REMEMBERING WHAT CHANGES THE FUTURE: LEARNING TO COMPRESS AGENT HISTORIES FOR LONG-HORIZON TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon agents interleave reasoning, actions, and observations, causing their interaction histories to grow until context budgets force compression. Summarization and truncation can reduce this cost, but a coherent account of the past does not reveal which facts, goals, or constraints remain necessary for future control. The central question is therefore whether a summary preserves information that can affect subsequent decisions, rather than whether it provides a fluent account of past events. We propose Policy-Aware Compression through Counterfactual Evaluation (PACE), a reinforcement-learning framework for learning future-relevant memory. At each compaction boundary, PACE compares continuations from the same environment state: one sees the full history and the other sees a candidate compact state. The one-sided difference in downstream utility gives a direct learning signal for harmful omissions, while task-state and safety signals preserve information that may matter beyond the immediate answer. The compactor is trained using candidate groups under hard memory budgets. A hybrid-execution analysis relates local future sufficiency to repeated compaction. In 8K single-boundary evaluations on HotpotQA and MuSiQue, PACE preserves substantially more downstream QA performance than prompt-based and learned compression baselines at the same compact-memory budget. The results support a simple principle: effective agent memory should prioritize what can change the future, rather than information that merely supports a coherent account of the past.

1 INTRODUCTION

Systems such as ChatGPT (OpenAI, 2026), Gemini (Google DeepMind, 2026), DeepSeek (DeepSeek-AI, 2026), and Claude (Anthropic, 2026) are advancing language agents that combine reasoning with tool use, evidence gathering, and plan revision (Yao et al., 2022). Long-horizon tasks require retaining facts, unresolved subgoals, constraints, and provenance that may become useful many steps later. Reinforcement learning (RL) optimizes this behavior from task outcomes, making retained information consequential for subsequent decisions (Wu et al., 2025; Lu et al., 2025; Li et al., 2026).

Accumulated histories increase inference costs and eventually exceed context budgets. Context compaction replaces an older prefix with bounded memory used alongside recent interactions. Existing agents consolidate memories, fold subtrajectories, summarize periodically, or manage context budgets explicitly (Zhou et al., 2026; Sun et al., 2026; Wu et al., 2025; 2026). Because compression changes a sequential policy's input, a terse record of an unresolved dependency can be more useful than a fluent synopsis that omits it. The objective is to remember what can change the future.

This objective creates four challenges. **C1: Future relevance.** Textual similarity and brevity do not measure whether memory retains the evidence and task state needed for later decisions. **C2: Delayed credit.** An omitted constraint may cause failure much later, making episode-level rewards difficult to assign to individual compactions. **C3: Asymmetric preservation.** Alternative successful continuations are acceptable, but improvements should not cancel the measured harm of omissions. **C4: Repeated use.** New observations and further compactions change memory requirements, so training and analysis must account for successive interventions.

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

We propose Policy-Aware Compression through Counterfactual Evaluation (PACE). At each boundary, paired full-history and compact-state continuations from the same environment state ground **C1** in downstream utility and assign local credit for **C2**. For **C3**, we measure performance losses and beneficial changes separately. Candidate groups provide within-boundary comparisons for budget-constrained RL. For **C4**, we train on fresh and recursively compacted histories and relate local losses to repeated-compaction return under stated assumptions.

Our contributions are as follows:

- We formalize context compaction as budgeted policy-state abstraction: bounded memory preserves information needed for future decisions, with future sufficiency measured through paired downstream task performance.
- We propose PACE, a reinforcement-learning framework that evaluates candidate memories through paired full- and compact-history continuations and optimizes them with within-boundary group comparisons, task-state and safety signals, and a hard memory budget.
- Experiments on two multi-hop QA benchmarks compare PACE with compression baselines and examine objective ablations, memory budgets, and delayed credit. PACE delivers strong downstream QA performance at 8K; 64K and 256K evaluations examine information preservation through repeated compression.

2 PRELIMINARIES

2.1 AGENT INTERACTION HISTORIES

Consider an agent interacting with a partially observable environment. Let o_t and a_t denote its observation and action at time t. The interaction history is

$$h_t = (o_0, a_0, o_1, a_1, \ldots, o_t). \tag{1}$$

An execution policy π_ϕ maps the active context to a distribution over actions and seeks to maximize the discounted task return. Because the latent state is not directly observed, the full history acts as the information state of the policy.

2.2 REINFORCEMENT LEARNING

Reinforcement learning optimizes a policy from interaction outcomes rather than a fixed target action at every state (Sutton et al., 1998). Let r_t denote the reward at time t. For a trajectory τ, policy-gradient methods maximize discounted return using an advantage A_t relative to a baseline:

$$J(\phi) = \mathbb{E}_\tau \left[\sum_t \gamma^t r_t \right], \qquad \mathcal{L}_{\mathrm{PG}} = -\mathbb{E}_\tau [A_t \log \pi_\phi(a_t \mid x_t)]. \tag{2}$$

PACE freezes the execution policy π_ϕ and instead treats the compactor c_θ as a policy whose action is an autoregressively generated state z_t. Downstream continuations, rather than textual targets, determine its advantage; Section 3.5 gives the group-normalized sequence objective used in training.

2.3 BUDGETED CONTEXT COMPACTION

A compactor c_θ receives a history and a token budget B_t and samples a textual or structured compact state

$$z_t \sim c_\theta(\cdot \mid h_t, B_t), \qquad \ell(z_t) \le B_t, \tag{3}$$

where ℓ counts serialized tokens. Let p denote the fixed instruction or immutable prompt. The deployed context may retain a recent uncompressed window:

$$x_t^z = p \oplus z_t \oplus \mathrm{tail}_L(h_t), \qquad x_t^f = p \oplus h_t, \tag{4}$$

where x_t^f is available only as a training reference once the full history would exceed the deployment budget. We focus on *lossy logical context abstraction*: omitted information is unavailable to the policy unless a later environment observation reveals it again. The learning objective is therefore not to reconstruct h_t or match every next action, but to minimize the loss in future task utility caused by replacing x_t^f with x_t^z, subject to context and safety constraints.

