Testing
XX(X):1–12

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

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

Taewoon Kim[1], Vincent François-Lavet[2], and Michael Cochez[3]

Abstract

Agents in partially observable environments require persistent memory to integrate observations over time. While KGs (knowledge graphs) provide a natural representation for such evolving state, existing benchmarks are not well suited to isolating long-term memory as the core challenge: they typically do not expose KG-shaped hidden states, do not require persistent tracking of time-varying facts, or do not permit direct inspection of what an agent remembers. We introduce the RoomKG Benchmark, a configurable neurosymbolic benchmark instantiated through a room environment whose hidden state is an RDF KG and whose observations are RDF triples. The agent may extend these observations into a temporal KG when storing them in long-term memory. The benchmark is easily adjustable in terms of grid size, number of rooms, inner walls, and moving objects.

To evaluate this benchmark, we define a lightweight temporal KG memory for agents, based on RDF 1.2 triple annotations with annotation properties (`:time_added`, `:last_accessed`, `:num_recalled`), and evaluate several symbolic baselines that maintain and query this memory under different capacity constraints. Two neural sequence models (Long Short-Term Memory (LSTM) and Transformer) serve as contrasting baselines without explicit KG structure. Agents train on one layout and are evaluated on a held-out layout with the same dynamics but a different query order, exposing train–test generalization gaps in long-term memory rather than in perception. In this setting, temporal annotations lead to more stable performance, and the symbolic TKG (temporal knowledge graph) agent achieves roughly fourfold higher test QA (question-answer) accuracy than the neural baselines under the same benchmark and query conditions. RoomKG Benchmark therefore provides a compact benchmark for studying when explicit long-term memory matters, what kinds of memory representations support it, and why existing benchmarks are insufficient for that purpose. The benchmark definition, agent implementations, and experimental scripts are archived and released as open-source software for reproducible research.

Keywords

neurosymbolic benchmark, partial observability, temporal knowledge graphs, agent memory

Introduction

Agents operating in partially observable environments must infer and maintain structured information about the world as it evolves over time (Kaelbling et al. 1998). Because the world evolves according to structured dynamics, agents must store and update facts observed many steps earlier. Knowledge graphs (KGs) provide a natural representational substrate for such settings: entities, relations, and annotations can express spatial structure, object locations, and temporal metadata in a uniform semantic framework (Ehrlinger and Wöß 2016; Hogan et al. 2021; Battaglia et al. 2018; Zambaldi et al. 2019). However, despite substantial work on relational and symbolic environments, most existing benchmarks are not designed to evaluate long-term memory itself. They do not jointly provide KG-shaped hidden states, temporally evolving facts that must be remembered over many steps, and an evaluation interface that makes the contents and use of memory inspectable (e.g., (Battaglia et al. 2018; Zambaldi et al. 2019; Côté et al. 2018; Hausknecht et al. 2020; Ammanabrolu and Hausknecht 2020; Wayne et al. 2018; Graves et al. 2014, 2016)). Recent discussions of benchmark quality in neuro-symbolic AI further emphasize the need for benchmarks that clearly specify their task, metrics, data generation process, and intended uses (Manhaeve et al. 2025; Bortolotti et al. 2024; Hu et al. 2025).

This paper introduces the RoomKG Benchmark, a deterministic and fully configurable neurosymbolic benchmark designed to fill this gap. The underlying environment's hidden state is an RDF KG whose entities include rooms, objects, walls, and the agent, and whose relations encode spatial adjacency and object locations. At each timestep the agent receives a symbolic observation consisting of RDF triples describing the local room structure and the objects currently present. The benchmark is easily adjustable in terms of grid size, number of rooms, pattern of inner walls, and the number and motion patterns of moving objects. A further component of the benchmark is a query at each step

[1]HumemAI, The Netherlands
[2]Vrije Universiteit Amsterdam, The Netherlands
[3]ELLIS Institute Finland and Åbo Akademi University, Finland

Corresponding author:
Taewoon Kim, HumemAI, The Netherlands.
Email: taewoon@humem.ai

requesting the current location of a named object. Accurate responses require maintaining an internal representation of the world that integrates partial observations over time. The point of the benchmark is not merely to expose structured observations, but to force agents to demonstrate persistent long-term memory under controlled neurosymbolic conditions.

While KGs and temporal KGs are widely used for representing evolving knowledge (Hogan et al. 2021; Dell'Aglio et al. 2017; Moreau and Missier 2012; García-Durán et al. 2018), their application as *agent memory* in interactive environments remains underexplored. Prior work on gridworlds, symbolic environments, and relational reasoning benchmarks typically either (i) does not provide a KG-shaped hidden state, (ii) gives agents unstructured symbolic or text observations without a defined KG memory model, or (iii) uses neural representations that do not expose explicit triples or annotations for inspection (Côté et al. 2018; Hausknecht et al. 2020; Ammanabrolu and Hausknecht 2020; Kim et al. 2023; Battaglia et al. 2018; Zambaldi et al. 2019; Graves et al. 2014, 2016; Wayne et al. 2018; Ha and Schmidhuber 2018). Crucially, these limitations make it hard to answer the benchmark question we care about: when an agent must remember a changing world over long horizons, what memory representation actually helps? As a result, there is limited understanding of how explicit temporal KGs function as internal state representations, how simple symbolic update rules perform relative to neural sequence models, and how temporal annotations influence generalization.

This benchmark gap is especially important because existing work on long-term memory has largely emphasized latent neural representations rather than explicit symbolic ones. Recent work on long-term memory in artificial intelligence has increasingly focused on neural vector-based representations, where stored information is encoded in continuous embeddings rather than explicit symbolic structures (Graves et al. 2014, 2016; Wayne et al. 2018; Ha and Schmidhuber 2018; Pritzel et al. 2017; Blundell et al. 2016). While such approaches are powerful, they often lack interpretability: the contents of memory and the basis of retrieved answers are not directly observable as explicit symbolic facts. One of the motivations for our contribution is to provide a setting where both the hidden state of the world and the agent's long-term memory are represented as explicit KGs. This enables interpretable memory inspection, deterministic updates, and transparent reasoning.

We propose a lightweight temporal KG memory model for agents that extends standard RDF triples with RDF 1.2 triple annotations capturing `:time_added`, `:last_accessed`, and `:num_recalled`. These annotation properties provide a simple but expressive mechanism for tracking the recency and usage of stored facts, enabling memory strategies that remain fully interpretable (Missier et al. 2013). Based on this TKG framework, we implement several symbolic baselines that update and query their memory using deterministic rules, with optional capacity limits and simple eviction heuristics. To provide a contrasting perspective, we include two neural sequence models (LSTM and Transformer) that receive exactly the same symbolic observations but maintain memory as a fixed-length sequence queue buffer without explicit KG structure.

We evaluate all agents on two deterministic layouts: a training environment and a held-out test environment with identical dynamics but a different question order. This setup enables a controlled examination of train-to-test generalization and highlights the importance of structured temporal memory. Across capacities, temporal annotations contribute to more stable behavior, and the symbolic TKG agent exhibits significantly higher test performance than the neural baselines under the same benchmark and query conditions. The benchmark is therefore intended not only as a new environment, but as a diagnostic instrument for studying long-term memory in neurosymbolic agents. The benchmark, agent implementations, and full reproducibility material are publicly available via Zenodo and GitHub.

