EVITRANS: COMPOSABLE STATE TRANSITIONS FOR VIDEO REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Video state tracking requires associating events with persistent entities and preserving their history-dependent effects. A fixed interval-level state change can become invalid when earlier records are revised. We present **EviTrans**, a framework that makes evidence-grounded, composable state transitions the reusable unit of video reasoning. Each interval is represented as an executable function of its entry state and boundary memory, rather than the change realized under one history, preserving state-dependent effects during temporal composition and revision. A typed observation-to-transition interface connects query requirements, visual entity binding, and event decoding to executable updates, while retaining evidence references and unresolved information. Integrating this representation with a persistent segment tree and deterministic query programs enables compatible questions to share an executable history and compiled records to be revised while preserving earlier versions. On VSTAT's 834 videos and 1,500 questions, the frozen, offline configuration achieves an overall score of 51.11, exceeding the matched direct video question answering baseline by 13.58 points, with unavailable inputs retained as failures. In backend diagnostics on 211 videos retaining 33 versions, persistent maintenance reduces edit-to-answer latency by 22.2% and retained memory by 52.1% relative to blocked sequential execution, while historical queries are slower under the measured caching policies.

1 INTRODUCTION

Video questions often concern the accumulated effects of events rather than the content of an individual frame (Girdhar & Ramanan, 2019). Counting how many distinct people performed both of two actions requires associating those actions with identities across time. Locating an object after repeated swaps requires preserving the order and consequences of earlier changes. These questions couple three responsibilities: observing visual evidence, maintaining historical state, and computing an answer. In direct video question answering (VQA), these responsibilities are handled within the model's answer-generation process (Bai et al., 2025; Zhang et al., 2024). Recent analysis on VSTAT identifies failures in event recognition, entity association, and state updating (Yu et al., 2026). This motivates an explicit representation of how visual evidence changes the state on which an answer depends.

Existing approaches structure video reasoning through symbolic representations, procedural execution, and hierarchical evidence organization. Neuro-symbolic methods connect visual perception and object dynamics with symbolic reasoning (Yi et al., 2019). Procedural VQA generates programs that invoke visual modules and compute answers (Choudhury et al., 2023). Hierarchical methods organize query-relevant frames and descriptions for language-model reasoning (Wang et al., 2025). These approaches provide mechanisms for representing visual content and performing question-level computation. The representation problem we address lies between these two stages: how to retain the effects of visual evidence on the predicted world state in a shared history that supports temporal composition, repeated queries, and subsequent revision.

Three gaps between evidence representation and state maintenance are central to this problem. First, the effect of visual evidence can depend on the preceding state. A video interval cannot always be summarized by a fixed, history-independent state change. For example, observing an object perform an action increases a distinct-object count only if that object has not been counted before. An

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.

unconditional count increment loses this dependency. Instead, the interval must specify a transition that can be applied correctly to different prior states (Yi et al., 2019). Second, visual predictions do not specify valid state changes. An object or action prediction must be associated with a persistent identity and interpreted as a new event, a continuing condition, or unresolved evidence (Choudhury et al., 2023). Without explicit update semantics, repeated observations can trigger duplicate updates, while missing evidence can be confused with an observed absence. Third, question-level execution does not provide a shared history-maintenance mechanism (Wang et al., 2025). Computing an answer and maintaining the history behind that answer are different operations. Reusing evidence across compatible questions or revising an earlier prediction requires executable dependencies between observations, state updates, and query results, together with a mechanism for retaining prior versions (Driscoll et al., 1986).

We introduce **EviTrans**, a framework that connects visual observation, historical state maintenance, and answer computation through *evidence-grounded, composable state transitions*. Our central contribution is a reusable interval representation: each interval retains an executable function over its entry state and the boundary memory required by the supported operations, rather than a fixed state change realized under one history, preserving history-dependent effects during temporal composition and revision. A typed query contract and a transition compiler connect visual predictions to identity-bound updates while retaining unresolved information. A world state executor and persistent segment tree maintain the resulting transition history, from which deterministic query programs compute answers. These components provide a common execution model for history-dependent reasoning, compatible-query reuse, and version-preserving replacement of predicted records.

EviTrans connects visual observation to state execution through a typed query contract, which specifies what to observe, what state to maintain, and how to query it. The visual state observer provides evidence-linked observations of relevant entities. Object binding and event decoding associate these observations with predicted entities and distinguish state changes from persistent conditions, while retaining unresolved evidence. The compiler converts the resulting changes into executable transitions by fixing evidence-derived operands while retaining entry-state-dependent reads and update conditions.

The World State Executor applies these transitions, while a standard persistent segment tree maintains their temporal composition as executable transformations rather than states precomputed under one history. Deterministic query programs compute answers from the predicted state, allowing compatible questions to reuse the same world representation. Replacing a compiled transition rebuilds only its path to the root while preserving earlier versions (Driscoll et al., 1986). Visual evidence, historical state, and answer computation are thus connected through executable records that support inspection and revision.

We evaluate a frozen EviTrans configuration on VSTAT, which comprises 834 videos and 1,500 questions (Yu et al., 2026). Under the official scoring rule, the evaluated configuration obtains an overall score of 51.11 (17.91 points above the matched direct Qwen3-VL-8B baseline and 6.71 points above the highest direct Gemini-3.1 Pro baseline), with 61.19% multiple-choice accuracy and a mean relative accuracy of 45.51 on numerical questions. The evaluation retains all 1,500 questions in the denominator, including 31 questions associated with unavailable inputs from 13 videos, which are scored as failures.

Our work makes three main contributions:

1. **History-aware composable state transitions.** We formulate video intervals as executable transitions over query-relevant states and their required memory, preserving entry-state-dependent effects during temporal composition and re-execution.

2. **Typed, evidence-grounded transition compilation.** We develop an observation-to-transition interface that connects constrained query requirements, visual entity binding, and state-change decoding, while retaining evidence references and unresolved information.

3. **Versioned world execution for video queries.** We integrate the transition representation with persistent segment tree maintenance and deterministic query programs, supporting evidence-linked answers, compatible-query reuse, and version-preserving replacement of predicted records.

2 Related Work

Visual state tracking and programmatic reasoning. Video state tracking benchmarks examine how models integrate events across time, covering compositional actions and object localization under occlusion (Girdhar & Ramanan, 2019), as well as state tracking across synthetic and real-world videos (Yu et al., 2026). On the modeling side, neuro-symbolic approaches connect visual perception and object dynamics with symbolic reasoning (Yi et al., 2019). Programmatic approaches organize visual operations around the question being answered by generating Python programs that invoke visual modules and compute answers (Surís et al., 2023; Choudhury et al., 2023). Related work in robotic question answering also connects visual observations with language-based reasoning to support task understanding and answer prediction in embodied settings (Sermanet et al., 2024; Ni et al., 2026).

EviTrans instead makes an evidence-linked interval transition the reusable unit of execution, preserving its dependence on the entry state across compatible queries and revisions.

Hierarchical video representations and structured memory. Trees support both temporal localization and selective access to video history. CPN uses a segment tree for coarse-to-fine temporal grounding (Zhao et al., 2021), whereas VideoTree constructs query-adaptive hierarchies of frames and captions for language-model reasoning (Wang et al., 2025). For longer histories, StreamForest organizes and compresses visual tokens into event-level memory trees (Zeng et al., 2025). Event-Causal RAG represents events and their preceding and resulting states in a graph, retrieving event chains and associated video evidence for answer generation (Yan et al., 2026). These structures organize visual features or semantic evidence for downstream prediction.

EviTrans builds on persistent data-structure design (Driscoll et al., 1986), using a segment tree to maintain versioned compositions of executable transitions. Query programs then compute answers directly from the resulting state.

Event semantics and state maintenance. Event-based formalisms give explicit semantics to changes over time. Event Calculus derives time-dependent facts from event descriptions and accommodates new information about past events (Kowalski & Sergot, 1986). Differential dataflow propagates input changes through dependent computations (McSherry et al., 2013), while persistent data structures retain access to earlier versions through structural sharing. These works establish foundations for temporal reasoning, incremental updates, and version management.

EviTrans connects these foundations to visual evidence by compiling identity-bound observations into composable interval transitions that retain entry-state dependencies and unresolved information. Its persistent tree shares unaffected subtrees across versions, avoiding full-tree reconstruction and copying when a compiled transition is replaced.

3 Methodology

As illustrated in Figure 1, EviTrans separates visual observation, historical state maintenance, and answer computation through evidence-grounded, composable state transitions. A typed query contract determines the observation requirements and execution semantics. Visual evidence is bound to entities and decoded into events, then compiled into interval transitions. The World State Executor applies these transitions, while a persistent segment tree maintains their temporal composition and versions. Deterministic query programs compute answers from the resulting predicted world.

