Short-Term-to-Long-Term Memory Transfer for Knowledge Graphs under Partial Observability

Taewoon Kim, Vincent François-Lavet, Michael Cochez

Keywords: neuro-symbolic reinforcement learning, partial observability, knowledge-graph memory, long-term memory, memory transfer

Summary

We study short-term-to-long-term memory transfer in reinforcement learning under partial observability with symbolic knowledge-graph memory. Our method treats each observed fact as a keep/drop decision and learns these decisions with a per-item Q-learning design that handles variable-sized short-term memory through shared parameters and matched temporal-difference (TD) updates. On the RoomKG benchmark at a long-term memory capacity of 128, learned transfer decisions outperform symbolic and neural baselines, including symbolic baselines with temporal annotations and history-based RL baselines. Graph neural network (GNN) ablations show that a lightweight local policy with short-term-only input is strongest in this regime. Behavioral analysis shows that the policy keeps navigation- and query-relevant facts while discarding lower-value candidate facts, supporting explicit and interpretable memory decisions.

Contribution(s)

1. We formalize short-term-to-long-term memory transfer as a neuro-symbolic reinforcement learning problem in partially observable knowledge-graph memory, where each observed fact receives an explicit keep/drop decision before long-term insertion.
   Context: The formulation isolates transfer decisions in the RoomKG benchmark while keeping question answering (QA), exploration, and eviction policies fixed to avoid policy-interaction confounds, so observed differences can be attributed to memory transfer.
2. We introduce a per-item Q-learning formulation for variable-cardinality short-term memory, using shared parameters to produce independent keep/drop Q-values as the number of candidate facts changes over time.
   Context: Methodologically, this contributes a practical value-based training recipe for variable-cardinality memory transfer decisions (temporal-difference (TD) updates over matched items when short-term observation counts differ across consecutive steps, with replay-driven stochastic pairing), complementing latent-memory RL approaches such as Deep Recurrent Q-Network (DRQN) and external-memory models (Hausknecht & Stone, 2017; Graves et al., 2014; 2016).
3. We provide controlled empirical and behavioral evidence on the RoomKG benchmark at a long-term memory capacity of 128: learned transfer decisions improve over symbolic and neural baselines, including symbolic baselines with temporal annotations, additional symbolic transfer baselines, and history-based RL baselines from prior RoomKG work (Kim et al., 2026).
   Context: Within the learned family, graph neural network (GNN)-based ablations show the strongest variant in this regime is a lightweight local policy that uses short-term input only, and step-level keep/drop analysis supports interpretable selection of task-relevant facts.

Short-Term-to-Long-Term Memory Transfer for Knowledge Graphs under Partial Observability

Taewoon Kim[1,2], **Vincent François-Lavet**[2], **Michael Cochez**[3]
taewoon@humem.ai, vincent.francoislavet@vu.nl,
michael.cochez@abo.fi

[1]**HumemAI**
[2]**Vrije Universiteit Amsterdam**
[3]**ELLIS Institute Finland & Abo Akademi University**

Abstract

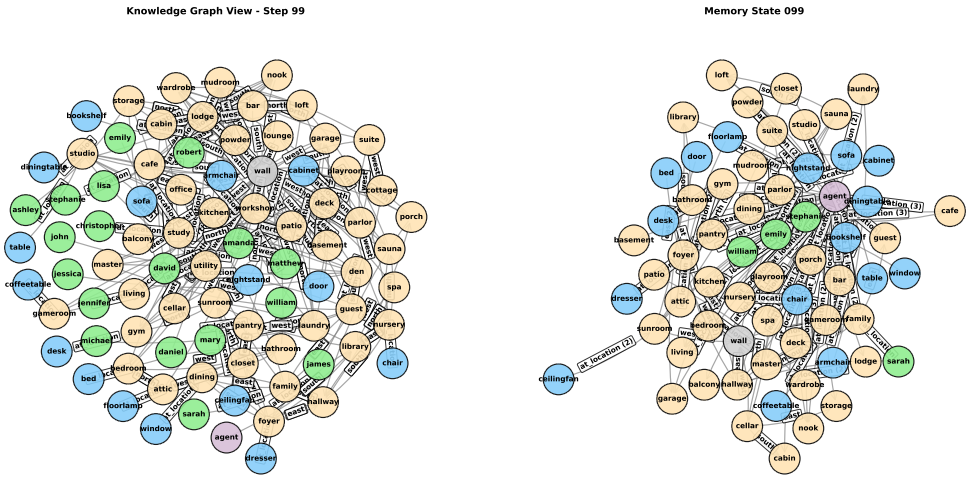
Reinforcement learning under partial observability requires deciding what information to retain, yet most memory-based approaches do not explicitly model short-term-to-long-term transfer of symbolic observations. We study this transfer process in a temporal knowledge-graph memory setting and cast it as a neuro-symbolic value-based decision problem: for each observed triple, the agent chooses whether to keep or drop it before long-term insertion. To handle variable-sized short-term buffers, we use a per-item Q-learning design with shared parameters and a practical temporal-difference update over matched items across consecutive steps. On the RoomKG benchmark at long-term memory capacity 128, learned transfer decisions outperform symbolic and neural baselines, including symbolic baselines with temporal annotations and history-based LSTM/Transformer baselines. Across transfer-policy ablations, a lightweight local short-term-only variant performs best, and step-level behavior shows that the policy keeps navigation- and query-relevant facts while discarding lower-value candidate facts, supporting explicit and interpretable memory decisions under memory constraints.

1 Introduction

Humans routinely act under partial observability, yet still make effective decisions by selectively moving information from immediate experience into longer-term memory. This memory-centric perspective motivates reinforcement learning settings where performance depends not only on action selection, but also on what information is retained.

In reinforcement learning under partial observability, many successful methods use recurrent or external-memory architectures to learn latent states (Hausknecht & Stone, 2017; Graves et al., 2014; 2016; Pritzel et al., 2017). While effective, these approaches usually do not model an explicit neuro-symbolic transfer decision: when an observation arrives, which facts should be transferred to long-term memory and which should be dropped.

