Why Knowing Both Hops Is Not Enough: Understanding Two-Hop Generalization in Language Models

Anonymous ACL submission

Abstract

Large language models (LLMs) can solve complex multi-hop problems, yet they exhibit a puzzling failure on seemingly simple two-hop queries: although a model may correctly store each individual hop, it often fails to combine them. In this paper, we study the internal mechanism in two-hop reasoning by training transformers from scratch in a controlled symbolic environment. Our experiments reveal a systematic pattern in two-hop generalization: Models generalize reliably when the second hop follows the same distributional patterns observed during training, but systematically fail when the second hop deviates-even though all required atomic facts are individually encoded. Mechanistic analysis shows that this failure arises from a mismatch across layers: lower layers correctly construct compact intermediate representations, while upper layers are specialized to only reason on representations produced within multi-hop trajectories seen during training. Consequently, correct intermediate information is not effectively consumed by upper layers during out-of-distribution two-hop inference. Motivated by this mechanistic misalignment, we use a recurrent-style training strategy that applies the same blocks to both input embeddings and intermediate hidden states, implicitly aligning their formats. This training strategy enables transformers to reuse their reasoning circuitry across input forms and substantially improves generalization on out-of-distribution two-hop queries.

1 Introduction

Large language models (LLMs) have achieved impressive performance across multiple tasks(Zhao et al., 2025a), yet their ability to perform multi-hop reasoning remains fragile and poorly understood(Huang and Chang, 2023). Even in the simplest case—two-hop reasoning—models exhibit a persistent failure: although they can correctly store each atomic fact (e.g., A →B and B → C), they often struggle to reliably combine them into a compositional inference (A →C) (Biran et al., 2024; Balesni et al., 2024).

Therefore, enhancing the ability of multi-hop reasoning has recently become a research focus (Li et al., 2024; OpenAI, 2024; Petty et al., 2024; Luo et al., 2024). Beyond explicit prompting strategies such as Chain-of-Thought (CoT) (Kojima et al., 2022; Wei et al., 2022), A growing line of work suggests that LLMs can answer compositional queries without generating intermediate steps, indicating the presence of implicit multi-hop reasoning (Deng et al., 2024; Yang et al., 2024). However, the underlying mechanisms governing how transformers internally synthesize stored atomic facts into compositional inferences remain not well understood, raising questions about the robustness and generalizability of this capability.

Prior studies have attempted to probe this phenomenon, but they face key limitations. Work analyzing pretrained LLMs suffers from limited experimental control, as it is difficult to distinguish between genuine multi-step reasoning and memorization, spurious correlations, or dataset artifacts given the opacity of pretraining data (Xu et al., 2022; Wang et al., 2023; Ju et al., 2024). As noted by Ye et al. (2025), this uncertainty makes pretrained models inappropriate for drawing definitive mechanistic conclusions. Although research that trains transformers in synthetic environments has uncovered interesting inductive biases (Wang et al., 2024), these studies frequently largely focus on behavioral outcomes, leaving the internal mechanisms underlying systematic two-hop generalization failures insufficiently explored.

We study two-hop reasoning in a symbolic setting by training transformers from scratch, isolating compositional behavior from prior knowledge. Our experiments reveal a consistent internal pattern: models first learn aligned, compact representations for intermediate "bridge entities", allowing gen-

eralization when test-time intermediate states resemble those seen during training. However, when the second hop must operate on an intermediate representation that never appears as part of a multi-hop trajectory, models fail despite they encode the correct underlying atomic fact. Mechanistic analysis shows that upper layers specialize in reasoning only over representation formats encountered during training; for out-of-distribution second-hop facts, they degenerate into simple mapping. Motivated by this, we introduce a recurrent-style training strategy that aligns lower-layer inputs and intermediate representations via shared transformer blocks, enabling the reuse of reasoning circuitry and substantially improving two-hop generalization.

2 Experimental Settings

We construct our training data in a symbolic environment to ensure that the model's behavior can be analyzed under strictly controllable conditions; the design of this environment follows the setups used in Wang et al. (2024).

Data Construction: We begin by defining a finite entity set $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$ and a relation set $\mathcal{R} = \{r_1, \ldots, r_{|\mathcal{R}|}\}$. We construct the **atomic facts** as followings: For each head entity $e_h \in \mathcal{E}$, we first randomly sample 20 distinct relations from $\mathcal{R}$. For each selected relation r, we then randomly choose a tail entity $e_t \in \mathcal{E}$ as the tail entity, yielding an atomic fact (e_h, r, e_t), which simulates simple real-world knowledge (e.g., $(America, capital\ is, Washington)$). All atomic facts are partitioned according to a proportion ϕ_1 into in-distribution (ID) and out-of-distribution (OOD) subsets. We then construct **two-hop facts** by composing pairs of ID atomic facts (e_h, r_1, e_b) and (e_b, r_2, e_t) whenever the tail entity of the first matches the head entity of the second. Each resulting composite takes the form (e_h, r_1, r_2, e_t), where the shared entity e_b serves as the bridge entity. The full set of two-hop ID composites is further split with ratio ϕ_2 into a training split (Train-II) and an ID evaluation split (Test-II). Applying the same compositional procedure to OOD atomic facts yields the OOD evaluation split (Test-OO). Following the setup in Ye et al. (2025), we additionally generate two cross-distribution evaluation sets: Test-IO, where the first hop originates from ID atomic facts and the second hop from OOD; and Test-OI, where the ordering is reversed. We provide more details in Appendix A.

Training Configuration: Our training split consists of all atomic facts (both ID and OOD) and the in-distribution two-hop composites in Train-II. All other two-hop splits (Test-II, Test-IO, Test-OI, and Test-OO) are used for evaluation to ensure that generalization is measured under controlled distributional shifts. We train a GPT2-style decoder-only transformer(Radford et al.) with 8 layers and a hidden dimension of 512. The model is optimized using AdamW with a learning rate of 1×10^{-4}. More detailed training configuration is provided in appendix B.

3 Training Results

Figure 1 shows the training dynamics across different data splits. We observe that:

- The model first achieves near-perfect accuracy on the in-distribution two-hop training set (Train-II) after a relatively small number of optimization steps, indicating that the compositional patterns present in the training data are quickly memorized.
- After continued training, the accuracy on the in-distribution two-hop test set (Test-II) begins to increase steadily, eventually approaching the performance achieved on Train-II.
- At a later stage of training, the model also exhibits noticeable gains on the Test-OI split, where the first hop is out-of-distribution while the second hop remains in-distribution. This indicates a limited form of generalization, in which the model can partially transfer its learned compositional structure to cases where only one component deviates from the training distribution.
- In contrast, performance on the Test-IO and Test-OO splits remains close to chance throughout training. Even with extended optimization, the model fails to generalize when the second hop originates from out-of-distribution atomic facts, highlighting a persistent asymmetry in its two-hop generalization behavior.

Overall, the model exhibits a highly asymmetric generalization pattern: it generalizes reliably in-distribution, does so only when the first hop is out-of-distribution (Test-OI), and consistently fails

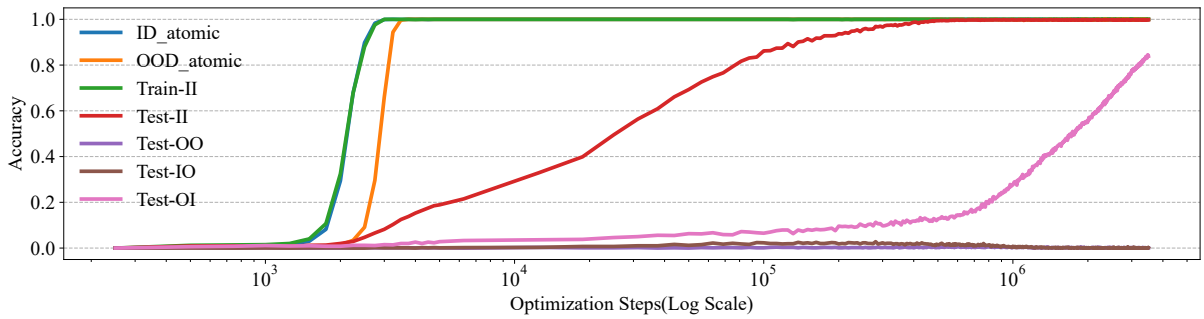


Figure 1: Training dynamics of a transformer trained on all atomic facts and in-distribution two-hop compositions. Accuracy is shown for atomic facts, the two-hop training set (Train-II), and multiple evaluation splits. The model rapidly fits the training data, followed by delayed generalization to the in-distribution test set (Test-II) and partial improvement on Test-OI. In contrast, performance on Test-IO and Test-OO remains near chance throughout training, indicating a failure to generalize when the second hop is out-of-distribution.

when the second hop is out-of-distribution (Test-IO and Test-OO). In the following section, we investigate the underlying reasons for this disparity by analyzing how intermediate representations are formed and processed during two-hop reasoning.

4 Mechanism of Two-Hop Reasoning

4.1 Identifying the Bridge Entity Representation in Hidden Space

To explain the asymmetric training dynamics observed above, we begin by identifying where and how the bridge entity e_2 is represented in the model. To this end, we conduct a logit lens (nostalgebraist, 2020) analysis at both the r_1 and r_2 positions. At the r_1 position, we track how the probability of e_2 evolves across layers, probing when the model acquires sufficient information about the intermediate entity. At the r_2 position, we examine the probabilities of the relation token r_2 itself and the target entity e_3, which together reflect the onset of second-hop reasoning.

As shown in Figure 2, Layer 5 emerges as a critical transition point. By this layer, the model has already accumulated strong evidence for the bridge entity e_2, as indicated by the high probability of e_2 at r_1 position. More importantly, after Layer 5 we observe a qualitative shift in behavior at the r_2 position: the probability of the surface token r_2 begins to decrease, while the probability of the final answer e_3 rises. This reversal indicates that the model transitions from encoding itself to integrating contextual information and performing the second hop of reasoning.