3.1 Typed Query Contract and World State

The same observed action can support different computations: counting occurrences, counting distinct participants, or locating a participant after a sequence of changes. These computations require different state fields and update rules. We specify the observation requirements and state semantics before extracting evidence, so that observation and execution operate on the same definitions.

Let $V = ((x_i, \tau_i))_{i=1}^{N}$ denote a video represented by N frames, where x_i is frame i and τ_i is its timestamp, and let q denote the original question, including answer options when provided. The

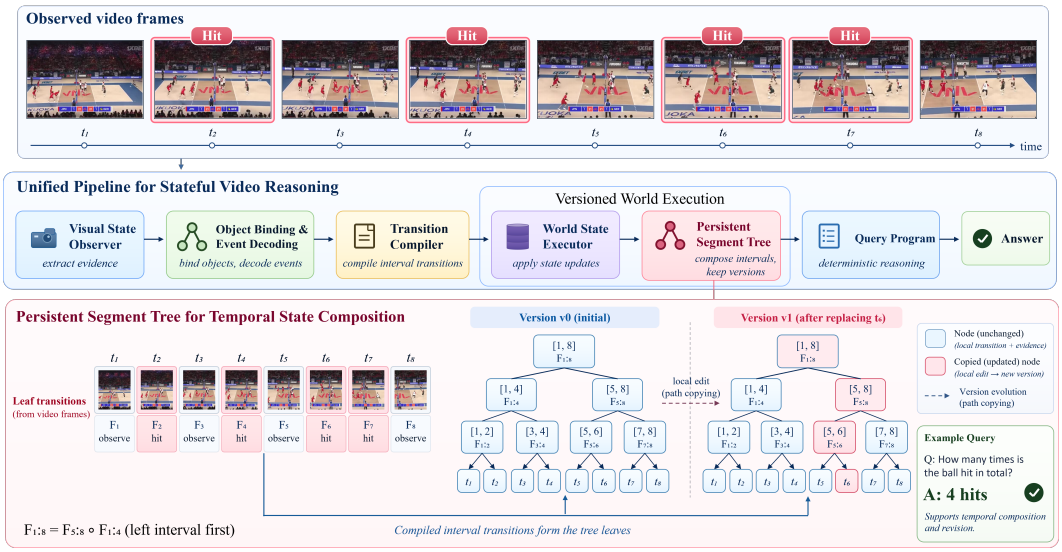


Figure 1: **Overview of EviTrans. Top:** Selected frames from an illustrative volleyball sequence, with four hit events highlighted. **Middle:** Visual observation, object binding, and event decoding supply evidence-grounded transitions to versioned world execution, followed by deterministic query answering. **Bottom:** Tree leaves store compiled interval programs and evidence references; internal nodes maintain chronological compositions, such as $F_{1:8} = F_{5:8} \circ F_{1:4}$, with the left interval executed first. Replacing the sixth compiled record creates a new version by copying the modified leaf and its ancestors. Unaffected subtrees are shared across versions, preserving the previous history; they are drawn separately for clarity.

processing timeline contains T chronologically ordered intervals $I_1, \ldots, I_T$. From the question semantics, EviTrans constructs a *typed query contract*

$$\mathcal{C}_q = (\mathcal{R}_q, \Sigma_q, \mathcal{A}_q, \pi_q), \tag{1}$$

where $\mathcal{R}_q$ contains the entity references to resolve, Σ_q specifies the state fields and their types and temporal meaning, $\mathcal{A}_q$ specifies the required visual observations, and π_q is a deterministic query program. The contract instantiates program-defined fields and operations for set updates, counters, permutations, and history queries. For the remainder of the section, we write $\Sigma = \Sigma_q$ for the active state schema.

Let $\mathcal{S}_\Sigma$ be the space of admissible states under Σ, including states with unresolved information. We write the predicted world state after interval t as

$$s_t = (w_t, h_t, u_t) \in \mathcal{S}_\Sigma, \tag{2}$$

where w_t contains the query-relevant world fields, h_t contains the boundary memory required to interpret subsequent observations, and u_t retains unresolved fields and evidence. This decomposition distinguishes the quantities being queried from the memory used to maintain them. World fields may include entity attributes, membership sets, counters, positions, and query-relevant historical records; boundary memory records conditions such as whether an event is already ongoing. The initial state s_0 is determined by the allowed question information, the task's initialization rules, and visual evidence. A zero counter or empty set is initialized when its scope establishes that value; an unobserved initial relation remains `UNKNOWN`.

3.2 Visual Evidence and State-Change Decoding

A visual observation becomes useful for state tracking only when its participants and temporal meaning are established. Repeated observations of one continuing condition should preserve that condition, whereas a new event may require an update. We retain entity references, temporal support, and evidence sources so that event decoding can distinguish these cases without treating missing observations as negative evidence.

The *Visual State Observer* extracts query-relevant evidence using a frozen vision-language model, complemented by geometric processing and frozen action experts where applicable. It produces local, structured observations of the entities, attributes, relations, and actions specified by $\mathcal{A}_q$, retaining their temporal support, participant references, and evidence sources. Let $\mathcal{O}_t$ denote the observations assigned to interval I_t. Each observation is represented as

$$o = (\theta, \mathbf{r}, p, z, \rho), \tag{3}$$

where θ is its temporal support, $\mathbf{r}$ is an ordered tuple of participant references, p is an observed property, relation, or action predicate, z is its typed value or `UNKNOWN`, and ρ identifies its evidence sources, including frame or clip references and the observation procedure. Processing is offline: evidence from multiple times can support relation calibration and event confirmation. Interval assignment records when a fact applies, while its supporting evidence may extend beyond that interval.

Object binding. Let $\mathcal{M}$ contain the entity mentions in the question and the observations, and let $\mathcal{E}$ be the set of predicted visual entities. Object binding associates mentions with these entities:

$$\beta : \mathcal{M} \to \mathcal{E} \cup \{\texttt{UNKNOWN}\}. \tag{4}$$

Here β denotes the binding map, and an unresolved binding retains its observation and evidence. Binding makes observations at different times refer to a common entity when the available evidence supports that association.

Event decoding. Using $\mathcal{O}_t$, the bindings, and the required entry-state information, event decoding interprets observations through state boundaries and transition preconditions. Confirmed persistence leaves the corresponding world field unchanged, while a supported change produces an update. Boundary memory is carried between intervals so that a continuing condition is not counted as a fresh onset at each boundary. Decisions whose outcomes depend on the entry state are retained as conditions in the compiled transition. Missing observations, ambiguous participants, and unresolved preconditions remain explicit in u_t; they are distinct from a confirmed negative observation or a confirmed absence of change.

3.3 History-Aware Transition Compilation

An interval must remain executable when its incoming history changes. Storing only the result obtained under one history would discard the conditions that produced that result. The transition compiler fixes evidence-derived operands while retaining state-dependent reads and update conditions, so the same interval can be applied to different admissible entry states.

For interval t, the *transition compiler* produces an ordered program $P_t = (a_{t,1}, \ldots, a_{t,m_t})$, where m_t is the number of operations and $a_{t,j}$ is operation j. Each operation carries its operation type, bound operands, transition preconditions, and evidence references. Let $f_{t,j}$ denote the deterministic state-update function implementing $a_{t,j}$ through its domain adapter. It takes an input state s and returns the updated state $s' = f_{t,j}(s)$, with both states belonging to $\mathcal{S}_\Sigma$. The interval transition is

$$F_t = f_{t,m_t} \circ \cdots \circ f_{t,1}, \qquad s_t = F_t(s_{t-1}), \tag{5}$$

where $\circ$ denotes function composition and the rightmost operation executes first. A program with no required update implements the identity transformation; unresolved evidence is represented explicitly rather than removed to obtain an empty program. Historical records and boundary memory are updated within the same state interface.

Entry-state-dependent effects. Consider a predicate p for an action of interest within the temporal scope specified by the contract. Let $M_p \subseteq \mathcal{E}$ be the resolved set of entities already counted for p, and let $c_p = |M_p|$ be its associated distinct-entity count. When the evidence confirms that entity $e \in \mathcal{E}$ performs the action, the operation has the semantics

$$M_p' = M_p \cup \{e\}, \qquad c_p' = c_p + \mathbf{1}[e \notin M_p], \tag{6}$$

where primes indicate updated values and $\mathbf{1}[\cdot]$ is the indicator of a true condition. The evidence fixes e and p, while the incoming M_p determines the count change. The compiler retains this membership update rather than replacing it with the increment realized during one execution. Unresolved entity identity or membership is handled through the state's unresolved-information representation.

Domain adapters specify operand types, state-field access, and unresolved-information handling for the common state-to-state interface (Appendix C.1).