We study this question in a temporal knowledge-graph memory setting (Hogan et al., 2021), where observations are explicit Resource Description Framework (RDF) triples (Kellogg et al., 2026a;b) (`head`, `relation`, `tail`) and transfer actions are directly inspectable. Prior work introduced the RoomKG benchmark and established it as a test bed for memory-centric decision making under partial observability (Kim et al., 2026). Building on that setting, we focus on *short-term-to-long-term memory transfer*: deciding keep/drop for each short-term observed triple before long-term insertion.



(a) Hidden state ($s_{t=99}$), not directly observable to the agent.

(b) Our agent's internal symbolic memory state at step 99.

Figure 1: Side-by-side comparison at step 99 in RoomKG: the environment hidden state (left) and our agent's internal symbolic memory state (right). Node colors indicate semantic categories: rooms (yellow), agent (purple), static objects (blue), moving objects (green), and walls (grey). In implementation, each memory is stored as a base triple together with temporal annotations. For readability in the memory-state plot, memories with the same main triple $(\texttt{head}, \texttt{relation}, \texttt{tail})$ are collapsed, and relation labels marked with (N) indicate that N distinct memories share that main triple (differing by temporal annotations).

The key technical challenge is variable cardinality: the number of short-term items changes across steps, so the transfer-action dimension is not fixed. We address this with a shared-parameter per-item Q-learning design and a practical temporal-difference (TD) update over matched items when consecutive short-term set sizes differ.

For controlled evaluation, we keep non-transfer components fixed. We compare against symbolic baselines with temporal annotations, prior RoomKG history-based baselines (LSTM (Hochreiter & Schmidhuber, 1997)/Transformer (Vaswani et al., 2017)), and two additional symbolic baselines (novel-only and random 50% transfer). At a long-term memory capacity of 128, learned transfer decisions outperform this suite. Within the learned family, ablations over local/global and short-term-only/full-context settings show that a lightweight local short-term-only policy is strongest in this regime. We position this paper as a focused mechanism-validation study of explicit memory transfer decisions, not a broad benchmark sweep.

2 Background and Problem Setup

2.1 Environment and Partial Observability Setup

We consider the RoomKG benchmark, a partially observable sequential decision-making environment where the full world state is hidden and the agent receives only local observations at each step (Kim et al., 2026). Let s_t denote the hidden state and o_t the observation at time t. The environment is intentionally designed around explicit graph structure: hidden states and observations are expressed as RDF graphs (Kellogg et al., 2026a;b), and memory augments base triples with temporal annotations, keeping contents explicit and auditable while still supporting RL training.

Concretely, the hidden world graph includes rooms, objects, walls, and the agent, while each o_t is a local induced subgraph around the current room. Since single-step views are incomplete, correct

behavior requires integrating information across time through memory rather than reacting to o_t alone.

Unless stated otherwise, we use the standard RoomKG configuration from prior work: a 7×7 grid (49 rooms), 18 static objects, 18 moving objects, 36 inner walls with periodic schedules, and 100 steps per episode. Following prior work, we use the standard `train`/`test` split: `train` for policy learning/model selection and `test` as a held-out evaluation environment with the same dynamics but different query ordering.

Figures 1a and 1b show, side by side, an example hidden-state snapshot (s_t) and the corresponding internal symbolic memory-state view of our agent at the same step. This gap between hidden state and local observation motivates explicit memory transfer decisions under partial observability.

2.2 Memory Representation and Transfer-Decision Problem

We model interaction as a partially observable control problem over $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \Omega, \gamma \rangle$, where the agent optimizes the discounted return. Here, $\mathcal{S}$ is the hidden-state space, $\mathcal{O}$ the observation space, $\mathcal{A}$ the action space, $P(s_{t+1} \mid s_t, a_t)$ the transition model, $R(s_t, a_t)$ the reward function, $\Omega(o_t \mid s_t)$ the observation model, and $\gamma \in [0, 1)$ the discount factor. The objective is $J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$, where T is the episode horizon and r_t is the reward at step t. The key design question is how to decide what to transfer to long-term memory under partial observability and finite capacity.

We use the standard POMDP tuple $\langle \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \Omega, \gamma \rangle$ to describe the hidden environment dynamics. The transfer controller operates on top of that environment as a structured internal decision process induced by the current short-term memory. If the current short-term set contains $n_t := |\mathcal{M}_t^{\text{short}}|$ items, then the transfer decision space at step t is $\mathcal{A}_t^{\text{tr}} := \{0, 1\}^{n_t}$, the set of binary keep/drop assignments with one entry per short-term item. This makes explicit that the variable-cardinality keep/drop decisions are not a fixed discrete action set of the underlying environment, but an internal fact-selection mechanism conditioned on the current observation set.

In RoomKG, reward is emitted only through question-answering correctness, so memory decisions are learned through their downstream effect on future QA outcomes. Intuitively, keep/drop transfer decisions shape the agent's effective internal state: not the true hidden state s_t, but a belief-like symbolic memory state built from partial observations. Better transfer decisions should produce memory states that more closely track task-relevant aspects of s_t, which in turn improves decision quality and return.

At each step t, the agent receives a short-term set $\mathcal{M}_t^{\text{short}} = \{m_{t,1}, \ldots, m_{t,n_t}\}$, where n_t varies over time, and maintains a capacity-limited long-term memory $\mathcal{M}_t^{\text{long}}$ with $|\mathcal{M}_t^{\text{long}}| \leq K$. The keep/drop policy outputs one binary action per short-term item, $a_{t,i}^{\text{tr}} \in \{0, 1\}$, deciding whether $m_{t,i}$ is kept (transferred/refreshed in long-term memory) or dropped.

We treat the agent's effective internal memory state as $M_t = \mathcal{M}_t^{\text{short}} \cup \mathcal{M}_t^{\text{long}}$, where $\mathcal{M}_t^{\text{short}}$ captures immediate local observations and $\mathcal{M}_t^{\text{long}}$ captures persistent, capacity-limited memory. This internal symbolic state is not the hidden state s_t; rather, it is a belief-like task state built from partial observations and memory updates.

Its evolution is induced by the next observation, transfer decisions, and fixed non-transfer policies. Writing the per-step transfer assignment as $\mathbf{a}_t^{\text{tr}} = (a_{t,1}^{\text{tr}}, \ldots, a_{t,n_t}^{\text{tr}})$, we can view the update abstractly as

$$M_{t+1} = U\big(M_t, o_{t+1}, \mathbf{a}_t^{\text{tr}}\big),$$

where U denotes deterministic short-term refresh, selective insertion into long-term memory, and any resulting capacity handling under the fixed eviction rule.