3 PACE

3.1 BUDGETED CONTEXT ABSTRACTION

Each compaction boundary b_t contains a replayable snapshot of the environment state s_t, history h_t, current observation, instruction p, and environment or user-simulator random inputs. The compactor emits z_t satisfying Equation 3. The backend supports snapshotting or deterministic replay, agent execution, utility evaluation, and optional task-state and safety signatures. Keeping π_ϕ fixed while training c_θ makes preservation a stationary target and supports open- or closed-weight agents; joint adaptation can follow.

Constrained decoding and post-generation validation enforce the hard token budget. We permit budgets $B_t \sim p_B$ and candidate ceilings within each budget. Required memory depends on unresolved subgoals, constraints, and conflicting observations rather than raw history length, so B_t is a ceiling with learned stopping, not a fixed compression ratio. The recent window $\mathrm{tail}_L(h_t)$ preserves local syntax and immediate tool feedback; z_t retains information relevant beyond it.

3.2 SUFFICIENCY-GATED ADAPTIVE MEMORY

An unconditional rate penalty can collapse heterogeneous histories to nearly the same short output. Instead, we sample candidates under several proposal ceilings $b_m \in \mathcal{C}_B$ while providing the same maximum budget B_t to the policy. Let $g_m \in \{0, 1\}$ indicate whether candidate m preserves continuation utility within tolerance, retains the required task-state signature, and introduces no incremental safety or unsupported-certainty failure. If at least one candidate is safe, let $\ell_b^* = \min_{m:g_m=1} \ell(z_m)$. We augment its continuation score with

$$\mathcal{A}_{\mathrm{rate}}(z_m; b_t) = \alpha g_m \left(1 - \frac{\ell(z_m) - \ell_b^*}{B_t}\right) - \kappa d^-_{\mathrm{state}}(z_m) - \beta \frac{[\ell(z_m) - \widetilde{B}(b_t)]_+}{\max(B_t - \widetilde{B}(b_t), 1)}, \tag{5}$$

where $\widetilde{B}(b_t)$ is a soft allowance derived from training-time state complexity rather than raw history length. The first term rewards the shortest *safe* member of the group and contributes to the scalarized candidate score alongside continuation utility and state-preservation terms. The final term penalizes only the length exceeding the state-dependent allowance. At inference, neither the sufficiency gate nor reference rollouts are needed: the compactor receives a maximum budget and its learned stopping behavior produces a variable-length state.

3.3 COUNTERFACTUAL CONTINUATION REGRET

Paired continuation. For a saved boundary and candidate z_t, we generate paired full-context and compact-context continuations of horizon K:

$$\tau^f_{t:t+K} \sim P_{\pi_\phi,\mathcal{E}}(\cdot \mid b_t, x^f_t, \omega), \tag{6}$$

$$\tau^z_{t:t+K} \sim P_{\pi_\phi,\mathcal{E}}(\cdot \mid b_t, x^z_t, \omega). \tag{7}$$

Both continuations begin from the same environment state and share exogenous randomness ω whenever supported by the backend, including user-simulator events, tool responses, retrieval results, and environment seeds. For stochastic policies, shared random inputs can couple token draws even when the contexts induce different token distributions; the sampling rule and treatment of divergent trajectories must be specified. Common random numbers reduce the variance of the utility difference when branch utilities have positive covariance. Each branch selects actions under its own context. For nonterminal environments, K may be sampled to mix short and long probes.

Bootstrapped utility. For continuation τ, define

$$U_K(\tau) = \sum_{j=0}^{K-1} \gamma^j r_{t+j} + \gamma^K V_\psi(q_{t+K}), \tag{8}$$

where q_{t+K} is a task-relevant outcome state and V_ψ estimates remaining utility under π_ϕ. For tasks completed within the probe, the bootstrap term is zero. The branches share a value estimator; Appendix D.3 bounds the contribution of both endpoint errors.

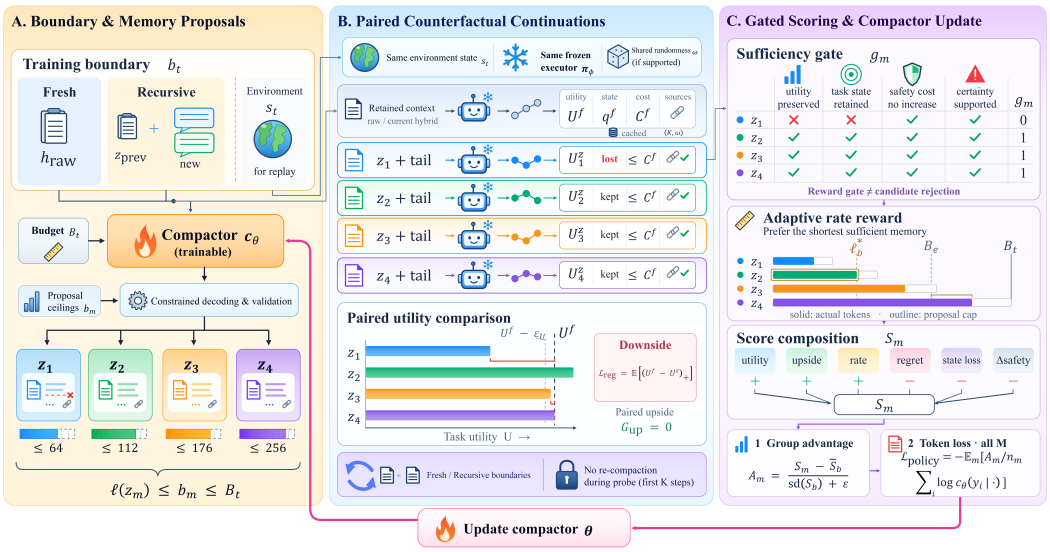

Figure 1: Overview of PACE training. (A) A boundary provides fresh or recursively compacted context, from which the trainable compactor proposes memories under candidate budgets. (B) Full and compact branches are evaluated with paired continuations from the same environment state. (C) Sufficiency gating, adaptive length rewards and regret-aware scores produce the grouped policy update. At inference, PACE emits one memory without reference rollouts or scoring branches.

