A Study of Modus Ponens in Transformer Models

Paulo Pirozelli PAULO.SILVA@MAUA.BR
Instituto Mauá de Tecnologia
Universidade de São Paulo

Fabio G. Cozman FGCOZMAN@USP.BR
Universidade de São Paulo

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

Transformer models are the backbone of modern natural language processing. However, whether they can truly perform logical reasoning remains uncertain. This paper examines transformers' capacity for logical inference in a controlled setting, isolating a single rule—*modus ponens*—and eliminating confounding factors such as semantic knowledge and linguistic complexity. We systematically vary architectural components, specifically the number of attention heads and layers, to assess their impact on logical inference. Our results show that attention heads enhance information propagation, whereas deeper architectures accelerate convergence but also introduce potentially redundant parameters. While transformers successfully handle level-2 inference tasks, their difficulties with higher-level and out-of-distribution problems suggest that they rely on heuristic "shortcuts" rather than explicit multi-step reasoning. Analysis of attention maps and ablation experiments indicates that these shortcuts function similarly to a *matching-aggregation* algorithm, where attention heads identify inference links, and the feed-forward network verifies if they form a valid chain. These findings highlight fundamental limitations in transformers' ability to perform structured logical reasoning.

Keywords: Transformers, Logical Reasoning, Modus Ponens, Mechanistic Interpretability

1. Introduction

Transformer models are the dominant neural architecture in modern natural language processing (Vaswani et al., 2017), driving state-of-the-art large language models such as GPT-4 (Achiam et al., 2023) and Llama 2 (Touvron et al., 2023). These models demonstrate remarkable performance across diverse tasks, whether through fine-tuning or in-context learning, often producing responses that seem to reflect complex reasoning (Bubeck et al., 2023). However, it remains unclear whether transformers genuinely engage in reasoning or merely approximate it by leveraging statistical patterns. Understanding the nature of their reasoning abilities is crucial for developing methods that enhance coherence and reliability in their outputs.

Most studies on reasoning in transformers focus on models trained on natural language, i.e., language models (Han et al., 2022; Tian et al., 2021; Clark et al., 2020). This poses a significant challenge, as it becomes difficult to disentangle genuine reasoning from stored knowledge or a combination of both. In this paper, we investigate a highly constrained logical task—*modus ponens*—using a synthetic dataset specifically designed to isolate this rule. The dataset employs symbols devoid of semantic content, ensuring that reasoning is assessed independently of linguistic priors or memorization. Furthermore, to remove any influence of pre-trained knowledge, we train all models from scratch on this dataset. Our goal is to determine whether transformers can engage in pure logical reasoning and what strategies they use to solve logical problems. The key contributions of this paper are:

- A synthetic dataset for logical reasoning that isolates a single operation—*modus ponens*—allowing precise control over the number and sequence of inferential steps.
- A series of interpretability experiments that analyze the internal mechanisms of transformers and their strategies for solving logical reasoning tasks.

All data and code are available in our GitHub repository. [1]

2. Related Works

When tested on natural language datasets, transformer models appear capable of solving logical reasoning tasks (Clark et al., 2020; Tian et al., 2021; Han et al., 2022). However, evidence suggests that they do not truly master logical reasoning. Instead, they tend to rely on statistical artifacts to infer theorems rather than employing general, rule-based procedures (Zhang et al., 2022). Moreover, their reasoning is brittle when confronted with abstract content or information that conflicts with prior knowledge (Dasgupta et al., 2022; Tang et al., 2023). Transformers also struggle to structure logical proofs (Saparov and He, 2023) and face significant challenges with multi-step and compositional reasoning (Rae et al., 2021; Dziri et al., 2023; Press et al., 2023).

One way to better study a complex phenomenon such as logical reasoning is to isolate it from other possible influences. This can be achieved by creating algorithmically generated datasets that focus on a single phenomenon, minimizing confounding factors (e.g., Power et al. 2022; Hanna et al. 2023). Our dataset follows a similar methodology to (Zhang et al., 2022) and (Clark et al., 2020) but imposes even stricter constraints, isolating a single rule—*modus ponens*—while deliberately excluding conjunction, negation, syntactic structure, and natural language templates. This simplified dataset is then used to explore the internal mechanisms of a transformer model, trained from scratch, as it attempts to solve the logical reasoning task.

Transformers' reasoning capabilities have been analyzed through various approaches, including behavioral evaluation (Clark et al., 2020; Han et al., 2022; Tian et al., 2021) and structural probing (Pirozelli et al., 2024). This study aligns with the field of mechanistic interpretability, which seeks to reverse-engineer neural networks (Cammarata et al., 2020; Wang et al., 2022; Kantamneni and Tegmark, 2025). Prior work has investigated model circuits (Elhage et al., 2021), causal tracing (Meng et al., 2022; Heimersheim and Nanda, 2024), and logit lens (Wang et al., 2024) to uncover the internal mechanisms of transformers. Our focus is predominantly on transformers' architecture and the role of their components in solving a constrained logical task.

3. Data

Reasoning can take many forms—such as causal, temporal, and mathematical—each offering a unique perspective on how conclusions are drawn. Among these, logical reasoning appears to be the most fundamental form of inference, as it does not depend on any particular subject matter. Traditionally, logical reasoning is divided into deductive and non-deductive forms, depending on whether the conclusion necessarily follows from the premises or holds with some degree of probability or plausibility, as in inductive and abductive reasoning (Groarke, 2024).

In what follows, we focus on the most fundamental deductive rule, ***modus ponens***—if p is true and p implies q, then q. Problems in our dataset involve the repeated application of this single

1. https://github.com/paulopirozelli/logicalreasoning.

