage, collection selection to detail, project to entity, commit to readback, and payload to changed field. Appendix A gives the complete guard table. Dependencies outside these families are not covered; within them, accepted records must have jointly compatible bindings and dependency relationships.

4.4 From Predicate State to Execution Control

Active evidence. Each predicate assessment is UNSEEN, SUPPORTED, REFUTED, or CONFLICT. Each predicate has a designated anchor $\alpha_t(p)$: normally the current snapshot, the

direct source collection for inherited filter/query/order evidence, or the admitted readback for persistence. A record is active only when it is not invalidated, matches this anchor, and resolves to a witness in W_t. The certificate view uses the newest active assessment,

$$\mathrm{StateAt}_t(p) = \mathrm{Status}\left(\operatorname*{arg\,max}_{r:\mathrm{Active}_t(r,p)} \mathrm{seq}_L(r)\right), \tag{8}$$

or UNSEEN if none exists. Snapshot changes, broken detail lineage, and later commits change which records remain active. A separate scope-level diagnostic view retains conflicting support and refutation until explicit invalidation or scope change. This history blocks completion only when a relevant guard emits an unresolved reason; it does not independently determine acceptance.

Operational acceptance. Let $\mathrm{Reasons}_G(X_t)$ collect unresolved reasons from the environment, task-constraint, coverage, answer-grounding, mutation, navigation, and terminal-mode checks. Let $\widehat{\mathcal{H}}_G(X_t)$ contain mode-specific hypothesis certificates with public-evidence verdicts and remaining obligations. A proof-gated completion certificate is accepted exactly when

$$\begin{aligned}\mathrm{Accept}_{\mathrm{cert}}(G, X_t) \iff\ & \mathrm{Reasons}_G(X_t) = \varnothing \\ & \wedge\, \exists \hat{h} \in \widehat{\mathcal{H}}_G(X_t) : \mathrm{Supported}(\hat{h}).\end{aligned} \tag{9}$$

This certificate rule evaluates predicate states and named guards rather than unrestricted proof search. The controller maps verdicts and obligations to completion, observation, exact-resource readback, or bounded grounded repair. Stop proposals and state-changing actions invoke CIV and the controller. Missing bindings request evidence rather than establishing task failure. Repair changes predicate support only through a new observation, outcome, or deterministic reassessment. The controller can also terminate without certification; Appendix E distinguishes this fail-open path from proof-gated acceptance.

4.5 Conditional Acceptance Consistency

Assumption A0—snapshot identity adequacy. Snapshot occurrences requiring different action-target bindings or transition provenance are distinguishable by the runtime's fingerprints.

A0 excludes hash collisions and *projection aliases* that merge different required bindings despite identical fingerprinted representations. Identifier tokens are retained in the hash input, so changing them is distinguishable under collision-free hashing. Identical tokens can nevertheless denote changed underlying bindings; A0 concerns binding validity, not textual equality.

Proposition 1—structural acceptance consistency. *Under A0, starting from an empty trace and ledger, runtime updates preserve source-bound action references and guarded cross-snapshot evidence windows. Certificate construction enforces consistent stored-key bindings and the supported dependency guards. Within the six supported families, a proof-gated certificate cannot reuse a stale action identifier, inherit collection evidence after its detail lineage breaks, carry mutation readback across a later commit, or assign different stored keys to the same contract variable.*

Proof sketch. The empty state satisfies the invariant. Action recording copies the source snapshot, and outcome construction retains before/after identifiers and dispatch status. Under A0, a target-binding change cannot pass the source-fingerprint check unnoticed. Opening edges require a unique ordered transition and an unbroken route-and-heading chain. Readback admits exact-resource probes only before the next commit. Ledger updates preserve provenance, while exact-anchor selection excludes expired evidence. Compatibility checks reject inconsistent bindings and unsupported dependency edges. Induction over updates preserves provenance and window invariants; the certificate checks yield the exclusions. Appendix B expands the argument. □

Interpreting operational acceptance as $\tau_{B,0:t} \models_B G$ additionally requires faithful extraction, valid identity keys, semantically correct checks covering the contract's dependencies, and no relevant external changes between paired observations. Extending this conclusion to $e_{0:t} \models U$ requires complete instruction coverage and faithful browser observability. Fail-closed treatment protects recognized obligations, not arbitrary contract omissions or hidden server properties.

Table 1: Complete-system task success (%). Counts in parentheses total 764 tasks. Overall is task-weighted. All systems use Gemini 3.1 Flash-Lite; bold marks the highest reported result.

Method	GitLab (180)	Reddit (106)	Shopping (187)	Shopping Admin (182)	Map (109)	Overall
AgentOccam	38.89	58.49	36.36	26.37	17.43	34.95
WebOperator	39.44	58.49	43.32	43.96	12.84	40.31
WebTactix	37.22	56.60	43.32	42.31	28.44	41.36
ELVA	**45.56**	**60.38**	**53.48**	**46.15**	**30.28**	**47.51**

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Benchmark and systems. We use WebArena-Verified v1.2.3, whose re-audited tasks, reference answers, and deterministic evaluator address ambiguous success criteria and brittle matching in the original WebArena (El hattami et al., 2025). The main comparison covers 764 tasks across five sites (Table 1). ELVA and author-run reproductions of AgentOccam (Yang et al., 2025), WebOperator (Dihan et al., 2025), and WebTactix (2026) use Gemini 3.1 Flash-Lite through Vertex AI and the same browser services while retaining their original agent designs.