Downside regret. We define paired continuation regret for each candidate as

$$\mathcal{L}_{\text{reg}}(z_t; b_t) = \mathbb{E}_{K,\omega}\Big[\big[U_K(\tau^f) - U_K(\tau^z)\big]_+\Big]. \tag{9}$$

The full-history rollout is a no-intervention reference. The positive part penalizes harmful omissions while accepting alternative continuations with equal or greater utility. Applying it before averaging prevents gains on some probes from canceling losses on others: $\mathbb{E}[(U_f - U_z)_+] \neq [\mathbb{E}U_f - \mathbb{E}U_z]_+$ in general. Appendix D.4 analyzes dependence on the specified continuation coupling. We record improvements separately as paired upside

$$\mathcal{G}_{\text{up}}(z_t; b_t) = \mathbb{E}_{K,\omega}\Big[\big[U_K(\tau^z) - U_K(\tau^f)\big]_+\Big]. \tag{10}$$

This term lets learning prefer a compact state that removes distracting or stale context. It is not subtracted when reporting downside regret: improvements on one probe do not cancel failures on another.

Task return can remain sparse within short probes. We therefore compare task-relevant outcome signatures:

$$\mathcal{L}^-_{\text{state}}(z_t; b_t) = \mathbb{E}\Big[d_q^-(q^f_{t+K}, q^z_{t+K})\Big], \tag{11}$$

where d_q^- is directional: it measures lost completed subgoals, missing confirmed bindings, newly invalid commitments, and harmful irreversible changes, but not additional valid progress made only by the compact branch. This avoids penalizing benign trajectory differences or improvements over the task state reached by the full-context reference.

Let $C(\tau) \geq 0$ be a cumulative safety or constraint cost. We measure only the increase over the paired reference:

$$\mathcal{L}_{\text{safe}}(z_t; b_t) = \mathbb{E}\Big[\big[C(\tau^z) - C(\tau^f)\big]_+\Big]. \tag{12}$$

This term measures incremental constraint cost under the same continuation coupling as the utility comparison. Costs shared by both branches cancel, so the penalty targets increases associated with the compact context. Next-action disagreement may be logged or used as early shaping for irreversible actions, but it is not part of the primary distortion: multiple action sequences can be behaviorally equivalent.

The future distortion is

$$\mathcal{D}_{\text{future}} = \mathcal{L}_{\text{reg}} + \mu_q \mathcal{L}^{-}_{\text{state}}. \tag{13}$$

We optimize compact-branch utility while controlling one-sided distortion:

$$\max_{\theta} \ \mathbb{E}_{b_t, z_t \sim c_\theta}[\lambda_u U_K(\tau^z) + \lambda_+ \mathcal{G}_{\text{up}} - \mathcal{D}_{\text{future}} + \mathcal{A}_{\text{rate}}(z_t; b_t)] \tag{14}$$

$$\text{s.t. } \ell(z_t) \le B_t, \qquad \mathbb{E}[\mathcal{L}_{\text{safe}}] \le \delta. \tag{15}$$

The absolute compact-branch utility provides a learning signal even when both branches have equally high or low utility. The optional upside bonus explicitly rewards beneficial denoising.

3.4 Training under repeated compaction

We mix *fresh* and *recursive* boundaries to match deployment histories containing earlier memories. Recursive samples compact a prefix, append observations, and compact the hybrid history again. Both branches receive identical subsequent observations and differ only in retaining or replacing the current boundary. This preserves paired credit while training recovery from stale or incomplete memories.

Compact states distinguish confirmed facts, unresolved hypotheses, provenance, and pending subgoals. When the observed prefix does not support a conclusion, unsupported certainty incurs a state cost. When later evidence becomes available, evidence-backed corrections receive utility rather than a consistency penalty. During early training we optionally reserve a small fraction of candidates for an uncertainty-marked proposal when evidence is incomplete. This guided proposal is evaluated using the same continuation reward and is not used at inference.

3.5 Regret-aware policy optimization

For each sampled boundary, we draw M candidates from $c_{\theta_{\text{old}}}$. We run the full-context continuation once for each (K, ω) pair, cache it, and compare all candidates against it. This requires one reference branch and M candidate branches rather than $2M$ branches from independent pairs. Local probes disable further compaction for their first K steps; otherwise a later summary could repair or amplify the current intervention and confound boundary-level credit.

For candidate m, an empirical score is

$$S_m = \lambda_u \widehat{U}^z_m + \lambda_+ \widehat{\mathcal{G}}_{\text{up},m} - \widehat{\mathcal{D}}_{\text{future},m} + \widehat{\mathcal{A}}_{\text{rate}}(z_m; b_t) - \eta\big(\widehat{\mathcal{L}}_{\text{safe},m} - \delta\big), \qquad A_m = \frac{S_m - \bar{S}_b}{\text{sd}(S_b) + \varepsilon}, \tag{16}$$

where scores are normalized across candidates from the same boundary. This reduces variation associated with differences in task difficulty. For a generated compact-state sequence $z_m = (y_1, \ldots, y_{n_m})$, we minimize a group-normalized, reward-weighted token loss

$$\mathcal{L}_{\text{policy}} = -\mathbb{E}_m \left[A_m \frac{1}{n_m} \sum_{i=1}^{n_m} \log c_\theta(y_i \mid y_{<i}, h_t, B_t) \right]. \tag{17}$$

The loss is a policy-optimization surrogate incorporating forced-prefix proposals, length-dependent sampling, and within-group normalization. Each group contains alternative memories at a shared boundary, enabling the update to use within-boundary differences in continuation quality. Per-candidate token averaging normalizes each memory's contribution by its length and gives each candidate equal weight in the outer expectation. The online updates use fixed penalty coefficients and continuation-based candidate scores. At inference, c_θ emits one compact state; no reference rollout, fork, or judge is required.

Appendix D formalizes future sufficiency and gives a local-to-global preservation bound under matched boundary occupancies and continuation rules, together with finite-continuation error analysis.

4 Experiments

We evaluate PACE on controlled long-context question answering. Each continuation terminates in an answer, enabling exact replay of the same boundary under full and compact contexts. Answer F1 measures downstream task utility.

We organize the study around four questions: how PACE compares with existing compression baselines across different history lengths; whether local paired feedback improves learning under delayed rewards; how performance changes with the memory budget; and how the regret penalty affects downstream performance.

4.1 EXPERIMENTAL SETUP

We use HotpotQA Distractor (Yang et al., 2018). Training uses a pool of 19,616 eligible questions sampled from HotpotQA, with distractor documents sampled from other questions in the dataset. The coefficient-sensitivity study uses separate training runs. The evaluation set contains 256 fixed questions disjoint from the training pool, with every question weighted equally. The main comparison covers 8K, 64K, and 256K history lengths, with a 256-token compact-memory ceiling. Each task contains the question, its ten original documents, the answer, and supporting-fact annotations. To isolate compression from retrieval variance in the 8K setting, we deterministically pad each history toward 8,192 prompt tokens with documents sampled from other questions and explicitly label them as unrelated records. Original and distractor documents are shuffled together to prevent document position from serving as a relevance cue.