This yields a variable-cardinality action structure: the number of transfer decisions at time t equals n_t. We therefore study memory transfer as a focused RL problem over symbolic memory items,

while holding non-transfer components fixed so that observed performance differences can be attributed to transfer decisions themselves.

3 Method

3.1 Method Overview and Scope

Our method learns only the transfer keep/drop policy. At each step, the agent receives the current symbolic memory state M_t and outputs one binary decision per short-term item in $\mathcal{M}_t^{\text{short}}$: keep (transfer to long-term memory) or drop. To isolate the effect of transfer decisions, non-transfer components (question answering, exploration, and eviction) are held fixed by symbolic heuristics during this method stage.

The learning signal remains task-level reward from the environment, so the keep/drop policy is optimized through delayed credit assignment: transfer decisions are judged by their downstream contribution to future QA correctness.

A naive alternative would treat transfer decisions as one joint combinatorial action over all short-term items. If there are n_t items at step t, this yields 2^{n_t} actions, which becomes quickly impractical and also assumes a fixed action indexing across time. Here, short-term memory is a set (not a sequence), so its cardinality and ordering are not stable decision coordinates. This motivates a set-compatible, per-item factorization rather than a single monolithic action head.

3.2 Per-Item Q-Value Parameterization

Using $n_t := |\mathcal{M}_t^{\text{short}}|$, we use a shared-parameter function Q_θ that maps the current symbolic state to per-item action values:

$$Q_\theta(M_t) \;\to\; \{\mathbf{q}_{t,i}\}_{i=1}^{n_t}, \quad \mathbf{q}_{t,i} \in \mathbb{R}^2.$$

Each row $\mathbf{q}_{t,i}$ contains Q-values for the two actions $a \in \{0, 1\}$ (keep vs. drop for item i). Shared parameters across items allow one model to score variable-length short-term sets without requiring a fixed action count. Here i is a step-local index over the current short-term set. Because $\mathcal{M}_t^{\text{short}}$ is a set, it has no canonical order; the environment therefore emits its items in a random order at each step. Thus, i does not identify the same memory item across time steps. Equivalently, when we write $Q_\theta(M_t, i, a)$ below, we mean the entry of the i-th output row indexed by action a, that is, $Q_\theta(M_t, i, a) = [\mathbf{q}_{t,i}]_a$.

In implementation, M_t is converted to a graph $G_t = (V_t, E_t, R_t, \mathcal{Q}_t)$, where $\mathcal{Q}_t$ denotes temporal annotation features, and passed through a graph neural network (GNN) encoder. Let $\mathbf{h}_v^{(0)}$ and $\mathbf{r}_p^{(0)}$ be initial entity and relation embeddings. After L message-passing layers:

$$\left\{\mathbf{h}_v^{(L)}\right\}_{v \in V_t},\ \left\{\mathbf{r}_p^{(L)}\right\}_{p \in R_t} = \mathrm{GNN}_\phi\Big(G_t, \{\mathbf{h}_v^{(0)}\}, \{\mathbf{r}_p^{(0)}\}\Big).$$

For each short-term memory item $m_{t,i} = (u_{t,i}, p_{t,i}, v_{t,i}, \xi_{t,i}) \in \mathcal{M}_t^{\text{short}}$, we build the item representation by concatenating the updated endpoint embeddings:

$$\mathbf{z}_{t,i} = \left[\mathbf{h}_{u_{t,i}}^{(L)} \,\|\, \mathbf{h}_{v_{t,i}}^{(L)}\right].$$

Because these node embeddings are produced by message passing on G_t, $\mathbf{z}_{t,i}$ already contains contextual information from neighboring short-term and long-term memory triples included in the encoder input. Thus, the final per-item transfer representation explicitly concatenates only head and tail node embeddings; relation and temporal-annotation effects are incorporated implicitly through the GNN updates that produced those node embeddings. Q-values are then produced by a small MLP head:

$$\mathbf{q}_{t,i} = \mathrm{MLP}_{\text{tr}}(\mathbf{z}_{t,i}) \in \mathbb{R}^2.$$

This design is related in spirit to independent Q-learning (Tan, 1993), because action values are computed per decision unit. However, unlike independent multi-agent formulations, we do not optimize separate MDPs: all per-item decisions are components of one agent solving the same underlying task MDP, and all components share parameters θ.

Action selection is per-item ϵ-greedy. For each i, $a_{t,i} = \arg\max_{a\in\{0,1\}} Q_\theta(M_t, i, a)$ with probability $1-\epsilon$, and a uniform random action otherwise.

3.3 TD Learning with Variable-Cardinality Matching

Transitions are stored in replay as $(M_t, \mathbf{a}_t, r_t, M_{t+1}, d_t)$, where $\mathbf{a}_t$ is the vector of per-item transfer actions and $d_t \in \{0,1\}$ is terminal status. Since $|\mathcal{M}_t^{\text{short}}|$ and $|\mathcal{M}_{t+1}^{\text{short}}|$ can differ, we compute TD updates on matched items only.

Because $\mathcal{M}_t^{\text{short}}$ is a set, there is no canonical item order for cross-step alignment. We therefore use a randomized, order-agnostic matching strategy in practice: observations are shuffled at environment emission time, replay samples transitions stochastically, and TD updates pair items index-wise up to the shared length ℓ_b. This avoids imposing an artificial sequence semantics on inherently set-valued memory.

For sampled transition b, let $\ell_b = \min\big(n_b, n_b'\big)$ with $n_b = |\mathcal{M}_b^{\text{short}}|$ and $n_b' = |\mathcal{M}_{b+1}^{\text{short}}|$. For each matched index $j \in \{1, \ldots, \ell_b\}$, where j denotes only the j-th sampled pair under this stochastic procedure and not the same persistent fact across consecutive steps:

$$y_{b,j} = r_b + \gamma(1-d_b)\,\hat{q}_{b+1,j},$$

where $\hat{q}_{b+1,j} = \max_{a\in\{0,1\}} Q_{\bar{\theta}}(M_{b+1}, j, a)$. The current estimate is $q_{b,j} = Q_\theta(M_b, j, a_{b,j})$.

