SEQUENCE-AWARE RECONSTRUCTION FOR LEARNING ACTION MODELS FROM NOISY ACTION-ONLY TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning symbolic action models from action-only traces is challenging when action identities, argument objects, and object types are uncertain and no states are observed. We introduce Plan-HMM, which uses a fixed-emission, type-specific hidden Markov model to reconstruct object-event traces and aligned pairwise evidence from weighted action candidates. These outputs support symbolic induction without requiring a complete reconstructed plan. Across nine planning domains, Plan-HMM improves transition recovery and invalid-plan rejection over selecting the highest-weight action candidate at each step, with slightly lower valid-plan acceptance on jointly completed settings. It also completes more settings under a fixed time limit. Additional experiments show improved performance under missing observations represented by candidate completions from a fixed training vocabulary.

1 INTRODUCTION

Symbolic action models describe the preconditions and effects needed to reason about actions and construct plans. Acquiring these models manually is costly, motivating approaches that learn them from execution data (Arora et al., 2018). However, records of actions are often easier to obtain than symbolic descriptions of the states before and after them. Event logs and recorded procedures may expose action order and object participation without providing intermediate states or a predefined predicate vocabulary (Gregory et al., 2017; Liu & Haslum, 2025). Learning from such action-only traces requires recovering the structure needed for planning from regularities in observed behavior.

LOCM and LOCM2 exploit this structure by projecting actions onto participating objects and inferring object-state machines from consecutive event pairs in object traces and from argument bindings (Cresswell et al., 2009; 2013; Cresswell & Gregory, 2011). SIFT further establishes that action models can be recovered from applicable traces under explicit structural assumptions (Gösgens et al., 2025). These approaches demonstrate the value of action-only evidence, but their clean-input formulations rely on correct action identities and bindings. Noisy action observations can distort the object traces and binding evidence used to infer predicates and action schemas.

Real-world action records can be noisy or incomplete. The same action may be described using different verbs, such as "pick up" and "lift" when referring to lifting a block from a table. Descriptions may also omit information that a reader can infer from context or commonsense knowledge. For example, "load the package into the truck, then unload it" leaves the truck implicit in the second action. Such variation and omission can make action names, participating objects, and object types difficult to identify consistently. Errors in these interpretations can disrupt consecutive event pairs in object traces and obscure the relations between objects needed for symbolic learning.

Existing extensions address related forms of imperfect evidence. LC_M repairs noisy or incomplete action-only traces using occurrence statistics and state-parameter consistency (Gregory et al., 2017), while POLOCM recovers ordering information from partially ordered actions (Liu & Haslum, 2025). Work on incomplete action observations also characterizes learning when some arguments are unobserved under specified observability assumptions (Jansen et al., 2025; Gösgens et al., 2026). We focus on ordered action traces in which action names, participating objects, and object types may be

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.

uncertain. The challenge is to use regularities across these traces to learn symbolic action models without observed states or access to clean action labels.

We introduce Plan-HMM, which uses sequence regularities to reconstruct object-event traces for symbolic action-model learning. A type-specific hidden Markov model captures dependencies between consecutive object events to resolve uncertainty over action names, participating objects, and object types. Reconstructed object traces and aligned pairwise evidence support LOCM2 without requiring a complete reconstructed action at every step. Missing information at known steps is represented by candidate completions from a fixed training vocabulary, allowing the same reconstruction procedure to handle both ambiguous and incomplete observations.

Our contributions are threefold. First, we study weighted action candidates with uncertain action identities, argument bindings, and object types, assuming known step order and no state observations. Second, we develop a virtual-evidence HMM that reconstructs object traces and partial binding evidence for LOCM2 without requiring complete grounded actions at every step. Third, we evaluate reconstruction quality and learned-model behavior across nine planning domains, varying training-data size, ambiguity, and missingness.

2 Problem Setting

We learn a typed STRIPS action model from action-only traces, without observed states, clean action labels, or a supplied predicate vocabulary. The model consists of lifted action schemas with typed parameters, preconditions, and add/delete effects. The input is a corpus of ordered traces $\widetilde{\tau} = (Q_1, \ldots, Q_N)$, where each step contains weighted grounded-action candidates:

$$Q_t = [(\widetilde{a}_{tk}, w_{tk})]_{k=1}^{K_t}, \qquad w_{tk} \geq 0, \qquad \sum_{k=1}^{K_t} w_{tk} = 1. \tag{1}$$

An argument position is a slot in an action's ordered parameter list; the object assigned to that slot is its argument object. Each candidate specifies an action name, distinct objects in ordered argument positions, and their candidate types. Candidates may disagree on any of these fields; their weights are local evidence, not necessarily calibrated probabilities of correctness. Object names identify entities within a trace. Missing fields, including entire actions at known steps, are represented by alternative completions from a supplied vocabulary while preserving observed fields. Step positions and order are known; recovering deleted steps or entities outside that vocabulary is outside our scope.

An *object event* $a.k$ identifies action a and its one-indexed argument position k. Projecting a deterministic trace onto object o retains its events in order with original step indices. Weighted candidates instead yield competing event, type, and nonparticipation hypotheses at each step.

LOCM and LOCM2 infer object-state machines from consecutive projected events, grouping objects into *sorts* by argument-position participation (Cresswell et al., 2009; 2013; Cresswell & Gregory, 2011). LOCM2 permits multiple machines over event subsets per sort. Recurring partner objects support *state parameters* encoding object relations; together, these define the STRIPS model. Candidate types are input hypotheses, whereas sorts are learner-inferred groupings. Binding evidence records argument positions referring to the same partner object.

3 Related Work

Action-only learning. The LOCM family described above provides our downstream symbolic learner. SIFT learns lifted STRIPS models from applicable action traces without state observations and establishes soundness and completeness under its structural assumptions (Gösgens et al., 2025). It assumes that the grounded actions in a trace are observed correctly, whereas our input associates each position with probability-weighted alternatives over action identity, arguments, and types. Bachor et al. (2025) show that unobserved state features can also be inferred from action sequences under plan non-redundancy assumptions and characterize the complexity of this inference; their result describes an identifiability boundary rather than a reconstruction algorithm for noisy candidates. ARMS omits intermediate states but uses initial and goal states and supplied domain information (Yang et al., 2007), so its observation regime is stronger.