For the 8K cross-dataset comparison, we sample a fixed set of 128 answerable examples from MuSiQue (Trivedi et al., 2022) and expose all 20 source paragraphs to the compactor. Gold answers, aliases, support labels and decomposition annotations are used only for evaluation. We transfer the HotpotQA-trained checkpoint directly to MuSiQue, retaining the 256-token memory ceiling, frozen executor, and padded-context construction. For the 64K and 256K comparisons, we construct fixed histories at these lengths from the same task format and evaluate 64 and 32 questions per dataset, respectively.

Baselines. We compare PACE with three learned compression baselines and a full-history reference. Full context/history feeds the uncompressed history directly to the frozen QA executor. LLMLingua-2 (Pan et al., 2024) performs task-agnostic token selection from the history without access to the task question. RECOMP-Ext (Xu et al., 2024) scores candidate sentences by their embedding similarity to the question and selects sentences under the memory budget. Provence (Chirkova et al., 2025) uses question-conditioned passage reranking and pruning to retain relevant text. We use the released baseline checkpoints with budget adaptations that enforce a 256-token retained-memory ceiling. The resulting memories are evaluated with the same frozen Qwen3.5-4B QA executor used by PACE. Baseline implementation and budget-adaptation details are provided in Appendix C.

Training and inference. The compactor and frozen executor use the same Qwen3.5-4B backbone; only the compactor receives rank-8 LoRA updates. Training uses 500 updates, four candidate ceilings (64/112/176/256 tokens), and a fresh/recursive/recursive boundary schedule. The executor compares full and compact boundary states with identical future evidence. We use bf16 AdamW with a peak learning rate of 2×10^{-6} and 60 warm-up updates. The reward combines answer quality, downside regret and state-quality terms with a sufficiency gate; all coefficients, boundary construction and compute accounting are in Appendix A. Evaluations use a 256-token memory ceiling and 32-token greedy executor answers, without test-time candidate selection.

Metrics and pairing. We report answer exact match (EM), token F1, one-sided counterfactual regret $[F_{\text{full}} - F_{\text{compact}}]_+$, supporting-fact token F1, memory length, and compression ratio. The full-context executor, learned compression baselines, and trained adapter use the same task IDs and frozen executor. We report nonparametric 95% bootstrap intervals over paired task IDs (10,000 resamples with a fixed seed of 2027). Method observations remain paired within each question. For the 256-question HotpotQA evaluation, we use stratified bootstrap resampling with fixed stratum sizes and preserve method pairing within each question. The reported intervals are pointwise and conditional on this fixed evaluation sample.

4.2 MAIN RESULTS AND BASELINE COMPARISONS

The 8K rows of Table 1 report the fixed 256-question HotpotQA comparison and a 128-question MuSiQue transfer evaluation using the same trained adapter. On HotpotQA, PACE reaches 81.90 F1 with 135.05 memory tokens from a mean 8,064.7-token compactor input, including the instructions

Table 1: QA performance across 8K, 64K, and 256K history lengths. All three lengths are included in the main comparison; scores are percentages. PACE uses the same trained checkpoint throughout. Bold marks the best compact-method point estimate within each displayed comparison (including ties).

8K histories

	HotpotQA				MuSiQue			
Method	F1 ↑	EM ↑	Regret ↓	Memory ↓	F1 ↑	EM ↑	Regret ↓	Memory ↓
Full context	80.58	69.14	0.00	7854.46	48.43	34.38	0.00	7860.84
LLMLingua-2	32.21	26.56	49.58	233.25	9.33	3.91	39.96	227.64
Provence	63.76	53.12	18.08	145.33	35.86	26.56	21.70	**118.57**
RECOMP-Ext	73.06	56.25	8.28	250.98	37.80	28.91	18.41	251.34
PACE (ours)	**81.90**	**68.36**	**4.66**	**135.05**	**55.91**	**42.97**	**12.36**	196.60

64K and 256K histories

	64K histories				256K histories			
	HotpotQA		MuSiQue		HotpotQA		MuSiQue	
Method	F1 ↑	EM ↑	F1 ↑	EM ↑	F1 ↑	EM ↑	F1 ↑	EM ↑
Full history	68.37	56.25	41.25	28.13	52.92	37.50	21.09	15.63
LLMLingua-2	18.95	9.38	11.23	1.56	20.94	15.63	3.65	0.00
Provence	63.21	48.44	29.81	20.31	70.40	**53.13**	31.18	28.13
RECOMP-Ext	70.30	**59.38**	33.90	23.44	64.61	43.75	18.23	15.63
PACE (ours)	**74.52**	53.12	**41.62**	**26.56**	**71.91**	**53.13**	**39.89**	**31.25**

Budget and evaluation. Memory in the 8K rows counts retained history tokens, excluding question and instruction wrappers. Compression methods use a 256-token memory ceiling. Full-history rows are direct frozen-reader references; counterfactual regret is not reported in this extended-history block.

and question. Uncompressed histories average 7,854.46 tokens; the Memory column of the table excludes the question and instruction wrappers.

The mean memory-to-compactor-input ratio of PACE is 1.67%. The answer F1 point estimate exceeds that of the full-context reference. This suggests that PACE compression can help the model focus on key answer-relevant information amid large amounts of irrelevant context. One-sided regret is 4.66%, measuring per-question downside separately from improvements on other questions. Memory length quantifies the retained context; compaction processes the full input before generating this memory.

External compression baselines. LLMLingua-2 reaches 32.21 F1; Provence reaches 63.76 under top-5 pruning and threshold calibration. PACE exceeds Provence by 18.14 F1 points (CI $[13.29, 23.09]$). Exact and normalized question-text matching found no matches in the public labeling archives for Provence. Budget adaptations and the treatment of empty Provence memories are detailed in Appendix C.

Scaling to long histories. Table 1 also reports the 64K and 256K settings on fixed question sets. Question IDs and source histories are fixed within each length, and the sets at the two lengths are disjoint. PACE processes each new 64K block together with the preceding compact state and replaces them with a new state under a 256-token ceiling. This rolling protocol continues until the entire history has been processed. The external baselines instead score the complete history with their native encoder batching and existing 256-token budget adapters. All methods use the same frozen QA executor.

