# On the Effect of Token Correlations on Semantic Transfer in Transformers

**Nevena Lazić**
Google DeepMind
nevena@google.com

**Julian Zimmert**
Google Research
zimmert@google.com

**Alan Malek**
Google DeepMind
alanmalek@google.com

**Tor Lattimore**
Google DeepMind
lattimore@google.com

**Csaba Szepesvári**
Google DeepMind
szepi@google.com

**András György**
Google DeepMind
agyorgy@google.com

## Abstract

We study issues which may arise when using transformer models to formally reason about problems stated informally using natural language. In particular, we focus on transfer learning between tasks that are semantically equivalent but phrased differently. We assume that examples with a particular *target label* are only available for a subset of tasks, and aim to transfer zero-shot the ability to predict this label to other tasks. Using controlled experiments on propositional logic problems, we show that transformers are capable of such transfer, but that it can be impeded by the presence of *task-specific tokens*, due to the model picking up on correlations between these tokens and the label early in the training. We observe a qualitatively similar (but weaker) effect when finetuning pretrained Gemma3 models. As a simple intervention, we show that introducing the target label later in the training dramatically improves transfer.

## 1 Introduction

Large language models (LLMs) built on the transformer architecture [Vaswani et al., 2017] are increasingly being used for solving problems in mathematics and science that require formal reasoning. These problems are typically presented to the models informally using natural language, which can contain many variations and even ambiguities. While models reasoning in natural language have achieved impressive results, multiple works have shown that their performance can be brittle and sensitive to changes in names and numerical values, context length, and irrelevant information [Jiang et al., 2024, Mirzadeh et al., 2024, Malek et al., 2025, Motwani et al., 2026].

In this work, we study the ability of models to transfer skills and knowledge between reasoning tasks that are semantically equivalent but phrased differently. Examples of rephrasing include changing variable names in a math problem, replacing words with synonyms, rewording, and translation to a different language. We will refer to each particular way of phrasing a problem as a task. We assume that the set of possible answers (or labels) for each problem is shared across tasks, and study a type of *label transfer*. Namely, we assume that training examples of problems with a particular label (which we will refer to as the *target label*) are only available for a proper subset of the tasks, and aim to transfer (zero-shot) the ability to predict this label to the other tasks.

A concrete example is provided in Figure 1, which contains propositional logic problems and their solutions, stated using three different text templates (corresponding to three tasks). These tasks differ in the variable names ($X_1, X_2, \ldots$ or $Y_1, Y_2, \ldots$) and/or keywords (*implies* or *if–then*). The possible answers to each problem are `true`, `false`, and `unknown`. The training data includes examples of the `unknown` label for only one task. At test time, we evaluate the model on examples whose label is `unknown`, phrased as in the other two tasks.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Train:
Facts: $Y_1$, $Y_1$ implies $Y_3$. Query: $Y_3$? Reasoning: New fact: $Y_3$. Answer: **true**
Facts: $Y_1$, $Y_1$ implies $Y_2$, $Y_2$ implies not $Y_3$. Query: $Y_3$? Reasoning: New fact: $Y_2$. New fact: not $Y_3$. Answer: **false**
Facts: $Y_1$, $Y_1$ implies not $Y_3$, $Y_2$ implies not $Y_3$. Query: $Y_2$? Reasoning: New fact: not $Y_3$. Answer: **unknown**

Facts: $X_1$, not $X_2$, $X_1$ implies not $X_3$. Query: $X_3$? Reasoning: New fact: not $X_3$. Answer: **false**
Facts: $X_1$, $X_1$ implies not $X_2$, not $X_3$ implies $X_2$. Query: $X_3$? Reasoning: New fact: not $X_2$. New fact: $X_3$. Answer: **true**

Facts: $Y_2$ false, if $Y_1$ then $Y_2$. Query: $Y_1$? Reasoning: By modus tollens $Y_1$ is false. Answer: **false**
Facts: $Y_1$ true, $Y_2$ false, if $Y_1$ then $Y_3$. Query: $Y_3$? Reasoning: By modus ponens $Y_3$ is true. Answer: **true**

Test:
Facts: $X_1$ implies not $X_2$, not $X_3$ implies $X_2$. Query: $X_1$? Answer: **unknown**
Facts: $Y_1$ true, if $Y_1$ then $Y_2$. Query: $Y_3$? Answer: **unknown**

Figure 1: Examples of propositional logic problems stated using different textual templates (different variable names or different wording). At training time, examples containing the `unknown` label available for only one of the tasks (here in magenta). At test time, we evaluate the models' ability to answer `unknown` correctly for problems stated using the other templates.