Metrics and protocols. We report official task success. Overall weights site-level rates by task counts, rather than averaging sites equally; differences are percentage points (pp). The main comparison measures complete-system performance. A separate diagnostic assesses predicate-evidence sufficiency, and ablations use a smaller task subset with matched runtime conditions. These comparisons answer different questions rather than isolating the verifier through overall task success alone.

5.2 MAIN RESULTS ON WEBARENA-VERIFIED

ELVA achieves **47.51% overall success**, exceeding the strongest reproduced baseline by **6.15 pp**. It has the highest reported rate on each site, although the margins over the strongest site-specific baselines on Reddit and Map are only 1.89 and 1.84 pp. Verification and runtime-stage effects are examined separately below.

5.3 DOES EVIDENCE LINEAGE IMPROVE VERIFICATION?

Diagnostic design. We use 1,600 predicate checkpoints from fixed trajectories: 800 with admissible evidence and 800 without it. Two annotators judge whether browser-visible evidence establishes the supplied predicate, not whether compilation covers the instruction or hidden backend state satisfies the task. A checkpoint is not an entire benchmark task.

The source trajectories, actions, states, contracts, and extracted records are fixed. We compare terminal-snapshot verification; unrestricted history; local transitions with resource/entity/selection keys masked; and CIV with those keys and compatibility checks at the same local horizon. The local contrast changes key availability and composition together, not checker logic on identical effective inputs.

FPR is the **false-completion rate** among inadmissible cases, and FNR is the **missed-valid-evidence rate** among admissible cases. Balanced accuracy averages the class-wise correct-decision rates. These class-conditional metrics concern evidence sufficiency, not deployment prevalence or independently measured task failure.

Results. Relative to key-masked local verification, CIV reduces FPR from **17.9% to 6.9%** and FNR from **21.1% to 10.6%**, increasing reported balanced accuracy from **80.5% to 91.3%**. Both errors decrease, rather than trading fewer false completions for more missed evidence. This supports the combined lineage-aware configuration, not individual keys or guards. A pure veto on an unchanged decision rule could only shrink its acceptance set. It would not recover previously rejected valid evidence, so the lower FNR cannot be explained by additional rejection checks alone.

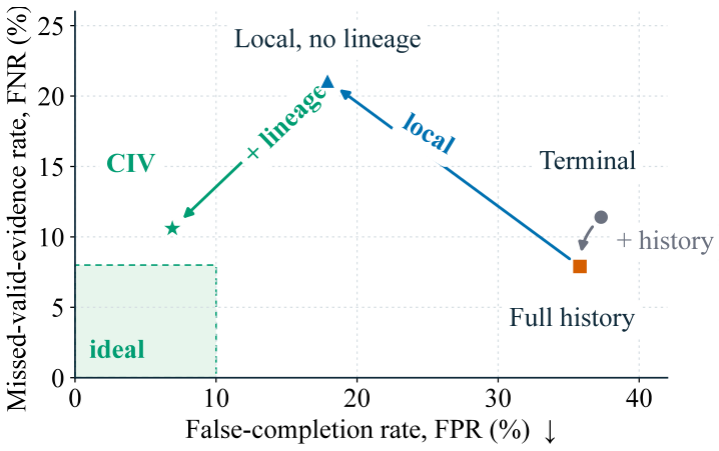

Figure 2: Verification on 1,600 balanced checkpoints. Source trajectories and extraction are fixed; policies differ in accessible context, key visibility, and composition. Lower FPR and FNR are better.

Table 2: Matched runtime-stage ablations on 151 tasks. Rates are percentages; Overall is task-weighted. Differences are ELVA-full minus ablation in pp. This subset and execution protocol differ from Table 1; 49.67% and 47.51% are not estimates on the same evaluation set.

Configuration	GitLab	Reddit	Shopping	Shopping Admin	Overall	Difference (pp)
ELVA-full	53.66	61.90	52.08	36.59	49.67	—
w/o APDC	43.90	57.14	45.83	24.39	41.06	+8.61
w/o evidence-gated control	46.34	38.10	47.92	34.15	42.38	+7.28
w/o CIV	43.90	57.14	47.92	41.46	46.36	+3.31
w/o APDC and CIV	48.78	52.38	45.83	21.95	41.06	+8.61

Unrestricted history improves FPR from **37.3% to 35.8%** and FNR from **11.4% to 7.9%** over terminal-snapshot verification. More history therefore preserves useful evidence. Relative to history, CIV lowers FPR by **28.9 pp** but raises FNR by **2.7 pp**: it rejects far more inadmissible proofs at some cost to evidence coverage. Neither a longer history nor temporal locality alone establishes joint admissibility.