At 64K, PACE reaches 74.52 F1 on HotpotQA and 41.62 on MuSiQue, compared with 68.37 and 41.25 for direct full-history execution. At 256K, after repeated rolling compaction, PACE reaches 71.91 and 39.89 F1, while direct full-history execution reaches 52.92 and 21.09. PACE retains answer-relevant information across successive compression steps as the history grows.

4.3 Ablation studies

We isolate the contribution of counterfactual-future feedback in a controlled repeated-compaction task, holding the policy class fixed.

Counterfactual-future feedback ablation. We ablate the local paired counterfactual feedback that operationalizes "what changes the future," replacing it with terminal-only reward in a controlled synthetic task. The comparison uses a scalar policy with hand-written memory alternatives. A deterministic tool task reveals eight one-time values across successive calls and requires an exact final submission after seven compaction boundaries. A scalar policy chooses between a sufficient and an information-dropping memory. Counterfactual-Future Feedback evaluates alternatives at the same replayable boundary; Terminal-Reward-Only RL receives only final binary success. Both use the same policy class and a learning rate selected before evaluation on the 64 confirmation seeds.

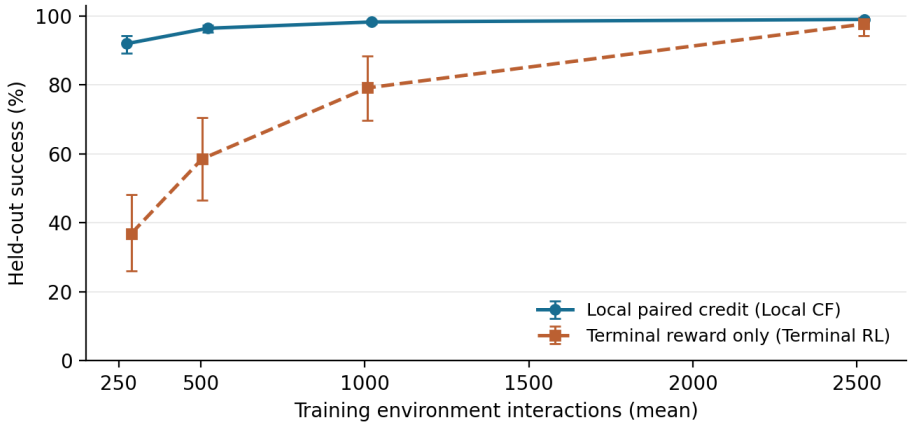

Figure 2: Counterfactual-future feedback ablation in the synthetic repeated-compaction task. Success is averaged over 64 training seeds; error bars are pointwise 95% seed-bootstrap intervals. The horizontal axis reports actual mean training interactions, including reference and candidate continuations. Lines connect the four measured budgets.

Figure 2 shows that training with counterfactual-future feedback is substantially more sample-efficient at nominal budgets of 250–1,000 interactions. Terminal-Reward-Only RL achieves comparable performance by 2,500 interactions. At the three smaller budgets, 40, 26, and 13 terminal-reward runs receive no nonzero update, respectively, whereas the paired probe always exposes a within-boundary contrast. The results show that counterfactual-future feedback improves sample efficiency in this sparse, delayed-reward task.

4.4 Hyperparameter sensitivity

Memory budget. We fix the seed-47 checkpoint and the same 128 official-validation histories, then vary the compact-memory cap from 64 to 256 tokens without retraining. PACE regenerates memories and answers at every cap. Table 2 reports answer quality, regret and realized memory length; caps are upper bounds rather than prescribed output lengths.

Table 2: PACE memory-budget sensitivity on 128 fixed HotpotQA questions. F1, EM, and regret are percentages; memory is measured in retained text tokens.

Cap	F1 ↑	EM ↑	Regret ↓	Memory tokens ↓
64	75.45	59.38	6.54	52.48
112	79.32	62.50	4.31	82.71
176	77.36	60.16	6.54	125.02
256	77.33	59.38	5.83	136.37

Realized memory increases with the cap, while F1 is non-monotonic. The 112-token condition has the highest F1 point estimate and the lowest regret among the measured caps. Across caps of 112–256

tokens, F1 remains between 77.33 and 79.32, indicating relatively stable performance over this budget range. This suggests that, with a sufficient token budget, PACE can retain the key information needed for answering without requiring a precisely tuned memory cap. Paired F1 intervals for each lower cap relative to 256 include zero.

Regret-weight sensitivity. We examine positive downside-regret coefficients $\lambda_r \in \{0.5, 1, 2, 3, 4\}$, holding the remaining training configuration fixed (seed 47, 500 updates). All final checkpoints are evaluated using the same 128 HotpotQA histories, frozen answer model, 256-token cap, and fixed evaluation prompt.

Table 3: Regret-weight sensitivity on the fixed HotpotQA subset. Scores are percentages; memory is measured in tokens. Bold marks the best point estimate in each metric (higher is better for F1/EM; lower is better for Regret/Memory).

λ_r	F1 ↑	EM ↑	Regret ↓	Memory ↓
0.5	77.25	60.94	5.16	137.07
1	77.33	59.38	5.83	136.37
2	77.50	**61.72**	5.00	**135.38**
3	**78.07**	**61.72**	4.78	139.38
4	77.88	60.94	**4.66**	140.30

Increasing the regret weight from $\lambda_r = 0.5$ to $\lambda_r = 3$ raises F1 from 77.25 to 78.07. EM reaches its joint maximum of 61.72 at weights 2 and 3. From weight 2 to 4, regret decreases from 5.00 to 4.66, while mean memory length increases from 135.38 to 140.30 tokens. This trend is consistent with the role of the coefficient: a larger regret weight increases the penalty on downside loss in the training objective. At $\lambda_r = 3$, PACE combines the highest F1 and joint-highest EM with 4.78 regret and 139.38 memory tokens. Increasing the weight to 4 further reduces regret to 4.66, while F1 and EM become 77.88 and 60.94, respectively. Across all tested weights, mean memory length remains within 135.38–140.30 tokens, a span of 4.92 tokens. These results identify a weight around 3 as a favorable setting for answer quality, while a larger weight places greater emphasis on reducing downside regret.