Taken together, these observations suggest that the output of Layer 5 at position r_1 encodes a stable

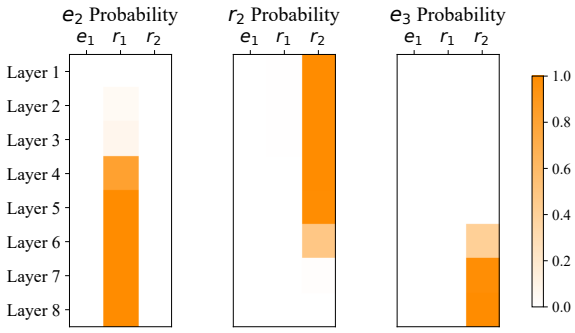


Figure 2: Tracking token probabilities reveals Layer 5 as a transition point: the bridge entity e_2 is already strongly represented at r_1 **(left)**, and after this layer the model shifts at r_2 from predicting the surface relation r_2 **(middle)** to the target entity e_3 **(right)**, signaling the onset of second-hop reasoning.

and functional representation of the bridge entity e_2, denoted as $\mathscr{R}$ in the following analysis, which is subsequently consumed by higher layers to complete the second-hop inference. In the remainder of this section, we adopt the Layer 5 hidden state as the definition of the bridge entity representation and use it as a basis for further mechanistic analysis.

4.2 Probing Bridge Entity Representation Consistency via Entity Patching

To explain why the model generalizes successfully on Test-II and Test-OI, we hypothesize that generalization arises from the emergence of consistent internal representations for the same bridge entity, even when it appears in different contexts.

To test this hypothesis, we introduce an **entity representation consistency patching experiment**, illustrated in Figure 3. We consider a two-hop

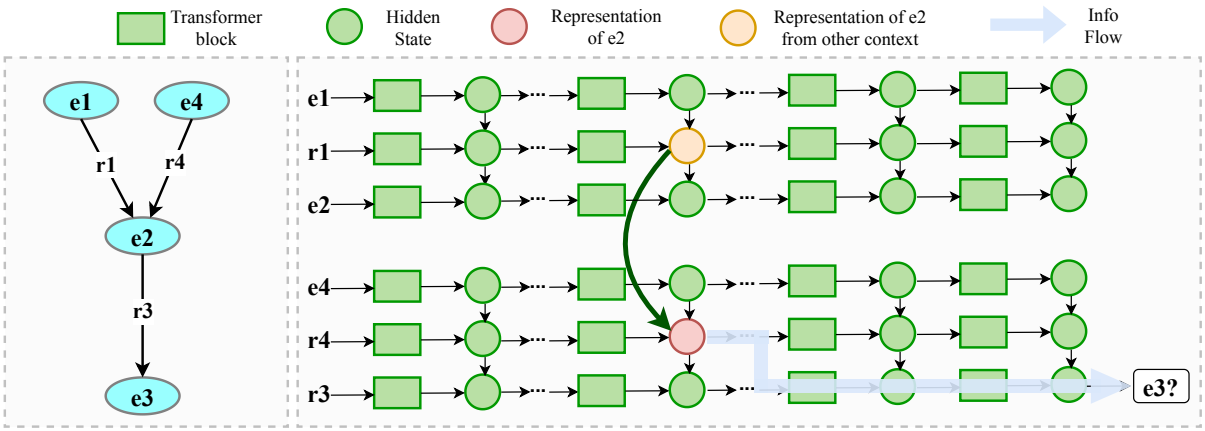


Figure 3: **(Left)** A two-hop query composed of two single-hop facts, where the bridge entity e_2 connects (e_4, r_4) and (e_2, r_3) to yield the target entity e_3. **(Right)** The entity patching procedure: the hidden representation of the bridge entity e_2 at Layer 5 is replaced with an alternative representation of the same entity extracted from a different single-hop context (e.g., (e_1, r_1)). The model's ability to recover the original answer after patching tests whether representations of the same entity are aligned across contexts.

query from the in-distribution training set (Train-II), where the bridge entity e_2 appears at position r_2 and is represented by the hidden state at Layer 5, following the definition established in Section 4.1. In this instance, the single-hop context that gives rise to e_2 is (e_4, r_4).

We then select another single-hop atomic fact with the same tail entity e_2 but a different context, e.g., (e_1, r_1). The representation of e_2 extracted from this context is used to replace the original bridge entity representation in the two-hop query, while keeping all other components unchanged[1]. After patching, we evaluate whether the model still produces the correct final answer e_3.

We perform this replacement using bridge entity representations drawn from two sources: (i) in-distribution (ID) atomic facts and (ii) out-of-distribution (OOD) atomic facts, and record the patching success rate throughout training. Intuitively, if the model has learned a context-invariant representation for e_2, substituting its representation from another context should preserve the correctness of the second-hop inference.

The results, shown in Figure 4, reveal a clear training-dependent pattern. In the early stages of training, patching success rates are low for both ID and OOD replacements, indicating that representations of the same entity remain highly context-dependent. During this phase, the model operates

[1]To exclude the possibility that the final answer is already encoded at position r_3 in early training, we replace r_3 with that from a different two-hop query whose answer is not e_3, ensuring that predicting e_3 depends on the patched representation rather than residual information from earlier layers.

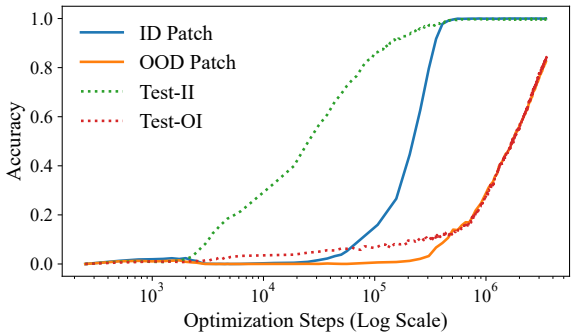


Figure 4: Patching success rates using ID and OOD entity representations during training. ID patching improves earlier and aligns with Test-II accuracy, while OOD patching improves later and tracks Test-OI performance, indicating that two-hop generalization emerges as bridge entity representations become more consistent across contexts.

in a memorization regime, encoding entity information in a fragmented and inefficient manner.

As training progresses, the success rate of ID-based patching increases first. Notably, this rise coincides with the improvement of performance on Test-II, suggesting that generalization within the in-distribution setting emerges once representations of the same entity become aligned across different ID contexts. With further training, the success rate of OOD-based patching also begins to increase, accompanied by a corresponding improvement on Test-OI.

These results support the assumption that generalization in two-hop reasoning is driven by the gradual alignment and unification of entity representa-

tions across contexts. Moreover, representations derived from in-distribution atomic facts converge more rapidly than those from out-of-distribution facts, explaining why generalization to Test-II precedes that to Test-OI.

4.3 Why Second-Hop OOD Fails: Reasoning vs. Mapping in Upper Layers

We next investigate why generalization emerges on Test-II and Test-OI, but consistently fails on Test-IO and Test-OO. An observation is that the essential difference between these test sets lies in whether the second hop corresponds to in-distribution or out-of-distribution knowledge. Given that the first-hop representation has already been shown to be well-formed and robust (Section 4.2), the remaining bottleneck must reside in the second-hop computation, namely how the model operates on intermediate representations.

In a two-hop query of the form $(e_1, r_1, r_2) \rightarrow e_3$, during the second-hop inference, the model is required to use the **representation** of the bridge entity e_2 formed at position r_1, and then reason through relation r_2 to infer the target entity e_3. As shown in the left side of Figure 5, this requires the upper layers to implement a computation that implements $\mathscr{R}(e_2) \xrightarrow{r_2} \mathbf{E}(e_3)$, where the input is an intermediate entity representation rather than a surface token.

In contrast, during single-hop training on atomic facts (e_2, r_2, e_3), the situation is fundamentally different. At position r_2, the upper layers already has direct access to the representation of e_3. Consequently, the upper layers **are not required** to perform relational reasoning; instead, they can learn a much simpler transformation: $\mathscr{R}(e_3) \longrightarrow \mathbf{E}(e_3)$, namely a mapping from an already-formed representation to the final output embedding space, as shown in the right side of Figure 5.

This distinction leads to a crucial hypothesis: Single-hop training primarily teaches the upper layers to perform representation mapping rather than representation-based reasoning. Under this view, the observed generalization behavior follows naturally. For Test-II and Test-OI, the second hop has appeared during training in the form of an intermediate representation (i.e., within Train-II). Consequently, the model learns to consume the representation of e_2 and reason through r_2, enabling it to transfer this representation-based reasoning capability to these test subsets. In contrast, for Test-IO and Test-OO, the second hop has never been trained in a representational form—it has only been encountered as a single-hop atomic fact. The upper layers therefore attempt to apply a learned mapping where reasoning is required, leading to systematic failure.

To validate this hypothesis, we design a linear probing experiment[2]. Specifically, for atomic training examples, we collect: the hidden state at the $\langle r \rangle$ position from an intermediate layer ℓ as input, denoted $\mathbf{h}_\ell(r)$; and the hidden state at the same position from the final layer L as output, denoted $\mathbf{h}_L(r)$. We then fit a linear model $\mathbf{h}_L(r) \approx \mathbf{W}\mathbf{h}_\ell(r)$ using least squares on a subset of these $(\mathbf{h}_\ell, \mathbf{h}_L)$ pairs, and evaluate its performance on held-out data, similar idea as Khandelwal and Pavlick (2025).

The evaluation results in Figure 6 reveal a clear layer-wise pattern: For lower layers, linear prediction performance is poor, indicating that the hidden state at $\langle r \rangle$ alone is insufficient and that the model still depends on information from entity positions. However, for upper layers (above layer 4), the linear fit becomes remarkably accurate. This suggests that, at these layers, the transformation from intermediate to final is well-approximated by a linear map, and no additional relational computation is required.

Taken together, these findings provide strong evidence that the upper layers of the model, as trained on atomic facts, specialize in representation mapping rather than relational reasoning. This specialization explains why second-hop generalization succeeds only when the second hop has been trained in a representational form, and fails otherwise.

5 Bridging the Representation-Reasoning Gap for Robust Two-Hop Generalization

Our analysis so far reveals that the failure of two-hop generalization does not stem from missing knowledge, but from a structural mismatch between how different layers of the model operate. Under standard training, the model naturally develops a functional separation: lower layers are responsible for constructing intermediate entity representations, while upper layers consume these representations to perform relational reasoning. However, the reasoning capability of the upper layers is only

[2]Besides, we also conduct an attention blocking experiment (Geva et al., 2023) in Appendix D, which yields the same conclusion.