5.4 Which Runtime Stages Contribute to Task Success?

Protocol. We select 151 tasks from different templates within GitLab, Reddit, Shopping, and Shopping Admin without consulting outcomes. Map is excluded. The single-agent policy loop, text-action interface, prompt, and budget are fixed; auxiliary critic and judge modules are disabled. Tasks are limited to 20 actions or 600 seconds. We remove APDC, evidence-gated control, or CIV individually and APDC/CIV jointly. Unlike the checkpoint diagnostic, these interventions can change subsequent trajectories.

Stage effects. Removing APDC lowers overall success from **49.67% to 41.06%**, and removing control lowers it to **42.38%**; both regress on every site. Removing CIV produces a smaller decrease to **46.36%**. These contrasts support evaluating verification with its upstream observations and downstream control, without isolating APDC's key-preservation role from its other observation changes.

Interaction and site heterogeneity. With control retained, CIV's contrast is 3.31 pp with APDC and 0.00 pp without it. Their additive interaction, $(49.67 - 46.36) - (41.06 - 41.06)$, is therefore 3.31 pp. APDC retains a 5.30 pp contrast without CIV. These whole-stage contrasts do not isolate key preservation from other observation changes. Without APDC, removing CIV improves GitLab

from **43.90% to 48.78%** but reduces Shopping Admin from **24.39% to 21.95%**, so pooled effects conceal opposing site-level changes.

With APDC enabled, removing CIV instead improves Shopping Admin from **36.59% to 41.46%**, despite decreases on the other sites. Additional readback or repair when successful progress lacks attributable evidence is one possible explanation, not an established cause. Section 6 distinguishes this task-level effect from diagnostic verification quality.

6 Analysis and Discussion

6.1 From Verification Decisions to Task Success

Rejecting an unsupported proof does not itself complete a task. The CIV ablation compares two rejection rules, not a gate against accepting every stop: other checks and evidence-gated control remain enabled without CIV. Consider a common distribution of proposed stopping points with preceding trajectories fixed. Let $Y_0 \in \{0, 1\}$ be the potential benchmark outcome of stopping, R_A, R_B indicate rejection by two gates, and $Y_1^A, Y_1^B \in \{0, 1\}$ their respective potential outcomes after bounded continuation. Their expected difference is

$$\Delta_{A,B} = \mathbb{E}\big[R_A(Y_1^A - Y_0) - R_B(Y_1^B - Y_0)\big]. \tag{10}$$

With shared continuation, $Y_1^A = Y_1^B = Y_1$, this becomes $\mathbb{E}[(R_A - R_B)(Y_1 - Y_0)]$: only disagreements contribute, without assuming nested rejection rules. Otherwise, continuation can differ even when both gates reject. Appendix C gives the recovered-failure/lost-success decomposition.

The decomposition distinguishes changing a decision from changing its eventual outcome. An additional rejection helps when continuation recovers an otherwise unsuccessful stop, and harms when continuation loses an already successful outcome. If both gates reject but request different readback or repair, their difference can arise from the subsequent actions rather than from the accept/reject decision itself. Acquiring stronger evidence without changing the benchmark outcome improves the justification for completion but leaves task success unchanged.

These are decision-level identities, not estimates of the ablation effect: runtime changes can alter encountered trajectories and stopping points. Diagnostic inadmissibility does not imply task failure ($Y_0 = 0$), so FPR/FNR cannot be substituted directly. CIV's diagnostic advantage and modest overall task contrast are therefore compatible: benefits depend on recovery and preservation of already successful outcomes, not rejection alone.

6.2 Limitations

The evaluation uses one benchmark and one policy model. The diagnostic compares internal evidence policies, not independent LLM-based or learned verifiers on shared checkpoints. Inference costs—including tokens, model calls, and latency—are not evaluated, so success gains do not establish efficiency. The 151-task ablation subset limits site-level and interaction estimates, and whole-stage removal does not isolate key preservation. The rule compiler is also limited by its recognized language patterns (Appendix D). Finally, Section 4.5 establishes structural consistency under an identity assumption, not unconditional semantic correctness.

7 Conclusion

Correct observations can still form an invalid completion proof when their relationships are lost. ELVA makes these dependencies executable across observation construction, action-bound evidence, admissible composition, and control. Its conditional invariant and complementary evaluations characterize structural consistency, verification behavior, and task-level effects. Reliable completion requires preserving not only facts but also their valid relationships—and acquiring missing evidence rather than guessing unsupported bindings.

AI USE STATEMENT

Generative AI assisted literature retrieval, figure drafting, manuscript organization, formal exposition, language editing, LaTeX preparation, and checks of notation and reported aggregate results during revision. Benchmark task-success and checkpoint scores were unchanged; offline implementation checks informed the reproducibility descriptions.