Incomplete actions, states, and order. POLOCM extends the LOCM object-event representation to partial action orders while assuming that observed action names and arguments are correct (Liu & Haslum, 2025); our setting retains total order but makes the grounded action at each position uncertain. Jansen et al. (2025) learn STRIPS+ models from traces in which action arguments and state predicates can be unobserved. Their subsequent framework covers partial information about both actions and states, including a case with no state observations (Gösgens et al., 2026). These methods address systematically missing or implicit information and retain full observability of selected action arguments; they do not resolve probability-weighted competing grounded actions as Plan-HMM does.

LC_M extends LOCM to missing and noisy action-only data without underlying states (Gregory et al., 2017). It uses action-occurrence statistics, state-parameter ratios, and constraint-based object-state-machine consistency to repair traces and induce a domain model. Plan-HMM instead learns type-specific event dynamics, performs sequence-level decoding independently for each object, and supplies reconstructed object traces and aligned pairwise evidence to LOCM2. The original LC_M implementation was not available, our paper-guided adapter accepts deterministic Top-1 traces, implements only argument-name substitution, returns repaired traces rather than a PDDL model, and then uses the same symbolic backend as the other inputs in each comparison. We label this restricted implementation the LC_M adapter. Its evaluation under a shared LOCM2 backend does not establish superiority over the complete published system.

Partial or noisy state observations. FAMA compiles action-model learning into planning and permits partially or fully unobserved actions, but still exploits observed states (Aineto et al., 2019). OffLAM$_{\mathrm{PT}}$ derives minimal lifted models from partially observable state–action traces and extends to partial actions (Lamanna et al., 2025), while PI-SAM and EPI-SAM provide safety and probabilistic-completeness guarantees under partially observable states (Le et al., 2024). AMLSI learns from feasible and infeasible sequences labelled using partial and noisy states (Grand et al., 2022); AMAN combines noisy action sequences with partially observed states (Zhuo & Kambhampati, 2013), and AMDN additionally permits disordered actions (Zhuo et al., 2019). Mourão et al. (2012) learn classifiers over noisy, incomplete state transitions and extract STRIPS rules; NOLAM estimates literals probabilistically from noisy state transitions (Lamanna & Serafini, 2024); and LAML lifts noisy state observations before inducing preconditions and effects (Gugliermo et al., 2025). These approaches are adjacent rather than direct baselines: even when actions are missing or noisy, observed state literals constrain action semantics, whereas Plan-HMM receives no state snapshots or predicate vocabulary.

4 PLAN-HMM

Plan-HMM learns event dynamics from weighted candidates and decodes object histories and compatible pairs for LOCM2, without requiring complete actions at every step (Figure 1).

4.1 OBJECT EVIDENCE

For one object in one input plan, let $q_t^c(e)$ sum the weights of candidates assigning it event $e = a.k$ with type c, and let α_t sum the weights of candidates not containing it. Because each candidate contains the object at most once,

$$\alpha_t + \sum_c \sum_{e \in \mathcal{E}_c} q_t^c(e) = 1, \tag{2}$$

where $\mathcal{E}_c$ is the set of action-position labels associated with candidate type c. Evidence is retained at every original step. An alternative type contributes to that type's event evidence, not to α_t: absence denotes nonparticipation, not type uncertainty.

Running example. In Figure 1, step 2 has candidates $\texttt{stack}(a, b)$ with weight 0.4 and $\texttt{unstack}(a, b)$ with weight 0.6; step 3 has $\texttt{pickup}(c)$ with weight 0.4 and $\texttt{pickup}(a)$ with weight 0.6. All objects have type block. For a, step 2 therefore gives event weights 0.4 for `stack.1` and 0.6 for `unstack.1`, with no absence. Step 3 instead gives weight 0.6 for `pickup.1` and absence weight 0.4.

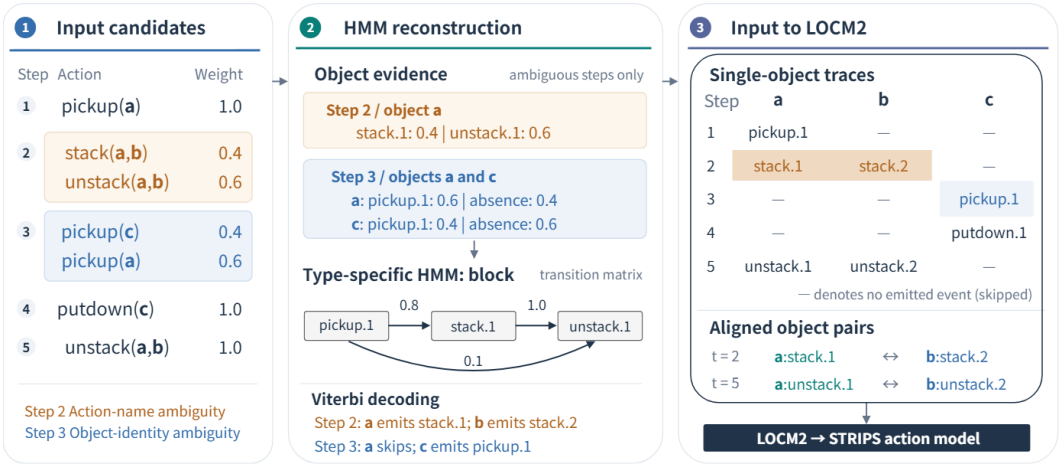

Figure 1: Plan-HMM on a five-step Blocksworld trace. Weighted candidates yield object-event and absence evidence; decoded events form single-object traces and within-step pairs for LOCM2. Selected transitions and parameters are illustrative, not learned from this trace. Appendix A provides the detailed example.

4.2 Type-Specific Event Model

We learn one HMM per candidate type c, shared by objects assigned that type. Its event-state set and output vocabulary are both $\mathcal{E}_c$. The learned parameters are the first-event distribution π^c and next-event probabilities A^c. Each object has one latent type throughout its input plan, with a uniform prior over its candidate types. Identity emissions $B^c_{ey} = \mathbf{1}[e = y]$ mean that a known event label would identify its state. Here no single label is observed: weighted candidates leave the event, participation, and type uncertain. Temporal inference resolves these alternatives while keeping the emission mapping fixed.