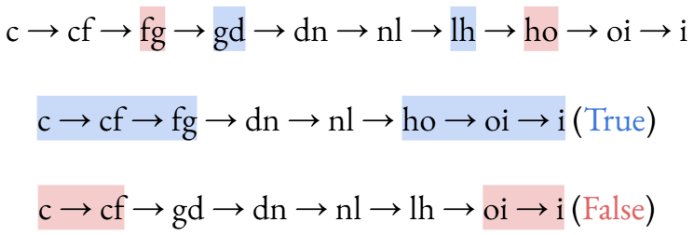

Figure 1: Example of a level-2 inference problem. The first row illustrates the original chain of rules, from which positive and negative examples are derived by removing specific rules. The second row (True) allows two consecutive applications of *modus ponens*, whereas the last row (False) permits only a single inference step.

rule. Even so, the inference process can become complex, as multiple paths may lead to the same conclusion, framing it as a search problem in an argumentative graph. To further simplify this setting, we constrain problems to linear chains of implications, ensuring a single valid deduction path. This setup significantly reduces problem complexity. With only one valid deduction path, the conclusion can be reached through a straightforward, sequential procedure.

Another challenge in determining whether transformer-based models can genuinely reason lies in their exposure to vast datasets, making it difficult to disentangle inference from stored knowledge. For instance, consider the argument: "All mammals are vertebrates. All dogs are mammals. Are dogs vertebrates?" Even if a model provides the correct answer, it is unclear whether this results from explicit reasoning (dogs → mammals → vertebrates), direct memorization, or indirect pattern recognition (e.g., an analogy with cats). To address this, we construct a synthetic dataset designed to isolate logical implication from linguistic structure and commonsense knowledge.

We define the **inference level** (I) of an argument as the minimal number of *modus ponens* applications required to distinguish positive from negative examples. An argument is labeled True if it permits exactly I applications of modus ponens starting from both the initial and final facts, and False if it permits at most $I - 1$. Figure 1 illustrates this concept for a level-2 inference problem. In the figure, letters c and i represent the initial and final facts, respectively, while each pair of letters denotes an inferential rule (e.g., cf means "c implies f"). The middle row illustrates a True example, where two consecutive applications of *modus ponens* can be performed, starting from the initial fact: c, $c \to f, f \to q$. In contrast, the last row represents a False example, where only one inference step is possible: c, $c \to f$, but no further conclusions can be drawn beyond this point. The same logic applies in reverse, starting from the final fact i.

3.1. Dataset Construction

The dataset is generated as follows. First, a letter is randomly sampled from the alphabet as the initial fact, which also serves as the antecedent of the first rule. This letter is then removed from the alphabet. Next, another letter is sampled as the consequent of the first rule. The process repeats iteratively: each newly chosen consequent becomes the antecedent of the next rule, ensuring a continuous chain of logical implications. This continues until a base chain with $R = 3I + 2$ rules is formed, after which the final fact is appended (first row in Fig. 1).

Both positive and negative examples are then created by selectively removing rules from the chain. In True examples, rules I and $R - I$ are removed, leaving exactly I inferential steps from both ends. In False examples, rules $I + 1$ and $R - I - 1$ are deleted, ensuring only $I - 1$ steps remain. After that, the rules are shuffled while ensuring that the order of the letters remains intact. The initial and final facts remain fixed in this process. The rules are then shuffled, preserving letter order, and each instance is represented as a concatenated string prefixed with *S*. For example, Sceynpybpzvscvz (True) and Smwomysjkqoggsq (False).

Each dataset instance, regardless of label, exhibits the following properties:

- $3I$ rules, derived from the original $3I + 2$ rules in the base chain after rule deletion;
- 1 initial fact, serving as the starting point for inference;
- 1 final fact, serving as the endpoint for inference;
- 3 letters never serving as consequents;
- 3 letters never serving as antecedents;
- $(I + 1) \times 2$ letters appearing as both antecedents and consequents in different rules.

Hence, all features in the dataset arise exclusively from the random sampling of letters during rule construction and the subsequent shuffling of the rules.

The total number of possible samples for inference level I is:[2]

$$samples = 2 \times P(26, 3I + 3) \times (3I)! \tag{1}$$

For this study, we generated a level-2 inference dataset consisting of 900K training samples, along with 50K additional instances for validation and testing. The dataset is perfectly balanced, with an equal number of samples of each label. General statistics of the dataset are provided in Appendix A. For an inference level of 2, the total number of possible samples is approximately 1.13×10^{12}. In contrast, our training set contains only 9×10^5 instances, with an additional 5×10^4 allocated for validation and testing. This vast discrepancy makes direct memorization or simple interpolation highly unlikely.

4. Transformers can solve *easy* logical problems

A key property of our classification task is that it admits an explicit algorithmic solution, outlined in Appendix B. This solution relies on a recurrent approach. However, transformers lack recurrence by design. This raises an important question: despite their lack of recurrence, can transformers effectively solve such problems?

Thus, we investigate whether transformer models can successfully solve constrained logical problems and how they organize reasoning. Our analysis focuses on the level-2 inference dataset

2. The **base chain**, representing the number of possible logical chains before rule deletion, is given by $P(26, 3I + 3)$, where $P(n, k)$ denotes the number of ways to sample k distinct elements from an alphabet of size n. The term $3I + 3$ arises because we need $3I + 2$ rules in the base chain, and each rule introduces a new element, plus one additional element for the final fact. The **factor of 2** accounts for the two possible labels. Finally, the **permutations** term $(3I)!$ reflects the number of possible orderings of the $3I$ rules after rule deletion.

described in the previous section.[3] Specifically, we examine the architectural requirements—such as the number of layers and attention heads—needed for transformers to accurately follow a reasoning chain.

Model Setup. We build several BERT models with varying numbers of layers and attention heads. As in a standard BERT architecture, we use an embedding size of 768, layer normalization, and a feedforward network with an intermediate layer four times the embedding dimension. ReLU is used as the activation function, and positional embeddings are added before the transformer blocks (Vaswani et al., 2017).

Each token is encoded as a 26-dimensional one-hot vector (one dimension per letter in the Latin alphabet), forming an input sequence of 15 tokens. For classification, a linear projection is attached to the start-of-sequence token, mapping its 768-dimensional embedding to a scalar output, which is then passed through a sigmoid function. This output is compared with the ground-truth labels for evaluation.