We study this problem from the perspective of simplicity bias and shortcut learning in transformers. We find that in the presence of *task-specific* tokens, label transfer becomes considerably more difficult. We show that this is at least in part because models rely on correlations and anti-correlations between the task-specific tokens and the target label to make the final prediction. Consistently with previous works [Shah et al., 2020, Hu et al., 2020, Lampinen et al., 2024], we find that these shortcuts relying on simple correlations are acquired in the early stages of training, and may persist even when more predictive features are available. We observe a qualitatively similar, but much weaker, phenomenon at the onset of finetuning pretrained Gemma3 models. As a simple remedy, we show that introducing examples with the target label later in the training dramatically improves performance.

## 2 Related work

**Simplicity bias and shortcut learning** Neural networks often generalize well despite being highly over-parameterized, motivating numerous studies into the *inductive bias* of the architecture and optimization method. Kalimeris et al. [2019], Hu et al. [2020], Shah et al. [2020] give empirical evidence that stochastic gradient descent (SGD) on small feed-forward and convolutional neural networks has a *simplicity bias*, progressively learning functions of increasing complexity. In particular, these works show that the initial performance gains obtained by SGD correspond to learning a linear classifier. Unfortunately, the same inductive biases can cause models to rely on *shortcuts*, corresponding to spurious correlations, for their predictive tasks [Lapuschkin et al., 2019, Shah et al., 2020, Zhang et al., 2023, Hermann et al., 2024, Lampinen et al., 2024]. For example, Shah et al. [2020] show on stylized problems that models tend to rely on simple features throughout training and ignore equally predictive features (that have a higher margin), which hurts robustness and generalization. Lampinen et al. [2024] find that models tend to rely more on features that are simpler to compute or learned first, even if all features are learned equally well, and Hermann et al. [2024] observe that the availability of the features during training also plays a role. Reliance on statistical shortcuts has also been observed in modern large language models [Tang et al., 2023, Malek et al., 2025, Gui and Ji, 2025]. Our work studies the case where certain tokens or token combinations have spurious correlations with the labels.

**Invariant representations** Multiple machine learning works have invested efforts into making models invariant to task-preserving transformations, such as image perturbations and rewriting text. For example, Arjovsky et al. [2019] study invariant risk minimization, where data is collected using different *environments*, defined as different measurements of examples sampled from the same distribution, and the goal is to generalize across all environments. Given a set of task-preserving transformations, Mohri et al. [2025] propose a self-improvement approach that projects the model onto a set of *coherent models* whose output is invariant to the transformations.

**Multilingual transfer** Blum et al. [2025] study cross-lingual knowledge transfer in a controlled setting with small transformer models. They find that models develop either unified or separate

representations early in the training, and that separation occurs when language itself is easy to recognize and useful in predicting the response. Languages are in turn easier to recognize in the presence of language-specific tokens. Thus, their results explain in part why transfer is better for languages with a shared script [Patil et al., 2022]. Other works related to multilingual transfer find that "code-switching" (alternating use of two or more languages) during pre-training helps transfer from English to Korean, Japanese, and Indonesian [Yoo et al., 2025], as well as Chinese [Wang et al., 2025]. Yoo et al. [2025] use a curriculum of code switching first at the token level, then at the sentence level, followed by monolingual corpora.

Our work considers the effect of task-specific tokens when learning to execute an algorithm on in-context data. We find that having semantically similar data (i.e., having the same labels for all tasks) early in the training helps transfer by removing spurious correlations between labels and task-specific tokens. Similarly to the code-switching works, we find that transfer becomes easier in the presence of some overlap between templates.

## 3 Problem definition

Let $\Sigma$ be an alphabet and let $f : \Sigma^* \to \Sigma$ be a next-token generator, where $\Sigma^*$ denotes the set of finite sequences (including the empty sequence) over $\Sigma$. Let $\overline{f}(x) : x \to append(x, f(x)) \in \Sigma^{|x|+1}$ be a function that outputs a concatenation of $x$ and $f(x)$. Define

$$f^{CoT}(x) = \overline{f} \circ \overline{f} \circ \ldots \circ \overline{f}(x)$$

as the function that applies $\overline{f}$ iteratively and halts after generating a token in some stopping set. Define $f^{E2E}(x) = f(f^{CoT}(x))$ as the function that outputs the final answer by applying $f$ one last time to the CoT-augmented input. We can think of the input $x$ as a question, $z = f^{CoT}(x)$ as chain-of-thought reasoning, and $y = f^{E2E}(x)$ as the final answer. We denote by $\mathcal{Y} \subset \Sigma$ the set of possible final answers.