REFERENCES

Tamer Abuelsaad, Deepak Akkil, Prasenjit Dey, Ashish Jagmohan, Aditya Vempaty, and Ravi Kokku. Agent-E: From Autonomous Web Navigation to Foundational Design Principles in Agentic Systems. *arXiv preprint arXiv:2407.13032*, 2024. URL https://arxiv.org/abs/2407.13032.

Peter Buneman, Sanjeev Khanna, and Wang-Chiew Tan. Why and Where: A Characterization of Data Provenance. In *Database Theory – ICDT 2001*, pages 316–330, 2001. doi: 10.1007/3-540-44503-X_20. URL https://doi.org/10.1007/3-540-44503-X_20.

Hyungjoo Chae, Sunghwan Kim, Junhee Cho, Seungone Kim, Seungjun Moon, Gyeom Hwangbo, Dongha Lim, Minjin Kim, Yeonjun Hwang, Minju Gwak, Dongwook Choi, Minseok Kang, Gwanhoon Im, ByeongUng Cho, Hyojun Kim, Jun Hee Han, Taeyoon Kwon, Minju Kim, Beong-woo Kwak, Dongjin Kang, and Jinyoung Yeo. Web-Shepherd: Advancing PRMs for Reinforcing Web Agents. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.15277.

Jiwan Chung, JiHyuk Byun, Vibhav Vineet, and Seon Joo Kim. Where Did It Go Wrong? Process-Level Evaluation of Web Agents with Semantic State Tracking. In *Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2606.15673.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Mahir Labib Dihan, Tanzima Hashem, Mohammed Eunus Ali, and Md Rizwan Parvez. WebOperator: Action-Aware Tree Search for Autonomous Agents in Web Environment. *arXiv preprint arXiv:2512.12692*, 2025. URL https://arxiv.org/abs/2512.12692.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. WorkArena: How Capable Are Web Agents at Solving Common Knowledge Work Tasks? In *International Conference on Machine Learning*, pages 11642–11662, 2024. URL https://proceedings.mlr.press/v235/drouin24a.html.

Amine El hattami, Megh Thakkar, Nicolas Chapados, and Christopher Pal. WebArena Verified: Reliable Evaluation for Web Agents. In *Workshop on Scaling Environments for Agents*, 2025. URL https://openreview.net/forum?id=94tlGxmqkN.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. WebVoyager: Building an End-to-End Web Agent with Large Multimodal Models. In *Annual Meeting of the Association for Computational Linguistics*, pages 6864–6890, 2024. doi: 10.18653/v1/2024.acl-long.371. URL https://aclanthology.org/2024.acl-long.371/.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Annual Meeting of the Association for Computational Linguistics*, pages 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Xing Han Lù, Amirhossein Kazemnejad, Nicholas Meade, Arkil Patel, Dongchan Shin, Alejandra Zambrano, Karolina Stańczak, Peter Shaw, Christopher J. Pal, and Siva Reddy. AgentRewardBench: Evaluating Automatic Evaluations of Web Agent Trajectories. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.08942.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Jiadai Sun, Xinyue Yang, Yu Yang, Shuntian Yao, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: Training LLM Web Agents via Self-Evolving Online Curriculum Reinforcement Learning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/c66e1fcc9691aae706250638f36f681b-Abstract-Conference.html.

Corby Rosset, Pratyusha Sharma, Andrew Zhao, Miguel González-Fernández, and Ahmed Awadallah. The Art of Building Verifiers for Computer Use Agents. In *Conference on Language Modeling*, 2026. URL https://www.microsoft.com/en-us/research/publication/the-art-of-building-verifiers-for-computer-use-agents/.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

WebTactix. WebTactix: Semantic Tree-Guided Parallel Multi-Agent Planning for Web Task. Public project and code artifact, 2026. URL https://paper-submission-anoymous.github.io/webtactix_introduction/.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An Illusion of Progress? Assessing the Current State of Web Agents. In *Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.01382.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. AgentOccam: A Simple Yet Strong Baseline for LLM-Based Web Agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.13825.

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://arxiv.org/abs/2210.03629.

Yujin Zhang and Daye Nam. HANSEL: Extracting Breadcrumbs from Web Agent Trajectories for Interactive Verification. *arXiv preprint arXiv:2606.18671*, 2026. URL https://arxiv.org/abs/2606.18671.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. GPT-4V(ision) is a Generalist Web Agent, if Grounded. In *International Conference on Machine Learning*, pages 61349–61385, 2024. URL https://proceedings.mlr.press/v235/zheng24e.html.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2307.13854.

A Runtime Details

A.1 Identity and Canonicalization

A snapshot is fingerprinted from its final public URL and normalized accessibility observation; a consecutive identical fingerprint is not appended twice. A stored action identifier requires the current fingerprint to match its source and the label and available destination still to agree. Redirects bind to their observed final URL. Later detail reuse matches the final route and heading; mutation readback matches the exact final resource. Ambiguous snapshot pairs, duplicate label destinations, missing keys, stale identifiers, and detected identity mismatches fail closed.