Training Details. We use a dropout rate of 0.1, a learning rate of 1×10^{-5}, and a batch size of 2048, which consistently yields stable convergence. Adam is used as the optimizer, with a one-epoch warm-up phase where the learning rate increases linearly from 0 to 1×10^{-5}. After warm-up, the learning rate decays linearly until reaching zero in the final epoch, up to a maximum of 1,000 epochs. The task is considered solved once the model achieves over 99% validation accuracy, after which it is evaluated on the test set. To estimate variability, each model configuration is trained ten times.

Single-Layer Models. Figure 2 shows the maximum validation accuracy attained by single-layer models with varying attention heads. Performance improves with more attention heads, with models exceeding 99% accuracy at 48 heads. This suggests that transformers solve this problem by efficiently propagating information **horizontally** across the input sequence.

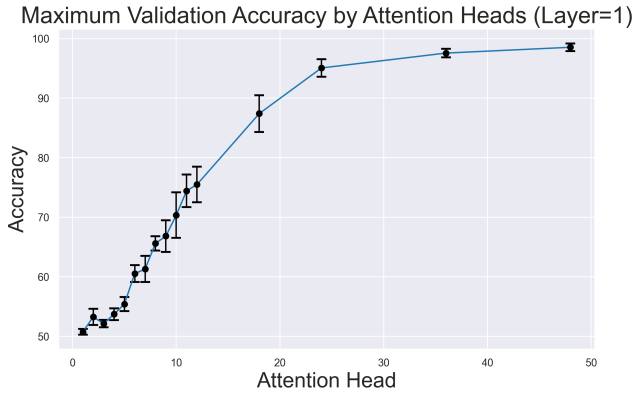

Figure 2: Maximum accuracy for single-layer models with varying attention heads. Each configuration was trained 10 times; error bars show confidence intervals.

3. A level-1 inference problem reduces to a simple lookup table.

Multiple-Layer Models. We also trained models with 2–6 layers and 1–6 attention heads, using the same training setup. Overall, 97.38% of these models converged within 1,000 epochs.[4] Figure 3 shows the number of epochs required for convergence. Deeper models generally converged faster (Wang et al., 2024), though gains diminished beyond five layers. Moreover, increasing the number of attention heads sped up convergence, especially in shallower models. These results suggest that transformers leverage both **horizontal** and **vertical** information flow to solve the task.

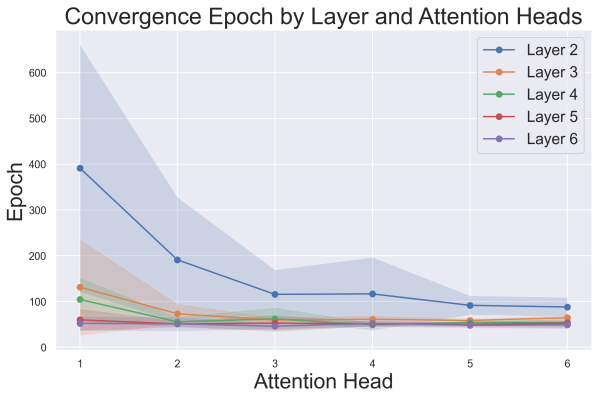

Figure 3: Epochs required for convergence in a level-2 inference task, across models with 2–6 layers and 1–6 attention heads. Each configuration was trained 10 times.

Grokking. The convergence process consistently followed a characteristic pattern: training accuracy gradually increased, and then, after prolonged training, both training and validation metrics abruptly surged to near-perfect levels. This sharp transition, known as grokking (Power et al., 2022), reflects a shift from memorization to strong generalization. Similar behavior has been observed in other tasks involving compositional reasoning (Wang et al., 2024). Examples of this and other convergence patterns are provided in Appendix C.

Conclusion. Single-layer models needed more attention heads to converge, indicating that horizontal information flow can compensate for shallow depth. In models with 2–4 layers, additional heads accelerated convergence, highlighting an interaction between horizontal and vertical flows. However, beyond five heads, additional heads provided no benefit, suggesting that vertical processing dominates in deeper models.[5]

5. Transformers *do not* make inferential steps

Transformer models can solve logical reasoning problems, but do they implement an algorithm akin to the one described in Appendix B? Two key observations suggest otherwise: their inability to generalize to out-of-distribution (OOD) data and their failure on level-3 problems.

4. Only models with 2 layers failed to converge. The fastest convergence, in 30 epochs, was achieved by a model with 6 layers and 2 attention heads.
5. Appendix D explores the trade-off between vertical and horizontal expansion in terms of parameter efficiency, using pruning techniques.

5.1. OOD Generalization

To assess whether transformers had truly learned a general solution, we designed a constrained OOD scenario. Specifically, we evaluated their ability to solve a level-2 inference problem for an unseen token. To achieve this, we filtered the original dataset for a single letter—*p*—excluding all instances containing this letter from the training set while retaining only instances with *p* in the validation and test sets. This filtering procedure reduced the dataset to 622K examples (588K for training, 17K for validation, and 17K for testing).

Results. We trained five models with 3 layers and 4 attention heads on the filtered dataset. Although some achieved up to 93% accuracy on the validation set, none fully solved the task involving the unseen token. This performance indicates a failure to generalize fully. This suggests that the models adopt a strategy distinct from our general algorithm.

5.2. Higher-Level Inference

We investigated whether transformers could handle logical problems requiring additional inference steps. To this end, we tested level-3 inference tasks, exemplified by 21-token sequences like `Shexoehvbnrogwpgupvbw` (True) and `Sjvlqzxtmyovlrzwrmjxw` (False). We experimented with various hyperparameters, including different numbers of layers, heads, learning rates, and optimization strategies. Models were trained for up to 10,000 epochs while monitoring validation accuracy. Despite achieving near-perfect accuracy on the training set, they consistently failed to generalize to the validation set.

Hypothesis. A likely explanation is that models adopt *shortcut* strategies—collapsing multi-step reasoning into a single, shallow step—instead of building explicit inference chains (Liu et al., 2023).

6. Partial Deductions

We hypothesize that transformer models do not execute a step-by-step chain of reasoning via explicit *modus ponens*. Instead, they approximate full inference chains through a **matching-aggregation** mechanism. In this process, attention heads identify individual inference links (e.g., $p \to q$, $q \to r$), while the feed-forward network verifies that these links form a valid chain. This procedure requires the model to locate individual rules and to memorize both positive and negative combinations of them. Appendix E provides additional details on this theoretical framework.