Let $f_*$ denote the target function we want to learn. Let $\mathcal{X} \subset \Sigma^*$ and $\mathcal{Z} \subset \Sigma^*$ denote the sets of questions and chains-of-thoughts, respectively, in some *canonical* format. Let $\psi : \mathcal{X} \cup \mathcal{Z} \to \mathcal{X}_\psi \cup \mathcal{Z}_\psi$ denote a function mapping a canonical-format problem to a new format. We call $\psi$ a *semantics-preserving transformation* if (1) it is invertible; (2) it is identity for the final answers;[1] (3) reasoning about the transformed problem is equivalent to a transformation of the reasoning about the original problem; and (4) the final answers to the original and transformed problems are the same:

$$\psi^{-1}(\psi(x)) = x, \quad \psi(y) = y \ \ \forall y \in \mathcal{Y}, \quad f_*^{CoT}(\psi(x)) = \psi(f_*^{CoT}(x)), \quad f_*^{E2E}(x) = f_*^{E2E}(\psi(x)).$$

Let $\mathcal{D} = \{(x_i, z_i, y_i)\}_{i=1}^N$ be a set of questions, reasoning traces, and answers in the canonical format. Let $\Psi = \{\psi_i\}_{i=1}^K$ denote a set of $K$ (unknown) semantics-preserving transformations (each defining a *task format*), and let $u \in \mathcal{Y}$ denote a unique *target label*. We are given a training set consisting of transformed examples $\mathcal{D}_T = \{\psi^{(i)}(x_i), \psi^{(i)}(z_i), y_i\}_{i=1}^N$, where $\psi^{(i)} \in \Psi$. We assume that the data is such that whenever $y_i = f_*^{E2E}(x_i) = u$, $\psi^{(i)} \neq \psi_1$, that is, we have no examples of label $u$ for transformation $\psi_1$ (in the example of Figure 1, we have no label `unknown` for transformations teal and blue). We are interested in approximating $f_*$ using a function $f_\theta$ parameterized by $\theta$ that generalizes to the label $u$ under transformation $\psi_1$. We define *label transfer accuracy* as

$$acc(\theta, u) = \mathbb{E}_{x:x\in\mathcal{X}, f_*^{E2E}(x)=u} \left[\mathbb{I}[f_\theta^{E2E}(\psi^1(x)) = u]\right],$$

where the expectation is taken over some data distribution conditioned on $f_*^{E2E}(x) = u$ and $\mathbb{I}$ denotes the indicator function. Typically, the next-token predictor $f_\theta$ will be an auto-regressive transformer [Vaswani et al., 2017], outputting a vector of *logits* corresponding to the tokens in the vocabulary, and trained using the next-token-prediction cross-entropy loss. We can make $f_\theta$ a deterministic next-token predictor by selecting the token greedily from the logits.

We assume that there exist multiple distinct functions matching the training data. Obviously, any function $f'_*$ that is identical to $f_*$ when either the task format is in $\Psi \setminus \{\psi_1\}$ or the answer is not $u$ achieves perfect training accuracy even when it is incapable of outputting label $u$ for format $\psi_1$. We study curricula and data-distributional properties which lead to generalization to $\psi_1$.

[1] We could also consider the case where applying $f_*^{E2E}$ to different problem transformations produces different answers (for example, translations into different languages). In that case, we can reduce to the shared-label setting by adding one last step to the reasoning chain that transforms the final answer to the canonical form (for example, the same language) to make the set of labels the same across tasks.

# 4 Label transfer in logical reasoning

In this section, we run controlled experiments on synthetic propositional logic problems, designed to demonstrate the effect of token-label correlations on label transfer. We convert problems to text using two textual templates, only one of which is used for the target label. We consider both varying the *data*, such as variable names, as well as the *syntax*, such as the keywords that describe variable relations or computations to be executed. In all cases, we have a set of task-specific tokens that can be used to uniquely determine the task identity for each example. We vary the training step in which the examples of the target label are introduced. As we will show, *delaying the target label* results in a significant improvement in transfer accuracy across settings, indicating that (spurious) label-token correlations early in the training can be detrimental to generalization.

