

Response to Reviewers, Tracking #966-1995 (RESUBMIT)

Title: A Neurosymbolic Benchmark for Temporal Knowledge-Graph Memory in Partially Observable Environments

Authors: Taewoon Kim, Vincent François-Lavet, Michael Cochez

We thank all three reviewers for their careful and constructive reviews. We have revised the paper substantially in response. Below we respond point by point; section, table, figure, and listing numbers refer to the revised manuscript. A copy of the paper with the changes marked against the submitted version, additions in green and removals in red, accompanies this response.

Summary of the largest changes:

1. A new early subsection, “**Agent interface and task protocol**” (Section 3.1), defines the observation–query–action protocol, the supported agent classes, and the role of the reward signal before any agent is introduced (Reviewer 1).
2. A new “**Reification and RDF 1.2**” discussion in Section 2.1, with a side-by-side Turtle example (Listing 1), states explicitly that the RDF 1.2 annotation model is the conceptual memory model while the released artifacts serialize it via standard RDF reification, and why (Reviewer 3).
3. New **memory-operation semantics** (Section 4.1) define insertion, repeated observation, short-term precedence, query-time update, traversal update, contradictions, and eviction and deletion (Reviewer 3).
4. A new “**Calibration Bounds**” subsection (Section 5.2) anchors the absolute scores between an analytic upper bound of 100, the symbolic agents at saturated capacity ($K=512$, unchanged at $K=1024$), a single best-neural bound over all eight trained configurations, and an observation-only lower bound (Reviewers 2 and 3).
5. The **spread over all 27 TKG policy variants** (worst/median/best per capacity) is now reported alongside the training-selected variant, both in the text and as dotted best/worst bounds in the redesigned Figure 3 (Reviewer 2).
6. A **calibration table** (Table 3) consolidates all reference bounds with exact numbers (Reviewers 2 and 3).
7. **Organization:** two Background subsections duplicated material that Related Work already covered, with the same citations. They have been removed, so Background now carries only the representational preliminaries (RDF and RDF 1.2, temporal KGs) and all comparative discussion sits in one place. Headings are also numbered in the revised manuscript, so the pointers in this letter resolve directly (Reviewer 1).

In addition, prompted by R2.5 we re-checked every comparative claim in Section 5 against the figure data, and found more of the same problem: with four neural curves now plotted, any sentence naming one of them as “the best neural agent” can be refuted by pointing at another curve. We therefore stopped comparing agent to agent. Section 5.1 now reports the neural family by a **bound**: across all eight trained configurations (two architectures $\times$ two sizes $\times$ with/without arrival-time features), no neural agent reaches more than **18.6 on train or 14.2 on test at any capacity**, and on the held-out environment none gains more than 0.6 points across the whole capacity range. Every claim in Section 5 about the neural agents is now stated against that bound, so no single neural agent can refute it. The symbolic comparison is made directly instead, because no capacity contradicts it: the TKG agent leads the KG agent at every

capacity from $K=32$ upward, on train and on test. The crossovers are then simple to state: both symbolic agents exceed that bound, the TKG agent from $K=64$ on train and $K=32$ on test, and the KG agent from $K=128$ and $K=64$.

Reviewer 1

R1.1: “... it would help to include a short subsection or introductory paragraph explaining the agent interface ... whether the benchmark is intended for symbolic agents, RL agents, neural sequence models, LLM-based agents, or any agent satisfying the observation–query–action protocol.”

We agree. The revised paper opens Section 3 with a new subsection, “Agent interface and task protocol” (Section 3.1), which defines: what the agent observes (o_t), what it must answer (the location query), what action it chooses, that private memory is unconstrained, and that the benchmark supports any agent satisfying the observation–query–action protocol (symbolic, RL, neural sequence, or LLM-based), before the benchmark internals and the experimental agents are introduced. The previous “Memory Interface (Agent-Agnostic)” subsection has been merged into it.

R1.2: “The discussion of ‘memory queries and exploitation policies’ needs more explanation ... distinguish clearly between (1) policies used to answer a query, (2) policies used for exploration or frontier selection, and (3) policies used for memory eviction.”

Done. Section 4 now introduces the three policy families explicitly, in exactly this three-way structure (“Three policy families” paragraph), and Section 4.1 adds a “Memory-operation semantics” block specifying how `:time_added`, `:last_accessed`, and `:num_recalled` are initialized and updated: insertion sets `time_added = last_accessed = t` and `num_recalled = 0`; a statement selected by the QA policy gets `last_accessed` refreshed and `num_recalled` incremented. Two further rules that were previously implicit are now stated: the same refresh applies to each edge statement an exploration step relies on, so access counts reflect navigation as well as answering; and a query answered from the agent’s current room bypasses the policies entirely and updates nothing, which is why an agent with no long-term memory still answers a few queries correctly. These values are what all three policy families rank over.