3.4 World State Executor and Temporal Composition

Combining intervals requires preserving both event order and the state passed across their boundaries. We compose interval transitions as functions over the complete maintained state, including boundary memory and unresolved information. A node represents how its interval transforms an entry state, allowing execution to use the history supplied by its preceding interval.

The *World State Executor* applies the domain-adapted operations of each P_t. For interval indices $1 \le l \le r \le T$, define

$$F_{l:r} = F_r \circ F_{r-1} \circ \cdots \circ F_l, \tag{7}$$

so $F_{l:r}$ maps the state before interval l to the state after interval r. For an empty interval range, $F_{l:r}$ is the identity map id_Σ, defined by $\mathrm{id}_\Sigma(s) = s$ for $s \in \mathcal{S}_\Sigma$.

A persistent segment tree indexes the T interval records. Leaf t stores P_t and its evidence references Λ_t. An internal node covering $[l, r]$ maintains a composed program or a supported semantics-preserving summary. With midpoint $m = \lfloor (l + r)/2 \rfloor$, its transition satisfies

$$F_{l:r} = F_{m+1:r} \circ F_{l:m}. \tag{8}$$

Thus the right child executes on the state produced by the left child. A stored summary must preserve this function on admissible entry states, including the effects on memory and unresolved information. The tree maintains transformations rather than states precomputed under one particular incoming history.

Execution consistency. Under these semantics, tree execution equals chronological sequential execution:

$$F_{1:T}(s_0) = F_T\big(F_{T-1}(\cdots F_1(s_0)\cdots)\big). \tag{9}$$

The identity holds for fixed predicted transitions; the inductive derivation is in Appendix C.2.

3.5 Persistent Maintenance and Revision

A correction to an earlier record can alter later states without changing the meaning of every later transition. Retaining those transitions as functions separates revision of the stored history from re-execution on the revised state. Persistence additionally preserves earlier histories for inspection by sharing the records and subtrees unaffected by a replacement.

Let v index successive versions and $\mathcal{T}^{(v)}$ denote the tree for version v. Compiled-record replacement uses fixed interval indexing, state schema, entity bindings, and initial state. Replacing leaf k, where $1 \le k \le T$, with a revised program $\widetilde{P}_k$ and evidence references $\widetilde{\Lambda}_k$ creates a new leaf and copies its path to the root. The new tree $\mathcal{T}^{(v+1)}$ shares all other subtrees with $\mathcal{T}^{(v)}$, whose records remain unchanged.

Let $\widetilde{F}_k$ be the transition implemented by $\widetilde{P}_k$, and let the superscript (v) identify transitions in version v. The revised full-history transition is

$$F^{(v+1)}_{1:T} = F^{(v)}_{k+1:T} \circ \widetilde{F}_k \circ F^{(v)}_{1:k-1}. \tag{10}$$

The unchanged suffix is applied to the revised state produced by the prefix and replacement, so its state-dependent effects are recomputed under that history. Equation 10 gives the same execution as sequentially replaying the revised programs. Queries on the previous tree retain the previous execution because its programs and evidence records are unchanged. When new visual evidence changes bindings or other compiled records, the affected records are recompiled and their corresponding tree paths are updated.

Maintenance cost. For $T \ge 2$, a compiled-leaf replacement rebuilds $O(\log T)$ nodes in $O(C_{\mathrm{node}} \log T)$ time, where C_{node} includes creating or rebuilding a node's stored program or summary. This structural bound excludes state execution, querying, and any visual re-observation or recompilation. Retained-version storage and representation-dependent execution costs are detailed in Appendix C.3.

3.6 Deterministic Query Execution

For fixed $\mathcal{C}_q$, β, and s_0, the predicted world is $\mathcal{W}^{(v)} = (\mathcal{C}_q, \beta, s_0, \mathcal{T}^{(v)})$. The deterministic program returns $\widehat{y}_q^{(v)} = \pi_q(\mathcal{W}^{(v)})$, leaving the answer `UNKNOWN` when required information cannot be resolved. An interval query preserves its preceding history through

$$s_r^{(v)} = F_{l:r}^{(v)}\big(s_{l-1}^{(v)}\big), \qquad s_{l-1}^{(v)} = F_{1:l-1}^{(v)}(s_0). \tag{11}$$

Further queries reuse this world only when their entity bindings, state fields, observation requirements, and temporal semantics are covered. Full interfaces, query examples, and evidence inspection are in Appendix C.4.

4 Experiments

We evaluate question answering, history-dependent execution, and version-maintenance efficiency.

4.1 Experimental Setup

Benchmark and inputs. We evaluate the frozen EviTrans configuration on VSTAT (Yu et al., 2026), comprising 834 videos and 1,500 original questions. Valid inputs are available for 821 videos and 1,469 questions. The remaining 31 questions associated with 13 unavailable videos receive zero credit in the full denominator. Inference uses videos, timestamps, and permitted question information, including supplied answer options. Official answers are used only for scoring.

Implementation. The Visual State Observer combines frozen Qwen3-VL-8B-Instruct (Bai et al., 2025) with geometric processing and frozen action experts where applicable. Observation is performed offline, allowing cross-time evidence for relation calibration and event confirmation. The typed query contract, transition compiler, World State Executor, and deterministic query programs follow Section 3. Selected visual-inference, decoding, runtime, and contract-interface settings are summarized in Appendix A.

Quality-based stratification. All 1,469 available-input questions follow the same execution pipeline. An independent, answer-blind audit of frozen front-end evidence defines the 356-question high-quality subset (211 videos); the other 1,113 questions form the lower-quality group. Inclusion requires every applicable completeness, grounding, binding, and temporal-interpretation criterion to pass. Auditors do not access reference answers, system answers, executed state trajectories, or evaluator scores. Grouping affects analysis only, membership is fixed across component comparisons, and unresolved outputs remain in the denominator. Appendix B.2 specifies the audit protocol.

Baselines and comparison protocol. Table 1 distinguishes published benchmark reference scores from our completed matched reruns of direct Qwen3-VL-32B, ProViQ-based, and VideoTree-based VQA. Published references retain their original settings. Our reruns used the same input-availability manifest and official evaluator, with matched visual-input and model-call budgets. These matched system-level comparisons are distinct from the observation-matched component comparisons in Section 4.3. Baseline adaptations and measured resource consumption are summarized in Appendix A.4.

Metrics and statistical reporting. We use the official evaluator and report the overall score, multiple-choice (MC) accuracy, and numerical mean relative accuracy (MRA), all on a 0–100 scale. The overall score averages official per-question scores over the full evaluation population. For the high-quality subset, resolved-answer coverage is the fraction of queries returning an answer other than `UNKNOWN`. Backend correctness checks include unresolved outputs. Backend timing and memory measurements follow Section 4.4.

4.2 Main Results

Table 1 reports an overall score of 51.11, with 61.19% MC accuracy and 45.51 numerical MRA. The overall score exceeds the published Gemini-3.1 Pro (low) reference by 6.71 points.

Table 1: VSTAT results.

Method	Overall	MC accuracy	Numerical MRA
Qwen3-VL-8B-Instruct	33.2	–	–
Gemini-3.1 Pro (low)	44.4	–	–
Qwen3-VL-32B	37.53	42.72	34.64
ProViQ	18.91	24.44	15.84
VideoTree	26.44	37.50	20.29
EviTrans	**51.11**	**61.19**	**45.51**

Table 2: Component analysis on the fixed 356-question high-quality subset. Overall scores aggregate the modified subset predictions with unchanged EviTrans predictions for the other 1,113 available-input questions and zero credit for 31 unavailable-input questions. A–D use the full Visual State Observer. E uses a frozen vision-language model (VLM)-only observation adapter with the same persistent backend and query programs.

Variant	Configuration	Overall	Subset
A	Chronological facts; LM answering	47.59	72.53
B	Explicit predicted state; LM answering	50.57	85.06
C	Sequential execution; query programs	**51.11**	**87.36**
D	Persistent segment tree; query programs	**51.11**	**87.36**
E	VLM-only observer; persistent segment tree	30.58	0.84

The high-quality subset, identified by the observation-quality threshold, contains 356 questions from 211 videos and scores 87.36. Of these queries, 345 produce execution answers and 11 remain unresolved. The remaining 1,144 questions score 39.83; this complement contains 1,113 lower-quality questions processed by the same execution pipeline and 31 unavailable-input questions scored as failures. All subset comparisons retain the same question identifiers and quality-group membership. The assessment procedure and quality-stratified results are documented in Appendix B.