## 4.1 Propositional logic problems

We create synthetic propositional logic problems consisting of (1) atomic facts, or true zeroth-order predicates ("A is true"); (2) definite clauses involving only two literals ("A implies not B"); and (3) a query, a literal whose truth value is to be determined. We do not allow any contradictions in the statements. New atomic facts can be derived iteratively through the application of *modus ponens* and *modus tollens* inference rules, also called *forward chaining*.[2,3] We randomly generate synthetic data as follows. Given a set of variable names $\mathcal{P}$, we select a subset of variables for each question, and assign `true`/`false` labels to a subset. We then randomly sample atomic facts and implications that are consistent with the labels and sufficient to derive them. The remaining variables are labeled `unknown`. Each problem has between 10 and 20 literals and at most 40 implications, including 5 to 10 *irrelevant* implications. The irrelevant implications do not lead to any new knowledge. `unknown` variables are either mentioned in the irrelevant implications, or not mentioned at all. Implications are given in the order in which they can be applied to derive new knowledge (this order is not unique). The CoT for all problems runs the forward chaining algorithm. In every reasoning step, it finds the first (leftmost) implication in the prompt that leads to a new fact, and applies it to derive that fact. The algorithm stops as soon as the label of the query is derived, or (in case of a query with `unknown` value) when no further facts can be derived. When the query is not mentioned in the context, the CoT is empty. We convert the problems (including the CoT corresponding to forward chaining) to text using simple templates such as those in Figure 1; we will refer to each textual template as a task.

## 4.2 Training details

We train 8-layer transformers with 4-head attention, dimension 512, MLP dimension 1024, tied token embeddings/unembeddings, and RoPE positional embeddings [Su et al., 2024]. We use a custom tokenizer. The exact number of model parameters varies depending on the tokenizer setup, and it is $\sim$21M for the experiments in Figure 2. We train using AdamW [Loshchilov and Hutter, 2017] with decay $10^4$, and a learning rate using linear warmup and cosine decay, with maximum value of $10^{-3}$. Our implementation relies on the Simply framework [Liang et al., 2025]. We train models to predict the CoT reasoning and the final answer conditioned on the problem, with next-token prediction. We choose not to train on the problem definitions because they are generated randomly and not predictable beyond syntax; including them in the loss leads to worse performance. Rather than training on a fixed dataset, we continually generate new random problems throughout training. We evaluate the models' accuracy on fixed test set of size 1024, where we generate reasoning and the final answer using greedy sampling.

## 4.3 Transfer learning experiments

We evaluate transfer on two-task datasets, where the tasks correspond to different textual templates. During training, the first template (task1) is only used for examples with labels `true` and `false`,

[2]Modus ponens states that if *A is true* and *A implies B*, then *B is true*. Modus tollens states that if *B is false* and *A implies B*, then *A is false*. Of course other inference rules also exist (although not used in our examples), such as that if *A implies B* and *not A implies B*, then *B is true*, or the transitivity of implications: if *A implies B* and *B implies C*, then *A implies C*.

[3]Computationally, solving this problem is equivalent to solving two 2-SAT problems, consisting of the original problem with an additional constraint that the query is true and that it is false. Satisfiability of only one of the problems implies that the query is true/false, and satisfiability of both implies that the query is unknown. A contradiction in the facts would result in neither problem being satisfiable; however we do not consider this setting.

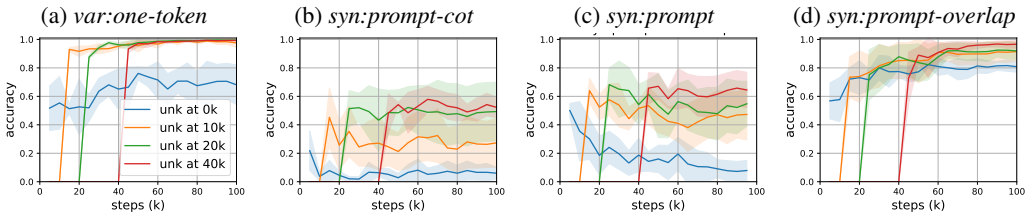


Figure 2: Transfer experiments on propositional logic problems formatted using two different textual templates. All training examples with the label `unknown` are formatted using only the task2 template, and we vary the step in which they are introduced in {0, 10k, 20k, 40k}. We evaluate transfer accuracy on examples whose label is `unknown`, formatted using the task1 template, using five random seeds for each setting. Transfer is considerably better if `unknown` examples are introduced later. In-distribution accuracy is near-perfect in all cases.