Contributions

- We introduce the RoomKG Benchmark, a deterministic and configurable neurosymbolic benchmark whose hidden state and observations are expressed as RDF knowledge graphs.
- We formalize a benchmark task, benchmark statistics, and benchmark metrics for object-location question answering under partial observability.
- We define a lightweight temporal KG memory with RDF 1.2 triple annotations and deterministic update rules.
- We compare symbolic TKG agents with neural sequence baselines under a shared interface and varying memory capacities.
- We release the benchmark artifact through a stable Zenodo record together with the benchmark software, baseline agents, and reproducibility scripts.

Background

RDF, RDF 1.2 Triple Annotations, and Knowledge Graph Representations

The Resource Description Framework (RDF) models facts as triples (s, p, o) with subject, predicate, and object drawn from IRIs or literals. An RDF graph therefore corresponds to a directed, edge-labeled multigraph in which relations capture structured connections between entities. This representation is well suited for spatially structured domains: rooms, objects, and adjacency relations can be expressed directly as triples without task-specific encodings.

RDF 1.2 extends RDF with triple terms, reifiers, and annotation syntax for *statements about statements* (Kellogg et al. 2026; Kellogg and Tomaszuk 2026). In concrete syntax, a triple such as $\ll s\ p\ o \gg$ can be annotated with metadata such as `:time_added` or `:last_accessed`, giving a concise notation for temporal or provenance-style annotations on a base fact. For agent memory, this provides a lightweight and standard-compatible way to associate temporal metadata with observed facts.

Temporal Knowledge Graphs

Temporal knowledge graphs (TKGs) represent facts whose validity or relevance changes over time. Common

approaches include interval-based, event-based, and update-based models, all of which associate each triple with temporal metadata reflecting when it was introduced or last observed. RDF 1.2 can express such metadata by attaching annotation key–value pairs to annotated triples, enabling fine-grained annotations such as `:time_added`, `:last_accessed`, or `:num_recalled` (Kellogg et al. 2026; Kellogg and Tomaszuk 2026; Moreau and Missier 2012). Temporal KGs have been widely used in the Semantic Web to model evolving data, streaming updates (Dell'Aglio et al. 2017), and temporally indexed knowledge sources.

In interactive settings, temporal annotations allow agents to track the recency and usage of stored information, to maintain evolving internal models, and to reason over past observations. These properties align naturally with the needs of partially observable environments, where an agent's internal state must integrate information gathered over many timesteps.

Symbolic Memory and Semantic State Tracking

Research on long-term memory for artificial agents has increasingly explored neural vector-based representations, where stored information is encoded implicitly in embeddings or hidden states (Graves et al. 2014, 2016; Wayne et al. 2018; Ha and Schmidhuber 2018; Pritzel et al. 2017; Blundell et al. 2016). While effective for high-dimensional signals, such representations are typically opaque: the contents of memory cannot be inspected directly as noted in neural-symbolic systems (Besold et al. 2017), and the basis for retrieval is difficult to trace.

In contrast, symbolic memory systems store explicit facts that can be examined, queried, and updated deterministically. KGs offer a particularly suitable substrate for this purpose, as they represent entities and relations at a semantic level and support principled operations for insertion, deletion, and reannotation. When temporal annotations are incorporated, symbolic memory becomes fully transparent: each fact carries a clear history of when and how it was observed.

Heuristic update strategies such as recency-based selection, frequency-based selection, and simple eviction rules such as first-in, first-out (FIFO), least recently used (LRU), and least frequently used (LFU) can operate directly over annotation values attached to embedded triples. Such heuristics preserve interpretability while providing meaningful signals for deciding which facts to retain when memory capacity is limited.

Semantic Representations for Agents and Interactive Environments

Semantic Web research has long explored structured representations, logical reasoning, and knowledge-driven decision processes. Several interactive or agent-based settings use symbolic descriptions of the world, but the underlying environments are rarely expressed directly as KGs, and the agent's internal memory is typically not a temporal KG. Existing gridworld-style or relational reasoning benchmarks often provide symbolic observations without specifying how those observations should be integrated into a structured, evolving memory.

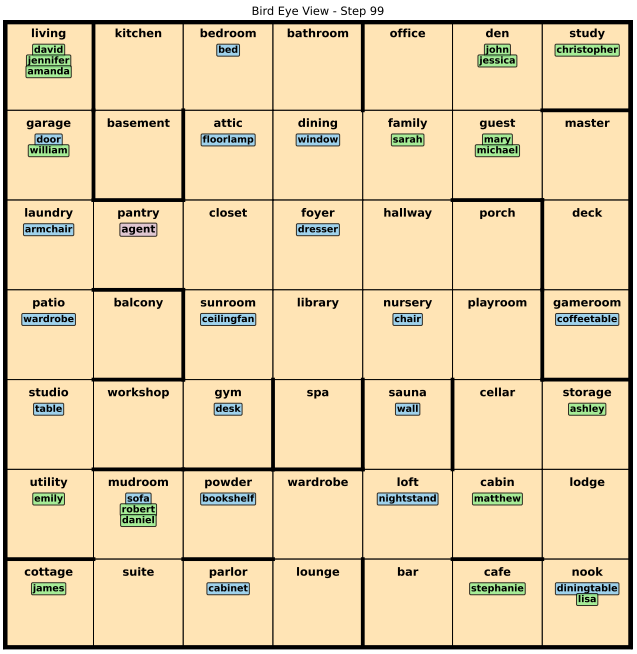

Figure 1. Bird's-eye schematic of the hidden state at $t = 99$ ($s_{t=99}$), showing spatial layout and entity placement. This view is only a schematic for intuition: the actual environment state and agent-facing world are represented in the RDF knowledge-graph world (Figure 2). The agent does not directly observe s_t; its observation o_t is the induced RDF subgraph of the current room and visible adjacency relations.

As a result, there is limited empirical understanding of how temporal KGs function as internal state representations for agents, how simple annotation-driven heuristics compare to unstructured sequence-based memory, and how temporal metadata affects generalization in deterministic but partially observable domains. The benchmark and memory model introduced in this paper address these gaps by providing a setting in which both the hidden state of the world and the agent's memory are explicit, inspectable knowledge graphs represented in standard RDF and RDF 1.2 annotation syntax.

RoomKG Benchmark

The RoomKG Benchmark (Figures 1 and 2) is instantiated through a configurable $g \times g$ grid world populated with named rooms, static and moving objects, and periodically changing inner walls. All benchmark components, namely hidden dynamics, observations, and the QA loop, are defined in RDF triples, making the benchmark a minimal but fully KG-shaped testbed for studying temporal memory.

The benchmark deliberately couples navigation and question answering. Because objects move and visibility is local, correct answers require exploration, accumulation of partial observations, and persistent memory of object trajectories. This design prevents trivial solutions: without navigation the task reduces to static KG lookup, and without queries exploration has no semantic purpose. Their combination creates a controlled setting in which temporal memory is both necessary and measurable.