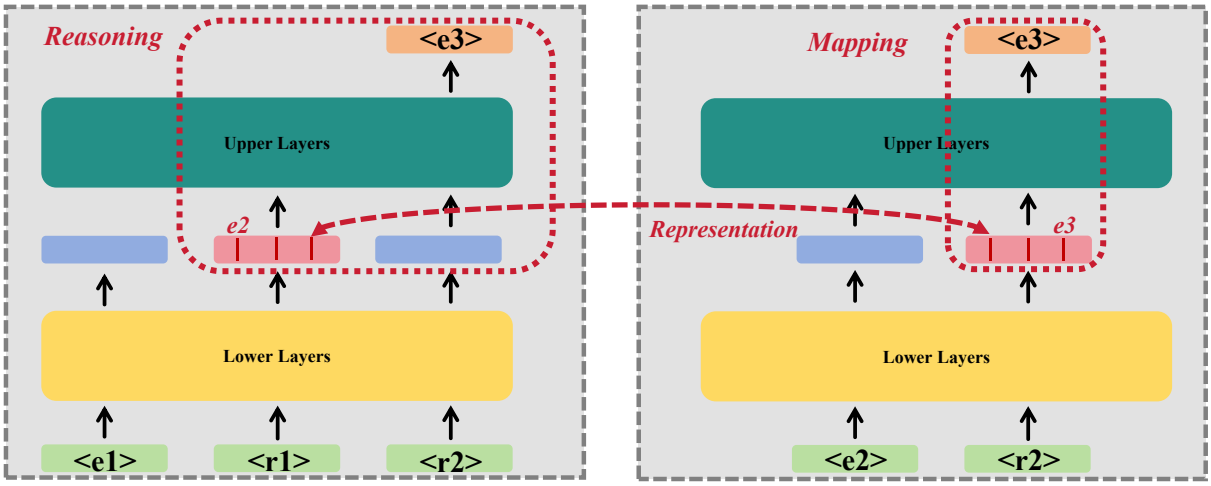


Figure 5: **(Left)**: In a two-hop query $(e_1, r_1, r_2) \rightarrow e_3$, lower layers first construct a representation of the bridge entity e_2 at the r_1 position. The upper layers must then reason over this representation by combining $\mathscr{R}(e_2)$ with relation r_2 to infer the target entity e_3. **(Right)**: In contrast, for a single-hop atomic fact (e_2, r_2, e_3), the representation of e_3 is already formed at the r_2 position in lower layers. The upper layers therefore only need to map this representation to the final output embedding $\mathbf{E}(e_3)$, without performing relational reasoning.

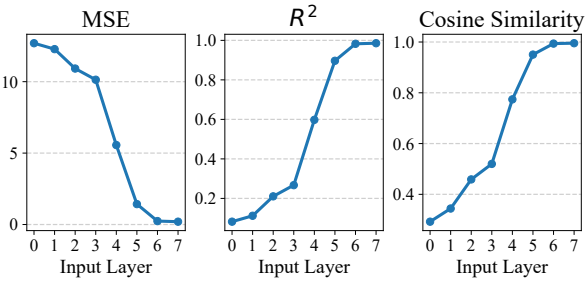


Figure 6: A linear probe predicts final-layer hidden states from intermediate layers, with MSE, R^2, and cosine similarity showing poor performance in lower layers but high accuracy in upper layers, which indicates that the upper layers mainly perform representation mapping rather than relational reasoning during atmotic-facts training.

reliable within the distribution of representational forms observed during training. When faced with out-of-distribution (OOD) knowledge at the second hop, the upper layers no longer perform reasoning; instead, they fall back to a simpler representation mapping behavior, leading to systematic failure.

This diagnosis suggests a natural direction for achieving robust generalization: explicitly bridging the gap between lower-layer representations and upper-layer reasoning. In the following, we explore two strategies to achieve this goal.

5.1 Explicitly training upper layers for representation-based reasoning

A direct approach is to equip the upper layers with the ability to reason over representations. Concretely, after standard training, we introduce an continual training process applied only to the upper layers (Layer 5 and above). In this auxiliary phase, the model is trained on OOD atomic facts, but presented purely in a representational form: the inputs to the upper layers are pairs of hidden states (h_e, h_r), rather than surface tokens[3]. We provide more details about the continual training procss in Appendix E

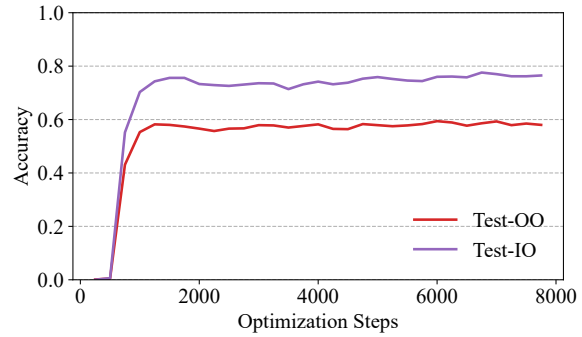
The results[4] are shown in Figure 7a. We observe that this additional process substantially improves generalization on previously failing settings, particularly Test-IO and Test-OO, indicating that the upper layers have indeed acquired the ability to perform reasoning over representations rather than relying on token-level shortcuts. However, this approach comes with notable drawbacks: it introduces additional training stages and increases computational cost.

5.2 Eliminating layer-wise mismatch via parameter sharing

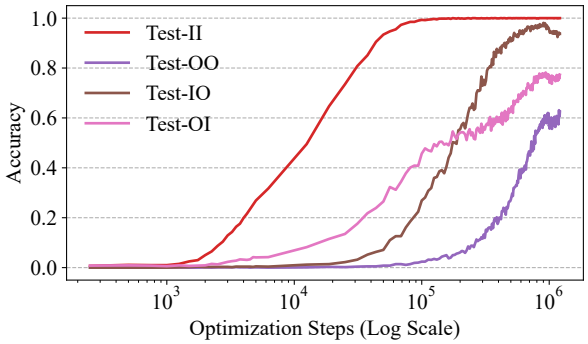
An alternative and more principled solution is to remove the structural mismatch altogether. Instead of explicitly training the upper layers on representational inputs, we enforce alignment between representations and lower-layer inputs by sharing param-

[3]This continual training only exposes the model to atomic facts and does not leak any two-hop supervision.

[4]We only depicte the process of continual training in Figure 7; the standard training process is the same as in Figure 1.



(a)



(b)

Figure 7: **(a)** Continual training of upper layers on atomic facts presented in a representational form improves two-hop generalization on previously failing OOD settings. **(b)** Looped training with shared parameters yields consistently strong generalization across all test splits by aligning intermediate representations with lower-layer inputs.

eters across layers[5], forming a **looped architecture** (Yang et al., 2023; Fan et al., 2024). In this design, the model's first forward pass produces an intermediate representation, which is then fed back as input to a second forward pass using the same set of parameters. Because both passes share weights, the model is forced to process surface tokens and intermediate representations in a unified format.

As a result, even when the model has never encountered OOD atomic knowledge in a representational form, the alignment between representations and lower-layer inputs allows it to reuse the same reasoning circuit. Figure 7b demonstrates that this looped training strategy yields strong and stable generalization across all evaluation settings, without requiring any additional supervision on OOD representations.

[5]Here, we share the parameters of the top four layers and the bottom four layers.

Training	**Compared Representations**		e_2	r_2
Looped	$(h_4(r_1), h_4(r_2))$ (E_{e_2}, E_{r_2})	vs.	0.6796	0.3141
	$(h_5(r_1), h_5(r_2))$ $(h_1(e_2), h_1(r_2))$	vs.	**0.8132**	**0.8487**
Standard	$(h_4(r_1), h_4(r_2))$ (E_{e_2}, E_{r_2})	vs.	0.3528	0.4121
	$(h_5(r_1), h_5(r_2))$ $(h_1(e_2), h_1(r_2))$	vs.	-0.0028	0.3052

Table 1: Cosine similarity between intermediate representations and lower-layer inputs under looped and standard training. Looped training induces strong cross-layer alignment.

5.3 Evidence of Representation-Input Alignment under Looped Training

The hypothesis behind looped training is that robust two-hop generalization emerges from an explicit alignment between intermediate representations and lower-layer inputs. In this subsection, we provide direct empirical evidence for this claim by quantitatively measuring representational similarity across layers.

We analyze the cosine similarity between the representations involved in the second-hop computation and the corresponding lower-layer inputs. Specifically, we consider the query of the second hop in two forms: (i) its representation at intermediate layers, and (ii) its form as a lower-layer input. We measure the cosine similarity between representations at the end of the loop (the fourth layer) and the input embeddings, as well as between higher-layer representations $(h_5(r_1), h_5(r_2))$ and the corresponding first-layer hidden states $(h_1(e_2), h_1(r_2))$. All similarities are averaged across evaluation examples. We compare these quantities under looped training and standard (non-looped) training.

The results in Table 1 reveal a striking contrast between the two training regimes. Under looped training, intermediate representations exhibit strong alignment with both input embeddings and lower-layer hidden states. In particular, the representation of the bridge entity e_2 shows consistently high cosine similarity across layers, and the representation of relation r_2 becomes highly aligned when comparing higher-layer outputs to lower-layer inputs. In contrast, under standard training, this alignment is weak or entirely absent: the similarity between higher-layer representations

and lower-layer inputs collapses, especially for the bridge entity representation.

These findings provide direct mechanistic evidence that looped training achieves robust two-hop generalization by aligning the representational formats of intermediate states with lower-layer inputs. By forcing surface tokens and intermediate representations to be processed through the same parameters, the model can reuse its existing reasoning machinery when encountering OOD facts, explaining the strong and stable generalization observed in Section 5.3.

6 Related Works

6.1 Mechanistic Interpretability

Mechanistic interpretability aims to reverse engineer how LLMs implement specific computations internally (Olah, 2022). A line of work studies hidden states across layers, including logit lens methods that project intermediate states through the unembedding matrix to examine token-level predictions (nostalgebraist, 2020; Dar et al., 2023; Yu and Ananiadou, 2024; Katz and Belinkov, 2023). Other approaches rely on causal interventions on hidden states to assess their impact on model outputs, enabling more direct localization of functionally relevant components (Stolfo et al., 2023; Meng et al., 2022; Vig et al., 2020). In addition, the superposition hypothesis motivates sparse autoencoders for extracting more interpretable features from activations (Scherlis et al., 2022; Elhage et al., 2022; Balcells et al., 2024; Zhao et al., 2025b). Together, these studies provide a toolkit for dissecting how complex behaviors emerge from layered computations.

6.2 Implict Reasoning

Recent studies show that LLMs can perform multi-step or compositional reasoning without explicitly generating intermediate chains, known as implicit reasoning (Kojima et al., 2022; Wei et al., 2022). Prior work provides evidence that transformers encode latent reasoning states internally (Deng et al., 2024), and pretrained models can answer multi-hop queries without exposing intermediate steps (Yang et al., 2024; Balesni et al., 2024). Mechanistic analyses further link the emergence of implicit reasoning to representation structure and grokking-like dynamics (Wang et al., 2024; Ye et al., 2025). Based on this line of research, we identify a layer-wise representation-reasoning mismatch as the cause of second-hop OOD failure, and show that aligning representational formats enables the two-hop generalization.