In a toy scenario where each attention head encodes precisely one rule (along with 26 heads each for initial and final facts), solving a level-2 inference problem would require 702 heads. However, real transformer models require far fewer heads due to the distributed nature of their learned representations: each head can capture multiple, often related, patterns. Rather than enforcing a rigid one-to-one mapping between heads and inference rules, the model compresses rule detection across multiple heads. In our experiments, models used at most 48 heads—substantially fewer than the naïve estimate. Pruning experiments further support this observation: removing a large fraction of model weights had minimal impact on performance (Appendix D). This suggests that transformer models do not depend on narrowly specialized heads. Instead, multiple heads contribute overlapping, often redundant information, improving robustness to pruning and enhancing generalization.

Multi-layer architectures also enable vertical distribution of rule detection. Earlier layers may identify some inference links, while later layers capture others. As a result, a multi-layer model can

spread the burden of rule detection across layers, reducing the need for many specialized attention heads within each layer compared to a single-layer model.

Beyond detecting individual rules, the model must also encode all valid and invalid rule combinations. In a level-2 inference task, the number of possible combinations is given by:

$$26 \times 650 \times 649 \times 2 = 21{,}936{,}200 \tag{2}$$

Here, 26 corresponds to the number of initial facts; 650, the number of possible first rules ($m \times (m-1)$); 649, the number of possible second rules; and the factor of 2 accounts for both forward and backward deductions.

A more efficient solution might restrict inference to a single direction and consider only the antecedent of the second rule, yielding:

$$26 \times 650 \times 25 = 422{,}500 \tag{3}$$

Nonetheless, even under this simplification, solving a level-3 inference problem would involve handling $26 \times 650 \times 649 \times 25 = 274{,}202{,}500$ distinct combinations—underscoring the intractability of exhaustive rule enumeration for problems of this kind.

7. Attention Maps

To analyze the implementation of the partial deduction strategy, we examine the model's attention maps. Each model learns a distinct solution. As is common with attention maps, the reasoning process is not always fully transparent. Therefore, we focus on two models whose deduction strategies are relatively clear.

We use **F** (*forward*) and **B** (*backward*) to denote deduction-relevant rules, and **N** (*none*) for irrelevant rules. **F0** and **B0** represent the initial and final facts, respectively. Numerical indices indicate a rule's position in the deduction sequence, while **(a)** and **(c)** specify the antecedent and consequent of a rule. If no clear pattern emerges, we label it as **NAN**. For instance, given the input `Srfmrtmpaytwybp`, the rules can be rearranged as follows:

- **Forward Deduction:** $\underset{\textbf{\textit{F0}}}{r}, \underset{\textbf{\textit{F1}}}{r \to t}, \underset{\textbf{\textit{F2}}}{t \to w}$
- **Irrelevant Rules:** $\underset{\textbf{\textit{N1}}}{a \to y}, \underset{\textbf{\textit{N2}}}{y \to b}$
- **Backward Deduction:** $\underset{\textbf{\textit{R2}}}{f \to n}, \underset{\textbf{\textit{R1}}}{n \to p}, \underset{\textbf{\textit{R0}}}{p}$

3 Layers, 1 Attention Head From the attention map, we observe that the first layer consistently focuses on the **third rule** (located at positions 7–8 in the input sequence). Rather than starting from either end of the sequence, the model initiates reasoning from this rule and follows its connections. Given that four out of the six rules are relevant for deduction (either forward or backward), the model has a **67% chance** of selecting a useful starting point.

The logical path the model follows depends on the initial rule it extracts. When the model starts with a relevant rule (**F1, F2, B1, B2**), the attention maps reveal the following sequences, progressing from the first to the third layer:

Case 1: F1 → F0 + F2(a)* → F0
Case 2: F2 → F1(c) → F0
Case 3: B1 → B0 → B2(c)
Case 4: B2 → B1(a) → NAN

The transition made by the model when starting with the third rule does not always follow the same direction. Sometimes, it progresses to the next rule in the deduction (e.g., **B1** → **B0**), while in other cases, it backtracks (e.g., **B2** → **B1**). An asterisk (*) denotes instances where a rule or token exhibited only weak activation. Figure 4 illustrates examples of the Case 4 and Case 2 strategies.

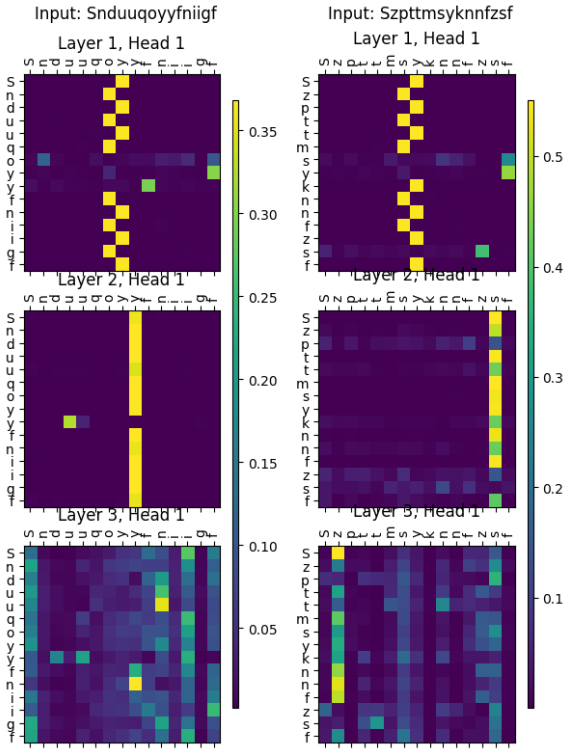

Figure 4: Attention maps for a 3-layer, 1-head model, illustrating Case 4 and Case 2 strategies, respectively.