The benchmark generalizes the earlier setting introduced by Kim et al. (Kim et al. 2023): instead of a single fixed layout, our benchmark supports arbitrary grid sizes,

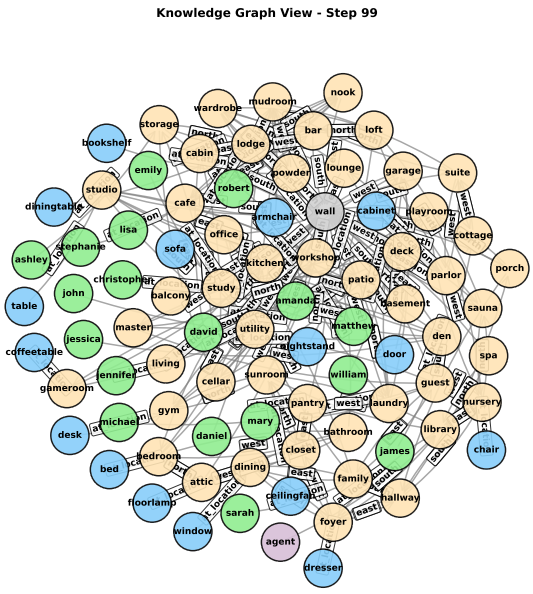

Figure 2. Knowledge-graph view of the same hidden state at $t = 99$ ($s_{t=99}$), expressed as RDF structure. Node colors indicate semantic categories: rooms (yellow), agent (purple), static objects (blue), moving objects (green), and walls (grey).

heterogeneous wall schedules, multiple moving objects with independent motion rules, and exposes the entire hidden state as an RDF KG; when used as long-term memory inside the agent, this can be extended into a temporal KG via annotations.

Benchmark Metadata, Scope, and Intended Use

The public benchmark artifact is archived at Zenodo as a software release with DOI `https://doi.org/10.5281/zenodo.18721817`, publication date 2026-02-21, and version `v3.0.6`. The archived benchmark software is based on the public repository `https://github.com/humemai/room-env`, which is released under the MIT license; the baseline-agent implementations used in this paper are hosted separately at `https://github.com/humemai/agent-room-env-v3` under the Apache License 2.0. Because the benchmark is procedurally specified rather than collected from external annotators, its source data consists of author-designed layouts, periodic wall patterns, object inventories, movement rules, and deterministic query schedules. Maintenance is therefore software-centered: new benchmark versions are created through versioned repository releases and archived snapshots.

In scope, RoomKG Benchmark targets neurosymbolic systems that must integrate symbolic observations over time, maintain explicit or implicit world models, and answer semantic queries under partial observability. The primary task is object-location question answering during navigation. The primary evaluation metric is QA accuracy; the benchmark also reports room coverage, triple coverage, and memory-state visualizations to expose how different memory representations support or hinder the task. We therefore intend the benchmark as a compact testbed for temporal memory, symbolic state tracking, and generalization studies, rather than as a full embodied-AI benchmark covering language, continuous control, or open-ended planning. For the same reason, the present comparison does not include large-language-model-based agents: their performance would conflate bounded memory effects with pretraining, prompt design, context-window management, and optional tool or retrieval configurations, which would blur the paper's focus on controlled long-term memory mechanisms under fixed symbolic observations.

With this scope in place, we next specify the benchmark dynamics and observation model that create these long-term memory demands.

Hidden State and Dynamics

At timestep t, the hidden state s_t consists of the on/off status of all inner walls, the positions of all static and moving objects, and the agent's current room.

Each room is a node, and cardinal directions are modeled as labeled edges (`north`, `east`, `south`, `west`). For a given room, a direction can point either to an adjacent room or to a wall, e.g., `(roomA, north, roomB)` or `(roomA, south, wall)`. This representation keeps blocked and unblocked directions explicit in the graph. Every moving object is equipped with a fixed, ordered preference list drawn from $\{\texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}, \texttt{stay}\}$. At each step it executes the *first feasible* move in its list; if none are feasible, it remains in place. This rule, combined with wall schedules, makes object motion *deterministic*.

The inner walls follow fixed periodic schedules: if inner wall i has a pattern of length $|p_i|$, then the joint wall configuration repeats every $P_w = \mathrm{lcm}(|p_1|, \ldots, |p_W|)$ steps. Moving objects induce an additional period P_m. After an initial transient T_{settle}, the global state obeys

$$s_{t+P} = s_t, \qquad P = \mathrm{lcm}(P_w, P_m),$$

so the benchmark environment is a finite deterministic system with a compact periodic orbit.

Although deterministic, the state space is combinatorially large. With $R = g^2$ rooms, M moving objects, and W inner walls with pattern lengths $|p_1|, \ldots, |p_W|$, a coarse upper bound on the state space is

$$|\mathcal{S}| \;\le\; R^M \prod_{i=1}^{W} |p_i|.$$

For our running configuration (g=7, R=49, M=18, W=36), this exceeds 6.6×10^{33}. Exact planning is infeasible; agents must instead reason over their own growing memory M_t.

The periodic wall schedule and deterministic object motion are intentional design choices: they ensure that the hidden state evolves in a structured but nontrivial way, making it impossible for an agent to answer location queries from the current observation alone. Because objects move and visibility is local, accurate question answering requires integrating partial observations over many steps. Periodicity keeps the benchmark fully deterministic and replayable while still enforcing genuine long-term memory demands.

A detailed description of the periodicity analysis, including closed-form bounds and deterministic replay examples, is provided in the public benchmark artifacts.

Observations as RDF Graph Fragments

At timestep t, the agent receives an observation o_t consisting of the RDF subgraph induced by its current room: the room node for the agent's location, any outgoing direction-labeled edges to adjacent rooms that are not blocked by walls, and object–location triples for all objects currently in that room. This yields 5–6 triples per step. Observations contain no information beyond the agent's immediate neighborhood, so long-horizon queries require integrating partial observations over time.

The benchmark couples these observations with an explicit query stream. Each timestep also issues a *location query* of the form: $(X,$ `:at_location`, $?)$. The agent must identify the current room of the queried object. This requires maintaining a temporal model of object motion as observed indirectly through partial glimpses, echoing aspects of KG-based QA (question-answer) (Yih et al. 2015).

Deterministic Question–Move Loop

At each timestep the agent first answers the query by returning the room of the specified object (reward $+1$ if correct, otherwise 0), and then executes a movement action chosen from $\{$`north`, `east`, `south`, `west`, `stay`$\}$.

Episodes last 100 steps, a length chosen to balance tractability with sufficient temporal depth. The full query sequence for each episode is pre-generated and fixed across all agents, ensuring strict comparability in evaluation.

Configurable Layouts and Test Split

The benchmark is fully parameterized by the grid length g, the number and placement of static and moving objects, the number and periodic patterns of inner walls, and the episode length.

We use two deterministic layouts: a training layout with fixed walls and object preferences, and a held-out test layout with identical dynamics but a different query order. This difference in question order yields a controlled train–test generalization setting in which agents must transfer their learned memory strategy rather than memorize a fixed query sequence.

Memory Interface (Agent-Agnostic)

The benchmark maintains only the hidden state s_t and the symbolic observation o_t emitted at each timestep. It does not prescribe any internal memory structure: agents may keep any private state they choose. The benchmark simply provides RDF observations, receives an action and an answer, and evaluates correctness against s_t.

In our experiments, we compare four agents. The *TKG agent* stores each observed fact as an RDF 1.2-annotated triple with annotation properties (`:time_added`, `:last_accessed`, `:num_recalled`). The *KG agent* stores the same observations as plain RDF triples without annotations. The two *neural baselines* maintain fixed-length buffers of tokenized observations without explicit KG structure; one uses an LSTM encoder and the other uses a Transformer encoder. All agents operate in the same benchmark and query conditions.

Default Configuration

Unless stated otherwise, experiments use a `grid_length` of 7, with 18 static objects, 18 moving objects, 36 inner walls following periodic patterns, an episode length of 100, and a fixed query list shared across all agents. The reward is simply the number of correct queries (maximum 100). All benchmark code, layouts, and deterministic replay scripts are included in the public benchmark artifacts for full reproducibility.