$\mathrm{CanonScope}$ lowercases scheme and host, removes fragments and pagination parameters, normalizes the trailing slash, and sorts remaining query pairs. $\mathrm{CanonRes}$ lowercases scheme and host and removes fragments, but retains the full query. Mutation resources require exact $\mathrm{CanonRes}$ equality. A collection entity key uses the absolute public destination when available, otherwise a hash of normalized label and actionable identifier interpreted with source scope and snapshot. Public-key equality does not by itself establish real-entity identity.

APDC audits $\forall u \in \mathcal{U}_{\mathrm{action}}(s),\ \mathrm{id}(u) \in \mathrm{IDs}(\mathrm{APDC}(s))$, where $\mathcal{U}_{\mathrm{action}}$ is the visible actionable-unit set. Use-time validation checks snapshot, label, destination, and transition. APDC identifier retention and adequacy of the fingerprint for distinguishing relevant execution occurrences are separate requirements.

A.2 Dependency Guards

The following table specifies the six dependency families introduced in Section 4.3.

Dependency	Required evidence relationship
Scope → filter, query, or order	The constraints use the same collection anchor.
Filter, order, or constraint → selection or coverage	Selection or coverage uses that collection's verified frontier.
Collection selection → detail	A unique opening transition connects the selected item to an unbroken detail lineage.
Project → entity	The opening transition retains the source project scope.
Commit → readback	An explicit probe returns to the exact resource before another commit.
Payload → changed field	The requested value matches the immediate semantic change or admitted readback.

The opening transition must be executed, nonfailed, and free of visible errors. Every later snapshot must retain the selected detail route and heading identity. Readback has no fixed step count: reload, revisit, goto, back, or forward probes are admissible only when they return to the exact resource before the next commit. Historical presence in the ledger does not itself preserve eligibility.

A.3 Ledger Views and Invalidation

Let $\alpha_t(p)$ be a predicate's designated anchor: normally the current snapshot, the direct source collection for inherited filter/query/order evidence, or the readback selected by the mutation guard for persistence. A record is active if not explicitly invalidated, anchored at $\alpha_t(p)$, and associated with a witness in W_t. With seq_L denoting append order, the certificate state is

$$\mathrm{StateAt}_t(p) = \begin{cases} \mathrm{Status}\left(\underset{r:\mathrm{Active}_t(r,p)}{\arg\max}\ \mathrm{seq}_L(r) \right), & \text{if an active record exists,} \\ \mathrm{UNSEEN}, & \text{otherwise.} \end{cases} \tag{A1}$$

A new snapshot changes current-state anchors; leaving the detail route removes the inherited collection, and another commit closes an earlier readback window. Reassessment at the same exact anchor

appends a record and changes the newest assessment. A repair changes support only through a new observation, outcome, or deterministic reassessment.

The separate scope-level diagnostic view retains conflicting support and refutation until explicit invalidation or scope change. A later support record does not automatically erase that conflict. Acceptance nevertheless uses exact-anchor state and named guards; a historical diagnostic conflict blocks completion only when a guard emits a corresponding unresolved reason.

B Structural Acceptance-Consistency Argument

This argument concerns the specified update and proof-gated certificate rules under Assumption A0 in Section 4.5. It does not infer semantic correctness from identifier equality.

Base case. The empty trace and ledger satisfy source binding and evidence-window membership vacuously.

Observations and actions. Snapshot construction records the public fingerprint. Under A0, deduplication cannot merge occurrences requiring different target bindings or transition provenance. Action recording copies the source snapshot. Outcome construction retains the before/after identifiers, dispatch status, semantic changes, and visible errors. These operations preserve attribution without establishing the semantic correctness of an extracted value.

Opening edges. The constructor admits only a unique ordered pair for an executed, nonfailed, error-free opening action. Continued reuse requires an unbroken route-and-heading chain. Breaking the chain removes the inherited source collection from the eligible window. Keeping an old ledger record cannot restore its expired edge.

Readback. The constructor admits explicit probes to the exact resource only before the next commit. Navigation away and back does not establish persistence unless the probe returns to that resource within this window. A later commit closes the earlier readback window.

Ledger and selection. Appending a record preserves earlier provenance. The certificate view selects the newest assessment among active records, excluding invalidated records, wrong anchors, and expired witnesses. Reassessment can update the newest exact-anchor state but cannot rebind its source. Compatibility requires selected partial substitutions to extend to a common typed assignment and every supported dependency to pass its guard. Different stored keys therefore cannot bind the same contract variable.

Induction over these operations preserves source binding and eligible lineage. The certificate checks then exclude stale source references, broken collection inheritance, readback across later commits, and incompatible stored-key assignments. Scope-level conflict retention is diagnostic and is not needed for these exclusions.

A semantic lift additionally requires complete task compilation into supported predicates and dependencies, faithful extraction and value checks, valid entity keys within scope, no relevant external changes between paired observations, and browser-visible state that faithfully exposes the instruction's required environment property. The structural argument does not independently establish these conditions.