R1.3: “The mention of reward appears before the paper has clearly introduced reinforcement learning ... clarify whether the reward is part of the benchmark interface for all agents, only used for RL-trained baselines, or simply an evaluation signal.”

Your reading is correct: it is primarily an evaluation signal. Section 3.1 now defines the per-step correctness signal $r_t \in \{0, 1\}$ first, as the evaluation metric for **all** agents, and then states that whether an agent additionally consumes it for training is up to the agent: the symbolic agents never do, while the neural baselines are DQN-trained on it. The question–move loop (Section 3.5) refers back to that definition, so the term “reward” no longer appears before it has been defined.

R1.4: “The paper would also benefit from a more explicit description of the query types.”

Section 3.1 now states explicitly that object-location queries ($X, :at_location, ?$) are the sole query class in the proposed benchmark; the Limitations section continues to list richer query families (temporal comparison, multi-hop, counterfactual) as future work.

R1.5: Minor: define `lcm`; typographical issues (“question-answer” vs “question answering”; `time_added` notation).

All fixed: `lcm` is expanded as “least common multiple” at first use (Section 3.3); “question-

answer” has been corrected to “question answering” throughout (including the abstract); annotation-property notation is now uniformly `:time_added` style.

Reviewer 2

R2.1: “...many of the details in the construction and the evaluation are provided externally to the paper.”

We agree that we might have deferred too much to the artifacts. The revision brings the representative material into the paper itself, taking your four examples in turn. **Deterministic replays:** Listings 2 and 3 are a two-step excerpt of one periodic orbit, the same room at $t = 99$ and $t = 0$ with its northern edge open in the first and a wall in the second, so the periodicity the text claims is visible on the page (Section 3.3). **Room coverage:** Figure 4 plots room and triple coverage per timestep for all four agents, and Section 5.3 reads the numbers (both symbolic agents reach all 49 rooms, the TKG agent at timestep 70 and the KG agent at 74). **Navigation:** the BFS exploration procedure and what the exploration policy does and does not decide are now specified in Section 4.1. **Evolution images:** Figure 5 shows the TKG agent’s memory at $t = 0$, $t = 50$ and $t = 99$, with the node counts narrated in Section 5.4. Alongside these the paper now shows a complete observation in Turtle (Listing 3), a hidden-state fragment (Listing 2), a side-by-side memory representation (Listing 1), the train/test protocol (Table 1), the 27-variant spread (Section 5.1), hyperparameter summaries (Section 5), and the calibration bounds (Section 5.2). We still kept some aspects in the artifacts: the code and replay scripts, the raw results for the neural and TKG runs, the serialized Turtle snapshots, and the memory-evolution series for the TKG agent at every timestep. Each is pointed to explicitly at the place it is used.

R2.2: “...the experiments are, to my view, underspecified and incomparable between neural and symbolic systems. The systems are evaluated with a memory capacity ranging from 0 to 512 ... 512 what?”

The revision adds a dedicated “Capacity semantics” paragraph (Section 4). K counts *stored entries*, and one entry is one observed triple together with the information the agent keeps alongside it: for the TKG agent an annotated occurrence of that triple, for the KG agent the bare triple, and for the neural agents one tokenized triple in a buffer of length K . What differs between these is how much each entry carries, and how fast entries accumulate: the KG agent’s store is a set, so re-observing a fact collapses onto the entry already held, whereas the TKG agent records each observation separately. We now made it explicit such that the reader is aware that comparisons between families have this caveat.

We also note that an episode emits only ~500–600 observation triples, so $K = 512$ approaches exhaustive retention for all families.

R2.3: “...the authors present, for each capacity, the best-performing system out of 27 options ... thus giving an unsafe advantage ... provide also information about the worst, or at least the median system.”

All numbers in this answer are the **TKG agent’s** 27 policy variants evaluated on the held-out environment. We would like to clarify the selection protocol first: the reported variant is selected by **training** accuracy and then evaluated once on the held-out test environment, i.e., standard model selection without access to test performance. That said, we agree the spread is informative, and the revision now reports worst/median/best over all 27 variants (Section 5.1): e.g., on test, (7.80 / 9.68 / 15.52) at $K=32$ and (45.28 / 45.72 / 46.52) at $K=512$. The spread shows that policy choice matters greatly under memory scarcity but that at high capacity all 27 variants converge to within 1.3 points, so the headline result is robust to the selection.