We minimize the expected squared TD error over replay samples and matched indices:

$$\mathcal{L}(\theta) = \mathbb{E}_{(M_t,\mathbf{a}_t,r_t,M_{t+1},d_t)\sim\mathcal{D},\,\text{match}}\left[\frac{1}{\ell_t}\sum_{j=1}^{\ell_t}(q_{t,j}-y_{t,j})^2\right],$$

with $\ell_t = \min\big(|\mathcal{M}_t^{\text{short}}|, |\mathcal{M}_{t+1}^{\text{short}}|\big)$. This keeps learning stable when short-term cardinality changes across steps while preserving dense supervision from each decision point. The expectation is thus over both sampled replay transitions and the stochastic emission/matching order. Normalizing by ℓ_t keeps the per-transition loss on a comparable scale when different transitions yield different numbers of matched items.

Each training iteration follows: (1) select per-item transfer actions by ϵ-greedy policy, (2) execute environment step and obtain reward, (3) store transition in replay, (4) sample mini-batch after warm start, (5) update online network by TD loss, and (6) periodically hard-update the target network. This is standard off-policy DQN training, specialized to variable-cardinality symbolic transfer decisions (Mnih et al., 2015).

4 Experimental Setup

4.1 Experimental Protocol

We evaluate on the RoomKG benchmark with the standard `train`/`test` split: `train` is used for learning/model selection and `test` is a held-out environment with the same dynamics but different query ordering (Kim et al., 2026). Unless stated otherwise, we use the default configuration from prior work (grid length 7, 49 rooms, 18 static objects, 18 moving objects, 36 periodic inner walls, and 100 steps per episode) and focus on a long-term memory capacity of 128. We choose the RoomKG benchmark and this long-term memory capacity as a focused setup where symbolic keep/drop decisions are directly auditable and memory pressure remains strong enough for short-term-to-long-term transfer to affect downstream QA performance. We view this as a controlled mechanism study rather than a broad robustness claim across environments or capacities.

4.2 Compared Methods and Metrics

We compare learned transfer decisions against a baseline suite consisting of: (i) baselines from prior RoomKG work, including simple symbolic systems with temporal annotations and end-to-end history-based neural baselines (LSTM (Hochreiter & Schmidhuber, 1997) and Transformer (Vaswani et al., 2017)), and (ii) additional symbolic transfer baselines introduced here (Novel-Only and Random-Transfer ($p = 0.5$)). For the learned transfer family, we additionally run an ablation grid over GNN encoder choices and transfer-policy modes. Encoder choices include graph convolutional networks (GCN) (Kipf & Welling, 2017), relational graph convolutional networks (R-GCN) to inject relation-type-specific message passing (Schlichtkrull et al., 2018), and StarE hyper-relational graph neural networks (StarE-GNN) to incorporate annotation features (Galkin et al., 2020). Transfer-policy modes are: *Local-Full* (per-item learned transfer decisions with full memory context), *Local-STM* (per-item learned transfer decisions with short-term-only encoder input), *Global-Full* (one pooled learned decision vector shared across short-term items using full context), and *Global-STM* (pooled shared learned decisions with short-term-only encoder input).

In all comparisons, non-transfer components are kept fixed so performance differences reflect transfer decisions. For the symbolic temporal-annotation baselines, these fixed components use most-recently-used (MRU) question answering, MRU exploration, and least-recently-used (LRU) eviction, computed from temporal annotations attached to memory entries (e.g., `time_added`, `last_accessed`, `num_recalled`); see Appendix 8 for the precise symbolic policy definitions and annotation conventions.

The primary metric is episode-level question-answering score (number of correctly answered queries per episode). We report mean and standard deviation across random seeds for both `train` and `test`, and use the held-out `test` split as the main generalization indicator. We use n=5 seeds as a transparent small-scale evaluation budget and report mean $\pm$ standard deviation. Baselines follow the RoomKG setup and aligned default training settings used in prior work where applicable; within our compute budget we kept training budgets broadly comparable across learned variants rather than performing architecture-specific exhaustive hyperparameter sweeps for every baseline. The implementation used in this paper is open sourced at `https://github.com/humemai/kg-memory-transfer`.

5 Results

5.1 Quantitative Results

Table 1 reports both `train` and `test` performance at a long-term memory capacity of 128 (mean $\pm$ std over n=5 seeds for each setting). The best result is obtained by the learned transfer-policy configuration with GCN encoder and *Local-STM* transfer policy.

Learned transfer decisions outperform end-to-end LSTM/Transformer baselines in both `train` and `test`, and also improve over symbolic baselines with temporal annotations on held-out performance with stable variance. Within learned variants, local per-item transfer decisions outperform pooled global decisions, and short-term-only input outperforms full-context input in this regime.

Quantitatively, the best setting (GCN + *Local-STM*) reaches 38.920 ± 3.090 on `test`, compared with 31.960 ± 1.255 for the strongest symbolic baseline with temporal annotations (*Novel-Only*), a gain of $+6.96$ points. Against end-to-end neural baselines, the margin is larger: $+27.12$ over Transformer and $+31.32$ over LSTM on `test`. This places the main improvement in explicit transfer decisions rather than in replacing symbolic memory with sequence-only models.

Within learned variants, architecture and transfer mode have very different effects. Keeping the same GCN encoder, *Local-STM* outperforms *Local-Full* (38.920 vs. 35.080) and both global variants (19.440 and 9.080), indicating that per-item local transfer is more effective than pooled global decisions in this regime. Across encoder families under *Local-STM*, GCN is markedly stronger than

Table 1: Main `train`/`test` results at long-term memory capacity 128. Scores are episode-level QA accuracy (higher is better), reported as mean ± std over 5 seeds. Variant names: *Always-Transfer* transfers every short-term item, *Novel-Only* transfers only items not already in long-term memory, *Random-Transfer (p=0.5)* transfers each short-term item with probability 0.5, *Local-Full* and *Local-STM* are per-item learned transfer policies, and *Global-Full* and *Global-STM* are pooled shared-decision learned transfer policies. End-to-end DQN rows use *Always-Transfer* memory transfer.