5 Related Work

Long-horizon agents and memory. MEM1, ReSum and related RL approaches learn compact states for continued reasoning (Zhou et al., 2026; Wu et al., 2025; Lu et al., 2025). Context Folding and ContextBudget impose explicit context constraints (Sun et al., 2026; Wu et al., 2026), while Memory-R1 learns external-memory operations (Yan et al., 2026). CompactionRL trains compaction with cross-segment RL credit (Li et al., 2026). PACE assigns local credit through paired continuations at a shared boundary and trains on recursive memories, connecting history compression with value-preserving abstraction (Paischer et al., 2022; Li et al., 2006).

Context compression and memory selection. Query-focused and summarize-then-answer systems retain QA evidence (Xu et al., 2023; Inoue et al., 2021). Our baselines include RECOMP's extractive compression (Xu et al., 2024), Provence's passage pruning (Chirkova et al., 2025), and LLMLingua-2's token selection (Pan et al., 2024). ACON optimizes compression guidelines from failures (Kang et al., 2026); Slipstream validates compactions against uncompacted continuations (Chen et al., 2026). PACE learns from paired continuation losses while accounting for memory cost and repeated use, complementing analyses of compaction error and safety-constraint loss (Tirmazi et al., 2026; Chen, 2026).

6 Conclusion

PACE learns future-relevant agent memory through paired continuation feedback and budgeted compression. Repeated-use analysis connects local omissions to downstream control, while QA and delayed-credit evaluations support this learning objective. Long-history results demonstrate evidence retention across successive compactions. Future work will explore integrating additional summarization methods to further reduce information loss during compaction.

AI USE STATEMENT

The language models and reinforcement learning methods used in this work are part of the research methodology and experimental systems described in the paper. We did not use generative AI tools to assist with research ideation, experimental design, implementation, data analysis, figure preparation, or the writing and editing of the manuscript. All research decisions, experiments, analyses, and manuscript preparation were carried out by the authors.

REFERENCES

Anthropic. Claude Opus 5.5 System Card. System card, 2026. URL https://www.anthropic.com/claude-opus-5-5-system-card.

Shiyang Chen. Governance decay: How context compaction silently erases safety constraints in long-horizon llm agents. *arXiv preprint arXiv:2606.22528*, 2026.

Zhuofu Chen, Rui Pan, Yinwei Dai, and Ravi Netravali. Slipstream: Trajectory-grounded compaction validation for long-horizon agents. *arXiv preprint arXiv:2605.08580*, 2026.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. Provence: efficient and robust context pruning for retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2025, pp. 38082–38106, 2025.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the Limits of KV Cache Compression. Technical report, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash/blob/main/DeepSeek_V41_Tech_Report.pdf.

Google DeepMind. Gemini 3.8 Flash Model Card. Model card, 2026. URL https://deepmind.google/models/model-cards/gemini-3-8-flash/.

Naoya Inoue, Harsh Trivedi, Steven Sinha, Niranjan Balasubramanian, and Kentaro Inui. Summarize-then-answer: Generating concise explanations for multi-hop reading comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6064–6080, 2021.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/66270.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Yujiang Li, Zhenyu Hou, Yi Jing, Jie Tang, and Yuxiao Dong. Compactionrl: Reinforcement learning with context compaction for long-horizon agents. *arXiv preprint arXiv:2607.05378*, 2026.

Miao Lu, Weiwei Sun, Weihua Du, Zhan Ling, Xuesong Yao, Kang Liu, and Jiecao Chen. Scaling llm multi-turn rl with end-to-end summarization-based context management. *arXiv preprint arXiv:2510.06727*, 2025.

OpenAI. GPT-6 Astra System Card. System card, 2026. URL https://deploymentsafety.openai.com/gpt-6-astra.

Fabian Paischer, Thomas Adler, Vihang Patil, Angela Bitto-Nemling, Markus Holzleitner, Sebastian Lehner, Hamid Eghbal-Zadeh, and Sepp Hochreiter. History compression via language models in reinforcement learning. In *International Conference on Machine Learning*, pp. 17156–17185. PMLR, 2022.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 963–981, 2024.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon agent via context folding. In *International Conference on Machine Learning*, 2026.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Hayder Tirmazi, Sam Markelon, Allison Bishop, and Michael Mitzenmacher. Context compaction theory. *arXiv preprint arXiv:2608.01326*, 2026.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Xixi Wu, Kuan Li, Yida Zhao, Liwen Zhang, Litu Ou, Huifeng Yin, Zhongwang Zhang, Xinmiao Yu, Dingchu Zhang, Yong Jiang, et al. Resum: Unlocking long-horizon search intelligence via context summarization. *arXiv preprint arXiv:2509.13313*, 2025.

Yong Wu, YanZhao Zheng, TianZe Xu, ZhenTao Zhang, YuanQiang Yu, JiHuai Zhu, Chao Ma, BinBin Lin, BaoHua Dong, HangCheng Zhu, RuoHui Huang, and Gang Yu. Contextbudget: Budget-aware context management for long-horizon search agents, 2026. URL https://arxiv.org/abs/2604.01664.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

Ruochen Xu, Song Wang, Yang Liu, Shuohang Wang, Yichong Xu, Dan Iter, Pengcheng He, Chenguang Zhu, and Michael Zeng. Lmgqs: A large-scale dataset for query-focused summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14764–14776, 2023.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, et al. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12805–12825, 2026.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Zijian Zhou, Ao Qu, Zhaoxuan Wu, Sunghwan Kim, Alok Prakash, Daniela Rus, Bryan Kian Hsiang Low, and Paul Liang. Mem1: Learning to synergize memory and reasoning for efficient long-horizon agents. In *International Conference on Learning Representations*, volume 2026, pp. 58413–58438, 2026.

A Implementation Details

A.1 Training configuration

Table 4 supplements the training configuration in Section 4.1 with the remaining implementation settings. The LoRA target modules are `q_proj`, `k_proj`, `v_proj`, `o_proj`, `in_proj_qkv`, `in_proj_z`, `in_proj_b`, `in_proj_a`, `out_proj`, `gate_proj`, `up_proj`, and `down_proj`.

The scalar reward coefficients are 1.0 for negative counterfactual regret, 0.5 for compact-answer F1, 0.2 each for upside, supporting-fact F1, and revision, 0.25 for uncertainty calibration, and 0.3 for

Table 4: Additional training hyperparameters.