6.3 Two-Hop Reasoning

Two-hop reasoning is a canonical setting in multi-hop question answering, requiring the retrieval and composition of multiple atomic facts into a single inference (Yang et al., 2018). Prior work identifies a persistent gap between factual recall and compositional reasoning in LLMs: Press et al. (2023) formalize this issue as the compositionality gap, where models fail on two-hop queries despite succeeding on the corresponding single-hop facts. Subsequent studies show that such failures are unstable across domains and settings, questioning whether LLMs robustly perform genuine multi-hop reasoning (Berglund et al., 2023; Yang et al., 2024). Mechanistic analyses further reveal that even when the bridge entity is correctly retrieved at the first hop, it is often not utilized by higher layers, indicating a disconnect between representation formation and reasoning (Biran et al., 2024; Yu et al., 2025). Related work also investigates two-hop reasoning circuits from theoretical, controlled finetuning, and in-context learning perspectives (Feng et al., 2024; Guo et al., 2025).

7 Conclusion

In this work, we present a controlled mechanistic study of two-hop reasoning by training transformers from scratch in a symbolic environment. We show that successful two-hop generalization depends on learning compressed and aligned representations of intermediate bridge entities. We further find that upper layers mainly perform representation mapping from atomic facts, rather than representation-based reasoning, explaining failures when the second hop is out-of-distribution. Motivated by this insight, we explore two strategies to improve two-hop reasoning generalization: explicitly training upper layers on representational inputs, and a simpler looped architecture that aligns intermediate representations with lower-layer inputs. Both approaches substantially improve out-of-distribution two-hop generalization, with the looped design offering a principled and efficient solution. Overall, our results offer a mechanistic explanation for asymmetric two-hop generalization and highlight representation alignment across layers as key to robust multi-hop reasoning.

Limitations

While our results demonstrate that improved two-hop generalization is closely tied to more compact and aligned intermediate representations, the underlying drivers of this representational compression remain partially understood. In particular, although in Appendix C, we show that representation compression emerges gradually during training and is strongly correlated with the structural properties of the training data, we do not fully characterize why models are incentivized to compress intermediate representations in this manner.

In addition, while our looped architecture effectively enables representation reuse and substantially improves generalization, our training procedure remains relatively simple: we adopt a standard training strategy and apply only a single loop. It is possible that alternative training schedules, multiple looping iterations, or more adaptive mechanisms could further enhance efficiency or performance (Zhu et al., 2025). Exploring these directions may lead to more effective architectures, but falls outside the scope of the present study.

References

Daniel Balcells, Benjamin Lerner, Michael Oesterle, Ediz Ucar, and Stefan Heimersheim. 2024. Evolution of sae features across layers in llms. *arXiv preprint arXiv:2410.08869*.

Mikita Balesni, Tomasz Korbak, and Owain Evans. 2024. Lessons from studying two-hop latent reasoning.

Lukas Berglund, Asa Cooper Stickland, Mikita Balesni, Max Kaufmann, Meg Tong, Tomasz Korbak, Daniel Kokotajlo, and Owain Evans. 2023. Taken out of context: On measuring situational awareness in llms. *arXiv preprint arXiv:2309.00667*.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. 2024. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In *Conference on Empirical Methods in Natural Language Processing*.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, Toronto, Canada. Association for Computational Linguistics.

Yuntian Deng, Yejin Choi, and Stuart Shieber. 2024. From explicit cot to implicit cot: Learning to internalize cot step by step. *arXiv preprint arXiv:2405.14838*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, and 1 others. 2022. Toy models of superposition. *arXiv preprint arXiv:2209.10652*.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. 2024. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*.

Jiahai Feng, Stuart Russell, and Jacob Steinhardt. 2024. Extractive structures learned in pretraining enable generalization on finetuned facts. *ArXiv*, abs/2412.04614.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Tianyu Guo, Hanlin Zhu, Ruiqi Zhang, Jiantao Jiao, Song Mei, Michael I Jordan, and Stuart Russell. 2025. How do llms perform two-hop reasoning in context? *arXiv preprint arXiv:2502.13913*.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards Reasoning in Large Language Models: A Survey. *Preprint*, arXiv:2212.10403.

Tianjie Ju, Yijin Chen, Xinwei Yuan, Zhuosheng Zhang, Wei Du, Yubin Zheng, and Gongshen Liu. 2024. Investigating multi-hop factual shortcuts in knowledge editing of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8987–9001, Bangkok, Thailand. Association for Computational Linguistics.

Shahar Katz and Yonatan Belinkov. 2023. Visit: Visualizing and interpreting the semantic information flow of transformers. *arXiv preprint arXiv:2305.13417*.

Apoorv Khandelwal and Ellie Pavlick. 2025. How do language models compose functions? *arXiv preprint arXiv:2510.01685*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Zhaoyi Li, Gangwei Jiang, Hong Xie, Linqi Song, Defu Lian, and Ying Wei. 2024. Understanding and patching compositional reasoning in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9668–9688, Bangkok, Thailand. Association for Computational Linguistics.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language

models by automated process supervision. *ArXiv*, abs/2406.06592.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372.

nostalgebraist. 2020. Interpreting gpt: The logit lens. LessWrong blog post.

Chris Olah. 2022. Mechanistic interpretability, variables, and the importance of interpretable bases. Transformer Circuits essay.

OpenAI. 2024. llm . https://openai.com/zh-Hans-CN/index/learning-to-reason-with-llms/. Accessed: 2025-02-14.

Jackson Petty, Sjoerd Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. 2024. The impact of depth on compositional generalization in transformer language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7239–7252, Mexico City, Mexico. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, Singapore. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. 2022. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *Conference on Empirical Methods in Natural Language Processing*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Neural Information Processing Systems*.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *arXiv preprint arXiv:2405.15071*.

Fei Wang, Wenjie Mo, Yiwei Wang, Wenxuan Zhou, and Muhao Chen. 2023. A causal view of entity bias in (large) language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15173–15184, Singapore. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Nan Xu, Fei Wang, Bangzheng Li, Mingtao Dong, and Muhao Chen. 2022. Does your model classify entities reasonably? diagnosing and mitigating spurious correlations in entity typing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8642–8658, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. 2023. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*.

Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. 2024. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Jiaran Ye, Zijun Yao, Zhidian Huang, Liangming Pan, Jinxin Liu, Yushi Bai, Amy Xin, Liu Weichuan, Xiaoyin Che, Lei Hou, and 1 others. 2025. How does transformer learn implicit reasoning? *arXiv preprint arXiv:2505.23653*.

Zeping Yu and Sophia Ananiadou. 2024. Interpreting arithmetic mechanism in large language models through comparative neuron analysis. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3293–3306, Miami, Florida, USA. Association for Computational Linguistics.

Zeping Yu, Yonatan Belinkov, and Sophia Ananiadou. 2025. Back attention: Understanding and enhancing multi-hop reasoning in large language models. *ArXiv*, abs/2502.10835.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2025a. A Survey of Large Language Models. *Preprint*, arXiv:2303.18223.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. 2025b. Steering knowledge selection behaviours in LLMs via SAE-based representation engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5117–5136, Albuquerque, New Mexico. Association for Computational Linguistics.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, and 1 others. 2025. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*.

A Dataset Details

Our training set is constructed following the approach of Wang et al. (2024), while the test set is built following the methodology of Ye et al. (2025). Specifically, we first construct 2,000 distinct entities (<e_id>) and 200 distinct relations (<r_id>).

For atomic knowledge construction, each entity is randomly assigned 20 distinct outgoing relations, where each relation points to another entity—for example, $(\langle e_0 \rangle, \langle r_0 \rangle) \rightarrow \langle e_1 \rangle$. This results in a total of 40,000 atomic facts. To create in-distribution (ID) and out-of-distribution (OOD) facts, 95% of the atomic facts are assigned to the in-distribution set (ID_atomic), and the remaining 5% are assigned to the out-of-distribution set (OOD_atomic).

Two atomic facts can be combined into a two-hop fact if the tail entity of one fact matches the head entity of the other. In a two-hop fact, the first-hop and second-hop can each come from either the ID or the OOD atomic set. We first consider two-hop facts where both the first and second hops are drawn from the ID atomic set. These two-hop facts are further divided: one portion is included in the test set as Test-II, and from the remaining portion, a number of two-hop facts corresponding to ϕ times the number of ID atomic facts are randomly selected and included in the training set.

Additionally, we construct Test-IO, Test-OI, and Test-OO to more comprehensively evaluate the model's generalization ability. Specifically, Test-IO contains two-hop facts where the first hop comes from ID atomic facts and the second hop comes from OOD atomic facts; Test-OI and Test-OO are defined analogously.

In this paper, we set $\phi = 7.2$ because, according to Wang et al. (2024), 7.2 represents a relatively intermediate value. This corresponds to a training set size that allows the model to generalize on Test-II while avoiding an excessively large dataset that would significantly increase training cost. Choosing $\phi = 7.2$ is appropriate for our study of why the model fails to generalize on Test-IO and Test-OO.

More detailed information about the dataset is provided in Table 2 and Table 3. Table 2 shows the formats of atomic and two-hop facts. Table 3 presents the number of facts in each data split.

Fact Type	Example
Atomic Fact	<e_0><r_0><e_1>
Two-hop Fact	<e_0><r_0><r_1><e_2>

Table 2: Example formats of facts in the dataset.

Source	Data Type	Quantity
Training Set	ID_atomic	38000
	OOD_atomic	2000
	Train-II	273600
Test Set	ID_atomic	3000
	OOD_atomic	2000
	Train-II	3000
	Test-II	3000
	Test-IO	3000
	Test-OI	3000
	Test-OO	1987

Table 3: Detailed statistics of the dataset.