If the model starts from an **irrelevant rule** (**N1, N2**), the attention heads follow these paths:

Case 5: N1 → N2(a) → NAN
Case 6: N2 → N1(c) → NAN

Since these paths lack the final step, the model cannot determine whether the example is positive or negative based solely on this approach. Thus, it must be relying on alternative mechanisms to reach a conclusion.

2 Layers, 6 Attention Heads For this model, the attention maps indicate a different deduction strategy. In the first layer, **Attention Head 6** focuses on the **fourth rule**. If this rule is relevant to the deduction, **Attention Heads 3 and 5** attend to the connected rule, following this pattern:

Case 1: F1 $\rightarrow$ F2(a)
Case 2: F2 $\rightarrow$ F1(c)
Case 3: B1 $\rightarrow$ B2(c)
Case 4: B2 $\rightarrow$ B1(a)

Discussion. Attention mechanisms remain crucial to model performance, even when their individual contributions are not directly interpretable. For example, the first model we analyzed successfully solves problems even when starting with irrelevant rules (Case 4). This behavior suggests that attention may operate in a more distributed or diffuse fashion, enabling the model to integrate information in ways not fully captured by attention maps. Alternatively, it may indicate that other architectural components—such as residual connections—also play a significant role. [6]

8. Conclusion

Logical reasoning is a cornerstone of human intelligence, and endowing AI models with this capability is crucial for their advancement. In this work, we introduced a synthetic dataset centered on deductive reasoning—specifically, the *modus ponens* rule—designed to precisely control the inference steps required for each problem. By focusing on tasks with well-defined algorithmic solutions, our goal was to assess whether transformer-based models could discover and replicate such reasoning processes.

We trained various transformer architectures to address these tasks and found that both the number of attention heads and the number of layers significantly impact model performance. Attention heads were particularly important for enabling convergence in shallower models, while deeper architectures benefited from faster convergence. Our results suggest that although only a subset of the model's parameters directly contributes to logical reasoning, the overall parameter count remains closely tied to architectural depth and width.

Experiments involving level-2 inference problems—and the models' consistent failure on OOD and level-3 inference tasks—indicate that these models are likely not engaging in genuine step-by-step reasoning. Rather, they appear to exploit statistical shortcuts. Through attention map analysis and ablation studies, we found evidence that transformers implement a form of **matching-aggregation** mechanism: attention heads identify applicable rules, and the feedforward layers aggregate the relevant information. The failure of transformers to perform true multi-step inference highlights potential limitations of this architecture for tasks requiring sequential reasoning.

Acknowledgements

F. G. Cozman is partially supported by CNPq grant Pq 305753/2022-3. The authors would like to thank the Center for Artificial Intelligence (C4AI-USP), with support from the São Paulo Research Foundation (FAPESP) grant 2019/07665-4, and from the IBM Corporation. This study was financed, in part, by the São Paulo Research Foundation (FAPESP), Brasil. Process Number #2019/26762-0.

6. Appendix F presents an ablation study exploring the role of each component in the task.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023. URL https://arxiv.org/abs/2303.12712.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. Thread: circuits. *Distill*, 5(3):e24, 2020.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*, 2020.

Ishita Dasgupta, Andrew K. Lampinen, Stephanie C. Y. Chan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. Language Models Show Human-Like Content Effects on Reasoning, 2022. URL https://arxiv.org/abs/2207.07051.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and Fate: Limits of Transformers on Compositionality, 2023.

Nelson Elhage et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

Leo Groarke. Informal logic. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2024 edition, 2024. URL https://plato.stanford.edu/archives/spr2024/entries/logic-informal/.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, et al. FOLIO: Natural Language Reasoning with First-Order Logic. *arXiv preprint arXiv:2209.00840*, 2022.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model, 2023.

Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition, 2025. URL https://arxiv.org/abs/2502.00873.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Paulo Pirozelli, Marcos M José, Paulo de Tarso P. Filho, Anarosa AF Brandão, and Fabio G Cozman. Assessing logical reasoning capabilities of encoder-only transformer models. In *International Conference on Neural-Symbolic Learning and Reasoning*, pages 29–46. Springer, 2024.

Alethea Power, Yuri Burda, Harrison Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *CoRR*, abs/2201.02177, 2022. URL https://arxiv.org/abs/2201.02177.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *CoRR*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446.

Cody Rushing and Neel Nanda. Explorations of self-repair in language models. *arXiv preprint arXiv:2402.15390*, 2024.

Abulhair Saparov and He He. Language Models Are Greedy Reasoners: A Systematic Formal Analysis of Chain-of-Thought, 2023.

Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. Large Language Models are In-Context Semantic Reasoners rather than Symbolic Reasoners, 2023.

Jidong Tian, Yitian Li, Wenqing Chen, Liqiang Xiao, Hao He, and Yaohui Jin. Diagnosing the First-Order Logical Reasoning Ability Through LogicNLI. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3738–3747, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.303. URL https://aclanthology.org/2021.emnlp-main.303.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

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

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. On the Paradox of Learning to Reason from Data. *CoRR*, abs/2205.11502, 2022. doi: 10.48550/arXiv.2205.11502. URL https://doi.org/10.48550/arXiv.2205.11502.

Appendix A. Data Distribution

Figure 5 shows the distribution of letter by position in the input string of the training set.

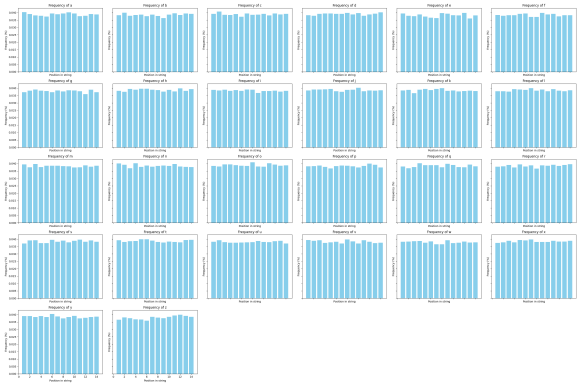

Figure 5: Frequency of letter by position in the input string. Training set.

Figure 6 shows there is almost no correlation between the letters that appear in strings in the training data; and that this correlation is smaller in magnitude for the labels.