Parameter	Value	Parameter	Value
Seed	47	LoRA alpha	16
LoRA dropout	0	Weight decay	0
Gradient clip	1.0	Sampling temperature	0.8
Top-p	0.95	Training GPUs	8 H100 80GB

unsupported-certainty cost. Adaptive-memory coefficients are 0.2 for state sufficiency, 0.1 for safe efficiency, 0.25 for complexity coverage, 0.5 for support shortfall, and 0.05 for excess length. The soft target ranges from 48 to 224 tokens. Its gate uses a 0.05 policy tolerance, 0.5 compact-F1 floor, and 0.5 minimum supporting-fact recall.

A.2 Boundary construction and training execution

A fresh training example compresses the first 8K-token segment of a roughly 16K-token history. A recursive example first compresses one segment greedily, combines that memory with the next segment, and trains at the second boundary of a roughly 24K-token history. One proposal exposes an `[UNRESOLVED]` field for incomplete evidence; the longest proposal receives a complexity-dependent minimum length.

Training uses eight H100 80GB GPUs for candidate generation, gradient computation, frozen execution, and gradient synchronization. The 167 fresh and 333 recursive updates use these devices for candidate throughput rather than model parallelism.

A.3 Compactor instruction

PACE uses the following evidence-grounded compactor instruction for the main long-history evaluations; the integer maximum replaces {`budget`}. The sensitivity experiments use a fixed instruction across all conditions.

> Maintain an evidence-grounded memory for answering the question later. The question guides relevance but is not evidence. Do not guess a final answer or fill missing relations using background knowledge. Previous memory is a generated, revisable summary rather than an authority; check new observations for evidence that corrects earlier claims, while retaining valid older facts that remain relevant. Preserve source-linked entities, relations, dates and quantities, and keep unresolved or conflicting links open. Use explicit `[EVIDENCE]` and, when needed, `[OPEN]` sections. Do not output an `Answer:` field, a readiness statement, a token count, or a reasoning narrative. Use as many tokens as necessary but no more, and never exceed {`budget`} tokens. Output only the compact memory.

B Training Seed Stability

To assess sensitivity to optimization randomness, we trained the same PACE configuration from three independent random seeds (47, 48, and 49). Each run completed 500 updates with the same backbone, data mixture, 8K boundary, and 256-token memory ceiling. We then evaluated the resulting checkpoints on the same 128-question HotpotQA evaluation set. Table 5 reports individual runs and their mean and sample standard deviation.

The compact-context F1 ranges from 77.01 to 78.23 across seeds (sample standard deviation 0.63), while regret remains between 5.05 and 5.83. These results indicate consistent performance across the three evaluated training seeds.

C Baseline Implementation Details

The external compressors use the shared evaluation protocol in Section 4.1. The following adapters enforce the 256-token budget using the Qwen executor tokenizer.

Table 5: PACE stability across three independent training seeds. F1, EM, and regret are percentages; memory is the mean number of retained tokens. The last two rows report the mean and sample standard deviation across seeds.

Seed	F1 ↑	EM ↑	Regret ↓	Memory ↓
47	77.33	59.38	5.83	136.37
48	77.01	60.16	5.41	136.62
49	78.23	60.94	5.05	149.01
Mean	77.52	60.16	5.43	140.66
Std. dev.	0.63	0.78	0.39	7.23

C.1 LLMLingua-2

The frozen LLMLingua-2 compressor receives the history without the task question or gold answers. We calibrate output length with the executor tokenizer, allowing up to three compression calls before applying a prefix cap when necessary. This adaptation satisfies the shared memory budget.

On the 128-question official-validation and development subsets, 12 and 4 outputs, respectively, require the final prefix cap. Every output remains in the evaluation. The aggregate results are reported in Table 1.

C.2 RECOMP-Ext

We use the public HotpotQA extractive checkpoint of RECOMP (Xu et al., 2024). The compressor encodes the question and each candidate sentence separately, using masked mean pooling and unnormalized embedding dot products. All history sentences, including distractor documents, are eligible. We split observations with English Punkt and retain the full observation header when scoring.

We greedily pack complete sentences in descending score order with stable tie-breaking, then restore selected sentences to source order. Headers and separators count toward the 256-token budget. If no complete sentence fits, a prefix fallback supplies the memory; no evaluated output required this fallback. The adapter uses fixed-budget selection without RECOMP's optional empty-memory selective augmentation. Selection uses the question and source text, with gold answers and supporting-fact annotations reserved for evaluation.

The 8K evaluation of RECOMP-Ext uses 128 HotpotQA questions and 128 MuSiQue questions.

C.3 Provence

We use `naver/provence-reranker-debertav3-v1` (Chirkova et al., 2025), a 430M-parameter DeBERTa-v3 model with passage-ranking and pruning heads, trained on MS MARCO and Natural Questions. Query-conditioned processing splits observations into blocks of at most 512 tokens, with complete observation headers as titles. Inference uses FP32, batch size 32, top-five passage reranking, and the native first-block title rule.

For each question, one set of encoder scores is reused across thresholds 0.10 through 1.00 in increments of 0.05, plus 0.99. We select the first threshold whose nonempty passages, joined with blank lines, fit the 256-token budget. Thresholds apply to token probabilities and then sentence selection fractions, following the native pruning procedure. Selection uses encoder scores and token counts; answer scores and supporting-fact labels are reserved for evaluation. Selected text is kept intact. Nine of the 256 HotpotQA questions yield empty memories at the final threshold and remain in the evaluation.

In addition to the F1 contrast reported in the main text, PACE improves EM by 15.23 percentage points (95% CI $[9.38, 21.09]$) and reduces regret by 13.43 points (CI $[-17.97, -9.07]$). The memory difference is -10.29 tokens (CI $[-23.13, 2.62]$). Intervals use the paired question-bootstrap protocol in Section 4.1.

D THEORETICAL ANALYSIS

We formalize the conditions under which low local regret preserves policy performance across repeated compactions. We fix the policy π_ϕ, task distribution, boundary rule, and environment coupling throughout the analysis.

D.1 FUTURE-SUFFICIENT CONTEXT ABSTRACTION

Let $U(\tau)$ denote the full remaining discounted utility from a boundary, and let paired full and compact continuations be defined on the same probability space.