To distinguish nonparticipation from an event, the memory state s stores the most recent event, or START before the first event. Let $P^c_s(e) = \pi^c_e$ when $s = \text{START}$ and $P^c_s(e) = A^c_{se}$ otherwise. With an equal skip/emit prior, the edge weights are

$$w^c_t(s, \text{skip}) = \tfrac{1}{2}\alpha_t, \qquad w^c_t(s, e) = \tfrac{1}{2}P^c_s(e)q^c_t(e). \tag{3}$$

Skipping preserves s without adding an event; emitting e updates s to e. Emitting the same label again adds an event and counts as a self-transition, whereas skipping does neither. The fixed weights $q^c_t(e)$ and α_t act as virtual-evidence likelihoods multiplying the normalized latent skip/emit process. The resulting outgoing edge weights need not sum to one and are not renormalized at each step.

4.3 Learning and Decoding

Let u index an object in an input plan, $\mathcal{C}_u$ its candidate types, and $L_{u,c}$ the sum of products of edge weights over all skip/event paths for type c. Baum–Welch estimates $\theta = \{A^c, \pi^c\}_c$ by maximizing

$$\ell(\theta) = \sum_u \log\left(\frac{1}{|\mathcal{C}_u|}\sum_{c\in\mathcal{C}_u} L_{u,c}(\theta)\right). \tag{4}$$

Forward–backward computes expected first-event and event-transition counts (Baum et al., 1970; Rabiner, 1989). Counts are pooled across objects, weighted by posterior type probabilities $\rho_{u,c} = L_{u,c}/\sum_{c'\in\mathcal{C}_u} L_{u,c'}$, and normalized to update π^c and A^c. Skip edges contribute neither kind of count. Parameters start uniformly; zero-count distributions retain their previous values, without pseudocounts. Candidate evidence and emissions remain fixed. For example, if c's step-4 `putdown.1` has types block and gripper with weight 0.5 each, these are local evidence weights, not posterior type probabilities: ρ depends on the entire trace. If another step certainly contains c as block, all gripper paths have zero weight because neither skipping nor a gripper event is supported.

Table 1: Benchmark settings. Budgets count actions; repeats are per domain and test plans per domain/repeat.

Parameter	Setting
Training budget	$20, 40, \ldots, 200, 300, 500$
Ambiguity X	$0.1, 0.2, \ldots, 0.9$
Repeats	10
Test plans	20 valid, 94–100 invalid

Otherwise, both types may remain possible (Appendix B.2). This objective treats objects as independent evidence channels, not as a joint model of candidate-list generation or globally consistent plans. Appendix B formalizes this likelihood and gives the updates and decoding recurrence.

Viterbi decoding selects each object's highest-weight type and complete path using the training edge weights (Viterbi, 1967), retaining original step indices and distinguishing skips from repeated emissions. Zero-weight edges are impossible; no local highest-weight fallback is used. In the example, `unstack` has the highest candidate weight at step 2, but the illustrative transitions give factors $0.4 \times 0.8 = 0.32$ for `stack.1` and $0.6 \times 0.1 = 0.06$ for `unstack.1` after `pickup.1`. These local factors illustrate the preference; Viterbi compares complete paths. Object a skips step 3 while c emits `pickup.1`, leaving a's events at steps 1, 2, and 5.

4.4 Object Traces and Pair Alignment

Every decoded event remains in its single-object trace, even when other objects disagree or leave arguments unresolved. An *aligned event pair* consists of two objects' events at the same input-plan step, with the same action name and distinct argument positions. Within each action and step, we rank events by input support (candidate weight for the selected object, type, and event), retain at most one event per object and argument position, and break ties lexicographically. Any two retained events form a pair. For example, a's `stack.1` and b's `stack.2` form a pair at step 2; their `unstack.1` and `unstack.2` form another at step 5. These are separate within-step pairs involving the same objects, providing evidence that b remains a partner of a across the two actions. LOCM2 uses single-object traces to infer FSMs and aligned pairs to test state-parameter hypotheses. Observed partners can support or contradict a hypothesis; missing partners provide neither, and at least one supporting observation is required. Missing arguments are never filled from clean traces. Complete argument coverage additionally yields an action a whose label $a.0$ enters LOCM2's auxiliary action-order trace, connecting retained actions across omitted steps. Because objects are decoded independently, pair alignment guarantees neither a globally consistent plan nor membership of every retained combination in the input candidate lists.

5 Experiments

We evaluate whether sequence-aware reconstruction improves object-event recovery and whether the resulting action models better distinguish valid plans from invalid plans. We also examine sensitivity to candidate ambiguity and training-data size. All methods supply evidence to the same LOCM2 backend.

5.1 Experimental Setup

Benchmark. We use a frozen benchmark spanning nine domains: Blocksworld, Childsnack, Ferry, Floortile, Miconic, Satellite, Sokoban, Spanner, and Transport. Table 1 summarizes the grid, which yields 9,720 noisy datasets and 1,080 unique Clean controls. Source plans are content-deduplicated before splitting. Each domain/repeat reserves 20 complete valid plans for evaluation; training segments contain at least five actions, with only the final selected plan clipped to reach the exact action budget. Repeats resample the split and noise, although source plans may recur across repeats.

Candidate construction. We add uncertainty to a fraction X of training steps. At each selected step, we keep the correct action and add three incorrect candidates that change only the action name, one argument object, or one argument's type, respectively. For each change, we select a synthetic name with probability 0.5 and an existing alternative otherwise; if no existing alternative is available, we use a synthetic name. We randomly assign the weights $(1/2, 1/4, 1/8, 1/8)$ to the four candidates. For example, a step whose correct action is $\texttt{pickup}(a : \texttt{block})$ may have the following weighted candidates:

$$Q_t = \begin{cases} (\texttt{pickup}(a : \texttt{block}),\ 1/4) & \text{correct action,} \\ (\texttt{lift}(a : \texttt{block}),\ 1/2) & \text{action change,} \\ (\texttt{pickup}(b : \texttt{block}),\ 1/8) & \text{object change,} \\ (\texttt{pickup}(a : \texttt{tool}),\ 1/8) & \text{type change.} \end{cases}$$

Here, $o : c$ denotes object o with candidate type c. In this example, Top-1 selects the incorrect action $\texttt{lift}(a : \texttt{block})$. Unselected steps contain only the correct action with weight one, so the expected Top-1 error rate is $3X/4$. Existing names are collected from the 500-action training prefix, and synthetic-name pools are constructed from this vocabulary; both are reused at all training budgets. Smaller datasets are prefixes of larger ones, and increasing X adds ambiguous steps without changing previously assigned candidates or weights. No action-applicability checks are used.