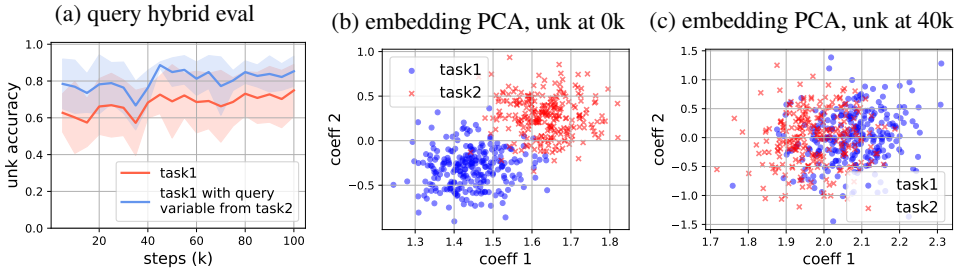


Figure 3: Additional evaluations for the *var:one-token* setting. (a) We evaluate label transfer on 1024 examples formatted using task1, and on the same examples where we change the query to a task2 variable name. This change corrects the output for a large fraction of the examples, and the gap remains large over iterations. (b), (c) Top two PCA coefficients of the embeddings (at the end of training) of task-specific tokens, when the target label is not delayed (b) or introduced at step 40,000 (c). These tokens are clearly separated when the target label is introduced at step 0. In this case, making a rank-1 update to the task1 tokens changes transfer accuracy from 59.5% to 99.7%.

while the second template (task2) is used for all three labels: `true`, `false`, and `unknown`. We evaluate the models' transfer accuracy, that is, their accuracy on problems whose label is `unknown` and are formatted using the first template. We consider several ways of defining the two transformations/templates. Either they differ in the used *variable names*, or they differ in the *syntax*, that is, in the way we formulate the propositional logic statements:

- ***var:one-token:*** We use task-specific single-token variable names, 250 per task. In this case, each task has 250 task-specific tokens. All other formatting remains the same.
- ***syn:prompt-cot:*** Different prompt and CoT syntax: we use task-specific syntax for stating implications ("A implies B" vs. "If A then B EndIf") and CoT inferences ("newfact B" vs. "fact B is proven"). We use the same 100 one-token variable names for both tasks.
- ***syn:prompt:*** Different prompt syntax: we only use different syntax for implications ("A implies B" vs. "If A then B EndIf"); CoT syntax and variable names are the same.
- ***syn:prompt-overlap:*** Different partially-overlapping prompt syntax: we construct five ways of stating an implication, and use implication templates 1-3 for task1 and implication templates 3-5 for task2. CoT and variable names are the same. See Appendix A for specifics. Implication template 3 is shared between the tasks and provides a hint about syntax equivalences.

**Target-label curriculum** For all settings, we vary the training step in which the examples with the target label (`unknown`) are introduced. All models achieve near-perfect accuracy in distribution, that is, on `true`/`false` labels and also on `unknown` labels in task2. Transfer results are shown in Figure 2. We observe that for all settings, transfer is much better if the target-label examples are delayed. This indicates that the model picks up on the anti-correlation between the target label and task identity early in the training, and that this reliance on task identity persists even after richer

features become available. We observe qualitatively similar results when the target label is `true`, shown in Appendix B.1 (this covers a slightly different scenario, as due to the difference in the stopping condition in the CoT, the reasoning behavior for the `true` and `false` labels is more similar than for `unknown`). Additionally, transfer appears easier for variable names than for syntax. While label curriculum (i.e., introducing the target label with some delay) considerably improves accuracy, it saturates at around 50%-60% accuracy for *syn:prompt-cot* and *syn:prompt*. Near-perfect accuracy for syntax transfer is achieved only in the *syn:prompt-overlap* case, where the data contains a hint about the equivalence of implication syntax.