Agents

We evaluate four agents: two *symbolic* agents that store explicit KG structures, and two *neural* agents that store tokenized observation histories. All agents receive the same query sequence and operate in the same benchmark, but their exploration behaviors cause them to generate different observation histories. We evaluate them under memory capacities from 0 to 512 entries, which covers the range where capacity meaningfully constrains what can be stored. A capacity of 0 means that no long-term memory is retained across timesteps, so agents must act only on the current observation and any non-persistent internal computation available within that step. For the TKG agent, the candidate query and exploration policies are most recently added (MRA), most recently used (MRU), and most frequently used (MFU), while the candidate eviction policies are FIFO, LRU, and LFU. Table 1 summarizes the four agents.

Symbolic agents

Both symbolic agents store explicit triples. Their difference is whether they attach temporal annotations.

KG agent (plain RDF triples). Memory is a bounded set of RDF triples (s, p, o). Because these entries carry *no* timestamps or usage counts, the agent has no basis for ordering them. Therefore, when memory is full, the only principled choice is to evict a *uniformly random* triple.

QA rule. Answers are obtained by plain SPARQL pattern matching:

```
SELECT ?o WHERE { <s> <r> ?o . }
ORDER BY RAND() LIMIT 1
```

If multiple candidates match, tie-breaking is uniform because the agent lacks temporal metadata.

Exploration. The current map is reconstructed from the triples stored so far. The agent performs BFS (breadth-first search) to the nearest unvisited room. BFS is used because the memory itself is a graph and BFS is the simplest deterministic traversal aligned with that structure.

TKG agent (RDF 1.2 annotations). Each observed triple (s, r, o) is stored together with annotation assertions over its annotated form, written here using the concise notation $\ll s\ r\ o \gg$ with:

`:time_added`, `:last_accessed`, `:num_recalled`.

This creates a temporal knowledge graph M_t.

Agent	Memory repr.	Annotations	QA rule	Eviction
Symbolic–KG	RDF triples	No	SPARQL	Random
Symbolic–TKG	RDF 1.2 annotations	Yes	MRA/MRU/MFU	FIFO/LRU/LFU
Neural–LSTM	Obs. history	No	Prediction head	FIFO
Neural–Transformer	Obs. history	No	Prediction head	FIFO

Table 1. Agent comparison. MRA = most recently added, MRU = most recently used, MFU = most frequently used.

QA rules. Each query $(s_q, r_q, ?)$ is answered by selecting the object o whose annotation value is maximal:

$$\text{MRA} : \arg\max \texttt{:time\_added},$$
$$\text{MRU} : \arg\max \texttt{:last\_accessed},$$
$$\text{MFU} : \arg\max \texttt{:num\_recalled}.$$

A concrete SPARQL 1.2 realization of the MRU rule is:

```
SELECT ?o WHERE {
    << <s> <r> ?o >> :last_accessed ?t .
}
ORDER BY DESC(?t)
LIMIT 1
```

Eviction. Using annotations permits structured removal:

$$\text{FIFO} : \arg\min \texttt{:time\_added},$$
$$\text{LRU} : \arg\min \texttt{:last\_accessed},$$
$$\text{LFU} : \arg\min \texttt{:num\_recalled}.$$

Exploration. The agent ranks frontier rooms using MRA/MRU/MFU on edge triples and performs BFS to the top-ranked frontier.

Thus, the TKG agent is entirely deterministic and fully traceable: every decision (answer, movement, eviction) is reproducible from M_t.

Neural agents

The neural agents store a fixed-length *tokenized observation history* without explicit graph structure. They differ only in the sequence encoder used.

Joint decision structure. Unlike the symbolic agents, which decompose question answering and exploration into two independent rules, the neural agents produce a *single joint operation* that simultaneously selects the answer and the movement. This results in a large option set (49 possible room answers $\times$ 5 movement choices) from which the prediction head chooses one element. Because answers referring to rooms that have not yet been observed are invalid, the option set is dynamically *masked* at each timestep so that only admissible answers remain. Consequently, the neural agents must learn end-to-end how to resolve queries and how to navigate, without access to explicit graph structure, annotations, or symbolic update rules. The neural agents learn this joint policy using a standard reinforcement-learning objective implemented via a Deep Q-Network (DQN) (Mnih et al. 2015).

Tokenization. Each triple (s, p, o) is converted to an embedding vector

$$\mathbf{e}_i = W_{\text{emb}} \left[\text{emb}(s) \parallel \text{emb}(p) \parallel \text{emb}(o) \right],$$

and the last L such tokens form the memory buffer. When full, the oldest token is removed (FIFO), reflecting the queue structure of a fixed-length sequence buffer.

LSTM neural agent

The sequence $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_L]$ is encoded by:

$$\mathbf{H} = \text{LSTM}(\mathbf{E}).$$

Because LSTMs process data sequentially, temporal order is included implicitly (Hochreiter and Schmidhuber 1997).

Question-conditioned pooling. Given a question triple $(s_q, r_q, ?)$, embed it as $\mathbf{q}$ and compute attention weights:

$$\alpha_i = \frac{\exp(\mathbf{q}^\top \mathbf{h}_i)}{\sum_j \exp(\mathbf{q}^\top \mathbf{h}_j)}, \qquad \mathbf{c} = \sum_i \alpha_i \mathbf{h}_i.$$

A feedforward prediction head maps $\mathbf{c}$ to the 245 possible operation options (answer $\times$ movement).

Transformer neural agent

Here, $\mathbf{E}$ is fed to a Transformer encoder:

$$\mathbf{H} = \text{Transformer}(\mathbf{E} + \text{PE}),$$

where PE are sinusoidal positional encodings that supply the temporal order (Vaswani et al. 2017), enabling long-range temporal reasoning. The same question-conditioned pooling and prediction head are used as in the LSTM agent.

Interpretability. Neural agents have no explicit triples, annotations, or traceable update rules, similar to episodic-control agents (Pritzel et al. 2017; Blundell et al. 2016). Their internal state is an opaque vector, and only soft attention weights give partial insight into which past observations were influential. Symbolic agents, in contrast, expose every stored fact and every eviction decision.

Experiments

We evaluate all agents in the deterministic RoomKG Benchmark under varying long-term memory capacities, from 0 to 512. Each configuration is trained and tested on 5 random seeds, and we report the mean and standard deviation. All raw results, hardware specifications, and hyperparameters are provided in the public benchmark artifacts. In particular, a long-term memory capacity of 0 denotes the no-memory setting: agents do not retain any long-term memory across timesteps.

The primary task is object-location question answering under partial observability. We report two quantitative views of performance: QA accuracy across memory capacities and coverage over time, measured in terms of visited rooms and stored triples. In addition, we provide a qualitative visualization of memory-state evolution for the TKG agent to illustrate how temporal symbolic memory develops over an episode.