B Training Details

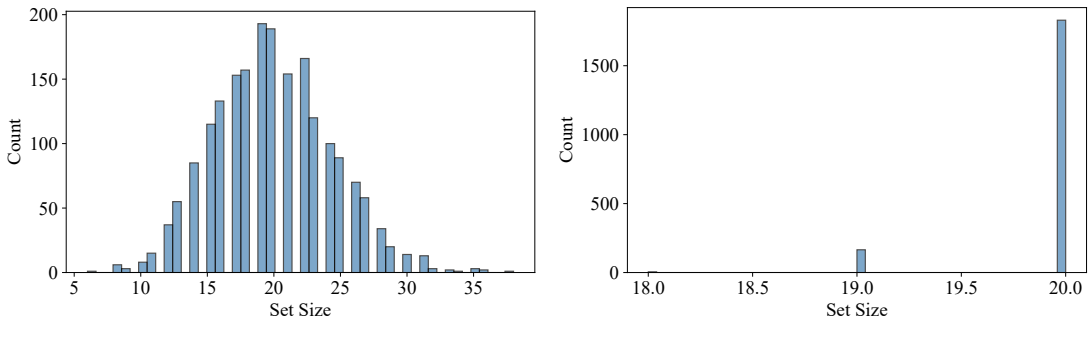
All experiments are conducted on Tesla V100-PCIE-32GB GPUs. Training is performed using four Tesla V100-PCIE-32GB GPUs, with a batch size of 1024 per GPU. We use a learning rate of 1×10^{-4} and a weight decay of 0.1 for optimization. Inference on the test set is conducted using greedy decoding. The number of optimization steps and corresponding training time for the main training experiments reported in this paper are summarized in Table 4.

Experiment	Steps	Training Time (h)
Standard Training	3,500,000	263.05
Rep.-Based Training	7,750	0.88
Looped Training	1,212,500	86.80

Table 4: Optimization steps and training time for the main training experiments in this paper. Standard Training corresponds to Section 3, Rep.-Based Training (denoting representation-based training) corresponds to Section 5.1, and Looped Training corresponds to Section 5.3.

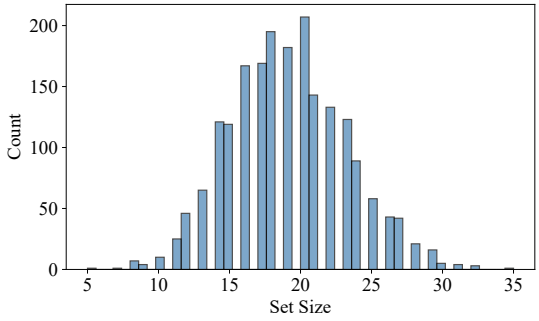
C Intermediate Representations of Bridge Entities Reflect Graph Structure

In this section, we explore the formation of entity representations in relation to the graph structure within the dataset. We first analyze the structure of the graph composed of all atomic facts, and then examine how this graph structure influences the formation of entity representations.

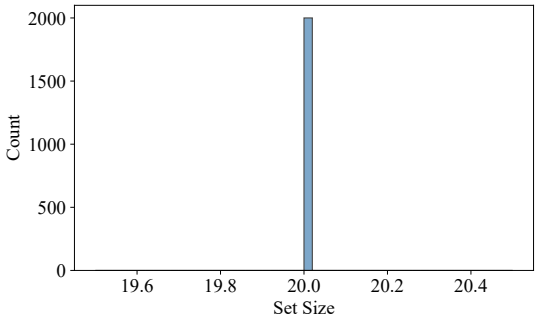


(a) Source entity set e_1

(b) Target entity set e_3



(c) Incoming relation set r_1



(d) Outgoing relation set r_2

Figure 8: Histograms of the set size distributions for the e_1, e_3, r_1, and r_2 sets across all entities.

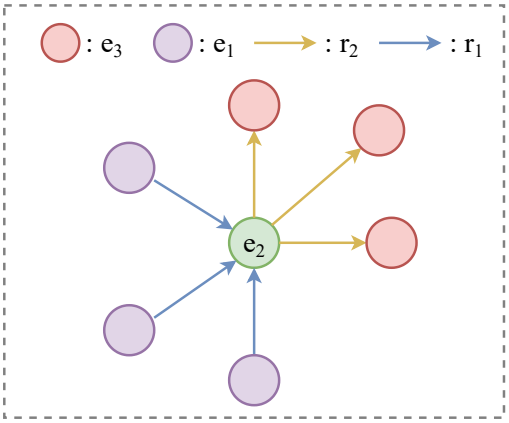


Figure 9: Illustration of the local graph structure centered at entity e_2, where r_1 and r_2 denote the sets of incoming and outgoing relations, respectively, connecting source entities e_1 and target entities e_3.

C.1 Graph Structure in the Dataset

The 40,000 atomic facts constructed in the dataset jointly form a graph structure. The resulting graph contains 2,000 nodes, where each node corresponds to an entity. The graph includes 200 edge types, with each edge type representing a distinct relation. Each node emits 20 outgoing edges connecting to other nodes, and every node is also the target of incoming edges from other nodes.

For ease of presentation, we consider an entity e_2. The edges emitted by e_2 define a set of outgoing relations r_2, and the target entities of r_2 form a set denoted as e_3. The edges pointing to e_2 define a set of incoming relations r_1, and the source entities of r_1 form a set denoted as e_1. Consequently, each entity is associated with four sets: a set of target entities e_3, a set of outgoing relations r_2, a set of source entities e_1, and a set of incoming relations r_1, as illustrated in Figure 9.

For all entities, we collect statistics on the sizes of the e_1, r_1, e_3, and r_2 sets, and plot histograms of the size distributions for each of the four sets, as shown in Figure 8. As can be clearly observed, the sizes of the e_1 and r_1 sets approximately follow a Gaussian-like distribution centered around 20. In contrast, the size of the r_2 set is exactly 20 for all entities, which is consistent with our atomic fact construction procedure. The size of the e_3 set is 20 for the majority of entities, with a small fraction having size 19. This deviation arises because, among the 20 outgoing edges of an entity, multiple edges may point to the same target entity, resulting in a reduced number of unique entities in e_3.

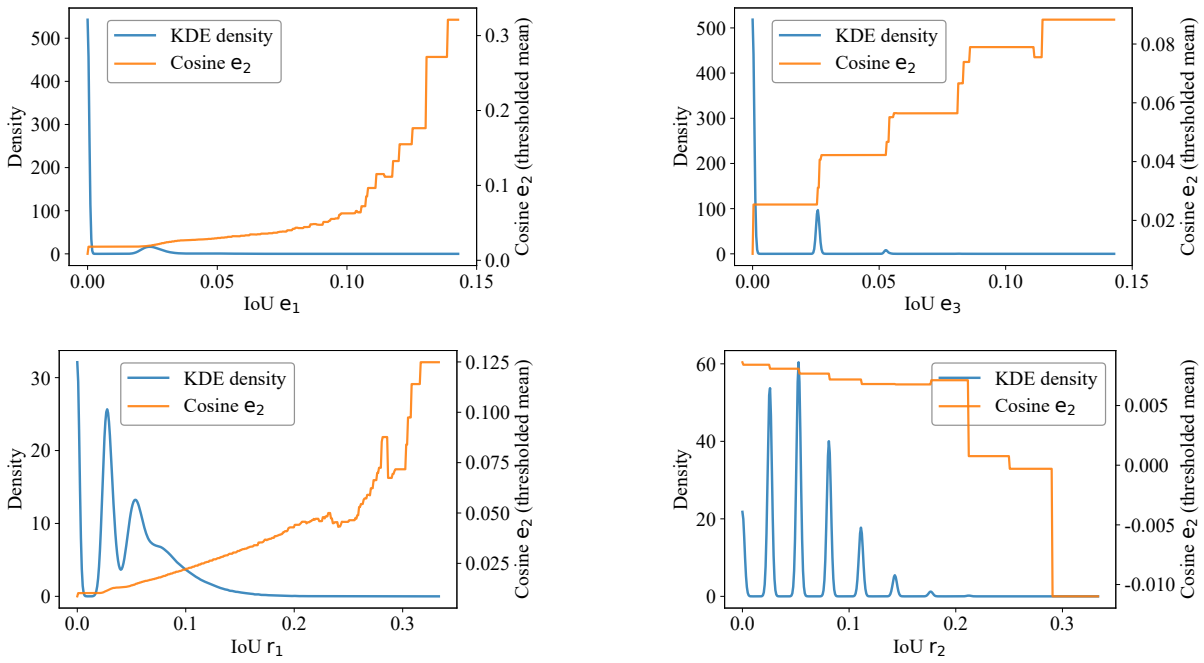


Figure 10: Relationship between pairwise set overlap and representation similarity for the four attribute sets e_1, e_3, r_1, and r_2. The x-axis denotes the intersection-over-union (IoU) between the corresponding sets of two entities. The left y-axis (blue curves) shows the kernel density estimation (KDE) of entity pairs at each IoU value. The right y-axis (orange curves) reports the average cosine similarity of the intermediate representations for entity pairs with IoU greater than or equal to the corresponding threshold.

C.2 How Does Graph Structure Shape the Formation of Entity Representations?

In this paper, we identify and define the output hidden state at r_1 in Layer 5 as the intermediate representation of the bridge entity. We further observe that, as training progresses, the intermediate representations of the same bridge entity in different contexts gradually converge. A natural question arises: what does this aggregated entity representation reflect, or what determines its position in the representation space?

An intuitive hypothesis is that the formation of entity representations is related to the graph structure within the dataset. Specifically, if two entities have similar surrounding attributes—i.e., high overlap in their e_1, r_1, e_3, and r_2 sets—then their representations in the representation space should also be close.

Based on this hypothesis, we design the following experiment to empirically examine the relationship between local graph structure and entity representation similarity:

1. For each entity, extract its surrounding attribute set, consisting of e_1, r_1, e_3, and r_2.

2. Collect the intermediate hidden state representations of all entities using ID atomic facts, and compute the mean representation for each entity.

3. Pair all entities and compute the overlap between their corresponding attribute sets, and plot kernel density estimation (KDE) curves for the resulting overlap distributions.

4. For multiple overlap thresholds, calculate the average cosine similarity between the representations of all entity pairs exceeding each threshold.

5. Plot the average cosine similarity as a function of the overlap threshold.

This experiment enables us to investigate how local graph structure influences the formation of entity representations. The results are shown in Figure 10. The following provides a detailed analysis of these results:

KDE distributions. We first examine the KDE curves. Most entity pairs concentrate at low set overlap values, indicating that only a small fraction of entity pairs share highly similar surrounding attribute sets.

Cosine similarity trends. Next, we consider the cosine similarity curves. For the e_1, e_3, and r_1 sets, the cosine similarity consistently increases as the overlap threshold grows, consistent with the expectation that greater structural overlap corresponds to more similar representations. In contrast, the r_2 set shows decreasing cosine similarity as the overlap threshold increases, appearing to contradict this expectation.