Definition 1 (Future sufficiency)**.** *A compact state mapping is ϵ-future-sufficient under boundary distribution d if*

$$\mathbb{E}_{b\sim d, z\sim c_\theta,\omega}\left[\left[U(\tau^f) - U(\tau^z)\right]_+\right] \le \epsilon. \tag{18}$$

This condition concerns utility rather than equivalence between action distributions or textual similarity. It bounds average paired downside for the served policy on the specified distribution. The positive-part expectation also admits a risk interpretation: for any $a > 0$, the Markov inequality gives $\Pr(U(\tau^f) - U(\tau^z) \ge a) \le \epsilon/a$. Thus low expected downside controls the frequency of large compaction failures.

D.2 LOCAL-TO-GLOBAL POLICY PRESERVATION

Consider an episode with N scheduled compaction boundaries. Define hybrid execution H^i to use compact contexts at its first i boundaries and oracle-restored raw histories thereafter. Thus, H^0 corresponds to full-context execution, and H^N corresponds to deployed repeated compaction. These are analytical interventions: a shadow log retains observations along the trajectory of each hybrid, even when those observations are unavailable to the policy. At boundary i, adjacent hybrids compare oracle restoration with an additional compaction of the currently available state. Both restore their respective raw logs at later boundaries. This oracle comparison may differ from a probe that retains only the current hybrid memory. Let J^i denote the expected return of this hybrid execution.

Proposition 1 (Hybrid compaction bound)**.** *Suppose the expected one-sided full-horizon return difference at the ith intervention is at most ϵ_i under the boundary occupancy induced by H^{i-1} and the continuation rule and coupling of the adjacent hybrids. Utilities are discounted to the start of the episode. Then*

$$J^0 - J^N \le \sum_{i=1}^{N} \epsilon_i. \tag{19}$$

Proof. Adjacent hybrids differ only at boundary i. Because $x \le [x]_+$, their expected return difference satisfies $J^{i-1} - J^i \le \epsilon_i$ by the assumed local regret bound. Telescoping over hybrids yields $J^0 - J^N = \sum_{i=1}^{N}(J^{i-1} - J^i) \le \sum_i \epsilon_i$. □

The bound requires local errors to be evaluated under the boundary occupancies, continuation rules, and coupling of the adjacent hybrids. These occupancies include states reached after earlier compactions, motivating recursive training on previously generated memories. The sum in Equation 19 describes how local losses can accumulate across successive interventions.

D.3 FINITE-CONTINUATION APPROXIMATION

For nonterminal probes, endpoint-value error bounded by δ_V contributes at most $2\gamma^K\delta_V$ to the regret error. A further residual-tail term ρ_K remains because positive-part regret is nonlinear in stochastic returns; exact values alone do not remove it. Lemma 1 in Appendix D.4 states both bounds and the full definitions. Our terminal QA probes incur no such truncation error. A global bound based on finite probes uses the same hybrid occupancies and continuation rules as Proposition 1.

D.4 ADDITIONAL PROOF DETAILS

Dependence on continuation coupling. For integrable paired utilities, $x_+ = (x + |x|)/2$ gives

$$\mathbb{E}[(U_f - U_z)_+] = \tfrac{1}{2}\big(\mathbb{E}U_f - \mathbb{E}U_z + \mathbb{E}|U_f - U_z|\big). \tag{20}$$

The mean utility difference depends only on the branch marginals, whereas the absolute difference also depends on their coupling. For two Bernoulli$(1/2)$ marginals, setting $U_f = U_z$ gives zero downside, independent draws give $1/4$, and setting $U_z = 1 - U_f$ gives $1/2$. All three have zero mean utility difference. Consequently, equality in distribution is not sufficient for zero paired regret, and changing the coupling changes the risk estimand, not merely the variance of its estimator. The hybrid bound therefore uses a specified coupling between adjacent executions.

For Proposition 1, define the random return difference between adjacent hybrids as $D_i = U(H^{i-1}) - U(H^i)$. Coupling the two executions up to boundary i makes their pre-boundary rewards identical. By construction, the conditional post-boundary difference is the full-versus-compact continuation difference, and $D_i \le [D_i]_+$. Taking the conditional expectation under the H^{i-1} boundary occupancy gives $\mathbb{E}[D_i] \le \epsilon_i$. The tower property and the telescoping identity $U(H^0) - U(H^N) = \sum_i D_i$ then give Equation 19, with each expectation taken under its corresponding hybrid occupancy.

Finite-continuation definitions and bound. Let $\widehat{U}_b$ be Equation 8 for branch $b \in \{f, z\}$, and $\bar{U}_b$ the same prefix with the exact endpoint conditional value V^π in place of V_ψ. The endpoint must include all state needed for this conditional value. Define $R = \mathbb{E}[(U_f - U_z)_+]$, $\widehat{R} = \mathbb{E}[(\widehat{U}_f - \widehat{U}_z)_+]$, and $\rho_K = \mathbb{E}[|(U_f - U_z) - (\bar{U}_f - \bar{U}_z)|]$.

Lemma 1 (Bootstrap and residual-tail error)**.** *If $|V_\psi(q) - V^\pi(q)| \le \delta_V$ at both branch endpoints, then*

$$\left|\widehat{R} - \mathbb{E}[(\bar{U}_f - \bar{U}_z)_+]\right| \le 2\gamma^K \delta_V, \qquad |R - \widehat{R}| \le 2\gamma^K \delta_V + \rho_K. \tag{21}$$

Both inequalities follow from the 1-Lipschitz positive-part map and the triangle inequality. A finite-probe version of the hybrid bound additionally requires matching hybrid occupancies and continuation rules, with the appropriate episode discount on each boundary error. Replacing expectations with sample means requires an appropriate sampling and concentration argument.

In this lemma, branch errors are measured relative to $\bar{U}_b$ (the prefix return plus the exact conditional endpoint value), not the realized return U_b. Lipschitz continuity bounds these errors by $2\gamma^K \delta_V$; a second application contributes ρ_K. To see why this term is needed, consider $\gamma = 1$, a prefix with zero reward, and equiprobable coupled future rewards $(U_f, U_z) = (1, 0)$ or $(0, 1)$. Both endpoint values are $1/2$: exact-value bootstrap regret is zero but full-return regret is $1/2$, despite $\delta_V = 0$. This error is distinct from finite-sample uncertainty.