Compared methods. We compare Clean (uncorrupted traces), Plan-HMM (sequence-aware reconstruction), Top-1 (highest-weight candidate per step), All-candidates, and LC_M adapter (restricted Top-1 argument repair). All-candidates supplies candidate-derived traces to the same learner, retaining at most 500 traces per task. When truncation is required, selection balances source plans and prioritizes transition coverage. The LC_M adapter uses occurrence and state-parameter ratio thresholds of 0.5 and at most five repair passes.

Training and computation. Plan-HMM initializes A^c and π^c uniformly and learns both with fixed identity emissions and an equal skip/emit prior. Training stops after 100 iterations or when the absolute log-likelihood change per object-step is at most 10^{-6}. Decoding uses the training edge weights. Each task has a 60-second limit covering reconstruction, learning, and evaluation, including a separate 50-second LOCM2 limit.

5.2 Evaluation Protocol

Reconstruction quality. Transition F1 measures recovery of consecutive object-event pairs. Let T_{rec} and T_{clean} contain the distinct pairs from reconstructed and corresponding uncorrupted training traces, with object type included in each pair's identity. Precision is the fraction of T_{rec} present in T_{clean}; recall is the fraction of T_{clean} recovered in T_{rec}. F1 is their harmonic mean, with zero for undefined ratios. Counts are pooled across types, objects, and plans without weighting repeated occurrences; the auxiliary zero-object channel is excluded. The reference comes directly from clean training traces before LOCM2 learning or filtering, not from the ground-truth domain model or the model learned by Clean.

Learned-model behavior. All methods share frozen evaluation suites of 20 held-out valid plans and 94–100 invalid plans per domain/repeat. To generate each invalid plan, we append one grounded action to a valid plan and verify that the ground-truth model rejects the appended action after accepting the prefix. We retain at most five distinct invalid plans per valid plan. Acceptance means passing replay with unknown initial state, not achieving a goal. Appendix C specifies action selection and replay semantics.

Let v_i indicate acceptance of valid plan i, and r_{ij} rejection of its jth associated invalid plan. With $m = 20$ and $K_i \geq 1$ invalid plans per valid plan, valid acceptance rate (VAR) and invalid rejection rate (IRR) are

$$\text{VAR} = \frac{1}{m}\sum_{i=1}^{m} v_i, \qquad \text{IRR} = \frac{1}{m}\sum_{i=1}^{m} v_i \left(\frac{1}{K_i}\sum_{j=1}^{K_i} r_{ij} \right). \tag{5}$$

IRR credits invalid rejection only when the corresponding valid prefix is accepted. Each prefix receives equal weight, including rejected prefixes with zero credit, so rejecting every plan yields zero

Table 2: Mean scores (percentages) under the 60-second limit. Each cell reports the full-grid mean with timeouts and failures scored zero, followed by the completed-only mean in parentheses.

Method	Completed / total	Transition F1	VAR	IRR	Behavioral F1
Clean	1,080 / 1,080	100.00 (100.00)	83.75 (83.75)	78.69 (78.69)	80.49 (80.49)
Plan-HMM	9,714 / 9,720	93.25 (93.31)	74.91 (74.96)	70.75 (70.79)	72.32 (72.37)
Top-1	8,794 / 9,720	46.50 (51.40)	71.90 (79.47)	36.19 (40.00)	40.58 (44.85)
All-candidates	2,946 / 9,720	10.99 (36.25)	26.00 (85.80)	11.93 (39.36)	13.72 (45.26)
LC_M adapter	8,307 / 9,720	46.69 (54.63)	67.29 (78.74)	42.23 (49.41)	45.59 (53.34)

IRR. We additionally compute behavioral $\mathrm{F1} = 2\,\mathrm{VAR}\,\mathrm{IRR}/(\mathrm{VAR} + \mathrm{IRR})$ per task before aggregation, with zero when both rates vanish. This is a harmonic behavioral summary, not classification or predicate-recovery F1. VAR and IRR remain primary to expose the acceptance–rejection tradeoff.

5.3 Results and Analysis

Reconstruction and invalid rejection. Table 2 reports full-grid means and completed-only means in parentheses. Among completed tasks, Plan-HMM reaches 93.31% transition F1 and 70.79% IRR, compared with 51.40% and 40.00% for Top-1. Behavioral F1 is 72.37% and 44.85%, respectively. These means cover different successful subsets: Plan-HMM completes 9,714 tasks and Top-1 completes 8,794. On their 8,793 common completed settings, the domain/repeat-balanced improvement is 40.63 percentage points in transition F1 (95% repeat-cluster interval [39.93, 41.32]), 30.52 points in IRR [28.65, 32.43], and 27.51 points in behavioral F1 [25.52, 29.56].

Figure 2 complements the aggregate results with variation across the 90 domain/repeat groups. Domain/repeat groups are matched across methods, but source-plan reuse means that their values should not be treated as independent samples.