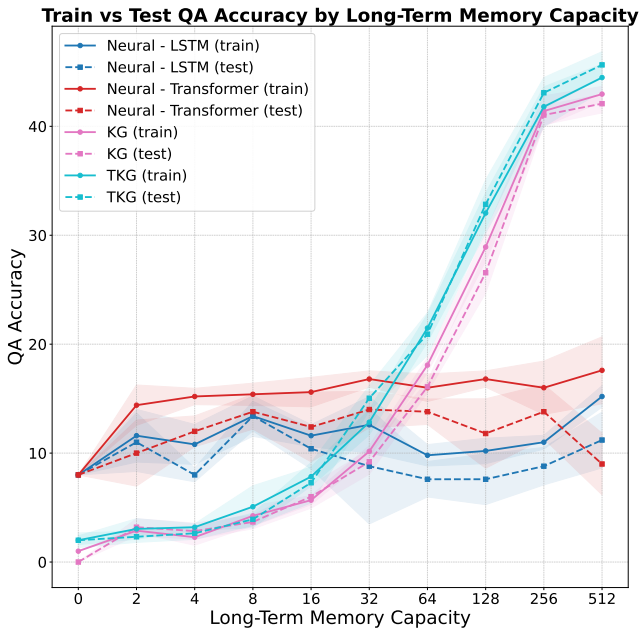

Figure 3. Train–test QA accuracy for all agents across long-term memory capacities. Mean and standard deviation are computed across 5 seeds. Raw values can be found in the public benchmark artifacts.

QA Accuracy Across Long-Term Memory Capacities

Figure 3 shows the train and test QA accuracy for all agents. Neural agents learn exploration and question answering jointly under a single end-to-end policy, yielding a large discrete action space and difficult credit assignment. As memory capacity increases, these agents also receive longer observation histories, making the underlying learning problem more computationally demanding; in principle, higher-capacity models may require substantially more training to benefit from the additional information. In our experiments, however, all memory-capacity settings were trained for the same number of episodes (200) to ensure a fair comparison, which likely contributes to the minimal performance gains observed in the higher-capacity regime.

Across almost all capacities, the Transformer agent slightly outperforms the LSTM agent, consistent with empirical trends in sequence modeling. However, neither architecture meaningfully exploits larger memory, and both exhibit substantial train–test gaps. In low-memory regimes (0–16), neural agents outperform symbolic ones—but this reflects random trial-and-error rather than informed reasoning. At capacity 0, where agents have no long-term memory at all, performance is predictably poor across the board: the agents often guess object locations without provenance and only gradually reinforce correct answers through reward.

Symbolic agents begin to outperform neural agents at capacity 32. Their accuracy increases with memory because operations such as BFS and graph queries add computation but not statistical difficulty. Neural agents, by contrast, must learn these structures from data and therefore require much more training and capacity to benefit from larger memory.

The TKG agent achieves better QA accuracy in most memory capacities. The TKG agent family consists of 27 variants, arising from all combinations of three annotation-based QA rules (MRA, MRU, MFU), three exploration priorities based on the same annotation properties, and three eviction heuristics (FIFO, LRU, LFU); these choices are independent because each stored fact carries the annotation properties `:time_added`, `:last_accessed`, and `:num_recalled`. We report the best-performing variant, selected by training accuracy and evaluated identically at test time. All 27 train/test combinations for every memory capacity are provided in the public benchmark artifacts. At capacity 512, the training-selected variant uses MRU for QA, MRU for exploration, and LFU for eviction. This suggests that recency is the most useful signal both for answer selection and for frontier prioritization, while LFU eviction helps preserve repeatedly useful facts under a fixed memory budget.

This setup favors the plain KG agent. Long-term memory capacity is counted in stored entries, so at the same nominal capacity the KG agent can retain more main triples than the TKG agent. By contrast, the TKG agent uses each entry for an annotated triple, combining the base fact with temporal metadata. Even with this storage disadvantage, the TKG agent performs better, indicating that temporal annotations make the memory state more informative and expressive than a larger store of unannotated triples.

At capacity 512, this MRU/MRU/LFU TKG variant reaches a QA accuracy of 44.48 on train and 45.64 on test, whereas the best neural agent (LSTM) reaches 11.2, a four-fold difference on test. These results highlight that explicit symbolic memory is considerably more effective for long-horizon, partially observable semantic reasoning. Statistical variation is shown as shaded regions.

Coverage Metrics: Room and Triple Coverage

Figure 4 shows, for the long-term memory capacity of 512, the coverage metrics of the four agents: KG, TKG, LSTM, and Transformer. Each subplot includes the cumulative number of unique rooms visited and the number of unique (s, p, o) triples stored in memory at each timestep. Raw values are provided in the public benchmark artifacts.

Symbolic agents. Both symbolic agents achieve full coverage of all 49 rooms in the benchmark. Moreover, the TKG agent reaches full room coverage slightly earlier (timestep 70) than the KG agent (timestep 74). The temporal annotations in the TKG memory allow it to track time-varying wall configurations more accurately, yielding a more up-to-date inferred map of the benchmark environment and enabling more efficient exploration. Triple-coverage results follow the same pattern: symbolic agents accumulate nearly complete internal maps because they continue to explore until all reachable structure has been observed.

Neural agents. In contrast, the LSTM and Transformer agents stop increasing room coverage long before the end of the episode (around step 50 for the Transformer and step 30 for the LSTM). A plausible explanation is that neural policies must couple exploration and answering within a single high-cardinality action space. Once training converges to a locally rewarding but suboptimal behavior, such as repeatedly answering queries with moderate reward or oscillating among familiar states, the agents no longer