**Hybrid evaluation** To further demonstrate that the transfer failure is due to task/label anti-correlation, we perform a hybrid evaluation of *baseline* models – those trained with no target-label delay. The hybrid tasks are constructed by gradually changing task1 components (either variable names or implication syntaxes) to task2 format. Figure 11 (in Appendix B.3) shows that that this results in a monotonic increase in accuracy for nearly all random seeds. For the *var:one-token* setting, we perform an additional evaluation on a hybrid task where the *query* comes from task2, and all other variables in the problem come from task1. In Figure 3a, we see that this change results in a significant increase in accuracy, and the gap persists throughout training. This suggests that for *var:one-token*, the spurious correlation between tokens and the target label is stored in the token embeddings, since this experiment effectively only changes the embedding of a single input token.

**Token embeddings can store task identity** For the *var:one-token* setting, we examine the manner in which task identity is stored in the token embeddings for models trained with no target-label delay. We perform principal component analysis (PCA) of all task-specific token embeddings (for one of the random seeds, at the end of training) and plot the top two PCA coefficients in Figure 3. We see that task1 and task2 tokens are well separated in this space when we train with the target label from the beginning (Figure 3b), and overlap when the label is delayed (Figure 3c). This suggests that a task-specific component is added to the embeddings in the early phase of training, and for task1 this component inhibits the target label. To further verify this, we applied a rank-1 update to the embeddings of task1-specific tokens so that the average of their second PCA coefficient matches that of the task2-specific tokens (by adding a constant to the second PCA coefficient of the task1 embeddings, equal to the difference between the means of the task2 and task1 coefficients), which changed the transfer accuracy from 59.5% to 99.7%. Unfortunately, with task-specific syntax, improving transfer via a similar low-rank intervention is more difficult, especially when meaning is determined via multi-token sequences such as If–then–EndIf statements.

**Effects of task-label correlations on CoT reasoning** Finally, we examine the ways in which failure manifests in the samples of models trained with no target-label delay (all figures and additional details are presented in Appendix C). For *var:one-token* and *syn:prompt*, we observe that incorrect samples are typically preceded by "motivated" reasoning – a short CoT that incorrectly derives the query to be `true` / `false` and does not follow the generation rules for the CoT otherwise learned for in-distribution examples; see Figure 12 for examples. This suggests that CoT reasoning may not be directly connected to computing the correct solution, reinforcing similar observations about larger models [Palod et al., 2025, Chen et al., 2025, Lindsey et al., 2025]. For the more difficult *syn:prompt-cot* setting, we observe both motivated reasoning and other failures (such as formatting failures, and the answer going into a loop); see Figure 17 for a few examples. On the other hand, correct answers are typically preceded by correct reasoning. For hybrid tasks, improvements in accuracy correspond to improvements in CoT correctness. Thus, our framework may also be useful in studying the effect of statistical shortcuts on motivated / performative reasoning.

For *var:one-token* and *syn:prompt-overlap*, conditioning on the correct reasoning (i.e., prompting the model with the problem and correct CoT in task1 format) results in near-perfect accuracy ($\geq 99.9\%$), and for *syn:prompt* the accuracy improves to over $90\%$; see Figure 15. Even conditioning on a few correct steps can significantly increase accuracy; see Figure 16. On the other hand, for *syn:prompt-cot* (where the CoT syntax differs between tasks), we observe a large variance in random seeds. Conditioning on the correct task1-format, reasoning improves transfer accuracy from $\sim$5% to $\sim$80% on 2/5 random seeds, and does not meaningfully change for the remaining 3 seeds. Conditioning on task2-format, reasoning makes accuracy above $\sim$90% for 4/5 random seeds; see Figure 15. Thus, task-label correlations can also be stored in the reasoning syntax.

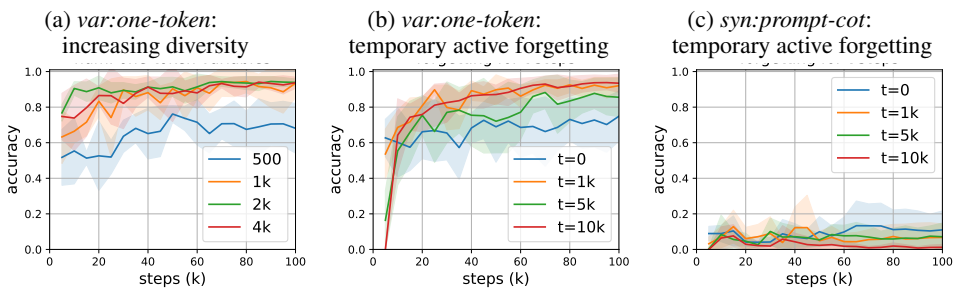


Figure 4: Comparison to other strategies for *var:one-token*. (a) Varying diversity (using the number of variable names as a proxy) improves transfer, but not to the same extent as delaying `unknown` examples. (b) Temporary active forgetting (resetting token embeddings every 100 steps for the first $t$ training steps) also improves transfer. (c) Temporary active forgetting does not help for templates varying syntax, such as *syn:prompt-cot*.