Magnitude and stability of effects. A closer inspection reveals an important difference. The cosine similarity curves for e_1, e_3, and r_1 remain above zero and exhibit relatively large variations, whereas the r_2 curve is roughly centered around zero, with both positive and negative effects and a much smaller range of variation. This suggests that the overlap of the r_2 set has only a negligible influence on the cosine similarity between entity representations.

Overall, higher overlap in the e_1, e_3, and r_1 attribute sets corresponds to greater similarity between the representations of two entities. This indicates that the intermediate entity representations capture implicit graph structures within the dataset, and that, as training progresses, the model tends to organize its hidden representations in a more structured and efficient manner.

D Layer-wise Attention Disabling Experiments

In Section 4.3, we design a controlled linear probing experiment and verify the following hypothesis: when trained on atomic facts, the upper layers of the model mainly learn a linear mapping from internal representations to the final output, rather than performing genuine reasoning.

To further support this hypothesis from a complementary perspective, we conduct an additional experiment based on an attention intervention. Specifically, we require the model to answer atomic fact queries while progressively disabling the attention from the relation token $\langle r \rangle$ to the entity token $\langle e \rangle$ in the top l layers during inference. We vary l from 1 to 8 and report the resulting accuracy as a function of l, as shown in Figure 11.

As can be clearly observed from Figure 11, accuracy on both in-distribution (ID) and out-of-distribution (OOD) atomic facts decreases as more top layers are disabled, which is consistent with intuition. Notably, masking the top three layers has only a marginal impact: the model still achieves

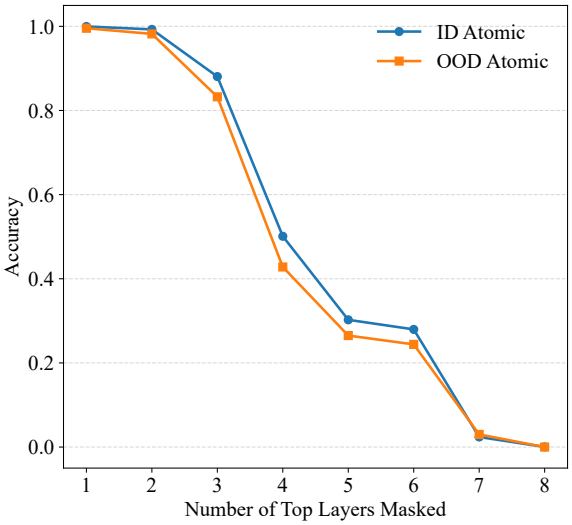


Figure 11: Accuracy variation when disabling attention layers from top to bottom.

over 80% accuracy. In contrast, disabling the top four layers leads to a sharp performance drop of approximately 40%.

This result indicates that, during atomic fact training, the top three layers do not substantially integrate information from the entity position to perform reasoning. Instead, they primarily learn a simple mapping from the representation at the relation token to the output space, further corroborating our findings in Section 4.3.

E Details of Representation-based Training Experiments

This section provides additional details on the training procedure described in Section 5.1. The representation-based training procedure is illustrated in Figure 12. In this experiment, besides the standard training procedure, we additionally train Layers 6–8 to learn OOD atomic facts (e_2, r_2) in a representation-based form (h_{e_2}, h_{r_2}). The procedure is as follows:

1. **Standard training:** We first train the model normally until it achieves high generalization performance on Test-II and Test-OI, as only when the model generalizes well on these benchmarks do the intermediate representations of the same entity become consistent and stable, which in turn provides a reliable foundation for subsequent representation-based training. For this experiment, we directly use the model after 3,500,000 optimization steps of standard training.

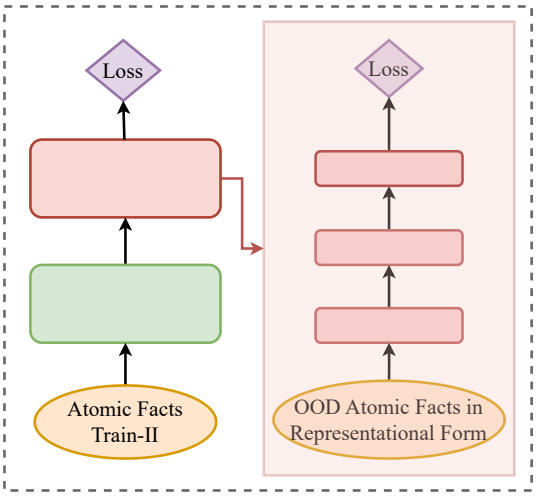


Figure 12: Illustration of the representation-based training process, where intermediate representations of entities and relations for OOD atomic facts are extracted from Train-II (Layer 5) and used to train Layers 6–8.

2. **Multi-round training with representation-based reasoning:** The following procedure is repeated for multiple rounds. In each round, we first perform standard training to maintain the model's generalization performance and prevent the additional representation-based reasoning training from degrading it. After this, we collect the intermediate representations from Train-II: $h_e = h_5(r_1)$ for all entities and $h_r = h_5(r_2)$ for all relations. These representations are then used to construct representation-based forms (h_{e_2}, h_{r_2}) for all OOD atomic facts (e_2, r_2). The pair (h_{e_2}, h_{r_2}) is fed into Layer 6 of the model and propagated through the subsequent layers, with supervision applied to guide the model to output the correct answers.

Test accuracy is evaluated throughout this process.

F Logit Lens Analysis of Looped Training Models

Figure 13 shows the logit lens results of the final model trained under the recurrent architecture when answering two-hop questions. For a token at position t in layer l, we extract its hidden state $h_l(t)$, apply a LayerNorm, and project it through the model's embedding transpose $E^\top$ to obtain the probability of each target entity or relation:

$$p_l(x \mid t) = \text{softmax}\Big(E^\top \text{LayerNorm}(h_l(t))\Big), \quad x \in \{e_2, e_3, r_2\}. \tag{1}$$

This allows us to inspect how the model's predictions for e_2, e_3 and r_2 evolve across layers.

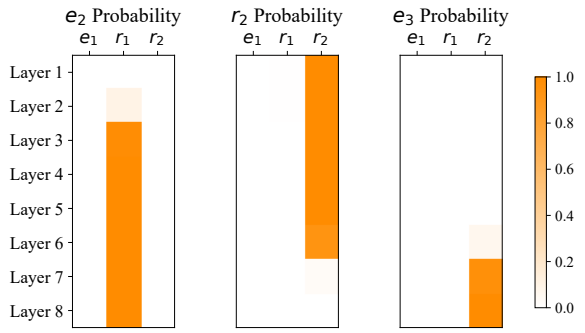


Figure 13: Logit lens of the final model trained under the looped architecture, showing hidden state predictions for e_2, e_3, and r_2. Layer 6 marks the start of second-hop reasoning.

From Figure 13, we observe that Layer 6 marks the onset of the second-hop reasoning: at the position of r_2, the probability of r_2 begins to decrease while the probability of e_3 starts to increase. This indicates that Layer 6 is the first layer where the model initiates second-hop inference. Following the approach in Section 4.1, we define the hidden state at Layer 5 corresponding to the r_1 position, denoted as $h_5(r_1)$, as the intermediate representation of the bridge entity e_2, and the hidden state at Layer 5 corresponding to the r_2 position, denoted as $h_5(r_2)$, as the intermediate representation of relation r_2. During the second-hop reasoning, the model performs inference based on these intermediate representations $h_5(r_1)$ and $h_5(r_2)$.

G Exploring Equivalent Alternatives to the Shared-Parameter Model

In Section 5.3, we empirically demonstrate that the shared-parameter model generalizes well on both Test-IO and Test-OO. We further explain its effectiveness: sharing the parameters of the lower and upper four layers forces the model to align the embedding-level representations and hidden-state representations of atomic facts, while naturally transferring the reasoning capability learned in the lower layers to the upper layers. As a result, even when the model is only exposed to embedding-based reasoning patterns of out-of-distribution

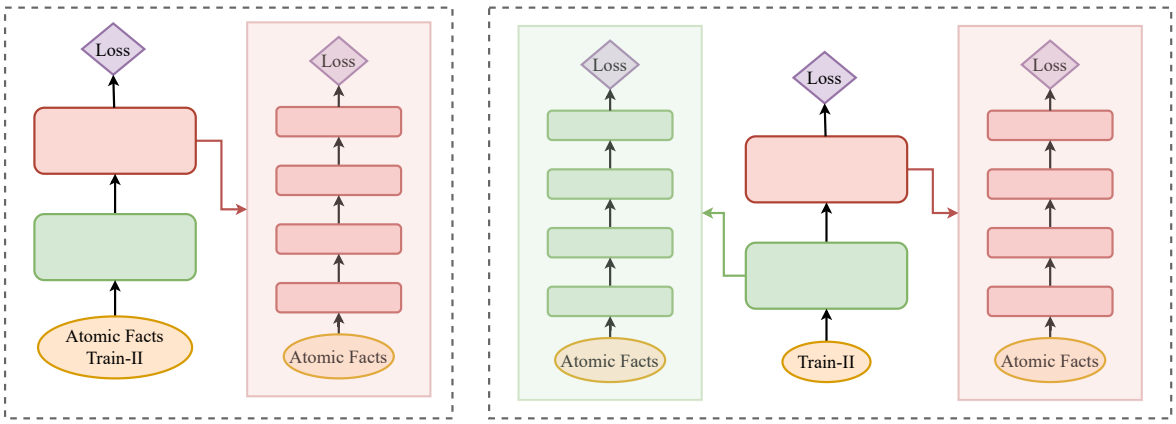


Figure 14: Comparison of two training strategies described in Appendix G. **Left:** the training strategy described in Appendix G.1, where in each training round, embeddings of atomic facts are additionally used to train the upper four layers. **Right:** the training strategy described in Appendix G.2, where atomic facts are removed from the original training set, and in each training round, embeddings of atomic facts are additionally used to train both the upper four layers and the lower four layers.

atomic facts during training, it can still perform reasoning over the hidden representations of out-of-distribution atomic facts at test time.

As a complement to Section 5.3, in this section we explore alternative mechanisms that can serve as equivalent substitutes for the shared-parameter model, with the aim of gaining a deeper understanding of the fundamental sources of its effectiveness. Specifically, we investigate two alternative training strategies that aim to replicate the key effects of parameter sharing without explicitly tying parameters across layers, as illustrated in Figure 14. The left panel of the figure corresponds to a strategy that trains only the upper layers using embedded atomic facts, while the right panel illustrates a strategy that trains both the upper and lower layers separately under the same embedded-atomic-fact supervision.