You also note that the selected system “could differ from capacity to capacity”. It does, and the revision now reports the pattern. The policy configuration (QA rule, exploration rule, eviction rule) is a hyperparameter of the symbolic agent, and tuning a hyperparameter per operating point on training data, then evaluating once on held-out data, is ordinary model selection rather than an advantage. The neural side is treated the same way, as noted above.

What varies is also instructive. LFU is the eviction rule of the selected variant at eight of the ten capacities: it wins outright at six, shares the top score with LRU at $K=0$, and ties with FIFO and LRU at $K=512$. FIFO is selected at $K=4$ and LRU at $K=256$. Eviction is therefore the most stable of the three dimensions without being fixed, and it has no bearing on the headline result. The QA and exploration rules, by contrast, change from capacity to capacity without a simple pattern. Under scarcity the variants differ widely (the spread at $K=32$ runs from 7.80 to 15.52) and which one wins is not stable, while at $K=512$ they converge to within 1.3 points and the choice stops mattering. Readers who want the un-selected picture have it: the worst/median/best spread is reported in Section 5.1, and the best/worst envelope over all 27 variants is drawn per capacity in Figure 3.

R2.4: “the results are provided without any analysis or reading of the numbers provided.”

The revision adds analysis throughout Section 5: a train-test-gap reading (the neural agents lose about a fifth to a quarter of their accuracy under the permuted query order, while the symbolic agents hold theirs: the TKG agent is slightly higher on test, inside its across-seed standard deviation, and the KG agent slightly lower), the capacity-scarcity interpretation of the 27-variant spread, and the calibration-bounds interpretation (what the remaining gap to the analytic upper bound consists of).

R2.5: “In page 7 the authors claim that the Transformer agent outperforms the LSTM agent, but this is not supported by the figure.”

Thank you for catching this; the sentence was imprecise and, worse, contradicted a later sentence that called the LSTM the best neural agent. Checking it against the figure data, which architecture leads varies with capacity and environment, so the sentence’s blanket ordering is not supported. Rather than restate it more carefully, we removed it: the paper no longer claims an ordering between the two neural architectures, because no result depends on one. Section 5.1 now reports what they have in common (their accuracy is nearly flat in capacity, and both lose about a fifth to a quarter of their accuracy from train to test) and bounds the neural family as described in the summary above.

R2.6: “if the run is deterministic, how could the neural methods not visit all rooms?”

Good question; the revision now addresses it explicitly (Section 5.3). The environment is deterministic, and at test time the neural agents are deterministic as well (greedy argmax over Q-values, no ϵ -exploration). Determinism, however, does not imply good exploration: what differs between agents is the policy itself. The symbolic agents follow a hand-designed BFS exploration procedure, whereas the neural agents must learn exploration end-to-end and converge to non-exploratory behavior. We deliberately do not equip neural agents with symbolic search; how well agents cope without built-in symbolic machinery is part of what the benchmark measures.

R2.7: “figure 4, put a) and b) on top, c) and d) below.”

Fixed; panels (a) and (b) are now the top row.

Reviewer 3

R3.1: *“The paper repeatedly refers to RDF 1.2-annotated triples ... In the supplement, the implementation appears to use old-style RDF blank-node reification ... The paper should include a small side-by-side example ...”*

You are right, and we thank you for inspecting the artifacts this closely. The revised paper states the situation explicitly in a new “Reification and RDF 1.2” discussion (Section 2.1): the RDF 1.2 annotation model is the *conceptual* memory model, and the released artifacts *serialize* it using standard RDF reification, for two reasons: (i) RDF 1.2 tooling support is only beginning to appear and has not reached the Python RDF ecosystem on which the implementation builds: rdflib does not parse RDF 1.2 syntax (verified against the current release), while Apache Jena does (the released RDF 1.2 snapshot files validate with Jena riot and can be queried with SPARQL 1.2 triple-term patterns); and (ii) each memory must be an independently addressable occurrence of a base triple, with its own annotation set and identifier, even when the same base triple is stored multiple times; a per-memory reification node plays exactly the role of an RDF 1.2 *reifier*. Listing 1 shows the same memory entry side by side in both forms; the encodings are one-to-one, and interchangeable for everything the paper does: the agents read base triples and annotation values, which are identical in both, and perform no RDF entailment, which is the one place the two would differ. The paper therefore describes the memory in RDF 1.2 terms and treats the reification as an implementation detail, which is also the honest chronology: the implementation predates RDF 1.2, which reached Candidate Recommendation in April 2026. We have also added the two remaining concrete Turtle examples you ask for: a hidden-state fragment at $t = 99$ (Listing 2, Section 3.3, “Hidden State and Dynamics”) and a complete local observation at $t = 0$ (Listing 3, Section 3.4, “Observations as RDF Graph Fragments”), so that all three requested artifacts (hidden-state fragment, observation, and memory entry) now appear in the paper itself. We added serialized Turtle snapshots of agent memories to the released artifacts in both serializations (the reification form that current tooling loads, and the equivalent RDF 1.2 annotation form, produced by a converter that is also included, together with a checker that verifies the two serializations carry identical memory occurrences), so the representation can be inspected without re-running code. The baseline repository is archived at <https://doi.org/10.5281/zenodo.21967778> (v0.1.5), so the snapshots and the raw results are reachable from a single citable record.