Family	Variant	Train score	Test score
Symbolic temporal-RDF	Novel-Only	32.120 ± 1.204	31.960 ± 1.255
Symbolic temporal-RDF	Always-Transfer	28.600 ± 1.702	29.680 ± 2.317
Symbolic temporal-RDF	Random-Transfer ($p = 0.5$)	21.480 ± 2.282	22.440 ± 3.567
End-to-end DQN	Transformer + Always-Transfer	16.800 ± 0.748	11.800 ± 3.187
End-to-end DQN	LSTM + Always-Transfer	10.200 ± 1.166	7.600 ± 2.332
DQN temporal-RDF	GCN + Local-STM	$\mathbf{37.240 \pm 1.713}$	$\mathbf{38.920 \pm 3.090}$
DQN temporal-RDF	R-GCN + Local-STM	12.920 ± 4.657	13.600 ± 3.132
DQN temporal-RDF	StarE-GNN + Local-STM	13.680 ± 2.847	13.760 ± 3.032
DQN temporal-RDF	GCN + Local-Full	34.800 ± 1.523	35.080 ± 3.037
DQN temporal-RDF	GCN + Global-Full	18.240 ± 8.306	19.440 ± 10.506
DQN temporal-RDF	GCN + Global-STM	8.880 ± 3.000	9.080 ± 3.645

R-GCN and StarE-GNN in this setup, suggesting that extra relational/annotation modeling does not automatically translate into higher QA reward for the present memory-transfer task.

5.2 What the Learned Transfer Policy Does

Unless stated otherwise, the analyses in Sections 5.2 and 5.3 are taken from a single trained **DQN temporal-RDF (GCN + Local-STM)** model (the best configuration in Table 1) on held-out `test` episodes.

To understand what the learned transfer policy actually retains, we analyze per-step keep-or-drop transfer decisions on held-out `test` episodes across 560 transfer decisions (100 steps; about 5–6 candidate short-term memories per step). The policy keeps 224 items and drops 336 items (overall keep rate 0.40). Figure 2 shows the global trajectory of this keep/drop behavior over time.

Three stable patterns emerge. First, the policy strongly preserves the agent's own position signal: `(agent, at_location, room)` is kept 98/100 times. This is not just a preference; in this environment it appears functionally necessary. Without reliable retention of the agent's current location and visited-room trace, the agent policy cannot localize itself or run the internal BFS traversal over the memory-built graph map toward unexplored rooms. Second, the policy retains question-relevant object locations: for objects appearing in the held-out query set, `at_location` facts are kept 58 times and dropped only 2 times. Third, the policy usually drops room-direction map links (for example, `north`/`south`/`east`/`west` links): it keeps 68 and drops 332. This behavior is similar for links to walls and links to other rooms, suggesting that many map-link facts are treated as noisy or less important under changing layouts.

Figure 1b complements this action-level view by showing the resulting internal memory snapshot at the final step. Together, the decision logs and the step-99 memory state indicate a selective memory policy: keep location facts that directly support navigation and question answering, while dropping many room-direction map links (for example, `north`/`south`/`east`/`west` edges).

5.3 Qualitative Step-Level Examples

At step 8, the queried object is `john`, and the policy keeps `(john, at_location, playroom)` (together with `(agent, at_location, playroom)`). In this case it also retains the full locally observed playroom neighborhood, showing that the policy does not simply keep isolated object-location facts: at some steps it preserves the co-observed room context

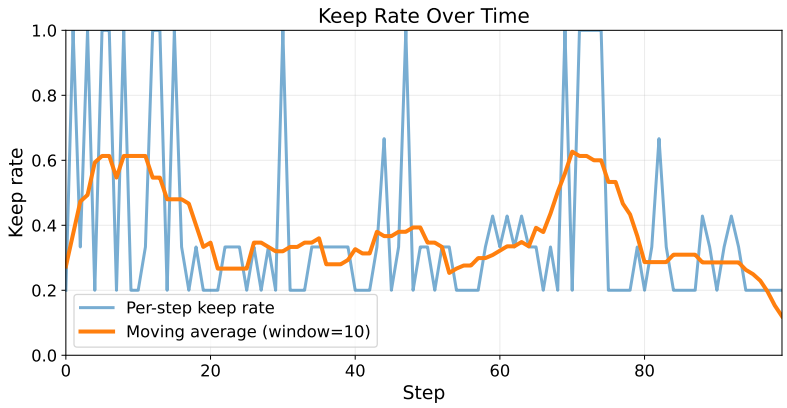


Figure 2: Keep-rate trajectory over held-out `test` steps (higher means more short-term items are transferred to long-term memory). The moving average highlights that transfer is adaptive rather than fixed-rate.

alongside the queried object and the agent's location. At step 26, the policy keeps `(agent, at_location, studio)` and `(table, at_location, studio)` while dropping four directional links from `studio`; this more directly illustrates the selective pattern highlighted by our aggregate analysis, namely preserving self-localization and object-location facts while compressing many map-direction links. At step 62, the queried object is `william`, and the policy keeps `(william, at_location, living)` while dropping four directional links from `living`; this again matches the pattern of preserving likely QA anchors while compressing many map-direction links.

An additional benefit is interpretability: because memory transfers are symbolic (explicit RDF triples with temporal annotations), we can directly inspect what is kept or dropped at each step. This is much harder in latent-memory approaches where internal state updates are distributed in hidden vectors (Hausknecht & Stone, 2017; Graves et al., 2014; 2016).

6 Related Work

Memory in RL under partial observability. Classic approaches to partial observability frequently rely on latent memory, including recurrent value-based agents and neural external-memory architectures (Hausknecht & Stone, 2017; Esslinger et al., 2022; Graves et al., 2014; 2016; Pritzel et al., 2017). These methods are strong at sequence modeling, but memory updates are hard to inspect at fact level. More broadly, batch RL under partial observability also raises stability concerns such as overfitting and asymptotic bias (Francois-Lavet et al., 2019). We target this gap with explicit keep/drop transfer decisions for each short-term symbolic fact.

Recent POMDP studies have also explored compact history summarization and explicit memory decisions, including memory traces as an alternative to fixed windows (Eberhard et al., 2025) and external-memory action formulations where the agent jointly chooses environment and memory-transfer actions (Icarte et al., 2020). Our formulation is closest in spirit to this line, but specializes it to symbolic triple-level updates in temporal KG memory with step-level inspectability.