4.3 Component Analysis

We compare component variants on the fixed high-quality subset and report both subset and full-benchmark scores (Table 2). The component interventions affect only these 356 questions. For the other 1,113 input-available questions, predictions from the unchanged unified EviTrans configuration are reused; the 31 unavailable-input questions retain zero credit. Quality-group membership is fixed across variants. Variants A–D use identical cached observations. A and B use the same frozen 32B language model (LM), while B and C supply the answer stage with identical predicted states and historical records. Contracts, question inputs, and unresolved-output handling remain fixed.

Explicit state maintenance (B versus A) improves the subset score by 12.53 points, corresponding to approximately 2.97 points overall. Deterministic query execution (C versus B) adds 2.30 subset points. Sequential replay (C) matches the persistent segment tree (D) on every question; revision and version-maintenance efficiency are evaluated in Section 4.4. The VLM-only adapter (E) produces valid answers for only 5/356 queries, reflecting an observation-adaptation failure rather than isolating the contribution of auxiliary experts. Additional mechanism audits, changed-entry-state and interval-boundary diagnostics, and complete-observation evaluations of operation combinations and longer histories are reported in Appendices D and E.

4.4 Persistent Maintenance

We compare backends on the 211 videos and 356 queries above, using 13–128 leaf records per video and 32 changed-record replacements per video (6,752 total). Version-retaining backends keep 33 versions, including the initial version. Edit-to-answer time includes new-leaf compilation, maintenance, state execution, and all original queries for the video. We take five-repeat medians per video, then average across videos. Interval application starts from an identical supplied entry state; entry-state retrieval is excluded. Memory denotes the total size of reachable Python objects, with

Table 3: Measured backend efficiency. Time is in milliseconds; memory is MiB per video.

Backend	Edit-to-answer	Interval application	Memory
Blocked sequential	0.2350	0.0772	1.730
Mutable segment tree	0.1793	0.0634	0.473
Full-index snapshots	0.2041	0.0632	1.822
Persistent segment tree	0.1828	0.0632	0.828

each object counted only once, rather than process resident memory. Full-index snapshots share immutable programs and evidence. Detailed measurement and caching policies are in Appendix F.

Relative to blocked sequential execution, persistence reduces edit-to-answer time by 22.2% and retained memory by 52.1%. Relative to full-index snapshots, the reductions are 10.4% and 54.6%; editing is 2.0% slower than with the current-version-only mutable tree. Historical-answer latency is 0.1427 ms versus 0.0322 ms for blocked sequential execution under the measured caching policies. All version-retaining backends pass reported historical-query checks, including `UNKNOWN`. Timing decomposition, expanded correctness audits, and measured scaling and cache-aligned workload results are reported in Appendix F. Shared-world query reuse is reported separately in Appendix G.

5 Limitations and Future Work

EviTrans is limited by visual-observation and entity-binding errors, program-defined state fields and operations, and offline cross-time evidence. Complete-observation diagnostics cover new operation combinations and longer histories, but do not establish end-to-end transfer to independent video benchmarks (Appendix E). Streaming operation requires event-confirmation and revision policies for arriving observations.

Backend gains depend on workload and caching: in the base suite, persistence reduces edit-to-answer latency and retained-version memory, but increases historical-query latency. Larger operator-trace results do not establish these gains on longer original videos (Appendix F). Revision timings include new-leaf compilation, maintenance, execution, and answering, but not visual re-observation or dependent-record recompilation after evidence or binding changes. Improved front-end evidence, compact semantics-preserving summaries, and selective state caching remain future directions.

6 Conclusion

We presented EviTrans, a framework that connects visual observation, historical state maintenance, and answer computation through evidence-grounded, composable state transitions. A typed query contract specifies observation and computation requirements, while the transition compiler preserves entry-state dependencies. The World State Executor, persistent segment tree, and deterministic query programs provide a shared execution model for querying and revising predicted histories. The evaluated frozen configuration achieves an overall score of 51.11 on all 1,500 VSTAT questions. Backend diagnostics show lower edit-to-answer latency and retained-version memory than blocked sequential execution, with a trade-off in historical-query latency. EviTrans makes the connection between visual evidence, state changes, and answers executable, supporting inspection, compatible-query reuse, and version-preserving revision.

AI Use Statement

In this work, we used generative AI tools for Language polishing. We have not used generative AI tools for experiment design. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Rohan Choudhury, Koichiro Niinuma, Kris M Kitani, and László A Jeni. Zero-shot video question answering with procedural programs. *arXiv preprint arXiv:2312.00937*, 2023.

James R Driscoll, Neil Sarnak, Daniel Dominic Sleator, and Robert Endre Tarjan. Making data structures persistent. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pp. 109–121, 1986.

Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions and temporal reasoning. *arXiv preprint arXiv:1910.04744*, 2019.

Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New generation computing*, 4(1):67–95, 1986.

Frank McSherry, Derek Gordon Murray, Rebecca Isaacs, and Michael Isard. Differential dataflow. In *CIDR*, 2013.

Tengjun Ni, Xin Yuan, Shenghong Li, Kai Wu, Ren Liu, Wei Ni, and Wenjie Zhang. Bolt: Decision-aligned distillation and budget-aware routing for constrained multimodal qa on robots. In *International Conference on Learning Representations*, volume 2026, pp. 2604–2626, 2026.

Pierre Sermanet, Tianli Ding, Jeffrey Zhao, Fei Xia, Debidatta Dwibedi, Keerthana Gopalakrishnan, Christine Chan, Gabriel Dulac-Arnold, Sharath Maddineni, Nikhil J Joshi, et al. Robovqa: Multimodal long-horizon reasoning for robotics. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 645–652. IEEE, 2024.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11854–11864. IEEE, 2023.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 3272–3283, June 2025.

Peizheng Yan, Yu Zhao, Liang Xie, Juntong Qi, Mingming Wang, and Erwei Yin. Event-causal rag: A retrieval-augmented generation framework for long video reasoning in complex scenarios. *arXiv preprint arXiv:2605.06185*, 2026.

Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, Antonio Torralba, and Joshua B Tenenbaum. Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*, 2019.

Sihyun Yu, Nanye Ma, Pinzhi Huang, Hyunseok Lee, Shusheng Yang, June Suk Choi, Ellis Brown, Oscar Michel, Boyang Zheng, Jinwoo Shin, and Saining Xie. Benchmarking visual state tracking in multimodal video understanding. *arXiv preprint arXiv:2606.03920*, 2026.

Xiangyu Zeng, Kefan Qiu, Qingyu Zhang, Xinhao Li, Jing Wang, Jiaxin Li, Ziang Yan, Kun Tian, Meng Tian, Xinhai Zhao, Yi Wang, and Limin Wang. Streamforest: Efficient online video understanding with persistent event memory. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 75804–75835. Curran Associates, Inc., 2025. doi: 10.52202/085713-2547. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6dd91fec726dbed8915a1fbadd91d1d2-Paper-Conference.pdf.

Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024.

Yang Zhao, Zhou Zhao, Zhu Zhang, and Zhijie Lin. Cascaded prediction network via segment tree for temporal video grounding. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4195–4204. IEEE, 2021.

Table 4: Selected visual inference and runtime settings.

Item	Setting
Vision-language model	Frozen Qwen3-VL-8B-Instruct; BF16.
Image resolution	Aspect-ratio-preserving resize; dimensions divisible by 32; image area $\leq 512^2$ pixels.
Inference	Offline; greedy decoding; one beam; PyTorch random seed 41.
Hardware	RTX 5090 (32 GB).
Software	Reconstructed environment record: Ubuntu 24.04.2 LTS; Python 3.11.15; PyTorch 2.9.1+cu128; Transformers 4.57.6.

Table 5: Selected contract, prompting, and execution settings.

Artifact	Evaluated specification
Contract construction	Deterministic parsing of question stems into a fixed typed-contract catalogue.
State vocabulary	Task-specific entities and fields; assignment, counting, reset, permutation, set updates, and history append.
Observation prompts	Request local facts and participant identities, not cumulative answers. Answer options are excluded from observation prompts; reference answers are used only for scoring.
Output handling	Short JSON records or enumerated labels with validity checks. Invalid or truncated outputs are rejected and failures recorded; no automatic semantic repair or retry.
Binding and memory	Task-specific identity association and event rules. Boundary state and unresolved evidence are retained; ambiguous identities remain `UNKNOWN`.
Query execution	Fixed typed queries over the maintained state. Results map to the original answer options or numerical output; queries abstain when required information remains unresolved.

A Implementation and Evaluation Details

A.1 Evaluated Configuration

The configuration record separates visual inference, task adaptation, and state execution. The evaluated Visual State Observer uses frozen Qwen3-VL-8B-Instruct (Bai et al., 2025), complemented by geometric processing and frozen action experts where applicable. The typed query contract, transition compiler, World State Executor, and deterministic query programs use the interfaces in Section 3. Observation is offline: evidence supporting relation calibration and event confirmation can span multiple intervals.