C Verification-to-Success Decomposition

Consider a common distribution of proposed stopping points with their preceding trajectories fixed. Let $Y_0 \in \{0,1\}$ be the potential benchmark outcome of stopping immediately, and $Y_1 \in \{0,1\}$ the potential outcome under a specified bounded continuation. With $R = 1$ indicating rejection and continuation, the outcome is $(1-R)Y_0 + RY_1$. Relative to accepting every proposed stop,

$$\begin{aligned}\Delta_{\text{gate}} &= \mathbb{E}[(1-R)Y_0 + RY_1] - \mathbb{E}[Y_0] \\ &= \Pr(R=1, Y_0=0, Y_1=1) - \Pr(R=1, Y_0=1, Y_1=0).\end{aligned} \tag{C1}$$

The first term counts recovered failures and the second lost successes. Acquiring stronger evidence while leaving the benchmark outcome unchanged has no effect on this success measure.

The CIV ablation instead compares two rejection rules. For gates R_A, R_B at common stopping points, with respective continuation outcomes Y_1^A, Y_1^B,

$$\begin{aligned}\Delta_{A,B} &= \mathbb{E}[(1-R_A)Y_0 + R_A Y_1^A] - \mathbb{E}[(1-R_B)Y_0 + R_B Y_1^B] \\ &= \mathbb{E}[R_A(Y_1^A - Y_0) - R_B(Y_1^B - Y_0)].\end{aligned} \tag{C2}$$

Under shared continuation, $Y_1^A = Y_1^B = Y_1$, the expression is $\mathbb{E}[(R_A - R_B)(Y_1 - Y_0)]$. Only gate disagreements contribute under that condition; nested rejection rules are unnecessary. With different continuation policies, the outcomes may also differ at stopping points both gates reject.

These identities do not estimate causal components from the reported aggregate ablation. Runtime interventions can change which stopping points are encountered, and diagnostic evidence sufficiency is not the potential benchmark outcome Y_0. The checkpoint FPR and FNR therefore cannot be substituted for the probabilities in these equations.

D Contract Compilation and Generalization

D.1 Compiler Construction and Inputs

The contract compiler is a hand-specified composition of regular expressions, lexical categories, and structured extraction functions in `elva_agent/goal_contract.py`. It is not an LLM call or a learned semantic parser. The entry point `compile_goal_contract` receives the instruction and optional public task context. At the online integration boundary, `_safe_task_config` retains only the public instruction, start URL, and site list; reference answers and evaluator specifications are not compiler inputs. The compiler uses the instruction for language interpretation and the start URL for allowed-origin and deictic-scope binding. It does not select a rule by task identifier or dispatch on a benchmark-site name.

Compilation first separates a trailing, explicitly headed output-protocol paragraph from website semantics. It then extracts intent, quantification, ordering, time windows, scope, numerical conditions, identity, query/filter constraints, answer requirements, mutation payloads, and terminal modes. Dependencies are assembled after these fields have been extracted. The rule vocabulary includes domain terms such as issue, project, product, review, and document, alongside generic relations such as named, within, before, and after. Consequently, site-neutral dispatch does not imply unrestricted linguistic coverage or establish transfer to unseen task distributions.

Public context is necessary for phrases such as "the current project." `_compile_deictic_scope_constraints` binds such phrases to the public start-route path. Without a usable route, it records unresolved deictic metadata instead of inventing a scope name. Relative dates use an explicit "today is" date in the instruction when present and the process date otherwise. Preserving the input instruction, public start context, compiler configuration, and clock context is therefore necessary to reconstruct the same contract.

D.2 Contract Fields and Dependency Assembly

`GoalContract.to_report()` exports an inspectable contract. Each predicate has an identifier, kind, expected value, mandatory flag, and prerequisite identifiers. The compiler always emits environment and answer predicates; their presence alone is not evidence that the task's material constraints were recognized.

Requirement	Compiled representation and assembly rule
Scope and query	Scope terms and typed scope constraints are recorded separately from free-text queries. Scope prerequisites are attached where constructed by the compiler.
Filtering	State, semantic-facet, facet-absence, and visibility constraints retain values and polarity. Navigation-specific extraction is controlled by the navigation-CIV configuration.

Requirement	Compiled representation and assembly rule
Ordering and selection	Sort specifications record field and direction. Extremum-selection predicates depend on the scope, filter, and numerical predicates already emitted.
Coverage	Count, sum, range, extremum, top-k, and universal requirements generate coverage predicates with the relevant scope/filter/order prerequisites.
Identity and answer	Explicit entity identity is represented separately from answer-field conditions and terminal presentation.
Mutation	Commit and readback predicates are emitted; readback depends on commit. Target identities, payload literals, requested states, and conditional branches are stored as structured fields.

The six dependency families in Appendix A are runtime verification families, not six mutually exclusive task classes. Some relationships are explicit predicate prerequisites; others require execution evidence, such as the collection-to-detail opening edge or the resource binding of a changed field. Counting contracts containing a prerequisite is therefore different from counting tasks whose complete instruction is representable and correctly compiled.