G.1 Training Upper Layers with Embedded Atomic Facts

In this section, we augment the standard training pipeline with an additional procedure. After each training epoch, we further train the upper four layers of the model using all atomic facts, including both in-distribution (ID) and out-of-distribution (OOD) atomic facts. Specifically, we directly feed the embeddings of atomic facts (E_e, E_r) into the fifth layer of the model, and supervise the model outputs using the target entities of these atomic facts. This procedure is designed to simulate a key property of the shared-parameter model, namely, enabling the upper layers to perform embedding-based reasoning.

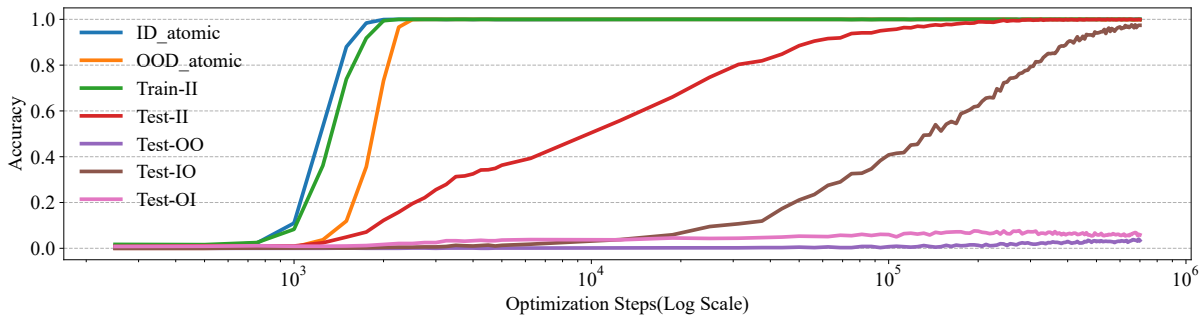
The training results are shown in Figure 15a. Compared with the standard training setting, the results differ in the following aspects:

- **Test-IO and Test-OO.** The augmented training procedure enables strong generalization on Test-IO, achieving nearly 100% accuracy, and weak but non-zero generalization on Test-OO, with around 4% accuracy. In contrast, under the standard training setting, the model achieves almost zero accuracy on both Test-IO and Test-OO.

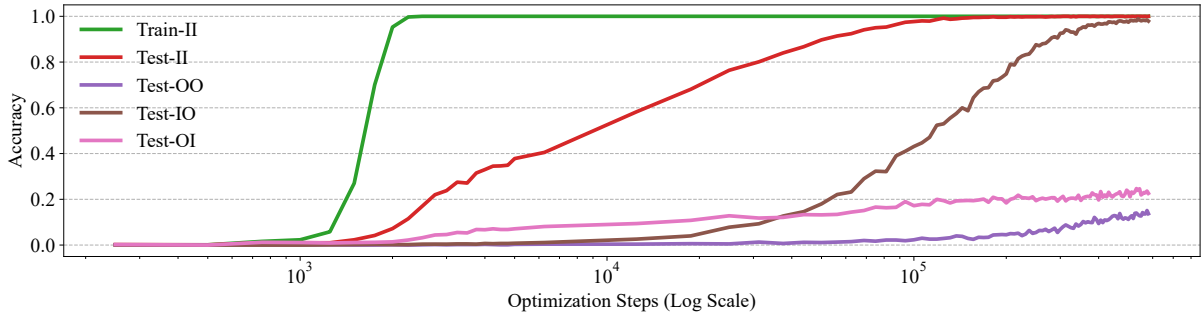
- **Test-OI.** Despite the improvements on Test-IO and Test-OO, the model only achieves limited generalization on Test-OI, with around 6% accuracy. By comparison, the standard training procedure yields over 80% accuracy on Test-OI.

We interpret these results as follows. Training the upper layers with embedding-based atomic facts reshapes their parameters and forces the intermediate representations of entities and relations to align with their corresponding embeddings. Moreover, since the upper layers are explicitly trained on OOD atomic facts, the model is able to generalize to Test-IO. However, introducing this additional training procedure also alters the parameters of the upper layers, which interferes with the model's ability to generalize on Test-OI.

Despite this limitation, the experiment remains informative. The fact that this training strategy enables generalization on Test-IO supports our



(a)



(b)

Figure 15: Training results for the two training strategies described in Appendix G: (a) the strategy corresponding to the left panel in Figure 14, where embeddings of atomic facts are additionally used to train the upper layers; (b) the strategy corresponding to the right panel in Figure 14, where embeddings of atomic facts are used to train both the upper and lower layers. Note that the accuracy on atomic facts is not shown in (b) because this training set does not include supervision that uses all layers to answer atomic facts.

hypothesis that the effectiveness of the shared-parameter model stems from endowing the upper layers with embedding-based reasoning capability.

G.2 Training Upper and Lower Layers Separately with Embedded Atomic Facts

In this section, we attempt to address the issue of disrupted generalization on Test-OI observed in Appendix G.1. According to our analysis in Section 4.2, the model's ability to generalize on Test-OI arises from the consistency of representations for the same entity across different contexts. Furthermore, as analyzed by Ye et al. (2025), the aggregation and alignment of representations generated from out-of-distribution atomic facts are enabled by the presence of in-distribution atomic facts that share the same target entity. In the training strategy adopted in Appendix G.1, the upper layers of the model are trained on atomic facts in two different phases: once through the standard dataset training and once through the additional embedding-based training. We hypothesize that this repeated training of atomic facts in different forms interferes with the model's ability to generalize on Test-OI.

To test this hypothesis, we decouple the standard atomic-fact training from the additional embedding-based training. Specifically, in each training epoch, we train the lower four layers and the upper four layers separately using all atomic facts, while removing all atomic facts from the original training set and retaining only Train-II. The training results are shown in Figure 15b. Compared with Appendix G.1, we observe that, while maintaining high accuracy on Test-IO, the model achieves 25% accuracy on Test-OI and 15% accuracy on Test-OO. Both results are approximately four times higher than those reported in Appendix G.1. These results support our hypothesis that the degradation in Test-OI generalization is indeed caused by interference introduced by the additional training procedure.

Although the proposed training strategy does not reach the accuracy achieved by the shared-parameter model on Test-OI and Test-OO (80% and 60%, respectively), it nonetheless attains a substantial level of performance and provides a closer functional approximation to shared-parameter training. Compared with the setting in Appendix G.1, this experiment more strongly suggests that the

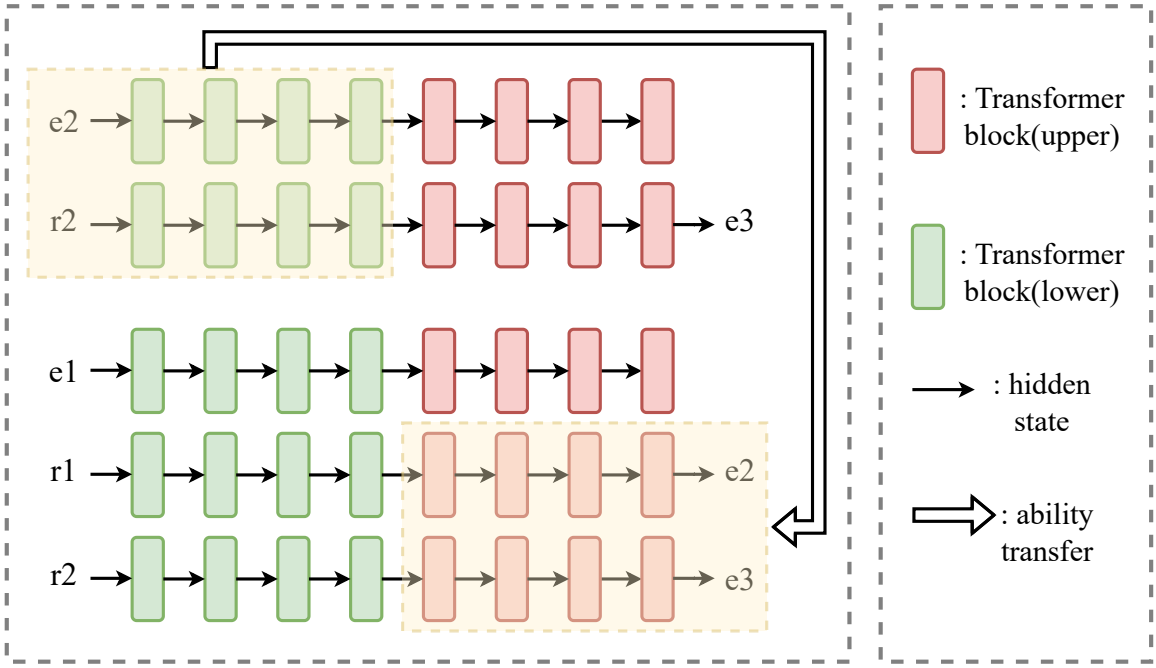


Figure 16: Visualization of how a model trained with a looped architecture transfers embedding-based atomic fact reasoning learned in lower layers to higher layers. The bottom part illustrates the reasoning process for a two-hop instance, while the top part shows the embedding-based reasoning process corresponding to the second hop of the same instance.

success of shared-parameter training stems from enabling both the upper and lower layers of the model to perform embedding-based reasoning over atomic facts.

H Detailed Evidence on Alignment under Looped Training

This section supplements Section 5.3. We provide a detailed description of how the results in Section 5.3 are computed and present additional evidence for *Representation–Input Alignment under Looped Training*. In Section 5.3, we only report the alignment results for (i) $(h_4(r_1), h_4(r_2))$ vs. (E_{e_2}, E_{r_2}), and (ii) $(h_5(r_1), h_5(r_2))$ vs. $(h_1(e_2), h_1(r_2))$. Here, we extend this analysis by reporting cosine similarities between hidden states across a broader range of layer pairs.

We first describe how the results in Section 5.3 are computed. For each two-hop instance from the training set (Train-II), we identify the atomic fact corresponding to its second hop, which is in-distribution (ID). For each two-hop instance from the test set (Test-IO), we identify the atomic fact corresponding to its second hop, which is out-of-distribution (OOD). This matching strategy allows us to examine whether the model reuses embedding-level representations of atomic facts when performing the second hop of reasoning, as illustrated in Figure 16.