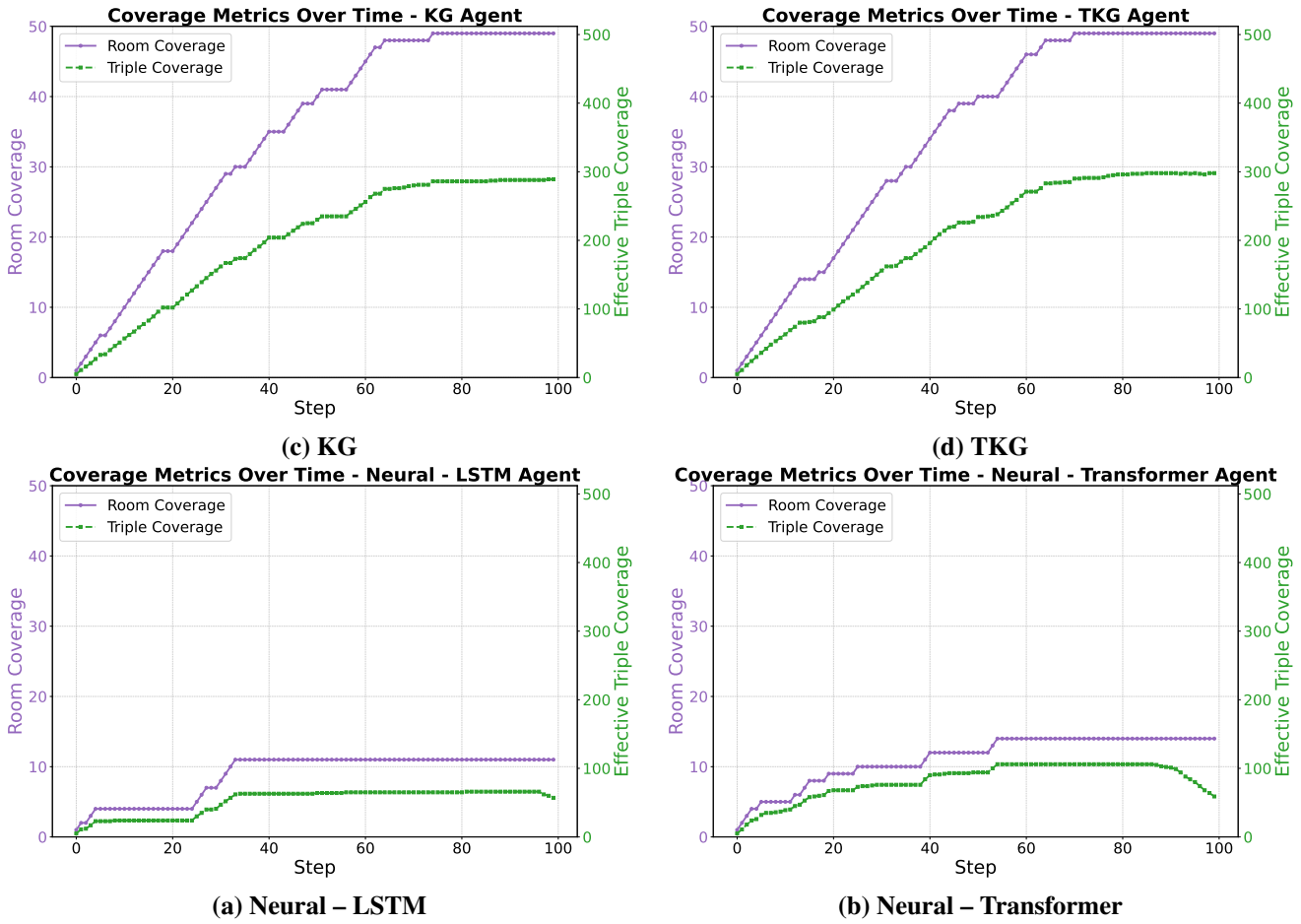

Figure 4. Coverage metrics for the four agents, at the long-term memory capacity of 512. Neural agents halt exploration early, while symbolic agents visit all 49 rooms and accumulate nearly all unique triples. Raw values can be found in the public benchmark artifacts.

discover new rooms. Because the reward signal is sparse and not explicitly tied to exploration, the policies collapse into low-entropy, non-exploratory behaviors. These plateauing exploration patterns also explain the limited triple coverage: neural agents simply encounter fewer states from which to build internal structure.

Memory-State Evolution (TKG)

To illustrate the expressive advantage of temporal symbolic memory, Figure 5 shows the TKG agent's long-term memory at timesteps $t = 0$, $t = 50$, and $t = 99$, using capacity 512. Each edge label includes the number of associated memories when more than one annotated triple exists. Node colors correspond to semantic categories (rooms, static objects, moving objects, walls, and agent).

The graphs show the internal map becoming increasingly complete over time. Static objects and room nodes accumulate in a way that reflects what the agent has actually observed by each timestep. At $t = 0$, the visualization contains only the current short-term observation: four room nodes (yellow), one wall node (grey), and one agent node (purple), with no stored object nodes yet. By $t = 50$, after both short-term and long-term memories have accumulated, the graph contains 43 room nodes, 14 static object nodes (blue), 5 moving object nodes (green), together with the wall and agent nodes. By $t = 99$, the graph contains all 49 room nodes, 16 static object nodes, and 6 moving object nodes, again together with the wall and agent nodes. This final snapshot shows that the agent has visited all rooms and has retained all room nodes in memory, while still storing fewer moving than static objects because moving objects are harder to observe consistently over time.

Full memory evolution images for all agents (from $t = 0$ to $t = 99$) are available in the public benchmark artifacts.

Related Work

Temporal and hyper-relational knowledge graphs. The Semantic Web stack provides a mature basis for representing structured data as RDF graphs and querying them with SPARQL (Seaborne and Harris 2013). RDF 1.2 and SPARQL 1.2 extend this model with triple terms, reifiers, and annotation syntax for *statements about statements* (Kellogg et al. 2026; Kellogg and Tomaszuk 2026; Hartig et al. 2026). Temporal and hyper-relational knowledge graphs build on these ideas by attaching time and other annotations to edges and have been extensively studied for tasks such as link prediction and event forecasting (e.g., (García-Durán et al. 2018)). In contrast, we use a temporal KG not primarily as a prediction target but as the *internal state* of an agent: every observation becomes an RDF 1.2-annotated statement with temporal annotations, and downstream decisions (answering

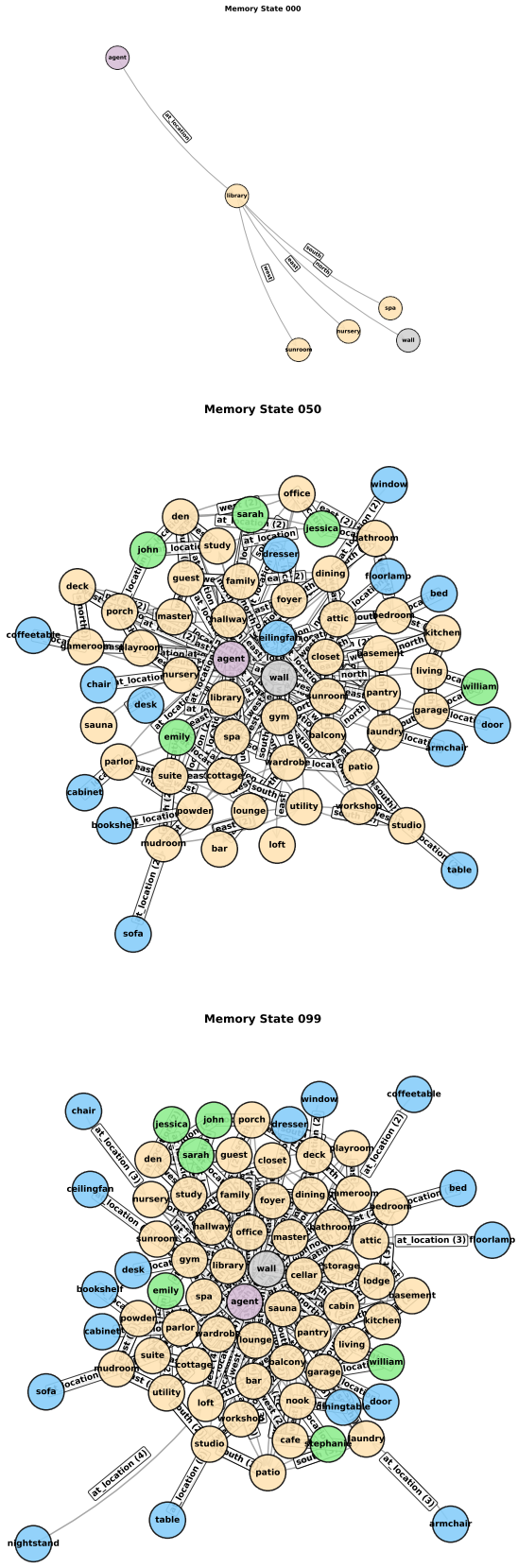

Figure 5. Memory state of the TKG agent at $t = 0$, $t = 50$, and $t = 99$ (capacity 512). Counts in parentheses (e.g., "`at_location (3)`") indicate how many TKG memories share the same main triple but differ in their temporal annotation values.

and exploration) are implemented as queries and updates over this KG.