Symbolic and knowledge-graph memory for RL. The RoomKG benchmark introduced a temporal knowledge-graph memory setting that exposes the memory-management challenge under partial observability (Kim et al., 2026). More broadly, RDF and knowledge-graph formalisms provide structured, auditable representations (Hogan et al., 2021). In our setting, temporal triple annotations keep stored content explicit and inspectable without changing the RL focus of the problem. This symbolic view differs from latent sequence memory in that stored content remains semantically typed and queryable. Earlier work by Kim et al. (2023) modeled a broader agent with short-term,

episodic, and semantic knowledge-graph memory systems; our paper isolates one specific mechanism in that design space, namely short-term-to-long-term transfer under fixed downstream policies. The central policy question is which observed facts to transfer under capacity constraints.

Related neuro-symbolic RL work has combined neural function approximation with symbolic structure or rules, including symbolic priors for RL, symbolic policy discovery, and hierarchical neuro-symbolic designs (d'Avila Garcez et al., 2018; Landajuela et al., 2021; Mitchener et al., 2022). Recent studies also study neuro-symbolic guidance and action-constraint mechanisms for improving sample efficiency in deep RL (Veronese et al., 2026; Han et al., 2026). Our setting is narrower: explicit triple-level transfer decisions for temporal KG memory under partial observability, with step-level inspectability.

Graph encoders for relational state representations. Our experiments use standard graph-neural backbones for relational encoding—GCN, R-GCN, and StarE-GNN (Kipf & Welling, 2017; Schlichtkrull et al., 2018; Galkin et al., 2020). They provide a controlled test bed for transfer-policy learning; our claim is about memory transfer decisions, not a new GNN.

Transfer decisions and model capacity. Most prior neural baselines in this setting either transfer by fixed rules or rely on end-to-end sequence memory. In contrast, we formulate short-term-to-long-term transfer as an RL decision problem over variable-cardinality short-term sets, trained with a practical off-policy temporal-difference (TD) recipe inspired by DQN-style learning (Mnih et al., 2015). The key technical focus is variable-cardinality transfer decisions: the number of short-term items changes across steps, so transfer decisions must be produced and trained robustly without a fixed action dimension. Compared with fixed symbolic transfer heuristics, our approach learns transfer decisions; compared with latent-memory baselines, it retains fact-level interpretability while improving transfer quality under memory limits.

7 Conclusion

We studied short-term-to-long-term memory transfer as a first-class RL decision in a partially observable temporal knowledge-graph setting. Instead of relying on opaque latent-state updates or fixed symbolic transfer rules, our method learns explicit keep/drop actions for each observed triple under memory constraints. Empirically, the learned policy improves QA over symbolic and neural baselines at long-term memory capacity 128, and the action-level analysis explains why: it preserves navigation- and query-critical facts while discarding many lower-value candidate facts. This supports our central claim that interpretable memory transfer decisions are effective in memory-constrained POMDP RL, while also highlighting scope limits: evaluation is limited to RoomKG and one main memory-capacity regime, non-transfer components are fixed for attribution, and temporal-difference (TD) matching for variable-cardinality short-term sets is practical rather than theoretically complete. Future work should test broader environments/capacities, stronger memory baselines, and joint end-to-end training that preserves fact-level interpretability.

Acknowledgments

This research was (partially) funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organization for Scientific Research, https://www.hybrid-intelligence-centre.nl/.

References

Artur d'Avila Garcez, Aimore Resende Riquetti Dutra, and Eduardo Alonso. Towards symbolic reinforcement learning with common sense, 2018. URL https://arxiv.org/abs/1804.08597.

Onno Eberhard, Michael Muehlebach, and Claire Vernade. Partially observable reinforcement learning with memory traces. ICML 2025 Poster (OpenReview), 2025. URL https://openreview.net/forum?id=c4zVRwxjDD.

Kevin Esslinger, Robert Platt, and Christopher Amato. Deep transformer q-networks for partially observable reinforcement learning, 2022. URL https://arxiv.org/abs/2206.01078.

Vincent Francois-Lavet, Guillaume Rabusseau, Joelle Pineau, Damien Ernst, and Raphael Fonteneau. On overfitting and asymptotic bias in batch reinforcement learning with partial observability. *Journal of Artificial Intelligence Research*, 65:1–30, 2019. DOI: 10.1613/jair.1.11478.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 2020.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. URL https://arxiv.org/abs/1410.5401.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John P. Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

Shuai Han, Mehdi Dastani, and Shihan Wang. Neuro-symbolic action masking for deep reinforcement learning, 2026. URL https://arxiv.org/abs/2602.10598.

Olaf Hartig, Andy Seaborne, Ruben Taelman, Gregory Williams, and Thomas Pellissier Tanon. SPARQL 1.2 query language. W3C working draft, W3C, March 2026. https://www.w3.org/TR/sparql12-query/.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps, 2017. URL https://arxiv.org/abs/1507.06527.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. *ACM Computing Surveys*, 54(4):1–37, July 2021. ISSN 1557-7341. DOI: 10.1145/3447772.

Rodrigo Toro Icarte, Richard Valenzano, Toryn Q. Klassen, Phillip Christoffersen, Amir massoud Farahmand, and Sheila A. McIlraith. The act of remembering: a study in partially observable reinforcement learning, 2020. URL https://arxiv.org/abs/2010.01753.

Gregg Kellogg, Olaf Hartig, Pierre-Antoine Champin, and Andy Seaborne. RDF 1.2 concepts and abstract data model. W3C candidate recommendation snapshot, W3C, April 2026a. https://www.w3.org/TR/rdf12-concepts/.

Gregg Kellogg, Andy Seaborne, and Dominik Tomaszuk. RDF 1.2 turtle. W3C working draft, W3C, April 2026b. https://www.w3.org/TR/rdf12-turtle/.

Taewoon Kim, Michael Cochez, Vincent François-Lavet, Mark Neerincx, and Piek Vossen. A machine with short-term, episodic, and semantic memory systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 48–56, 2023. DOI: 10.1609/aaai.v37i1.25075. URL https://ojs.aaai.org/index.php/AAAI/article/view/25075.

Taewoon Kim, Vincent François-Lavet, and Michael Cochez. Temporal knowledge-graph memory in a partially observable environment, 2026. URL https://arxiv.org/abs/2408.05861.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5979–5989. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/landajuela21a.html.