R3.2: *“... better position its temporal representation against prior temporal and spatio-temporal RDF work ... temporal validity should be distinguished from memory-management metadata.”*

Section 2.2 now positions the representation against Temporal RDF (Gutierrez et al.), logic-based validity time (Motik), named graphs/quads (Carroll et al.), stRDF/stSPARQL (Koubarakis & Kyzirakos), OWL-Time, and RDF stream processing. Our annotations are memory-management metadata (observation/access bookkeeping), not temporal validity; deciding which stored occurrence to trust is the job of the memory policies. This places the representation closer to provenance-style annotation than to validity-time models.

R3.3: *“The temporal memory operations are under-specified ... how insertion, repeated observation, update, query-time metadata update, eviction, deletion, and outdated or contradictory facts are handled.”*

Section 4.1 now contains a “Memory-operation semantics” block covering many cases explicitly, including the case you highlight: contradictory facts (two locations for one object) coexist in memory and are resolved by the QA policy at query time, so stale facts are superseded functionally rather than deleted. This also explains the occurrence counts visible in Figure 5.

R3.4: “A compact train/test table should specify which environment components are shared and which are changed.”

See Table 1. All components (layout, wall schedules, object inventories and motion rules, episode length, query pool) are identical between train and test. We also included that the same set of five seed is used for both instances. The only difference is the query order.

R3.5: “The paper should add baselines such as a hidden-state oracle, a perfect-memory symbolic agent, an observation-only baseline, or a baseline simply using last-seen facts.”

The new “Calibration Bounds” subsection (Section 5.2), with Table 3, anchors three of your four suggestions; the fourth, last-seen facts, is one of the 27 TKG variants whose spread Section 5.1 reports. Two of the four turn out not to be separate systems, which is itself informative:

- **Hidden-state oracle.** No experiment needed: an agent reading s_t will answer all 100 queries correctly by construction, since correctness is evaluated against that same state. We state the bound in the text and record it as the top row of Table 3, labelled analytic so it is not read as a measured run.
- **Perfect memory.** $K=1024$ is that baseline. An episode emits ~500-600 observation triples, so at $K=1024$ neither symbolic agent evicts anything. The reported TKG variant scores what it scores at $K=512$, and no variant of the 27 moves by more than 0.16 points. The KG agent reaches the condition sooner still: its store is a set, so repeated observations collapse to roughly 300 distinct entries and it never evicts at $K=512$ either. Perfect memory is therefore not an agent to build but a capacity the sweep already passes through, and reaching it changes nothing. That is the substantive finding: at these capacities eviction is not the binding constraint.
- **Last-seen facts.** This is our MRA retrieval with a memory large enough such that nothing is evicted; one of the existing 27 variants. It scores 45.44 on test against 45.64 for the variant we report, so answering with the most recently observed location is within noise of the tuned policies. Same convergence as above: once everything is retained, the retrieval rule stops mattering and what remains is whether the fact was stored at all.
- **Observation-only.** $K=0$, the lower-bound row of Table 3: with no long-term memory an agent answers only when the queried object is in the room it currently occupies, which holds for at most 2 of the 100 queries.

Together these make the absolute scores interpretable. The gap from the best saturated symbolic score (~46) to the analytic upper bound (100) is the irreducible cost of partial observability, since even an agent that retains everything must revisit rooms to refresh facts about moving objects. The gap from $K=0$ to $K=512$ isolates the contribution of long-term memory itself.

R3.6: “...the experiments compare KG versus TKG on the symbolic side, but do not include an analogous ‘temporal informed neural agent’.”

We would note first that the neural baselines are already temporally aware by construction, which the revision now makes explicit where they are introduced: the LSTM encodes order through recurrence, and the Transformer through sinusoidal positional encodings (Section 4.2). Sequence order is therefore available to them, and the KG-versus-TKG contrast on the symbolic side is not mirrored by a temporally-blind neural side.