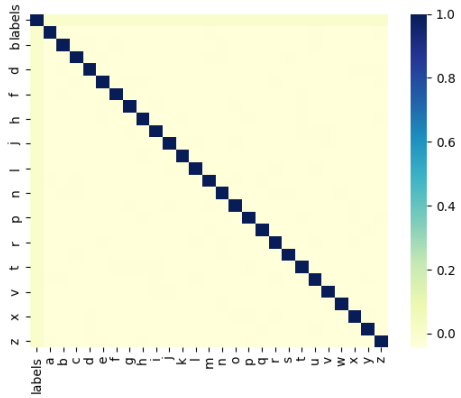

Figure 6: Correlation between the letters and labels of training data.

Appendix B. Algorithmic Solution

A key property of our classification task is that it admits an explicit algorithmic solution, outlined in Algorithm 1.footnoteFor simplicity, we ignore the start-of-sequence token. The algorithm runs in $O(I \cdot m)$ time, where I is the inference level (the number of loop iterations) and m is the length of the list L.

Algorithm 1 Inference Algorithm Pseudocode

Data: List L, inference level I
Result: Boolean label
$consequent\_pos \leftarrow 0$
$consequent \leftarrow L[consequent\_pos]$
$L[consequent\_pos] \leftarrow$ None
for $i \leftarrow 1$ **to** $I - 1$ **do**
  $antecedent \leftarrow consequent$
  $antecedent\_pos \leftarrow L$.index$(antecedent)$
  $L[antecedent\_pos] \leftarrow$ None
  $consequent\_pos \leftarrow antecedent\_pos + 1$
  $consequent \leftarrow L[consequent\_pos]$
  $L[consequent\_pos] \leftarrow$ None
end
if $consequent \in L$ **then**
  $label \leftarrow$ True
end
else
  $label \leftarrow$ False
end

B.1. Time and Space Complexity Analysis

Notation. Let I be the inference level (number of reasoning steps), and let m be the length of the input list L. The list contains $3I$ rules and two atomic facts (start and goal), with each rule encoded as two characters. Hence:

$$m = 6I + 2 = \Theta(I). \tag{4}$$

Initialization (Lines 1–3). These operations involve:

- Assigning scalars and retrieving $L[0]$: $\Theta(1)$
- Setting $L[0] \leftarrow$ None: $\Theta(1)$

Total: $\Theta(1)$

Loop Body (Lines 4–10). Executed for $I - 1$ iterations. Each iteration includes:

- One `index()` lookup: $\Theta(m)$
- A constant number of assignments and list accesses: $\Theta(1)$

Total: $\Theta(I \cdot m) = \Theta(I^2)$ (since $m = \Theta(I)$)

Final membership test.

- Checking whether the final consequent appears in the list: $\Theta(m)$.
- Assigning the final label: $\Theta(1)$.

Total: $\Theta(m) = \Theta(I)$

Overall Time Complexity.

- Dominated by the loop: $\boxed{\Theta(I^2)}$

Space Complexity.

- The list L is modified in-place: $\Theta(m) = \Theta(I)$
- Only a constant number of scalar variables are used: $\boxed{\Theta(I)}$

Appendix C. Model Convergence

Below, we show some typical patterns of model convergence observed in our experiments. In most cases, we observe a grokking behavior, where the validation accuracy suddenly jumps from random chance to perfect performance, indicating a transition from memorization to actual problem-solving. The cases differ in the distance between training and validation curves (Figs. 7, 8) and the number of eureka moments (Figs. 9, 10). We also note some cases of gradual learning (Figs. 11). It is important to note that these patterns were not discovered via a systematic analysis, but were instead observed through careful manual inspection of selected training runs.

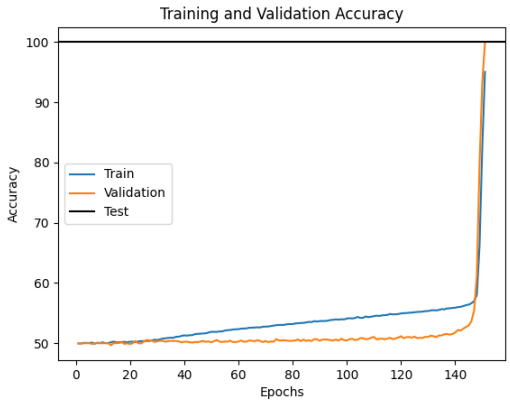

Figure 7: **Grokking**. Model with 3 layers and 4 attention heads.

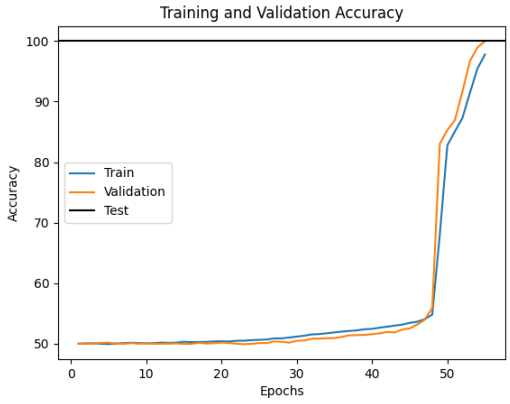

Figure 8: **Train and Validation Aligned**. Model with 3 layers and 4 attention heads.

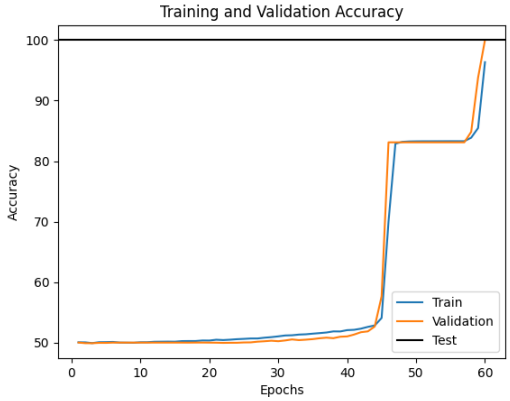

Figure 9: **Single Bump.** Model with 3 layers and 4 attention heads.