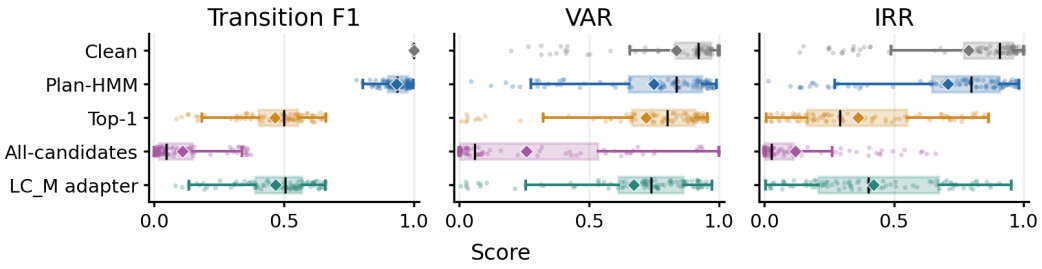

Figure 2: Score distributions over 90 domain/repeat groups. Group means average budgets and ambiguity levels; Clean uses unique budget controls. Boxes show quartiles and medians, whiskers reach the outermost observations within 1.5 IQR of the box, diamonds mark means, and dots show all groups. Failures and timeouts score zero.

Valid acceptance and completion coverage. On the full grid, Plan-HMM achieves higher VAR than Top-1 (74.91% versus 71.90%), with failures and timeouts scored zero. Among completed tasks, however, VAR is 74.96% for Plan-HMM and 79.47% for Top-1. On their common completed settings, the domain/repeat-balanced difference is -1.85 percentage points (95% interval $[-3.34, -0.37]$). Thus, Plan-HMM's higher full-grid VAR reflects greater completion coverage, while valid-plan acceptance remains lower on matched successful settings.

Clean completes all 1,080 controls and reaches 83.75% VAR, 78.69% IRR, and 80.49% behavioral F1. The shared learner therefore does not achieve perfect behavioral recovery even with uncorrupted evidence. Plan-HMM's completed-task means remain below these controls on all three behavioral measures.

Comparison with baselines. The restricted LC_M adapter completes 8,307 of 9,720 tasks, with 417 failures and 996 timeouts. On the 8,306 settings completed by both LC_M adapter and Plan-

HMM, domain/repeat-balanced differences favor Plan-HMM by 37.72 percentage points in transition F1, 21.47 points in IRR, and 19.32 points in behavioral F1, alongside 2.01 points lower VAR.

All-candidates completes 2,946 of 9,720 settings. Its full-grid transition F1, IRR, and behavioral F1 are 10.99%, 11.93%, and 13.72%, respectively, compared with completed-only means of 36.25%, 39.36%, and 45.26%. Among completed tasks, VAR reaches 85.80%, whereas IRR is 39.36%, indicating that high valid-plan acceptance does not ensure effective invalid-plan discrimination. The full-grid scores reflect both model quality and completion coverage, with 2,946 completed settings and 6,774 timeouts.

Sensitivity to ambiguity and training budget. Figure 3 shows that transition F1 and IRR generally decrease as ambiguity increases, while Plan-HMM remains above the noisy-input baselines on both metrics. Figure 4 shows that larger training budgets generally improve Plan-HMM's VAR and IRR, even when transition F1 is already high at smaller budgets. All settings contribute, with failures and timeouts scored zero.

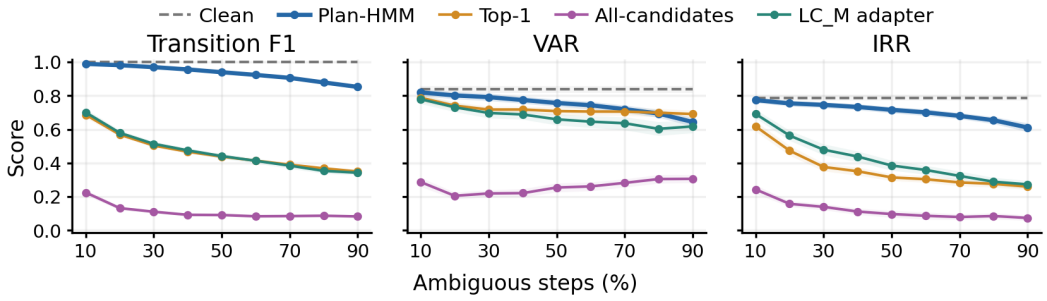

Figure 3: Performance versus ambiguous-step fraction X, averaging budgets within domain/repeat groups, then repeats and equally weighted domains. Bands show pointwise 95% repeat-cluster bootstrap intervals (2,000 draws). Clean reuses unique controls across X; failures and timeouts score zero.

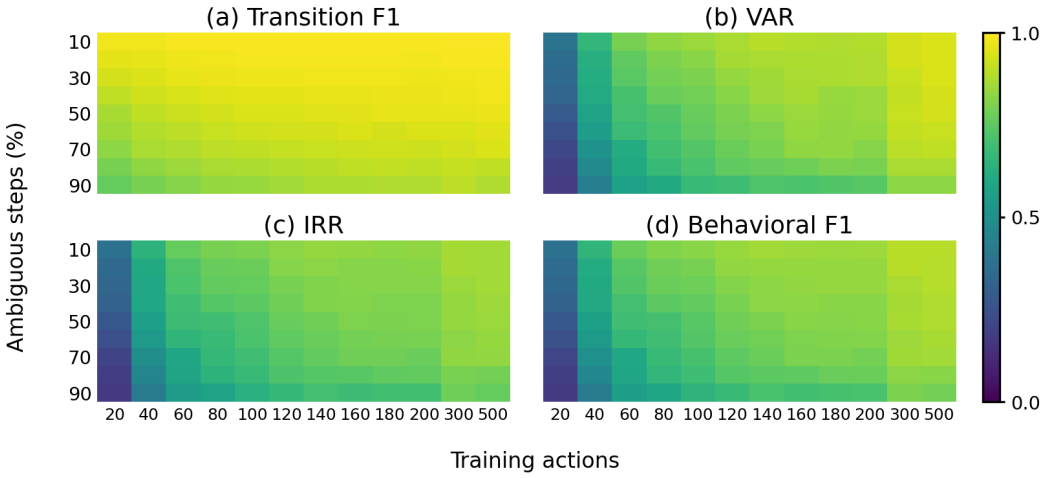

Figure 4: Plan-HMM performance across training budgets and ambiguity levels. Cells average ten repeats per domain, then weight domains equally. All panels use a 0–1 color scale; budget positions are categorical. Failures and timeouts score zero.