The question is, however, about something more specific than order, namely the absolute arrival time that `:time_added` records, so we ran the control. The revision adds LSTM and Transformer variants whose observation tokens additionally carry an explicit arrival-timestep feature. They

were trained under exactly the protocol used for the other neural agents (all ten capacities, five seeds each, 200 episodes), in two model sizes per architecture (11.9k and 52.1k parameters for the LSTM, 14.1k and 69.2k for the Transformer), which is 200 additional training runs. The larger variant of each architecture is plotted in Figure 3; Table 3 reports the bound over all of them.

Two points on how the control is set up, since both bear on whether it isolates what you intend.

First, it is **size-matched**. Each architecture is trained at two sizes, with and without the feature, under the identical protocol; the temporal variant adds only the timestep embedding (288 parameters at the smaller size, 1,088 at the larger, i.e. about 2%). So the comparison varies the feature, not model capacity.

Second, the result. Across all eight configurations, no neural agent reaches more than 18.6 on train or 14.2 on test at any capacity, and on the held-out environment none gains more than 0.6 points across the whole capacity range. Arrival-time features move agents around inside that band but never out of it, against 44.48 and 45.64 for the TKG agent at $K=512$. The effect is not even reliably positive: averaged over capacities the feature gains about a point on train and gives it back on the held-out environment, in both architectures, with every difference inside the across-seed spread (Section 5.1 gives the four values). We read this as evidence that the symbolic advantage is not access to arrival-time information but the ability to query and maintain those annotations explicitly.

On the two annotations we did **not** turn into input features: `:last_accessed` and `:num_recalled` are not properties of the world, they are functions of the agent’s own retrieval history. Supplying them to a sequence model would require giving it an addressable store that records its own accesses, which is precisely the mechanism under test, so the control would no longer be a control. Arrival time is the one signal of the three that is well-defined for a sequence model without importing the symbolic machinery, which is why it is the one we implemented. This is now also explained in the paper.

R3.7: “The term s_t should be clearly distinguished from the local observation o_t and the agent’s internal memory M_t . The deterministic question-move loop could also be formalized.”

s_t and o_t are defined in Sections 3.3–3.4; the revision additionally defines $M_t = (M_t^{\text{short}}, M_t^{\text{long}})$ (Section 4) and formalizes the within-step order as an enumerated loop (Section 3.5). Six of the seven events you list are fixed by the benchmark, and Section 3.5 gives them in that order, including that the answer is graded before any movement and that objects move through the wall layout in force at t .

The seventh, the memory update, is internal to the agent, which is why it is not in the loop: fixing when an agent writes to memory would build one design choice into the benchmark. Section 3.5 says so, and records that the agents studied here update on receiving o_{t+1} .

R3.8: “...room coverage is likely strongly influenced by the exploration policy and thus should not be treated as direct evidence for memory quality alone.”

Agreed, and Section 5.3 now says so. The gap between the two symbolic agents, timestep 70 against 74, is not direct evidence about memory quality: they also differ in how their searches are implemented. The paper puts the annotation explanation no more strongly than “consistent with”. We would add one qualification rather than concede the general case: coverage is not independent of memory either. The set of visited rooms a search consults is read from long-term memory, so an agent with no long-term memory cannot tell where it has already been, which is part of why the $K=0$ agents cover so little.

R3.9: Figures 1, 2, 3, 5 readability.

Figure 3 has been redesigned. The submitted version was a single-column plot carrying all eight series in one panel; it is now a full-width, two-panel plot (train / held-out test) with a colorblind-validated palette and distinct markers per agent, and its embedded title has moved into the caption. It also carries two things the submitted version did not: dotted lines bounding the best and worst of the 27 TKG policy variants, and the temporal neural controls added for R3.6. Figure 2 is complemented by a concrete Turtle fragment of the same hidden state at the same timestep (Listing 2). For what is now figure 4, the panels are reordered with added row spacing. Figures 1 and 2 have been recreated with larger fonts (cell labels 12pt to 16pt) from the same recorded episode that underlies Figure 5 and the Memory-State Evolution analysis, so the state figures and the reported node counts now all describe a single trace. Figure 5’s three panels are refitted: they are separated by rules and vertical spacing, the $t=0$ panel is cropped to its content rather than floating in margins sized for the full graph, and their embedded titles have moved into the caption, which now gives the top-to-bottom order. All figures are vector graphics and remain losslessly zoomable in the electronic version.