Memory under partial observability. In partially observable environments, agents must aggregate information over time. A large body of work uses neural memories to maintain an internal state proxy, including recurrent architectures, memory-augmented networks such as Neural Turing Machines and Differentiable Neural Computers (Graves et al. 2014, 2016), and long-context models with compression or retrieval (e.g., MERLIN and related world-model agents (Wayne et al. 2018; Ha and Schmidhuber 2018)). These approaches typically store history in dense vectors or key–value tensors, which makes it difficult to inspect what exactly is remembered at a given time. Our work instead treats the agent's long-term memory as an explicit temporal knowledge graph in RDF 1.2 annotation syntax: each stored fact is a verifiable triple with annotations, and the effect of capacity limits or update rules on question answering can be inspected directly at the graph level. Alongside this, we include neural baselines that operate on observation histories without explicit KG structure, highlighting the behavioural differences between latent and symbolic memories (Kimura et al. 2021). What is missing from this literature is a benchmark expressly constructed to make long-term memory failures visible, measurable, and comparable across symbolic and neural representations.This same issue appears when one looks at interactive environments and KG-based agents.

Graph-structured environments and KG-based agents. Several interactive environments expose graph-structured or relational state (Chaplot et al. 2020), and neural architectures explicitly designed for graph-structured reasoning (Yun et al. 2019). TextWorld and Jericho provide text-based games where an evolving symbolic world model can be extracted and used for control, leading to agents that query or learn over dynamic knowledge graphs (Côté et al. 2018; Hausknecht et al. 2020; Ammanabrolu and Hausknecht 2020; Oh et al. 2017). Agents with multiple KG-based memory stores have also been explored (Kim et al. 2023). Other systems use KGs as world models for planning, question answering, or recommendation (Hogan et al. 2021; Ehrlinger and Wöß 2016; Bordes et al. 2015; Yih et al. 2015). More recent benchmark-oriented work in neuro-symbolic AI has argued for explicit task definitions, benchmark metadata, and diverse evaluation views, as illustrated by benchmarking surveys and suites such as those of Manhaeve et al. (Manhaeve et al. 2025), Bortolotti et al. (Bortolotti et al. 2024), and Hu et al. (Hu et al. 2025). Compared to these settings, our focus is deliberately narrow: we design the Room KG Benchmark where (i) the hidden state and observations are *already* KGs, (ii) the question format is fixed, and (iii) the benchmark parameters (number of rooms, walls, objects, episode length) are easily adjusted. This allows us to isolate the role of temporal KG memory and to compare symbolic and neural baselines under controlled conditions rather than to optimize a full task pipeline. Existing environments and benchmark suites are valuable for their own aims, but they are not sufficient when the goal is to evaluate whether an agent can build, maintain, and use explicit long-term memory about a changing symbolic world.

Semantic Web, agents, and explainability. Within the Semantic Web community, KGs are increasingly used to support decision-making and explainable reasoning in multi-step processes, e.g., planning, data analytics, and interactive decision support. Traceability and provenance are central: KG representations allow one to link conclusions back to the underlying facts (Anyanwu and Sheth 2003). Our design follows this line by making the agent's long-term memory itself a temporal KG; question answering is implemented as SPARQL 1.2 retrieval over that memory, and memory contents can be visualized as graphs over time. This provides a compact testbed for studying how temporal annotations (e.g., recency and recall counts) interact with partial observability and capacity limits, and how symbolic KG memories compare to neural baselines in terms of generalization from training to held-out question orders.

Limitations

RoomKG Benchmark is intentionally narrow. First, its hidden dynamics are deterministic, procedurally specified, and fully symbolic, which makes the benchmark well suited for controlled studies of memory but less representative of noisy embodied settings. Second, the current benchmark task is restricted to object-location question answering; it does not yet cover richer query families such as temporal comparison, counterfactual reasoning, or multi-hop compositional questions. Third, our released benchmark views are derived from synthetic layouts rather than from human-authored or naturally occurring datasets, so benchmark diversity is controlled by parameterization rather than by broad real-world variation. Finally, although we include both symbolic and neural baselines, future work can evaluate a wider range of agents, training regimes, and evaluation protocols than those studied here.

Conclusion

We presented the RoomKG Benchmark in which both the hidden state and the agent's memory are represented as knowledge graphs, enabling a controlled study of temporal memory under partial observability. Using a lightweight RDF 1.2-based memory with temporal annotations, symbolic agents achieved full room coverage and substantially higher QA accuracy than neural sequence baselines under the same benchmark and query conditions. The results highlight the effectiveness and interpretability of temporal knowledge graphs as agent memory, and show why a dedicated benchmark is needed: without a setting that makes long-term memory explicit, structured, and inspectable, it is difficult to distinguish whether an agent succeeds because it remembers well or merely because the task under-tests memory. RoomKG Benchmark provides a compact testbed for future work on semantic representations, temporal update rules, and neuro-symbolic memory models, and is intended to serve as a reference point for evaluating long-term memory in neurosymbolic agents.

References

Ammanabrolu P and Hausknecht M (2020) Graph constrained reinforcement learning for natural language action spaces. In: *International Conference on Learning Representations*. URL https://openreview.net/forum?id=B1x6w0EtwH.

Anyanwu K and Sheth A (2003) ρ-queries: Enabling Querying and Ranking Complex Relationships. In: *Proceedings of the 12th International World Wide Web Conference (WWW '03)*, WWW '03. New York, NY, USA: Association for Computing Machinery. ISBN 1581136803, pp. 690–699. DOI:10.1145/775152.775249. URL https://doi.org/10.1145/775152.775249.

Battaglia PW, Hamrick JB, Bapst V, Sanchez-Gonzalez A, Zambaldi V, Malinowski M, Tacchetti A, Raposo D, Santoro A, Faulkner R, Gulcehre C, Song F, Ballard A, Gilmer J, Dahl G, Vaswani A, Allen K, Nash C, Langston V, Dyer C, Heess N, Wierstra D, Kohli P, Botvinick M, Vinyals O, Li Y and Pascanu R (2018) Relational inductive biases, deep learning, and graph networks. URL https://arxiv.org/abs/1806.01261.

Besold TR, d'Avila Garcez A, Bader S, Bowman H, Domingos P, Hitzler P, Kuehnberger KU, Lamb LC, Lowd D, Lima PMV, de Penning L, Pinkas G, Poon H and Zaverucha G (2017) Neural-symbolic learning and reasoning: A survey and interpretation. URL https://arxiv.org/abs/1711.03902.

Blundell C, Uria B, Pritzel A, Li Y, Ruderman A, Leibo JZ, Rae J, Wierstra D and Hassabis D (2016) Model-free episodic control. URL https://arxiv.org/abs/1606.04460.

Bordes A, Usunier N, Chopra S and Weston J (2015) Large-scale simple question answering with memory networks. URL https://arxiv.org/abs/1506.02075.

Bortolotti S, Marconato E, Carraro T, Morettin P, van Krieken E, Vergari A, Teso S and Passerini A (2024) A neuro-symbolic benchmark suite for concept quality and reasoning shortcuts. In: *Proceedings of the 38th International Conference on Neural Information Processing Systems*. pp. 115861–115905. URL https://dl.acm.org/doi/10.5555/3737916.3741595.

Chaplot DS, Salakhutdinov R, Gupta A and Gupta S (2020) Neural topological SLAM for visual navigation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 12875–12884. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Chaplot_Neural_Topological_SLAM_for_Visual_Navigation_CVPR_2020_paper.html.

Côté MA, Ákos Kádár, Yuan X, Kybartas B, Barnes T, Fine E, Moore J, Tao RY, Hausknecht M, Asri LE, Adada M, Tay W and Trischler A (2018) Textworld: A learning environment for text-based games. In: *Proceedings of the Workshop on Computer Games*. pp. 41–75. URL https://arxiv.org/abs/1806.11532. Presented at CGW@IJCAI 2018, Stockholm.