Robustness to missing observations. We evaluate missing observations across nine domains and ten repeats, using 200 training actions and the same held-out suites. We mask 10–90% of steps, removing either the complete action, one typed argument, or one argument type while retaining step

positions and order. Completions use a fixed vocabulary from the pre-corruption 500-action training prefix, with uniform weights over schemas, objects, or types and their corresponding completions, without consulting erased values.

We consider pure missing observations and missingness applied after 30% ambiguity, allowing overlap. This benchmark uses existing-vocabulary alternatives, without the synthetic names used in the main experiment. We compare Plan-HMM, Top-1, and All-candidates against matched Clean controls under the same 60-second limit. All-candidates selects at most 500 traces, balancing source plans and prioritizing transition coverage.

Figure 5 reports behavioral F1 with failures and timeouts scored as zero. We average masking conditions within domain/repeat groups, then repeats and equally weighted domains; Blocksworld excludes type-only masking. Plan-HMM completes 4,677/4,680 tasks, versus 3,981 for Top-1 and 1,328 for All-candidates. At 90% missingness, its F1 is 0.450 under pure missing and 0.383 with ambiguity, compared with 0.337 and 0.213 for Top-1. These results show improved fixed-budget performance under missing observations, with degradation at high missingness.

Our experiments evaluate the complete pipeline; the individual contributions of temporal learning, skipping, and pair alignment remain unisolated.

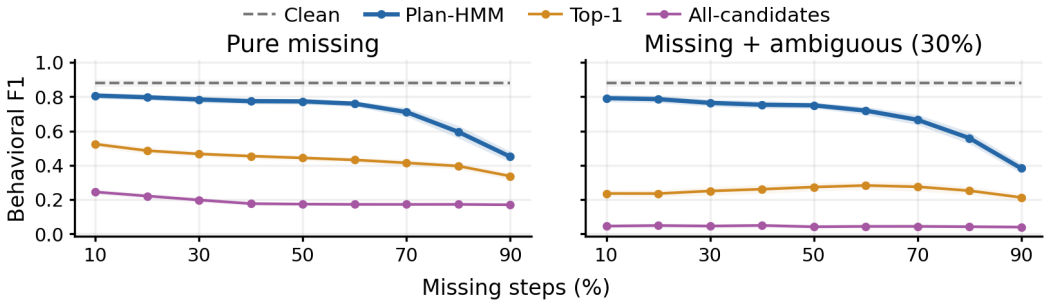

Figure 5: Behavioral F1 under pure missingness and missingness with 30% ambiguity, using 200 training actions. Masking conditions are averaged within domain/repeat groups, then repeats and equally weighted domains. Failures and timeouts score zero. Dashed lines are matched Clean controls; bands are pointwise 95% repeat-cluster bootstrap intervals.

6 Conclusion

Plan-HMM reconstructs object traces and aligned pairwise evidence from weighted action candidates for symbolic action-model learning. Across nine planning domains, it improves transition recovery and paired invalid-plan rejection over Top-1 on jointly completed settings, while slightly reducing valid-plan acceptance. Its higher full-grid VAR reflects greater completion coverage under the fixed time limit. The same reconstruction procedure handles missing observations through candidate completions, improving fixed-budget behavioral F1 over Top-1.

These findings concern controlled ambiguity and missingness with known step positions, a supplied training vocabulary, and the correct action retained in the candidate set. They do not establish robustness to missing correct candidates, inserted or deleted steps, or errors from real extraction systems. Independent object decoding also leaves cross-object consistency unresolved. Future work should examine these settings and improve consistency without discarding useful partial evidence.

AI use statement

We used LLMs to polish writing, assist with literature discovery, and draft portions of the manuscript. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final paper.

REFERENCES

Diego Aineto, Sergio Jiménez Celorrio, and Eva Onaindia. Learning action models with minimal observability. *Artificial Intelligence*, 275:104–137, 2019. doi: 10.1016/j.artint.2019.05.003.

Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty. A review of learning planning action models. *The Knowledge Engineering Review*, 33:e20, 2018. doi: 10.1017/S0269888918000188.

Pascal Bachor, P. Maurice Dekker, and Gregor Behnke. Is this plan necessarily redundant? on the computational complexity of unobserved domain learning. In *Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 11–20, 2025. doi: 10.1609/icaps.v35i1.36096.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970. doi: 10.1214/aoms/1177697196.

Stephen Cresswell and Peter Gregory. Generalised domain model acquisition from action traces. In *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 42–49, 2011. doi: 10.1609/icaps.v21i1.13476.

Stephen Cresswell, Thomas Leo McCluskey, and Margaret Mary West. Acquisition of object-centred domain models from planning examples. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 338–341, 2009. doi: 10.1609/icaps.v19i1.13391.

Stephen Cresswell, Thomas Leo McCluskey, and Margaret Mary West. Acquiring planning domain models using LOCM. *The Knowledge Engineering Review*, 28(2):195–213, 2013. doi: 10.1017/S0269888912000422.

Jonas Gösgens, Niklas Jansen, and Hector Geffner. Learning lifted STRIPS models from action traces alone: A simple, general, and scalable solution. In *Proceedings of the Thirty-Fifth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 189–197, 2025. doi: 10.1609/icaps.v35i1.36117.

Jonas Gösgens, Niklas Jansen, and Hector Geffner. Learning lifted action models from traces with minimal information about actions and states. In *Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 929–939, 2026. doi: 10.24963/kr.2026/87.

Maxence Grand, Damien Pellier, and Humbert Fiorino. An accurate PDDL domain learning algorithm from partial and noisy observations. In *Proceedings of the 34th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 734–738, 2022. doi: 10.1109/ICTAI56018.2022.00113.

Peter Gregory, Alan Lindsay, and Julie Porteous. Domain model acquisition with missing information and noisy data. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2017.

Simona Gugliermo, Uwe Köckemann, Erik Schaffernicht, and Alessandro Saffiotti. Learning lifted action models for planning domain acquisition in noisy environments. *IEEE Access*, 13:214452–214466, 2025. doi: 10.1109/ACCESS.2025.3645682.

Niklas Jansen, Jonas Gösgens, and Hector Geffner. Learning lifted action models from traces of incomplete actions and states. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 832–842, 2025. doi: 10.24963/kr.2025/80.