D.3 Coverage, Omission, and Compiler Error

Compiler assessment distinguishes successful construction of a contract object, representability within the supported families, and faithful translation of the instruction. For a set of independently annotated material clauses, let a task be *representable* when every required relationship belongs to a supported family. Let a task be *faithfully compiled* when every material clause is correctly represented and no additional unsupported requirement is introduced. Representability does not imply faithful compilation, and neither is measured by task success alone.

A clause-level audit records the requested scope, value, polarity, entity binding, ordering, quantifier, terminal mode, and dependency. An omitted clause has no corresponding requirement; an incorrect clause has a representation with the wrong semantics. Missing prerequisites and spurious constraints are recorded separately. Task-level compilation error counts a task when any material clause is omitted or mistranslated, a required dependency is absent, or a spurious mandatory constraint is added. An omission rate uses annotated material clauses as its denominator, whereas task coverage uses distinct tasks. The companion audit utility exports the actual contracts and summarizes these quantities only when independently supplied clause annotations are available; it does not treat a nonempty predicate list as a correct contract.

D.4 An Inspectable Compiler Example

The instruction `Open the latest open issue in project "acme/widgets".` produces a detail-mode contract with seven predicates: environment, scope, filter, extremum selection, ordering, latest-item coverage, and answer. The selection depends on scope and filter; coverage depends on scope, filter, and ordering; the answer depends on coverage. This illustrates how extracted constraints are connected before browser evidence is collected.

By contrast, the instruction in the artifact's smoke test—`Find the first open issue under the active bug label and report its title.`—produces only the generic environment and answer predicates. The first-item, open-state, and label requirements are not separately represented. The same omission is observed with navigation-CIV enabled. The smoke test checks importability, nonempty output, and an answer terminal mode; it is not a clause-coverage test. These examples are offline compiler inspections, not an estimate of benchmark-wide error. They expose the distinction required by the semantic assumptions: fail-closed evidence checks protect recognized predicates but cannot recover obligations that compilation omitted.

E CERTIFICATES, REPAIR, AND THE POLICY INTERFACE

E.1 TERMINAL HYPOTHESES AND CERTIFICATE CONSTRUCTION

A terminal hypothesis specifies the required finishing mode, not a guessed reference answer. The compiler supports answer, detail, collection, mutation, and draft modes. Explicit draft-without-submit language fixes draft mode; a mutation intent otherwise fixes mutation mode. For ambiguous read-only requests, hand-specified language scores initialize answer/detail/collection at 0.4/0.2/0.2 and add weights for explicit queries, collection requests, information questions, navigation verbs, singleton selection, and aggregate requirements. The implementation applies softmax with temperature 1.15, retains modes with weight at least 0.06 together with the maximum-weight mode, and renormalizes. These are heuristic mode weights, not learned probabilities of task success.

`verify_stop_certificate` checks public environment and scope evidence, query/filter/order constraints, coverage, selected-entity presentation, mutation consequences, and navigation requirements. `_verify_terminal_hypotheses` then produces a `HypothesisCertificate` for each retained mode, containing its weight, supported/refuted flags, score, reasons, and obligations.

Terminal mode	Mode-specific evidence checks
Answer	A nonempty answer, admissible coverage or absence of unresolved reasons under the implemented branch, and rejection of unsupported absence claims.
Detail	A detail surface or admitted navigation lineage, selected-entity identity, selection constraints, attributes, coverage, and no presentation conflict.
Collection	A collection surface with the requested query, filters, ordering, and coverage.
Mutation	A supported mutation certificate linking the requested effect to its public commit and readback evidence.
Draft	A supported draft-state mutation certificate, respecting the requested non-submission terminal mode.

A supported mode receives score 1, a refuted mode 0, and an unresolved mode 0.35. Their weighted sum is the controller's stop-value heuristic. This value does not authorize a proof. Certificate reasons are deduplicated, and acceptance requires no unresolved reason and at least one supported hypothesis. If the highest-weight hypothesis has weight at least 0.5 but is unsupported, its reasons are retained as well. The defensive runtime predicate `_certificate_is_proven` checks both the certificate's `ok` flag and the existence of a supported hypothesis. Deterministically reconstructed aggregate answers replace actor wording only on this proven-certificate path.

E.2 GROUNDED REPAIR CANDIDATES AND RANKING

`plan_certificate_repair` turns certificate reasons and obligations into candidate actions on the current public interface. Candidate sources include restoring a previously assessed page, returning to the source collection, removing an unrequested query, establishing scope, setting filters or ordering, changing page size, following pagination, and opening a selected detail. Candidate fields record the concrete action, its kind and target label, expected predicate changes, rationale, reversibility, and eligibility for automatic execution.

Candidates are deduplicated by action and filtered against attempted actions and semantic signatures. The signature combines action kind, normalized label, and expected predicate changes, so regenerated element identifiers do not by themselves make the same failed repair a new candidate. Destructive-label candidates are filtered, and automatic detail opening requires visible identity evidence. Where navigation evidence already supports some predicates, a local repair is preferred to a route change that would discard them.