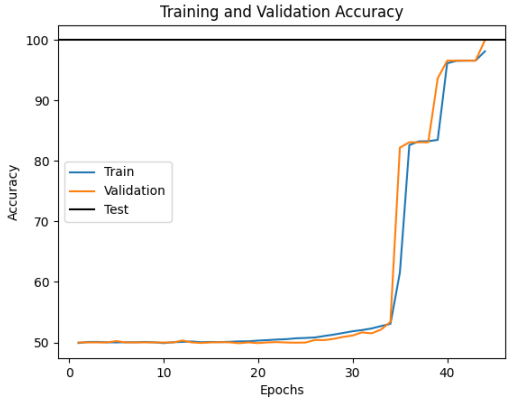

Figure 10: **Multiple Bumps.** Model with 6 layers and 6 attention heads.

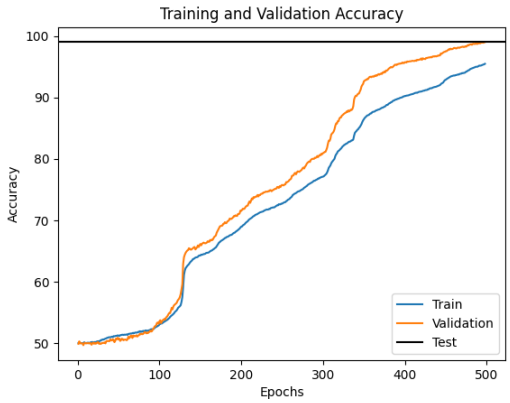

Figure 11: **Gradual Learning.** Model with 2 layers and 1 attention head.

Appendix D. Pruning

To further investigate the role of layers and heads in level-2 inference, we conducted a pruning experiment with two objectives. First, we aimed to quantify the minimum number of effective (non-zero) parameters required for optimal performance. Second, we examined whether our architectural insights—specifically, that fewer layers but more heads facilitate multi-step reasoning—hold under systematic pruning. We employed unstructured pruning, a technique that randomly eliminates individual weights instead of entire components (Cheng et al., 2024). Unstructured pruning is architecture-agnostic and straightforward to implement.

Training Setup. We experimented with transformer models spanning 2 to 6 layers and 1 to 6 attention heads.[7] We use a dropout rate of 0.1, a learning rate of 1×10^{-5}, and a batch size of 2048, which consistently yields stable convergence. Adam is used as the optimizer, with a one-epoch warm-up phase where the learning rate increases linearly from 0 to 1×10^{-5}. After warm-up, the learning rate decays linearly until reaching zero in the final epoch, up to a maximum of 1,000 epochs. After each epoch, we randomly set 0.1% of the weights to zero, permanently deactivating them. The network then underwent a full epoch (49 iterations at a batch size of 2048) to adapt to the newly pruned weights, leveraging its capacity for self-repair (Rushing and Nanda, 2024). Each model configuration was run five times.

Results. Despite the progressive deactivation of weights, the network demonstrated remarkable robustness. On average, models achieved near-perfect accuracy with only 4.4% of their original parameters ($SD = 0.96$). This suggests that a relatively small subnetwork is sufficient to match the full model's performance (Frankle and Carbin, 2019). The most parameter-efficient model, with 2 layers and 6 heads, required just 607,342 active weights—only 4.9% of the original 14,813,199. In terms of percentage, the most compressed model was a 4-layer, 6-head transformer, which retained just 3% of its weights.

To investigate how layers and heads influence these active subnetworks, we estimated a linear regression expressing the number of remaining (non-zero) weights W as a function of the number of layers L, the number of attention heads A, and their interaction, $L \cdot A$:

$$\begin{aligned} W = {} & 9.5 \times 10^5 + 7.2 \times 10^4 L - 1.7 \times 10^5 A \\ & + 3.8 \times 10^4 L \cdot A. \end{aligned} \tag{5}$$

Conclusion. Increasing the number of layers (L) raises the number of effective parameters required for solving the task (W), and this effect intensifies at higher attention head counts (A) due to their positive interaction. Conversely, the effect of attention heads on W depends on depth. For small L values (e.g., $L = 1$ or $L = 2$), the negative coefficient on A (-1.7×10^5) outweighs the positive interaction term ($3.8 \times 10^4 L \cdot A$), meaning that increasing A reduces W. However, once $L > 4.47$, the interaction term becomes dominant, making the overall effect of A over W positive. This finding aligns with our earlier observation that *horizontal* expansion (more heads) enhances efficiency in shallow models. These experiments reveal a fundamental tradeoff: **increasing the number of heads reduces parameter requirements, while greater depth accelerates learning**.

7. Models with a single layer proved difficult to converge, as seen in previous experiments.

Appendix E. Partial Deduction Algorithm

We show how a single-layer transformer can implement a solution to a two-level inference problem. This solution implements a **matching-aggregation** approach: attention heads detect specific patterns corresponding to logical rules, while the downstream feedforward network aggregates these signals to verify the inference.[8]

Suppose each token x is constructed by concatenating a letter one-hot vector with a position one-hot vector:[9]

$$x = [\text{token-hot} \,\|\, \text{position-hot}] \in \mathbb{R}^d.$$

Attention Pattern-Matching Assume we wish to detect the rule "$e \to b$" appearing in a specific pair of positions—say, positions 3 and 4. In our formulation, we design an attention head in which the query matrix Q is tuned to identify the antecedent (the letter e in position 3) and the key matrix K is tuned to detect the consequent (the letter b in position 4). For instance, if the columns of Q are set equal to the token embedding corresponding to e in position 3, then the dot product $Q \cdot x$ might yield:

- $+2$ if both the token is e and it is in position 3 (with each attribute contributing +1),
- $+1$ if either the token is e in a different position or the token in position 3 is not e,
- 0 otherwise.

An analogous construction applies to the key matrix K, which is tuned to output a positive value only when the consequent letter b appears in position 4. Consequently, when both conditions are met, the dot product $Q_i^\top K_j$ reaches its maximum value (in this case, $+2 \times +2 = +4$).