Leonardo Lamanna and Luciano Serafini. Action model learning from noisy traces: a probabilistic approach. In *Proceedings of the Thirty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 342–350, 2024. doi: 10.1609/icaps.v34i1.31493.

Leonardo Lamanna, Luciano Serafini, Alessandro Saetti, Alfonso Emilio Gerevini, and Paolo Traverso. Lifted action models learning from partial traces. *Artificial Intelligence*, 339:104256, 2025. doi: 10.1016/j.artint.2024.104256.

Hai S. Le, Brendan Juba, and Roni Stern. Learning safe action models with partial observability. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, pp. 20159–20167, 2024. doi: 10.1609/aaai.v38i18.29995.

Chunjie Liu and Patrik Haslum. Learning action models from partially ordered action traces. In *ICAPS Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2025.

Kira Mourão, Luke Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. Learning STRIPS operators from noisy and incomplete observations. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 614–623, 2012.

Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967. doi: 10.1109/TIT.1967.1054010.

Qiang Yang, Kangheng Wu, and Yunfei Jiang. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, 171(2–3):107–143, 2007. doi: 10.1016/j.artint.2006.11.005.

Hankz Hankui Zhuo and Subbarao Kambhampati. Action-model acquisition from noisy plan traces. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2444–2450, 2013.

Hankz Hankui Zhuo, Jing Peng, and Subbarao Kambhampati. Learning action models from disordered and noisy plan traces. *CoRR*, abs/1908.09800, 2019.

A Worked Example

For example, consider the trace

$$\tau = \langle \texttt{pickup}(a), \texttt{stack}(a,b), \texttt{pickup}(c), \texttt{putdown}(c), \texttt{unstack}(a,b) \rangle.$$

The corresponding object traces are

$$\tau^a = \langle \texttt{pickup.1}, \texttt{stack.1}, \texttt{unstack.1} \rangle, \qquad \tau^b = \langle \texttt{stack.2}, \texttt{unstack.2} \rangle.$$

For object a, steps 2 and 5 yield consecutive events because the intervening actions involve only c. Step indices are omitted here but retained for alignment.

A.1 Candidate evidence

Figure 1 extends the five-step Blocksworld trace above. We use two candidates here for clarity; the experimental benchmark uses four candidates per ambiguous step. Steps 2 and 3 are ambiguous:

$$Q_2 = [(\texttt{stack}(a,b), 0.4), (\texttt{unstack}(a,b), 0.6)],$$
$$Q_3 = [(\texttt{pickup}(c), 0.4), (\texttt{pickup}(a), 0.6)].$$

Steps 1, 4, and 5 retain their original actions with weight one. All objects have candidate type block. At step 2, the candidates disagree on the action name. For object a, projection gives

$$q_2^{\mathrm{block}}(\texttt{stack.1}) = 0.4, \qquad q_2^{\mathrm{block}}(\texttt{unstack.1}) = 0.6,$$

with absence weight $\alpha_2 = 0$. For object b, the corresponding events are `stack.2` and `unstack.2`.

At step 3, the candidates disagree on the participating object. For object a, the weight of `pickup.1` is 0.6 and the absence weight is 0.4. For object c, these weights are 0.4 and 0.6, respectively. Thus, an uncertain object assignment creates competing event and nonparticipation hypotheses for both objects. At step 4, only c participates, so object a has absence weight $\alpha_4 = 1$.

A.2 ILLUSTRATIVE DECODING

At step 2, the highest-weight candidate is `unstack` with weight 0.6. The illustrative transition probabilities in Figure 1 instead favor `stack.1` after `pickup.1`: the corresponding factors are $0.4 \times 0.8 = 0.32$ and $0.6 \times 0.1 = 0.06$. These are local factors; Viterbi compares complete paths, including subsequent events.

At step 3, decoding selects a skip for object a and `pickup.1` for object c. For object a, emitting `pickup.1` after `stack.1` has zero probability under the illustrative transition matrix, so decoding skips step 3. Skipping preserves `stack.1` as the context for `unstack.1` at step 5. For c, emitting `pickup.1` establishes the context for `putdown.1` at step 4. With the illustrative parameters $\pi_{\texttt{pickup.1}} = 0.44$, $\pi_{\texttt{putdown.1}} = 0.01$, and $A_{\texttt{pickup.1,putdown.1}} = 0.1$, the two paths for c have relative weights

$$\begin{aligned} \text{emit at step 3:} \quad & 0.4 \times 0.44 \times 0.1 = 0.0176, \\ \text{skip at step 3:} \quad & 0.6 \times 0.01 = 0.006. \end{aligned}$$

Common factors are omitted. These parameters illustrate decoding and are not fitted to the displayed trace.

A.3 TRACE AND PAIR OUTPUTS

Each object's trace contains all its decoded events in their original order, even when other objects disagree on the action or leave arguments unresolved. For example, if decoding recovers the actions in the example above, the indexed traces include

$$\begin{aligned} \widehat{\tau}^a &= \langle (1, \texttt{pickup.1}), (2, \texttt{stack.1}), (5, \texttt{unstack.1}) \rangle, \\ \widehat{\tau}^b &= \langle (2, \texttt{stack.2}), (5, \texttt{unstack.2}) \rangle. \end{aligned}$$

LOCM2 uses consecutive events in these traces to construct object FSMs.

Object-pair traces. An aligned event pair joins two objects' events within the same input plan and step, with the same action name and distinct argument positions. At step 2, $(a, \texttt{stack.1})$ pairs with $(b, \texttt{stack.2})$; at step 5, $(a, \texttt{unstack.1})$ pairs with $(b, \texttt{unstack.2})$. Events are paired within each step, not between steps 2 and 5. The recurrence of partner b across these pairs supports a state parameter linking the objects between `stack` and `unstack`. If b instead emits `unstack.2` at step 2, that pair is unavailable, but both events remain in their single-object traces.

B ESTIMATION AND DECODING DETAILS

For object-plan channel u and type $c \in \mathcal{C}_u$, let $L_{u,c}$ be the virtual-evidence likelihood defined in the main text. Forward–backward also computes posterior expected event-transition counts $C_{ij}^{u,c}$ and first-event counts $I_j^{u,c}$ conditional on this type.