For a certificate with terminal hypotheses, candidate action a is ranked by estimated residual evidence value rather than benchmark reward:

$$J(a) = \rho_{k(a)} \sum_{h} w_h (1 - s_h) \mathbf{1}[a \text{ addresses } h] - c(a). \tag{E1}$$

Here w_h is a terminal-mode weight, s_h its current score, and $k(a)$ the candidate kind. The factor $\rho_k = (\alpha_k + u_k)/(\alpha_k + \beta_k + u_k + v_k)$ is updated from within-trajectory increases in certificate stop value: u_k counts improvements greater than 10^{-6} and v_k counts non-improvements. Its prior is $(19, 1)$ for restoring the best state, $(5, 1.5)$ for detail opening, $(6, 1.5)$ for restoring the source collection, and $(3, 1.5)$ otherwise. These updates use evidence scores, not the official evaluator. For a hypothesis-free certificate with a nonempty expected change, the implementation instead initializes the predicted gain from the candidate grounding score clipped to $[0, 1]$ before calibration.

The action penalty is 0.03 for sort controls, scope-route changes, refresh, and filter-field/operator/selector operations; 0.04 for detail opening, state restoration, and removing an unrequested filter; 0.05 for scope refinement, query changes, page-size changes, and collection navigation; 0.06 for pagination and source-collection restoration; and 0.07 for a collection-filter action. Unlisted kinds use 0.08. Nonreversible candidates receive an additional 0.25 penalty. These quantities are dimensionless decision heuristics, not latency or token-cost measurements. The plan retains the five highest-ranked candidates. Its conservative automatic-action selector requires reversibility, an automatic-safety flag, $J(a) > 0.02$, a grounding score of at least 0.94, and a margin of at least 0.04 over competing actions of the same kind.

E.3 Mapping Verdicts to Actions

The controller reevaluates the plan against remaining action capacity, the number of dispatched repairs, origin constraints, and hard contradictions. It is not a direct mapping from a confidence score to task success.

Branch	Condition and effect
Certified completion	A proven certificate ends the proof episode. Additional speculative repair does not override an already proven state.
Grounded repair	With remaining capacity, execute a reversible, admissible candidate whose value exceeds the controller threshold, or one that addresses a hard contradiction.
Delegation	Return explicit navigation or mutation obligations to the LLM policy when the corresponding verifier is enabled and delegation capacity remains.
Uncertified termination	Preserve the actor's stop under `fail_open`, for example when no useful grounded repair is available, capacity is exhausted, or an unresolved transaction makes generic navigation unsafe. This does not mark the certificate as proven.
Budget/loop abort	Stop the episode with an abort record and `ok=False`, rather than a successful completion certificate.

The controller's default repair-value threshold is 0.08; a hard contradiction can activate a repair below that threshold. Candidate admissibility is checked again before dispatch, so the planner's automatic-action suggestion is not the final authority. Default caps are five repair actions and five rejected-stop interventions, with two actor delegations for navigation and two for mutation. At the repair boundary, the implementation can reserve a final reversible restoration to the strongest non-refuted checkpoint. These are controller caps rather than the benchmark task's overall action limit.

For unresolved mutations and drafts, generic route-changing repair is withheld to avoid abandoning unsaved state. Mutation-CIV may delegate commit/readback obligations to the actor when permitted; otherwise the controller can terminate without certification. The implementation records `accept`, `repair`, `delegate`, `fail_open`, or an abort decision together with its reason. Consequently, a recorded stop and a proven completion certificate are different events. Proposition 1 applies to the proof-gated certificate path, not to every episode termination.

E.4 LLM Policy and Rule-System Interface

The AgentOccam integration wraps observation, action dispatch, status, and terminal processing. The policy consumes the objective, public accessibility observation, and system feedback describ-

ing unresolved obligations. It returns ordinary text actions such as `click [id]`, `type [id] [value] [1]`, `goto [url] [1]`, and `stop [answer]`. The runtime parses those actions, binds dispatched commands to their source snapshot, records outcomes, and intercepts terminal proposals for certificate construction.

A rejected stop becomes a no-op with certificate feedback, keeping the environment available for the next policy decision. A grounded automatic repair is passed through a private runtime capability and revalidated as a current admissible action; matching text copied from page content is insufficient to force execution. The actor preprocessing hook preserves system feedback when rebuilding its accessibility text. Conversely, ledger observation capture removes the system-feedback prefix when the browser's `RootWebArea` payload is available, so suggested answers and repair instructions are not treated as browser evidence.

When scope selection is delegated, a constrained LLM call chooses among currently visible namespace candidates. Its output is parsed and validated against those candidates before use. It does not generate the final certificate. Pre-dispatch rule guards can instead fill a visibly empty required field from an instruction literal, select a visible commit destination, redirect typing to the grounded writable editor, or block an action that would destroy unrequested document content. Unresolved mutation obligations remain actor-owned rather than being implemented by a generic navigation repair.