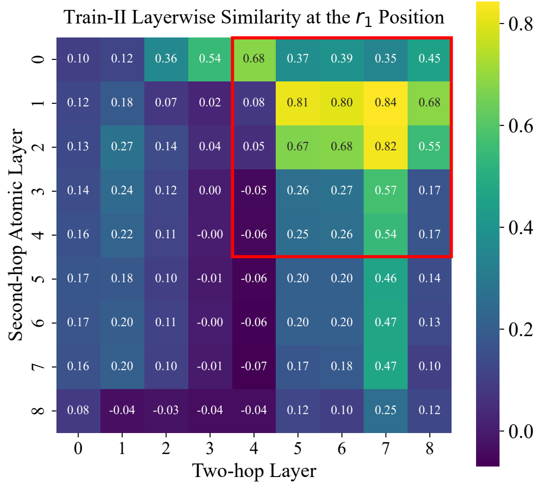
Specifically, for a two-hop query $\langle e_1, r_1, r_2 \rangle$, we extract the hidden states at the position of r_1 from all Transformer layers, and compare them with the hidden states at the position of e_2 from the corresponding second-hop atomic fact $\langle e_2, r_2 \rangle$. We compute a layer-wise cosine similarity matrix of size $(L+1) \times (L+1)$, where L denotes the number of Transformer layers[6]. Similarly, we extract the hidden states at the position of r_2 in $\langle e_1, r_1, r_2 \rangle$ and compare them with the hidden states at the position of r_2 in the atomic fact $\langle e_2, r_2 \rangle$, again computing a $(L+1) \times (L+1)$ cosine similarity matrix.

Finally, we average the cosine similarity matrices over all matched two-hop instances to obtain the final mean cosine similarity matrices. We present the averaged similarity matrices separately for Train-II and Test-IO, and compare models trained with the looped architecture against normally trained models. Figure 17 visualizes the resulting mean cosine similarity matrices.

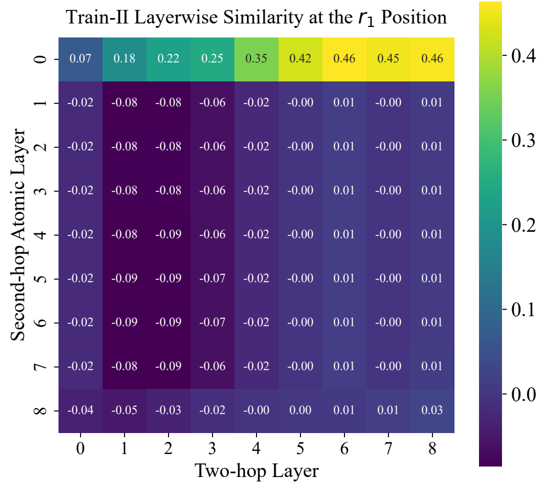
From Figure 17, it is clear that for models trained

[6]Layer 0 corresponds to the input embedding without positional encoding.

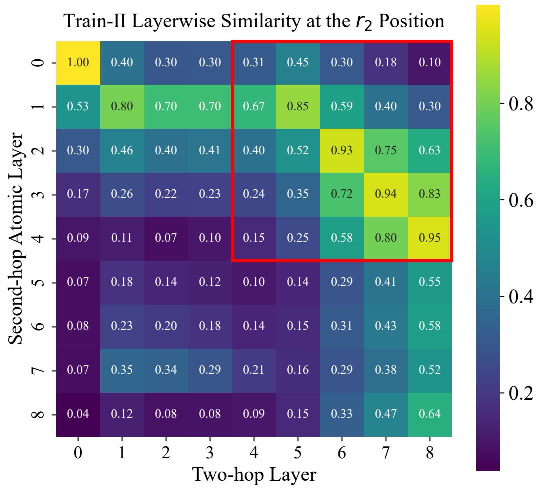
under the looped architecture (left column), the cosine similarity matrices at positions r_1 and r_2 for both Train-II and Test-IO ((a), (c), (e), and (g)) show significantly higher values along the diagonal directions within the red-square-highlighted regions. Notably, the entries at position (1,5) (row, column) exhibit particularly strong similarity across all subfigures. In contrast, models trained in the standard manner do not display this pattern. This indicates that for models trained under the looped architecture, the implicit second-hop reasoning in the two-hop queries is aligned with the reasoning over the corresponding single-hop atomic facts (based on embeddings), achieving strong alignment at the fifth layer. Under this input alignment condition, the model can naturally transfer the reasoning ability over out-of-distribution atomic facts learned in lower layers to higher layers, enabling effective generalization to both Test-IO and Test-OO.


Train-II Layerwise Similarity at the r_1 Position
Second-hop Atomic Layer
Two-hop Layer
0.0 0.2 0.4 0.6 0.8

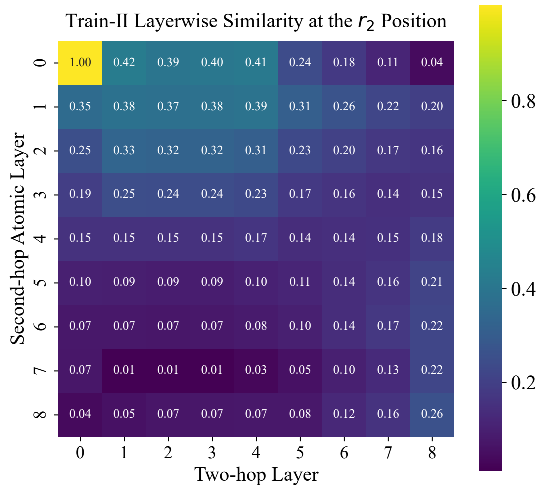
(a)


Train-II Layerwise Similarity at the r_1 Position
Second-hop Atomic Layer
Two-hop Layer
0.0 0.1 0.2 0.3 0.4

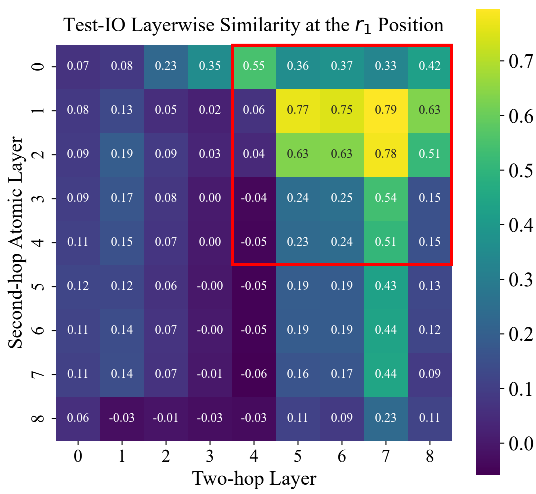
(b)


Train-II Layerwise Similarity at the r_2 Position
Second-hop Atomic Layer
Two-hop Layer
0.2 0.4 0.6 0.8

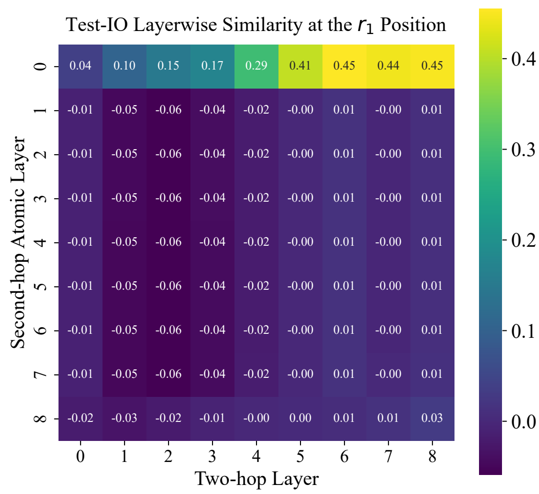
(c)


Train-II Layerwise Similarity at the r_2 Position
Second-hop Atomic Layer
Two-hop Layer
0.2 0.4 0.6 0.8

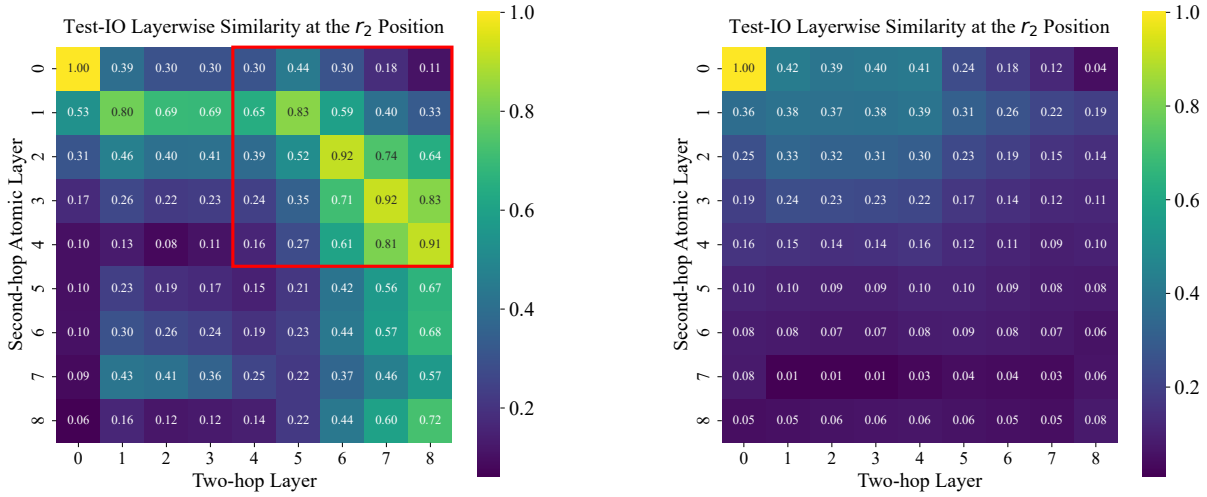
(d)


Test-IO Layerwise Similarity at the r_1 Position
Second-hop Atomic Layer
Two-hop Layer
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

(e)


Test-IO Layerwise Similarity at the r_1 Position
Second-hop Atomic Layer
Two-hop Layer
0.0 0.1 0.2 0.3 0.4

(f)



(g) (h)

Figure 17: Layerwise cosine similarity heatmaps comparing models trained under a recurrent architecture (left column) and models trained in the standard manner (right column). For each two-hop query $\langle e_1, r_1, r_2 \rangle$, hidden states at the position of r_1 are compared with the corresponding hidden states at e_2 in the atomic fact $\langle e_2, r_2 \rangle$, and hidden states at the position of r_2 are compared with the hidden states at r_2 in the atomic fact. Subfigures (a) and (b) show Train-II at position r_1, (c) and (d) show Train-II at position r_2, (e) and (f) show Test-IO at position r_1, and (g) and (h) show Test-IO at position r_2.