Contract and observation records. The typed query contract specifies entity references, state fields, observation requirements, and a query program. Structured observations retain temporal support, participant references, typed values, and evidence sources. The implementation record includes the exact program-defined vocabulary and its mapping to domain adapters. An interval's temporal assignment and the temporal extent of its supporting evidence are recorded separately.

A.2 Input Availability and Evaluation Manifest

The evaluation uses the official VSTAT collection of 834 videos and 1,500 original questions (Yu et al., 2026). Valid inputs cover 821 videos and 1,469 questions. The remaining 13 videos correspond to 31 questions; these questions receive zero credit and remain in the full denominator. Inference receives video frames, timestamps, and permitted question information, including supplied options. Official answers are used for scoring.

The evaluation manifest associates each question with its video, input-availability status, processing route, prediction, and evaluator output. Missing inputs, unresolved execution, and answer parsing failures are recorded as separate conditions.

Table 6: Evaluation population and input availability.

Population	Videos	Questions
Official collection	834	1,500
Available inputs	821	1,469
Unavailable inputs	13	31
high-quality subset	211	356

A.3 Answer Metrics and Statistical Reporting

Let $\mathcal{Q}$ be an evaluation set of questions, y_q the official answer to question $q \in \mathcal{Q}$, and $\widehat{y}_q$ the system output. Let $g_q \in [0,1]$ be the score assigned by the frozen official evaluator to $(q, \widehat{y}_q, y_q)$. The overall score is

$$S(\mathcal{Q}) = \frac{100}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} g_q. \tag{12}$$

Multiple-choice questions use accuracy, and numerical questions use the official mean relative accuracy (MRA) calculation. Numerical parsing, relative-error thresholds, and zero-target handling follow the official evaluator. Its nonnumerical accuracy group can also contain text-match items. The main result reports MC accuracy of 61.19 and numerical MRA of 45.51. Scores are aggregated from individual questions within each reported population.

Execution resolution. Let $\mathcal{Q}_{\mathrm{res}}$ contain questions for which execution returns a valid typed answer other than `UNKNOWN`. Resolved-answer coverage is

$$C_{\mathrm{res}}(\mathcal{Q}) = 100 \frac{|\mathcal{Q}_{\mathrm{res}}|}{|\mathcal{Q}|}. \tag{13}$$

The score in equation 12 and this coverage measure are reported separately. Execution traces retain `UNKNOWN`.

A.4 Baseline Adaptations and Resource Accounting

Published benchmark reference scores retain their original reported settings. We completed matched reruns of direct Qwen3-VL-32B video question answering, the ProViQ-based baseline, and the VideoTree-based baseline. The reruns used the same input-availability manifest and official evaluator, with matched visual-input and model-call budgets. Baseline-specific adaptations were recorded separately from the original methods. All evaluations retained the full denominator of 1,500 questions, with the 31 unavailable-input questions receiving zero credit.

Programmatic video question answering. The ProViQ-based baseline executed question-specific programs that invoked visual tools and computed answers from the returned evidence. Its measured resource consumption is reported in Table 7.

Hierarchical evidence organization. The VideoTree-based baseline organized visual evidence in a query-adaptive hierarchy for question answering. Its measured resource consumption is reported in Table 7.

Resource measurements. We recorded unique decoded frames, total frame presentations to models, model calls, generated tokens, expert invocations, wall time, and peak accelerator memory. Table 7 summarizes the measured model calls, generated tokens, wall time, and peak accelerator memory. Wall time is reported in seconds per video, and peak accelerator memory in GiB. Per-question and per-video costs were accounted for under each method's reuse policy. End-to-end timing includes visual inference and answer computation, including compilation and state execution where applicable. Budget matching controls the permitted computation; the reported resource measurements reflect each method's actual consumption.

Table 7: Measured resource consumption of the completed matched reruns. Wall time is reported in seconds per video; peak accelerator memory is reported in GiB.

Method	Model calls	Generated tokens	Wall time (s/video)	Peak memory (GiB)
Direct VQA	1,469	3,070	16.23	27.29
ProViQ-based question answering	94,153	3,791,707	36.70	24.32
VideoTree-based question answering	28,424	1,695,639	26.93	26.75
EviTrans	5,345	197,249	14.42	27.25

Table 8: VSTAT answer scores for the full benchmark, the high-quality subset, and the remaining questions. All scores are on a 0–100 scale. The high-quality subset includes its 11 unresolved queries. The remaining population includes 1,113 lower-quality questions and 31 unavailable-input questions, with the latter scored as failures.

Population	Questions	Overall	MC accuracy	Numerical MRA
Full benchmark	1,500	51.11	61.19	45.51
high-quality subset	356	87.36	99.33	78.64
Remaining questions	1,144	39.83	46.37	36.50

Observation-matched component comparisons. The component comparisons in Section 4.3 used a shared observation cache for variants A–D, with fixed question inputs, contracts, and quality-group membership. These within-system comparisons are separate from the matched baseline reruns, which evaluate each method's own evidence-processing and answer-computation pipeline.

B Additional Results and Quality-Stratified Evaluation

B.1 Full-Population and Subset Results

Let $\mathcal{Q}$ denote all 1,500 questions, $\mathcal{Q}_{\text{avail}}$ the 1,469 questions with available inputs, and $\mathcal{Q}_{\text{miss}}$ the 31 unavailable-input questions. Using the observation-quality score κ_q and threshold η defined in Appendix B.2, the quality groups are

$$\begin{aligned} \mathcal{Q}_{\text{H}} &= \{q \in \mathcal{Q}_{\text{avail}} : \kappa_q \geq \eta\}, \\ \mathcal{Q}_{\text{L}} &= \mathcal{Q}_{\text{avail}} \setminus \mathcal{Q}_{\text{H}}. \end{aligned} \tag{14}$$

The high-quality subset $\mathcal{Q}_{\text{H}}$ contains 356 questions from 211 videos, and the lower-quality subset $\mathcal{Q}_{\text{L}}$ contains 1,113 questions. Both subsets use the same execution pipeline. Let $\mathcal{Q}_{\text{R}} = \mathcal{Q}_{\text{L}} \cup \mathcal{Q}_{\text{miss}}$ denote the remaining 1,144 questions in the reported complementary population. Thus $\mathcal{Q} = \mathcal{Q}_{\text{H}} \cup \mathcal{Q}_{\text{R}}$, with disjoint question sets; their video sets can overlap.

The complete score is the question-weighted combination of the two subset scores. Applying their reported rounded values gives 51.11045, consistent with the reported 51.11. All component and backend comparisons on $\mathcal{Q}_{\text{H}}$ retain these same question identifiers, including unresolved cases.

B.2 Answer-Blind Observation-Quality Assessment

Purpose and information boundary. The quality partition characterizes the evidence supplied by the full Visual State Observer; it does not select an inference route. Let $\mathcal{Q}_{\text{avail}}$ denote the questions with available video inputs, and let Z_q contain the frozen observations, entity bindings, event-decoding records, and evidence references associated with question q. The audit examines these front-end records rather than the states or answers subsequently produced by execution.

Auditors receive the original video, the question stem without answer options, and the observation-facing contract fields

$$C_q^{\text{obs}} = (R_q, \Sigma_q, A_q).$$

They do not receive reference answers, system answers, executed state trajectories, evaluator scores, or correctness labels. Audit annotations are used only to assign analysis groups and are never supplied to the transition compiler, executor, or answer program.

Independent evidence inventory. Before inspecting Z_q, auditors construct a reference inventory of the local evidence required by the question and its observation contract. The inventory covers the required initial observations, relevant entities, attributes, relations, and event occurrences over the specified temporal scope, including preceding observations needed to interpret that scope. It records local facts and occurrences rather than cumulative answers.

The inventory is constructed from the video rather than from the system's predicted event list. Consequently, an event omitted by the observer remains an auditable omission. A missing detection is not treated as a confirmed absence. When the video does not establish a required fact or identity, the corresponding item is marked unverified rather than assigned a guessed value.

Audit criteria. For each question, the audit assigns four binary indicators $c_q^d \in \{0, 1\}$, with $d \in \{\mathrm{cov}, \mathrm{grd}, \mathrm{id}, \mathrm{tmp}\}$:

Observation completeness (cov)**.** Every required local fact and event in the independent inventory is represented in the front-end records with sufficient information for the declared observation requirement. A missing required event, initial observation, or participant causes this criterion to fail. An empty event record passes only when the required temporal scope is covered by evidence supporting the absence of relevant events.

Visual grounding (grd)**.** Every resolved, query-relevant assertion is supported by its cited visual evidence, including its predicate and typed value. A valid evidence pointer alone is insufficient. Unsupported assertions, incorrect values, and spurious events cause this criterion to fail.