Ludovico Mitchener, David Tuckey, Matthew Crosby, and Alessandra Russo. Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework (extended abstract). In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 5314–5318. International Joint Conferences on Artificial Intelligence Organization, 7 2022. DOI: 10.24963/ijcai.2022/742. Sister Conferences Best Papers.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. DOI: 10.1038/nature14236.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web: ESWC 2018*, pp. 593–607. Springer, 2018.

Ming Tan. Multi-agent reinforcement learning: independent versus cooperative agents. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ICML'93, pp. 330–337, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1558603077.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, pp. 5998–6008, 2017.

Celeste Veronese, Daniele Meli, and Alessandro Farinelli. Sample-efficient neurosymbolic deep reinforcement learning, 2026. URL https://arxiv.org/abs/2601.02850.

Supplementary Materials

The following content was not necessarily subject to peer review.

8 Policy Definitions Used in Experiments

This section defines the symbolic non-transfer policies used while learning and evaluating transfer decisions.

Question-answering policy. Given an RDF/SPARQL-style query triple (Kellogg et al., 2026a; Hartig et al., 2026) with one missing element (e.g., `(john, at_location, ?)` → missing tail), the agent ranks matching candidates in memory using temporal annotations such as recency and frequency and returns the top-ranked answer. `mra` denotes *most-recently-added* (rank by `time_added`), `mru` denotes *most-recently-used* (rank by `last_accessed`), and `mfu` denotes *most-frequently-used* (rank by `num_recalled`).

Exploration policy. The agent uses memory-filtered graph search for navigation. Annotation-based ranking resolves conflicting room/edge memories, then BFS is run on the resulting graph map; the environment action is the first move on the selected path.

Eviction policy. When long-term memory reaches capacity, one memory entry is evicted. `fifo` denotes *first-in-first-out* (evict oldest inserted entry), `lru` denotes *least-recently-used* (evict minimum `last_accessed`), and `lfu` denotes *least-frequently-used* (evict minimum `num_recalled`).

Transfer policy. Main baselines are *Always-Transfer* (transfer every short-term item), *Novel-Only* (transfer only items not already present in long-term memory), and *Random-Transfer (p=0.5)*. Learned variants are *Local-Full*, *Local-STM*, *Global-Full*, and *Global-STM*.

9 Formal Training Algorithm (Transfer Decisions)

Algorithm 1: Per-item DQN training with variable-cardinality matching

1. Initialize online parameters θ, target parameters $\bar{\theta} \leftarrow \theta$, replay buffer $\mathcal{D}$.
2. For each environment step t:
   (a) Build current memory state $M_t = (\mathcal{M}_t^{\text{short}}, \mathcal{M}_t^{\text{long}})$.
   (b) For each short-term item $i \in \{1, \ldots, n_t\}$, choose $a_{t,i}$ by ϵ-greedy from $Q_\theta(M_t, i, \cdot)$.
   (c) Execute transfer decisions, receive reward r_t, next state M_{t+1}, and terminal flag d_t.
   (d) Store transition $(M_t, \mathbf{a}_t, r_t, M_{t+1}, d_t)$ in $\mathcal{D}$.
   (e) If replay warm start is reached, sample minibatch transitions from $\mathcal{D}$.
   (f) For each sampled transition b, set $\ell_b = \min(|\mathcal{M}_b^{\text{short}}|, |\mathcal{M}_{b+1}^{\text{short}}|)$, then sample a stochastic step-local matching and pair matched items index-wise for $j = 1, \ldots, \ell_b$ and compute

$$y_{b,j} = r_b + \gamma(1 - d_b)\,\hat{q}_{b+1,j}, \quad q_{b,j} = Q_\theta(M_b, j, a_{b,j}).$$

   where $\hat{q}_{b+1,j} = \max_{a \in \{0,1\}} Q_{\bar{\theta}}(M_{b+1}, j, a)$.
   (g) Minimize

$$\mathcal{L}(\theta) = \mathbb{E}_{\text{replay, match}}\left[\frac{1}{\ell_b}\sum_{j=1}^{\ell_b}(q_{b,j} - y_{b,j})^2\right].$$

   The expectation is over both replay sampling and the stochastic matching.

Table 2: Core training settings used for the reported long-term memory capacity 128 experiments.

Setting	Value
Episode horizon	99 (100 steps/episode)
Training episodes	200
Training iterations	20,000
Replay buffer size	20,000
Warm start	2,000
Batch size	32
Target update interval	50
ϵ schedule	max 1.0 to min 0.01 over 10,000 iterations
Discount factor γ	0.95
Learning rate	10^{-4}
Double DQN	True
Gradient clipping	True (clip value 10.0)
Seeds	$\{0, 5, 10, 15, 20\}$

Table 3: Model-size settings used for the main long-term memory capacity 128 comparisons.

Component	Value
GCN / R-GCN / StarE embedding dim	16
GCN / R-GCN / StarE layers	2
R-GCN num bases	20
MLP hidden layers	1

(h) Every fixed interval, update target network: $\bar{\theta} \leftarrow \theta$.

10 Hyperparameters and Protocol

For the best-performing GCN + Local-STM model in this paper, the total number of trainable parameters is **8,339**, indicating a relatively compact model.

11 Runtime and Compute

All experiments were run on CPU only: **AMD Ryzen 9 7950X (16-Core Processor)** with **128GB DDR5 RAM**. The best-performing **DQN temporal-RDF GCN + Local-STM** model required about **9 hours** of training on this hardware.

12 Additional Hidden-State and Memory Snapshots

For spatial intuition at the same late-episode point used in the main text, Figure 3 shows a bird's-eye schematic of the hidden state at step $t = 99$. Figure 4 complements this view with two internal memory-state snapshots from held-out test execution (steps 0 and 50).