In practice, verifying whether an input instance satisfies the conditions for a two-step modus ponens inference would require three attention checks:

- The initial fact (x) (*forward*) or the final fact (y) (*backward*) appears in the second or last position, respectively;
- The rule $x \to y$ must be present among the candidate rules;
- The token y appears in an odd position greater than one (*forward*), which is reserved for rule antecedents, or the token x appears in an even position, which is reserved for rule consequents (*backward*).

A fully general, brute-force solution for a level-2 inference task might allocate heads as follows. One would need 26 heads to verify that a specific initial fact (one per letter) appears in its designated slot. To detect all possible rules $x \to y$, one could imagine assigning $26 \times 25 = 650$ heads (one for

8. Attention heads and feedforward computation could directly identify full inference patterns from the tokens, but this clearly produces a combinatorial explosion. It is computationally more efficient to recognize basic components first and aggregate them next. We could also imagine that, instead of single rules, the network first regconizes multistep sequences, e.g., e, $e \to b$, $b \to k$, and then simply verifies whether a given multistep rule is present. Again, this produces a much larger combinatorial space than the matching-aggregation algorithm we hypothesize.
9. In our models, we add both token and positional encodings. This is a simplification for exposition purposes. A similar demonstration could be made for the summation case.

each possible rule pair, assuming self-loops like $x \to x$ are excluded), plus an additional 26 heads to confirm that a candidate antecedent appears in its proper slot. Moreover, if backward deduction is considered (i.e., checking final facts and their corresponding positions, such as odd-numbered slots), an extra 26 heads might be required, yielding a total on the order of 702 distinct heads.[10]

Rule Aggregation. Once each attention head "activates" upon detecting its designated pattern, their outputs are aggregated and passed to a downstream feed-forward network. This network performs a logical combination—such as summing the positive signals and comparing the total to a threshold—to determine whether the overall inference is valid. One possible interpretation is that the attention mechanism handles the "matching" step, while the feed-forward layers integrate these signals to assess each potential chain of reasoning. These chains could follow structures like antecedent-rule-rule (e.g., $x, x \to y, y \to something$) or consequent-rule-rule (e.g., $something \to y, y \to x, x$).

For a level-2 problem, the number of necessary rule instances would be given by:

$$2 \times 26 \times 650 \times 649 = 21,936,200 \tag{6}$$

More generally, for a level-*I* problem, the number of possible rule combinations is:

$$\begin{aligned} &2 \times 26 \times R \times (R-1) \times \cdots \times (R-I) \\ &\quad = 2 \times 26 \times \frac{R!}{(R-(I+1))!} \end{aligned} \tag{7}$$

A simplification arises if one considers only a single direction, focusing solely on either the antecedent or consequent of the final rule. In this reduced form, a level-*I* problem allows:

$$\begin{aligned} &R \times (R-1) \times \cdots \times (R-I+1) \times m \\ &\quad = \frac{R!}{(R-I)!} \times m \end{aligned} \tag{8}$$

where *m* represents the vocabulary size.

Deeper Inferences. It is worth noting that while the above mechanism can be engineered to handle two-level inferences, extending it to support level 3 inferences (or deeper) presents significant challenges. As the depth of the inference increases, the number of possible combinations of initial facts, rules, and intermediate deductions grows combinatorially. This expansion would demand many more neurons in the dense layers than are typically available in standard transformer architectures, potentially explaining why models struggle with deeper logical reasoning tasks. This is perhaps why we were not able to train models to solve a 3-level task.

10. Higher-level inferences do not require more heads, since the combination of possible two-letter rules is still 650. That could change if a model considered multi-step rules (e.g., $x \to y \to z$). This would make the aggregation task lighter; however, the number of multi-step rules attended would be much larger, following $R \times (R-1) \times \cdots \times (R-k)$, where *k* represents the number of steps in a single rule. For instance, for a three-step rule (e.g., $x \to y \to z$), a model would need to identify $650 \times 649 \times 648 = 273{,}358{,}800$ rules. This would make the matching process significantly more complex.

Appendix F. Ablation

To better understand the information flow in our model, we conducted an ablation study, systematically removing specific components to assess their individual contributions to overall accuracy. The ablated components included attention heads (both layer-wise and individually), residual connections (before both the attention and feedforward networks), the feedforward network itself, entire transformer blocks, and positional embeddings.

Table 1 presents the results for a 3-layer, 4-attention-head model. The baseline corresponds to the full model with no components removed. In general, individual attention heads were not essential, except for Head 2 in the second layer, whose removal caused a significant drop in performance. Residual connections were crucial in the first two layers but had a lesser impact in the final layer. Interestingly, skipping the first transformer block did not entirely disrupt the model's ability to make predictions.

Component	Layer 1	Layer 2	Layer 3
Baseline		98.90	
Attention			
All	97.40	50.00	52.16
Head 1	98.39	98.89	98.55
Head 2	98.44	50.00	88.17
Head 3	98.90	98.88	91.53
Head 4	98.42	98.91	82.10
Residual C.			
First	51.93	53.22	92.40
Second	49.53	48.98	98.23
Feedforward	89.74	91.55	51.25
Transf. Block			
All		50.00	
Single	72.05	50.18	48.39
Pos. Emb.		49.54	

Table 1: Ablation of different components in a model with 3 layers and 4 attention heads. Test set accuracy is reported.

Through a qualitative assessment across multiple models, we identified the following key findings, highlighting how each component contributes to overall performance.

Attention Heads

Full Layer Removing all heads in a layer often led to accuracy dropping to chance level.

Individual Heads Single heads were usually non-critical, indicating that attention was distributed across multiple heads.

Head Sensitivity Models with fewer heads were more sensitive to head removal.

Feedforward Network

Shallow Models	Removing the feedforward network greatly reduced accuracy.
Deep Models	The contributions of feedforward layers were spread across multiple layers, making individual layers less crucial.

Other Findings

Positional Embeddings	Removing positional embeddings reduced performance to chance, highlighting their key role in encoding word order.
Transformer Redundancy	Some transformer blocks in deeper models were redundant and could be removed without significantly affecting performance.
Residual Connections	Residual connections were crucial in lower layers but had less impact in higher layers.