B.1 VIRTUAL-EVIDENCE LIKELIHOOD

For fixed type c, define a normalized latent process that chooses skip with probability $1/2$ and emits e with probability $P_s^c(e)/2$. Introduce an auxiliary binary evidence variable V_t at each step, with

$$P(V_t = 1 \mid \text{skip}) = \alpha_t, \qquad P(V_t = 1 \mid \text{emit } e, c) = q_t^c(e).$$

These are valid Bernoulli probabilities because the candidate weights imply $0 \le \alpha_t, q_t^c(e) \le 1$; the probabilities for $V_t = 0$ are their complements. Conditioning on $V_t = 1$ at every step yields exactly the edge weights in Equation 3. Summing over latent paths gives $L_{u,c} = P(V_{1:N} = 1 \mid c, \theta)$ for this fixed evidence model. Thus Equation 4 is a likelihood for the auxiliary evidence under a uniform type prior, not a generative likelihood for the original candidate lists. Local renormalization of the weighted edges would change this objective.

The complete-data log likelihood depends on π^c and A^c only through first-event and event-transition counts: skip probabilities, type priors, and evidence factors are fixed. Forward–backward and posterior type weighting compute their conditional expected counts, and the normalized updates below

maximize this EM auxiliary function over each probability simplex. Hence exact EM does not decrease the evidence log likelihood when it is finite; a zero-count distribution can retain its previous value. This statement concerns the objective, not convergence to a global optimum or an empirical benefit from any individual component.

B.2 Type evidence and certainty

Consider a local variation of Figure 1: replace step 4 by two `putdown`(c) candidates, assigning c type block or gripper with weight 0.5 each. For this object, $\alpha_4 = 0$ and $q_4^{\text{block}}(\texttt{putdown.1}) = q_4^{\text{gripper}}(\texttt{putdown.1}) = 0.5$. The posterior compares the complete-trace likelihoods under the two type models, not just these equal local weights. In the otherwise unchanged example, step 3 assigns c type block when it participates but also permits its absence. Thus a gripper path can skip step 3 and first emit at step 4, provided the gripper model gives that first event positive probability. A type-specific transition preference need not alone determine the type.

More generally, if a step t has $\alpha_t = 0$ and $\sum_e q_t^{c^\star}(e) = 1$, then every alternative type has zero event weight at that step and cannot skip it. Hence $L_{u,c} = 0$ for all $c \neq c^\star$; if $L_{u,c^\star} > 0$, its posterior is one. Certainty conditional on participation is weaker: if $\alpha_t > 0$, alternative types may survive by skipping. Conflicting mandatory types at different steps, or incompatible zero transition probabilities, can make every fixed-type path impossible; the posterior is then undefined, and the implementation reports no feasible fixed type rather than silently changing the object's type or using local selection.

B.3 Expected-count updates

The posterior type probability is

$$\rho_{u,c} = \frac{L_{u,c}}{\sum_{c' \in \mathcal{C}_u} L_{u,c'}}. \tag{6}$$

We pool expected counts using these probabilities:

$$\bar{C}_{ij}^{c} = \sum_{u : c \in \mathcal{C}_u} \rho_{u,c} C_{ij}^{u,c}, \qquad \bar{I}_j^{c} = \sum_{u : c \in \mathcal{C}_u} \rho_{u,c} I_j^{u,c}. \tag{7}$$

The parameter updates are

$$A_{ij}^{c} = \frac{\bar{C}_{ij}^{c}}{\sum_k \bar{C}_{ik}^{c}}, \qquad \pi_j^{c} = \frac{\bar{I}_j^{c}}{\sum_k \bar{I}_k^{c}}. \tag{8}$$

Skip edges contribute neither event-transition nor first-event counts.

Parameters are initialized uniformly. Distributions with zero total expected count retain their previous values; no pseudocounts are added. The candidate evidence and emission mapping remain fixed. These normalized-count updates maximize the EM auxiliary function, so exact EM does not decrease ℓ when the evidence likelihood is positive. Updating A^c and π^c changes posterior type probabilities and inferred participation in subsequent iterations.

The objective treats object traces as independent evidence channels; it neither models the generation of candidate lists nor enforces a globally consistent grounded plan.

B.4 Viterbi Decoding

Viterbi decoding selects the highest-weight type and event path for each object using the same edge weights as training (Viterbi, 1967). For fixed type c, let $D_t^c(s)$ be the highest log weight of a path ending in memory state s after original step t. Initialize $D_0^c(\mathrm{START}) = 0$ and all other scores to $-\infty$. START persists only through skipping:

$$D_t^c(\mathrm{START}) = D_{t-1}^c(\mathrm{START}) + \log(\alpha_t/2). \tag{9}$$

For $e \in \mathcal{E}_c$,

$$\begin{aligned} D_t^c(e) = \max\big\{ & D_{t-1}^c(e) + \log(\alpha_t/2), \\ & \max_s [D_{t-1}^c(s) + \log w_t^c(s,e)] \big\}, \end{aligned} \tag{10}$$

where s ranges over START and the event states. Backpointers distinguish skipping from emitting the same event again. We compare the best paths across candidate types under the uniform prior and retain the selected events with their original step indices. Zero-weight edges are impossible; there is no fallback to local highest-weight selection.

C Evaluation Details

Each domain/repeat uses 20 complete held-out valid plans. To construct an invalid plan, we append one grounded action to a valid plan. Candidate appended actions come from a problem-specific catalog of actions that are inapplicable after that plan in the original PDDL problem. We replay the valid plan under the ground-truth model using the evaluator's unknown-initial-state semantics, then retain only appended actions rejected at the next step. After deduplication, a fixed hash-based ordering selects at most five such actions per valid plan. Each selected action produces a separate invalid plan whose prefix is valid and whose final action is inapplicable. This yields 94–100 invalid plans per suite; fewer cases are retained when fewer verified actions are available. All methods use the same frozen suites.

Acceptance means that an action sequence passes replay under unknown-initial-state semantics; goal achievement is not evaluated. A positive precondition fails if its atom has been added or deleted by an earlier action and is currently false. A negative precondition fails if its atom is currently true.