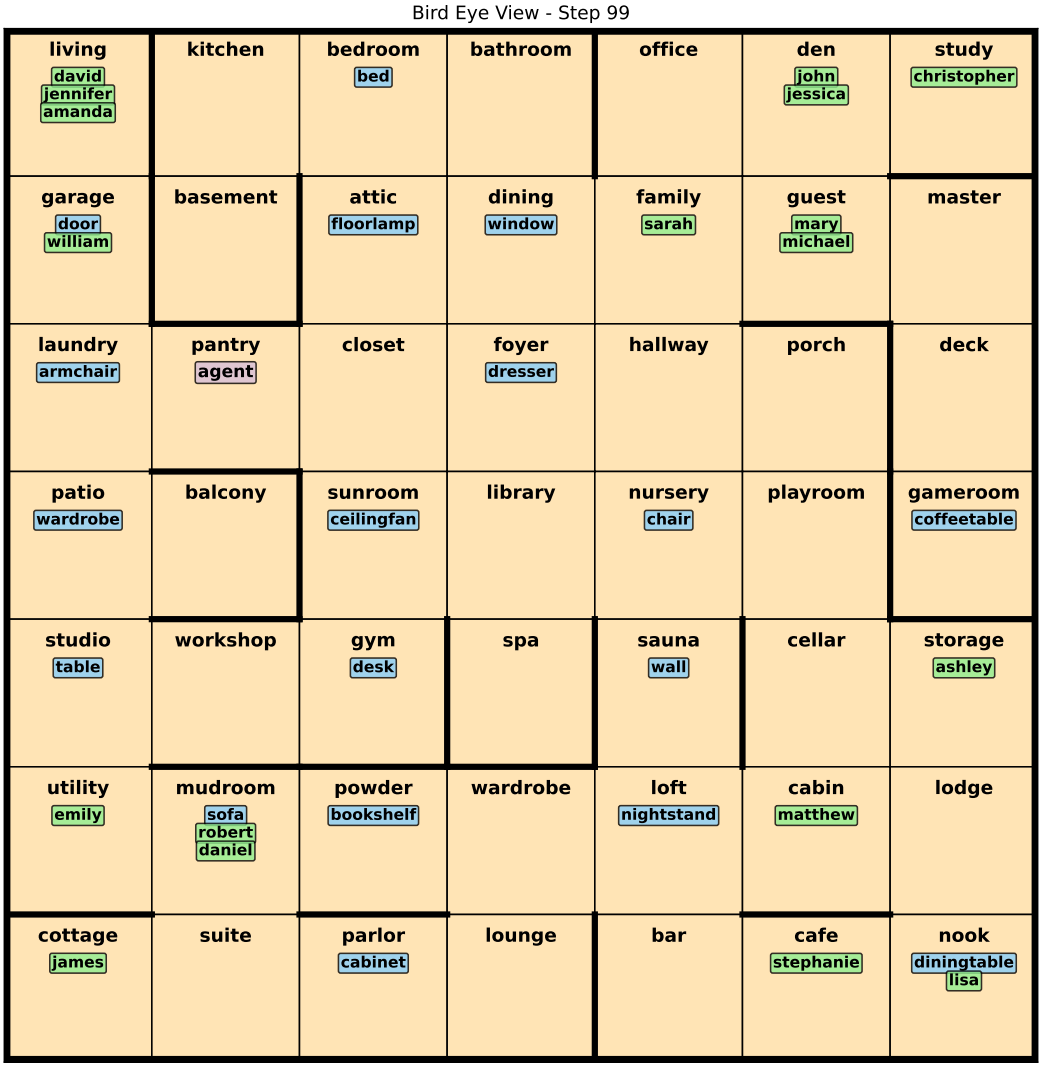
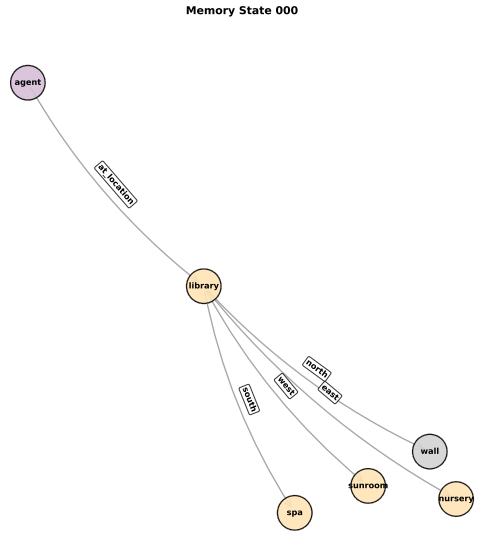
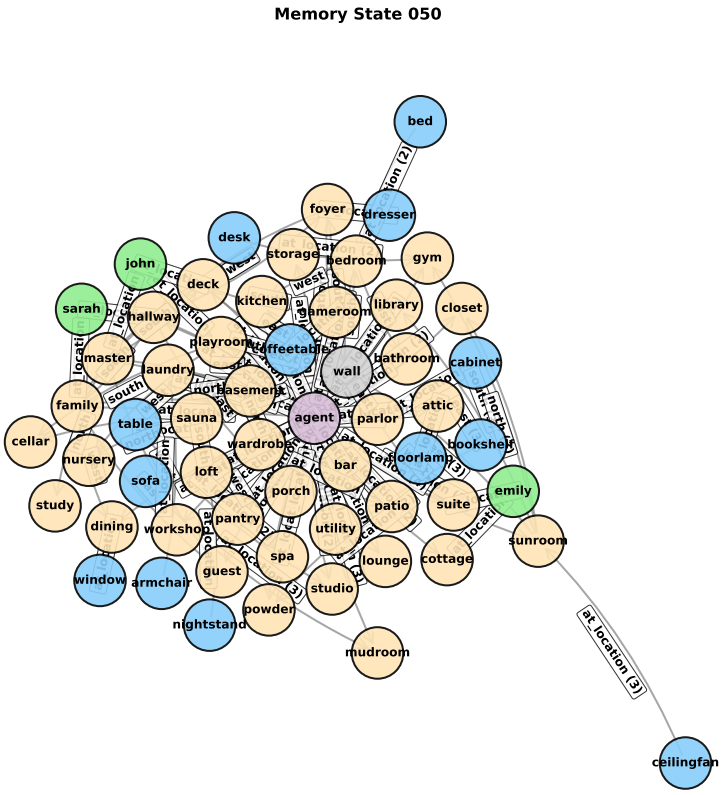


Figure 3: Bird's-eye schematic of the hidden state at $t = 99$ ($s_{t=99}$) in RoomKG, showing spatial layout and entity placement. This view is provided for intuition only: the actual environment state and agent-facing world in our method are represented as RDF knowledge-graph structures (Figure 1a). The agent does not directly observe s_t; instead, its observation o_t corresponds to a local induced RDF subgraph around the current room and visible adjacency relations.



(a) Step 0



(b) Step 50

Figure 4: Agent internal memory-state snapshots at two time points (steps 0 and 50) in a held-out test episode.