Dell'Aglio D, Della Valle E, van Harmelen F and Bernstein A (2017) Stream reasoning: A survey and outlook—a summary of ten years of research and a vision for the next decade. *Data Science* 1(1-2): 59–83. DOI:10.3233/DS-170006.

Ehrlinger L and Wöß W (2016) Towards a definition of knowledge graphs.

García-Durán A, Dumančić S and Niepert M (2018) Learning sequence encoders for temporal knowledge graph completion. In: *Proceedings of the 2018 Conference on Empirical Methods*

in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics, pp. 4816–4821. DOI: 10.18653/v1/D18-1516. URL https://aclanthology.org/D18-1516/.

Graves A, Wayne G and Danihelka I (2014) Neural turing machines. Technical report, arXiv preprint arXiv:1410.5401.

Graves A, Wayne G, Reynolds M, Harley T, Danihelka I, Grabska-Barwińska A, Colmenarejo SG, Grefenstette E, Ramalho T, Dyer C et al. (2016) Hybrid computing using a neural network with dynamic external memory. *Nature* 538: 471–476.

Ha D and Schmidhuber J (2018) World models. *CoRR* abs/1803.10122. URL http://arxiv.org/abs/1803.10122.

Hartig O, Seaborne A, Taelman R, Williams G and Tanon TP (2026) SPARQL 1.2 query language. W3C working draft, W3C. Https://www.w3.org/TR/2026/WD-sparql12-query-20260323/.

Hausknecht M, Ammanabrolu P, Côté MA and Yuan X (2020) Interactive fiction games: A colossal adventure. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34. pp. 7903–7910. DOI:10.1609/aaai.v34i05.6297. URL https://doi.org/10.1609/aaai.v34i05.6297.

Hochreiter S and Schmidhuber J (1997) Long short-term memory. *Neural Computation* 9(8): 1735–1780. DOI:10.1162/neco.1997.9.8.1735.

Hogan A, Blomqvist E, Cochez M, D'amato C, Melo GD, Gutierrez C, Kirrane S, Gayo JEL, Navigli R, Neumaier S, Ngomo ACN, Polleres A, Rashid SM, Rula A, Schmelzeisen L, Sequeda J, Staab S and Zimmermann A (2021) Knowledge graphs. *ACM Computing Surveys* 54(4): 1–37. DOI:10.1145/3447772. URL http://dx.doi.org/10.1145/3447772.

Hu M, Chen T, Zou Y, Lei Y, Chen Q, Li M, Mu Y, Zhang H, Shao W and Luo P (2025) Text2world: Benchmarking large language models for symbolic world model generation. In: *Findings of the Association for Computational Linguistics: ACL 2025*. Vienna, Austria: Association for Computational Linguistics, pp. 26043–26066. DOI:10.18653/v1/2025.findings-acl.1337. URL https://aclanthology.org/2025.findings-acl.1337/.

Kaelbling LP, Littman ML and Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1): 99–134. DOI: https://doi.org/10.1016/S0004-3702(98)00023-X. URL https://www.sciencedirect.com/science/article/pii/S000437029800023X.

Kellogg G, Hartig O, Champin PA and Seaborne A (2026) RDF 1.2 concepts and abstract data model. W3C candidate recommendation snapshot, W3C. Https://www.w3.org/TR/rdf12-concepts/.

Kellogg G and Tomaszuk D (2026) RDF 1.2 turtle. W3C working draft, W3C. Https://www.w3.org/TR/2026/WD-rdf12-turtle-20260407/.

Kim T, Cochez M, François-Lavet V, Neerincx M and Vossen P (2023) A machine with short-term, episodic, and semantic memory systems. In: *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press. ISBN 978-1-57735-880-0. DOI:10.1609/aaai.v37i1.25075. URL https://doi.org/10.1609/aaai.v37i1.25075.

Kimura D, Ono M, Chaudhury S, Kohita R, Wachi A, Agravante DJ, Tatsubori M, Munawar A and Gray A (2021) Neuro-symbolic reinforcement learning with first-order logic. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 3505–3511. DOI:10.18653/v1/2021.emnlp-main.283. URL https://aclanthology.org/2021.emnlp-main.283/.

Manhaeve R, Giannini F, Ali M, Azzolini D, Bizzarri A and Borghesi A (2025) Benchmarking in neuro-symbolic ai. In: *Learning and Reasoning: 4th International Joint Conference on Learning and Reasoning, IJCLR 2024, and 33rd International Conference on Inductive Logic Programming, ILP 2024, Proceedings*. DOI:10.1007/978-3-032-09087-4_17. URL https://doi.org/10.1007/978-3-032-09087-4_17.

Missier P, Belhajjame K and Cheney J (2013) The w3c prov family of specifications for modelling provenance metadata. In: *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13. New York, NY, USA: Association for Computing Machinery. ISBN 9781450315975, p. 773–776. DOI:10.1145/2452376.2452478. URL https://doi.org/10.1145/2452376.2452478.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S and Hassabis D (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533. DOI:10.1038/nature14236. URL https://doi.org/10.1038/nature14236.

Moreau L and Missier P (2012) Prov-dm: The prov data model .

Oh J, Singh S and Lee H (2017) Value prediction network. In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 6120–6130. URL https://dl.acm.org/doi/10.5555/3295222.3295360.

Pritzel A, Uria B, Srinivasan S, Badia AP, Vinyals O, Hassabis D, Wierstra D and Blundell C (2017) Neural episodic control. In: *Proceedings of the 34th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, volume 70. PMLR, pp. 2827–2836. URL https://proceedings.mlr.press/v70/pritzel17a.html.

Seaborne A and Harris S (2013) SPARQL 1.1 query language. W3C recommendation, W3C. Https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L and Polosukhin I (2017) Attention is all you need. In: *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., pp. 5998–6008. URL https://papers.nips.cc/paper/7181-attention-is-all-you-need.

Wayne G, Hung C, Amos D, Mirza M, Ahuja A, Grabska-Barwinska A, Rae JW, Mirowski P, Leibo JZ, Santoro A, Gemici M, Reynolds M, Harley T, Abramson J, Mohamed S, Rezende DJ, Saxton D, Cain A, Hillier C, Silver D, Kavukcuoglu K, Botvinick MM, Hassabis D and Lillicrap TP (2018) Unsupervised predictive memory in a goal-directed

agent. *CoRR* abs/1803.10760. URL http://arxiv.org/abs/1803.10760.

Yih Wt, Chang MW, He X and Gao J (2015) Semantic parsing via staged query graph generation: Question answering with knowledge base. In: Zong C and Strube M (eds.) *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, pp. 1321–1331. DOI:10.3115/v1/P15-1128. URL https://aclanthology.org/P15-1128/.

Yun S, Jeong M, Kim R, Kang J and Kim HJ (2019) Graph transformer networks. In: *Advances in Neural Information Processing Systems 32*. DOI: 10.5555/3454287.3455360. URL https://proceedings.neurips.cc/paper/2019/hash/9d63484abb477c97640154d40595a3bb-Abstract.html.

Zambaldi V, Raposo D, Santoro A, Bapst V, Li Y, Babuschkin I, Tuyls K, Reichert D, Lillicrap T, Lockhart E, Shanahan M, Langston V, Pascanu R, Botvinick M, Vinyals O and Battaglia P (2019) Deep reinforcement learning with relational inductive biases. In: *7th International Conference on Learning Representations (ICLR 2019)*. URL https://openreview.net/forum?id=HkxaFoC9KQ.