Entity binding (id)**.** The identity associations required by the contract agree with the visible entities and their cross-time correspondences. Incorrect participant assignments, identity switches, and inappropriate merging or splitting of identities cause this criterion to fail. A required identity that cannot be verified does not pass.

Temporal interpretation (tmp)**.** The decoded records preserve the occurrence, temporal assignment, and ordering of the required events and conditions. Repeated observations of one continuing condition are permitted, but they must not be decoded as separate event occurrences. Distinct occurrences must not be merged.

Temporal assignments are checked at the resolution of the frozen processing intervals. For a point event, the assigned interval must contain its independently annotated occurrence time. Intervals are half-open, except for the final interval, which includes its endpoint. For a continuing condition, onset, persistence, and termination are checked at the same resolution. When the contract requires finer temporal distinctions, the audit uses that declared resolution. Supporting evidence may extend beyond an event's assigned interval, consistent with offline observation. Uncertainty that prevents verification of a required temporal assignment is recorded as unverified.

Criterion applicability is determined from the observation contract before inspecting the system records. A genuinely inapplicable criterion is assigned one; a required but unverified criterion is assigned zero. State-dependent execution results are not used to judge any criterion.

Criterion-level assessment. Each applicable criterion is marked as pass, fail, or unverified, with supporting video references and a recorded reason. Its binary indicator is one only for a verified pass; failed or unverified criteria receive zero. These indicators determine the quality score and subset membership according to the selection rule below.

Score and selection rule. We define

$$\kappa_q = \min_{d \in \{\mathrm{cov},\mathrm{grd},\mathrm{id},\mathrm{tmp}\}} c_q^d, \qquad \eta = 1. \tag{15}$$

The quality groups are

$$\begin{aligned} \mathcal{Q}_H &= \{q \in \mathcal{Q}_{\mathrm{avail}} : \kappa_q \geq \eta\}, \\ \mathcal{Q}_L &= \mathcal{Q}_{\mathrm{avail}} \setminus \mathcal{Q}_H. \end{aligned} \tag{16}$$

Table 9: Additional VSTAT breakdowns. Coverage refers to membership in the high-quality subset.

Axis	Category	Questions	Score	Subset coverage (%)
Source	Synthetic	550	66.98	54.55
	Recorded	100	32.40	0.00
	YouTube	850	43.05	6.59
State element	Count	747	47.58	27.58
	Location	384	49.11	13.02
	Attribute	369	60.35	27.10
State structure	Atomic	710	48.21	25.92
	Sequence	211	60.24	23.70
	Set	249	57.83	20.08
	Dictionary	330	46.45	21.82

Thus, a question enters $\mathcal{Q}_H$ only when all applicable criteria pass. The score is an audit-pass indicator, not a calibrated probability of answer correctness. The complementary group includes both identified evidence failures and cases whose required evidence could not be verified. Unavailable inputs are recorded separately.

Evaluation and reproducibility. All available-input questions retain their original execution pipeline and predictions. The audit neither repairs observations nor changes the treatment of unresolved information. A final answer of `UNKNOWN` does not itself determine group membership and remains in the score denominator of its assigned group.

Membership is determined from the full observer and held fixed across component and backend comparisons, including variants with a different observation adapter. This is a conditional diagnostic of the full observer's evidence, not an estimate of performance on a representative benchmark subset. Full-benchmark results remain the primary evaluation.

The audit manifest records question and video identifiers, frozen record checksums, reference evidence, per-criterion decisions, decision reasons, rubric version, and final group membership. Reporting includes group sizes, answer scores, resolved-answer coverage, and composition by video source and task category.

B.3 Category Breakdown

Table 9 reports results by video source, state element, and state structure. Each category retains all of its questions, including unavailable inputs and unresolved outputs. Source, element, and structure are separate grouping axes.

C Transition Semantics and Execution Properties

C.1 State and Interval Semantics

We use the notation of Section 3. The active state schema is Σ, and $\mathcal{S}_\Sigma$ is its admissible state space. A state $s = (w, h, u)$ contains world fields w, boundary memory h, and unresolved information u. Interval t, with $t \in \{1, \dots, T\}$, contains an ordered compiled program P_t whose denotation is the deterministic transition $F_t : \mathcal{S}_\Sigma \to \mathcal{S}_\Sigma$. Evidence references associated with that program are Λ_t.

For indices $1 \le l \le r \le T$, the inclusive range transition is

$$F_{l:r} = F_r \circ \cdots \circ F_l, \tag{17}$$

where the rightmost function executes first. The identity transition id_Σ leaves every component of s unchanged. It represents an empty range. All operations read and write the state fields specified by their domain adapters.

Entry-state dependence. Let $\mathcal{E}$ be the set of predicted entities, p an action predicate, and $M_p \subseteq \mathcal{E}$ the resolved set of entities already counted for p within the contract's temporal scope. Evidence that entity $e \in \mathcal{E}$ performs p updates the set by $M_p \leftarrow M_p \cup \{e\}$. The change in its cardinality is

$$|M_p \cup \{e\}| - |M_p| = \mathbf{1}[e \notin M_p], \tag{18}$$

where $\mathbf{1}[\cdot]$ is one when its condition holds and zero otherwise. Consequently, identical evidence has different count effects under different entry sets. The compiled operation retains the entity and membership test. This is the semantics of counting distinct entities; counting separate action occurrences additionally uses event boundaries and the required memory.

Persistence, change, and missing evidence. A confirmed continuing condition preserves its current world value; boundary memory records that the condition remains active. A supported change applies the corresponding update. An unresolved participant or precondition remains represented in u. Whether later evidence can resolve an earlier unknown follows the domain adapter. The adapter record in Table 5 specifies these rules, including how unresolved information affects each query operation.

Transition interface. Set updates, counter updates, permutations, and history appends share the same state-to-state interface. Their domain adapters specify how operands are typed, which state fields are read and written, and how unresolved information is retained. This interface permits different observation procedures to use the same composition mechanism while preserving the semantics of their supported operations.

C.2 Composition and Revision Consistency

Composition property. Suppose every leaf implements its declared F_t and every internal node implements the chronological composition of its children over $\mathcal{S}_\Sigma$. Then, for any admissible entry state, execution at the root equals chronological sequential execution of the leaves.

Derivation. The property holds at a leaf by its definition. For a node covering $[l, r]$, let $m = \lfloor (l + r)/2 \rfloor$ be its midpoint. By induction, the children implement $F_{l:m}$ and $F_{m+1:r}$. Their parent applies $F_{m+1:r}$ to the output of $F_{l:m}$, yielding $F_{l:r}$ in Equation 17. Applying the argument up to the root gives $F_{1:T}(s_0)$, where s_0 is the initial state. Associativity changes the grouping of compositions while preserving execution order. A stored summary satisfies the same property when its denotation equals this composition on admissible entry states.

Revision property. Let v index a retained version, and let $F^{(v)}_{l:r}$ denote its range transition. Replacing leaf k by a transition $\widetilde{F}_k$ under the same schema, bindings, and initial state gives

$$F^{(v+1)}_{1:T} = F^{(v)}_{k+1:T} \circ \widetilde{F}_k \circ F^{(v)}_{1:k-1}. \tag{19}$$

Empty prefixes and suffixes use id_Σ. The unchanged suffix receives the revised entry state, so its state-dependent reads are evaluated under the revised history. Path copying preserves earlier roots and shares unaffected subtrees. With immutable shared programs and evidence, the previous version continues to implement its previous function. These properties apply to the predicted transitions and include their handling of boundary memory and unresolved information.

C.3 Structural and Execution Costs

A balanced tree over T leaves has $O(T)$ index nodes and height $O(\log T)$ for $T \geq 2$. A single-leaf replacement creates a new leaf and its path of ancestors. Let C_{node} bound the cost of creating or rebuilding an affected node, including its composition representation. Structural maintenance costs $O(C_{\mathrm{node}} \log T)$. Retaining K versions produced by $K - 1$ single-leaf replacements uses $O(T + (K - 1) \log T)$ index nodes and K root references.

Let C_{apply} be the cost of applying the required composed transition and C_{query} the cost of answering the requested queries from its result. Obtaining revised answers additionally incurs $C_{\mathrm{apply}} + C_{\mathrm{query}}$. New-leaf compilation is measured separately. A range lookup selects $O(\log T)$ canonical nodes; composition and application costs depend on their stored representations. This distinction follows

the dependence of segment tree complexity on the merge operation. The measured backend includes the actual `then` composition cost. These are structural counts; total memory also includes transition payloads and evidence. For composed programs, execution can visit their constituent operations; for supported summaries, its cost follows the corresponding summary representation. Visual re-observation and recompilation are accounted for separately when a revision requires them.

C.4 Deterministic Query Interface

A maintained history becomes reusable when questions read its state through explicit operations. Separating these operations from visual observation makes answer computation reproducible and permits compatible questions to share their evidence and state updates. The query contract defines which computation is valid for the maintained fields and temporal scope.

For a fixed contract, binding context, and initial state, define the predicted world in version v as

$$\mathcal{W}^{(v)} = (\mathcal{C}_q, \beta, s_0, \mathcal{T}^{(v)}). \tag{20}$$

The query program π_q accesses the required states and historical records through this world. A historical state at interval r is $s_r^{(v)} = F_{1:r}^{(v)}(s_0)$. Execution over $[l, r]$ uses its corresponding entry state:

$$s_r^{(v)} = F_{l:r}^{(v)}\big(s_{l-1}^{(v)}\big), \qquad s_{l-1}^{(v)} = F_{1:l-1}^{(v)}(s_0). \tag{21}$$

This preserves dependencies on events preceding the queried interval.

Let $\mathcal{Y}_q$ denote the answer domain specified by the question, such as numerical values or supplied option labels. The predicted answer is

$$\widehat{y}_q^{(v)} = \pi_q(\mathcal{W}^{(v)}) \in \mathcal{Y}_q \cup \{\texttt{UNKNOWN}\}. \tag{22}$$

Query operations include counting, filtering, set intersection, position lookup, and history lookup. For example, let p_A and p_B be two action predicates and let $M_{p_A}^{(v)}$ and $M_{p_B}^{(v)}$ be the resolved sets of entities satisfying them within the contract's temporal scope. A query asking how many distinct entities satisfy both returns $|M_{p_A}^{(v)} \cap M_{p_B}^{(v)}|$. Required unresolved information propagates through the corresponding query operations, leaving an answer `UNKNOWN` when the computation cannot resolve it.

A further question can reuse $\mathcal{W}^{(v)}$ when its entity bindings, required state fields, observation requirements, and temporal semantics are covered by that world. Its query program then operates on the existing transition history. The queried states are connected to transition records and their evidence references, providing an inspection path from the computed answer to the visual observations used to maintain it.

D Component Analysis and Mechanism Audits

D.1 Observation-Matched Comparisons

The attribution protocol uses the fixed 356-question subset, including unresolved cases. Variants A–D share the same cached observations, question information, and contract. Variants A and B use the same frozen 32B language model. Variants B and C provide their answer stages with identical predicted states and historical records.

D.2 Independent Evidence and Execution Audits

Table 11 reports entity-binding checks on 32 targets, temporal-boundary error on 3 shared matches, duplicate-update checks on 32 videos, and answer agreement on 356 questions.

Observation quality. On 32 audited targets, both the comparator and EviTrans achieve 100.00% entity-binding accuracy and 0.00% unresolved binding rate. On 3 shared matches, the reported matched-boundary errors are 0.236 s for the comparator and 0.021 s for EviTrans.

Table 10: Component analysis on the high-quality subset.

Variant	Configuration	Score	Resolved-answer coverage (%)
A	Full observer; chronological facts; LM answering	72.53	94.10
B	Full observer; explicit state; LM answering	85.06	97.47
C	Full observer; sequential execution; query programs	87.36	96.91
D	Full observer; persistent segment tree; query programs	87.36	96.91
E	Frozen VLM only; persistent segment tree; query programs	0.84	1.40

Table 11: Mechanism-level audit results.

Measurement	Audit size	Comparator	EviTrans
Entity-binding accuracy (%)	32 targets	100.00	100.00
Unresolved binding rate (%)	32 targets	0.00	0.00
Matched-boundary error (s)	3 shared matches	0.236	0.021
Duplicate-update rate (%)	32 videos	0.00	0.00
Answer agreement with predicted state (%)	356 questions	93.26	100.00

State maintenance. The duplicate-update audit examined whether repeated evidence of the same continuing condition incorrectly triggered additional state updates. On the 32 audited videos, the reported duplicate-update rate was 0.00% for both the comparator and EviTrans (Table 11). This diagnostic concerns duplicate state effects; event-occurrence counting and distinct-entity counting have different semantics, as described in Appendix C.

Answer computation. On 356 questions, answer agreement with the supplied predicted state is 93.26% for the comparator and 100.00% for EviTrans. This measures consistency with the predicted state, rather than accuracy against the video's ground-truth answer.

E History Dependence and Generalization

E.1 Changed Entry States and Interval Boundaries

We evaluated transition execution under changed entry states, interval repartitioning, and repeated stable observations. In the changed-entry-state tests, the observation sequence was held fixed while the history supplied to the transition was varied. Each diagnostic trace specifies an entry state, observations, an expected exit state, and a typed query result. Independent chronological sequential execution provided the reference.

Entry-state pairs. Three diagnostics varied membership histories, event-boundary memory, and initial arrangements, respectively, with 64 traces per diagnostic. These tests examined whether the same evidence retained its appropriate state-dependent effects when executed under different incoming histories.

Boundary changes and repeated evidence. The interval-repartitioning diagnostic moved legal interval boundaries while preserving the order, temporal support, and content of the observations. Required boundary memory was carried across the repartitioned intervals. The repetition diagnostic duplicated stable-condition observations while preserving the underlying events and used queries whose answers should remain unchanged by such repetition. Each diagnostic contained 64 traces. Across the three entry-state diagnostics and these two observation-trace diagnostics, the evaluation comprised 320 diagnostic instances.

Agreement metrics. The checks compared the required world fields, boundary memory, unresolved information, and typed answers against the independent sequential reference. Table 12 reports exact state-and-answer agreement for EviTrans and the fixed-realized-update and LM-maintained-history comparators. EviTrans obtained 100.00% agreement in each of the five diag-

Table 12: History-dependence diagnostics. Entries are exact state-and-answer agreement percentages on the specified trace population.

Diagnostic	Traces	Fixed realized updates	LM-maintained history	EviTrans
Changed membership histories	64	0.00	96.88	100.00
Changed event-boundary memory	64	0.00	92.19	100.00
Changed initial arrangements	64	20.31	28.12	100.00
Repartitioned intervals	64	100.00	89.06	100.00
Repeated stable observations	64	0.00	85.94	100.00

Table 13: Execution on complete-observation compositions and longer histories. Entries report exact query accuracy.

Input	Evaluation	Questions	Matched baseline	EviTrans
Complete observations	128 new compositions	128	62.50	100.00
Complete observations	$1\times$ / $2\times$ / $4\times$	32 / 32 / 32	100.00 / 90.62 / 75.00	100.00 / 100.00 / 100.00

nostics. These trace-level execution results are reported separately from the original-video question-answering scores in Table 1.

E.2 Operation Combinations and Longer Histories

The composition evaluation uses 128 new combinations of supported operations supplied as complete observations. Exact query accuracy is 62.50% for the reported matched baseline and 100.00% for EviTrans (Table 13).

The longer-history evaluation uses the reported $1\times$, $2\times$, and $4\times$ length settings, with 32 questions per setting and 96 questions in total. All inputs reported in Table 13 are complete observations; the table reports exact query accuracy at each length.

F Backend Efficiency, Correctness, and Scaling

F.1 Measured Workload and Backend Definitions

The measured workload contains 211 videos and 356 queries from the high-quality subset. Each video has 13–128 predicted leaf records and 32 replacements, giving 6,752 replacements in total. Every replacement changes the stored record. Version-retaining backends keep $K = 33$ versions, including the initial version. Controlled replacements use a fixed interval layout, state schema, entity-binding context, and initial state.

Blocked sequential execution. This backend maintains chronological records, block compositions, and state checkpoints. An edit refreshes the affected suffix state checkpoints during maintenance. Its interval implementation composes boundary records and complete blocks, then applies the resulting transition once. In the expanded 64-edit-per-video configuration, the block size is 32 leaves and block-prefix states are cached (Table 14).

Mutable segment tree. This backend maintains a single current tree and updates its affected path in place. Its measured memory contains the current version. Historical versions are not retained.

Full-index snapshots. This backend retains each version by copying all index nodes. Immutable compiled programs and evidence are shared across snapshots. Each snapshot supports execution on its corresponding history.

Table 14: Backend timing and memory configuration for the enlarged workload. Timings exclude front-end processing and disk I/O.

Item	Setting
Evaluation population	211 videos; 356 questions; 13–128 leaf records per video.
Edits and versions	64 record-changing replacements per video; 13,504 total. 65 versions including the initial version; the mutable tree retains only the current version.
Repetitions and aggregation	Five repetitions per video/backend in one process run. Median across repetitions, then equal-weight mean across videos.
Timer and warm-up	`perf_counter_ns`; one build-and-answer warm-up per case/backend. Primary timing uses 64 edits, 64 ranges, and 16 historical reads.
Execution controls	Backend order shuffled per repetition with seed 41; CPU affinity set to core 0. CPU frequency unlocked; background load not fully controlled.
Garbage collection	Disabled in primary timed sections; default settings for larger-record workloads.
Checkpoint and edit sampling	32-leaf blocks; seeded uniform index draws. Each recorded replacement must change the record.
Caching policy	Blocked execution caches block-prefix states. Blocked and snapshot backends share immutable programs. Final-answer caches are used only in explicitly cache-aligned repeated-query tests.
Memory measurement	`deep_bytes` over reachable Python objects, with each object counted only once based on its identity. Measures retained backend objects, not process RSS.

Persistent segment tree. This backend retains a root for each version and shares unaffected subtrees. Replacement compiles the new leaf and rebuilds its path to the root, including the actual `then` merges. State execution and query answering follow the structural update.

F.2 Timing and Memory Protocol

Edit-to-answer timing covers new-leaf compilation, structural or checkpoint maintenance, state execution, and all original questions for the video. Let $\ell_{b,d,j}$ denote the per-video latency statistic for backend b, video d, and timing repetition j. With $D = 211$ videos and five repetitions, the reported statistic is

$$\overline{\ell}_b = \frac{1}{D} \sum_{d=1}^{D} \mathrm{median}_{j \in \{1,\ldots,5\}} \, \ell_{b,d,j}. \tag{23}$$

Each timing component uses its own repetition median. Components therefore need not sum exactly to the separately aggregated total.

Interval application. All backends receive the same entry state and the same ordered range of leaf records, then compute its exit state. This operator-level measurement excludes entry-state retrieval. Method notation uses an inclusive range $[l, r]$. The expanded configuration uses 64 range applications per case/backend (Table 14). Interval-query measurements including entry-state recovery are reported separately in Table 20.

Historical queries. The base-suite historical-answer measurements use each backend's retained versions and caching policy (Table 17). The expanded configuration uses 16 historical reads per case/backend and permits final-answer caches only in explicitly cache-aligned repeated-query tests (Table 14). Repeated original queries with aligned answer caches and new compatible historical queries are reported separately in Table 20.

Memory accounting. Memory is the total size of all reachable Python objects retained by the system, including indices, summaries, complete evidence, and states or checkpoints, with each object counted only once. Shared questions, code, and visual models are excluded. MiB denotes 2^{20} bytes. The expanded configuration uses `deep_bytes` (Table 14). This measures retained backend objects rather than process RSS.

Table 15: Measured backend efficiency. Latencies are milliseconds; memory is MiB per video. The mutable tree retains the current version; other backends retain 33 versions.

Backend	Edit-to-answer	Interval application	Memory	Historical checks
Blocked sequential	0.2350	0.0772	1.730	Pass
Mutable segment tree	0.1793	0.0634	0.473	N/A
Full-index snapshots	0.2041	0.0632	1.822	Pass
Persistent segment tree	0.1828	0.0632	0.828	Pass

Table 16: Edit latency decomposition in milliseconds. Maintenance includes actual transition composition. Query time covers all original questions for the video.

Backend	New-leaf compilation	Maintenance	State execution	Queries
Blocked sequential	0.0156	0.1829	0.0002	0.0362
Mutable segment tree	0.0155	0.0124	0.1088	0.0426
Full-index snapshots	0.0156	0.0347	0.1110	0.0429
Persistent segment tree	0.0155	0.0153	0.1089	0.0430

F.3 Measured Results and Timing Decomposition

The original-video workloads share 211 videos, 356 questions, and 13–128 predicted leaf records per video. We distinguish two measurement suites. The base timing and memory suite uses 32 record-changing replacements per video, giving 6,752 replacements and 33 retained versions. The expanded replacement and correctness audit uses 64 replacements per video, giving 13,504 replacements and 65 retained versions.

Tables 3, 15, 16, and 17 refer to the base suite. Tables 14, 18, and 19 refer to the expanded suite. Their configurations, counts, and measurements are not pooled.

The persistent segment tree reduces edit-to-answer latency by 22.2% and retained memory by 52.1% relative to blocked sequential execution. Relative to full-index snapshots, the reductions are 10.4% and 54.6%. Its edit-to-answer latency is 2.0% above the mutable tree. These comparisons use ratios of video-averaged measurements.

Blocked sequential execution performs suffix-state work during maintenance, explaining its near-zero subsequent execution time. The persistent tree performs this computation after structural maintenance. Historical-query latency takes 0.1427 ms for persistence and 0.0322 ms for the measured blocked-sequential implementation. Initial processing takes 1.8181 ms and 1.5541 ms, respectively. The measured trade-off therefore combines lower edit latency and retained-version memory with higher initial and historical-read costs.

F.4 Edit Semantics and Execution Correctness

The expanded audit covers 13,504 record-changing replacements involving operations, operands, or preconditions, with no evidence-only replacements. Table 18 checks stored-record changes, transition behavior on probe states, realized world-state changes, and changes to original query answers. Behavioral comparisons are scoped to the tested probe states.

In the expanded correctness audit, blocked sequential execution, full-index snapshots, and the persistent segment tree each have 45,924 checked queries; the mutable tree has 23,140. All four backends obtain 100.00% state agreement and 100.00% answer agreement with independent replay. The three version-retaining backends also pass historical-query checks, including unresolved outputs (Table 19).

Table 17: Initial processing and historical-version answering in milliseconds. Initial processing includes compilation, construction, and answering. Historical timing uses the measured caching policies.

Backend	Initial processing	Historical-query latency
Blocked sequential	1.5541	0.0322
Mutable segment tree	1.6533	N/A
Full-index snapshots	1.5728	0.1428
Persistent segment tree	1.8181	0.1427

Table 18: Replacement audit over 13,504 edits in the expanded 64-edit-per-video suite. Percentages use 13,504 as their denominator.

Audit item	Count	Proportion (%)
Stored record changed	13,504	100.00
Transition behavior changed on probe states	13,302	98.50
Realized world state changed	6,061	44.88
At least one original query answer changed	812	6.01
Evidence-only replacement	0	0.00
Operation, operand, or precondition replacement	13,504	100.00

F.5 Scaling and Cache-Aligned Workloads

Table 20 reports additional editing and query workloads, including operator traces with 256 leaf records and 33 retained versions, and 1,024 leaf records and 65 retained versions. It also reports repeated original queries with aligned answer caches, new compatible historical queries, interval queries including entry-state recovery, and edit/read workloads. These operator-trace settings are listed separately from the original-video timing measurements.

Timing logs. Earlier runs containing identical-record replacement requests and additional interval measurements remain in the following logs:

```
outputs/timings.jsonl
strong_interval_timings.jsonl
```

The reported main table uses changed-record replacements and the stronger blocked interval baseline.

G Shared-World Queries

G.1 Compatible-Query Reuse

The reuse workload groups 1, 5, 10, and 20 query requests by cycling the original questions for each video. The groups contain 1–1, 1–5, 1–6, and 1–6 distinct questions per video, respectively. Comparisons include independent processing, shared blocked sequential execution, and the persistent segment tree (Table 21). Compatibility requires that entity bindings, state fields, observation requirements, and temporal semantics are covered by the existing world.

Table 19: Execution-correctness record. Historical checks include unresolved outputs. Agreement percentages refer to independent replay.

Backend	Historical checks	Checked queries	State agreement (%)	Answer agreement (%)
Blocked sequential	Pass	45,924	100.00	100.00
Mutable segment tree	N/A	23,140	100.00	100.00
Full-index snapshots	Pass	45,924	100.00	100.00
Persistent segment tree	Pass	45,924	100.00	100.00

Table 20: Additional backend workloads(ms/request).

Workload	Blocked sequential	Full-index snapshots	Persistent segment tree
Repeated original queries, aligned answer caches	0.000022	0.000022	0.000022
New compatible historical queries	0.128526	0.125080	0.125612
Interval queries including entry-state recovery	0.261460	0.257111	0.258241
One edit followed by one historical read	0.368762	0.534809	0.519672
Mixed edit/read distribution	0.329508	0.289795	0.273241
Larger operator trace: T=256, K=33	0.307411	0.180292	0.096630
Larger operator trace: T=1024, K=65	0.783584	2.095853	0.110677

Table 21: Shared-world reuse. Time includes world construction and all queries in the group. Columns report wall time in seconds.

Queries	Query population	Independent processing	Shared blocked sequential	Persistent segment tree
1	Original queries cycled; 1-1 distinct/video	0.001367	0.001372	0.001574
5	Original queries cycled; 1-5 distinct/video	0.005982	0.001449	0.001935
10	Original queries cycled; 1-6 distinct/video	0.011671	0.001552	0.002381
20	Original queries cycled; 1-6 distinct/video	0.023127	0.001759	0.003289