### 4.4 Comparison to other strategies for improving transfer

**Data-distributional effects: diversity and overlap** Multiple works on in-context learning (ICL) show that generalization to new problem instances, or even grokking, can be achieved by increasing the diversity of the training data [Raventós et al., 2023, He et al., 2024, Chan et al., 2025, Lazić et al., 2026]. Hence we evaluate the effect of diversity on label transfer. We focus on the *var:one-token* setting, with task-specific one-token variable names. We use the number of variable names as a proxy for diversity, and vary it in {500, 1000, 2000, 4000}. Figure 4a shows that diversity indeed improves transfer, though introducing the label with delay is still more effective (compare to Figure 2). Diversity is more difficult to evaluate systematically for tasks with different syntax, as it would require using the same phrases (e.g., "implies", "means that") in multiple different contexts to collapse their representations. However, our *syn:prompt-overlap* results show the effectiveness of a related data-distributional change: models can discover equivalences of task-specific statements through a common (equivalent) statement. Similar effects have been observed in multilingual training works [Yoo et al., 2025, Wang et al., 2025], where mixing of languages was found to improve transfer. Our *syn:prompt-overlap* setting can be thought of as a simplified model for the multilingual training of Yoo et al. [2025].

**Temporary active forgetting** Anand et al. [2025] study the task of *structural ICL*, which they define as the ability of a model to execute a symbolic reasoning algorithm on an input where the symbols correspond to arbitrary tokens. The main idea is that the algorithm should generalize based on the problem structure, rather than the token embedding. This is closely related to our setting with task-specific one-token variable names. Anand et al. [2025] proposed a strategy called *temporary active forgetting*, where the embedding parameters of the model are periodically reset in the first phase of training, and which has also been found helpful in symbolic logical reasoning [Lazić et al., 2026]. In our setting, resetting the token embedding reduces the correlation between task-specific tokens and labels. We evaluate this technique by resetting the embeddings every 100 steps for the first $t \in \{0, 1k, 5k, 10k\}$ training steps, and evaluating transfer accuracy as before. In Figure 4b, we observe that temporary active forgetting indeed improves transfer in the *var:one-token* setting where output needs to be invariant to variable names. However, as shown in Figure 4c, this strategy is not useful in the case of task-specific syntax, where it may hinder or delay the collapse of representations of equivalent phrases (such as "implies", "if–then").

## 5 Label transfer without reasoning

In this section we evaluate label transfer on a simple *copy/abstain* task, which does not require CoT and is similar to *var:one-token* logic problems with correct CoT given. We will show qualitatively the same behavior as before, and provide some intuition for why label curriculum is helpful.

Figure 5 shows some examples of the copy/abstain task. We are given in-context a list of variable names followed by their binary labels, as well as a special $\star$ variable followed by the `unknown` label. The last token is the query variable. If the query appears earlier in the context, the model needs to

Train:
<BOS> ⋆ unknown $A_5$ true $A_6$ false $A_2$ true $A_4$ true $A_6$, false
<BOS> ⋆ unknown $A_2$ false $A_4$ true $A_1$ false $A_3$ false $A_4$, true
<BOS> ⋆ unknown $B_1$ true $B_2$ false $B_3$ false $B_4$ false $B_1$, true
<BOS> ⋆ unknown $B_8$ false $B_2$ false $B_7$ true $B_9$ true $B_8$, false
<BOS> ⋆ unknown $B_3$ false $B_0$ false $B_5$ true $B_1$ false $B_6$, unknown
Test:
<BOS> ⋆ unknown $A_2$ false $A_4$ false $A_9$ false $A_6$ false $A_1$, unknown

Figure 5: Illustrative examples of the copy/abstain task. The context contains a list of variables followed by their values. The last token corresponds to a query variable. The output is the value of the variable if it is present earlier in the context, and unknown otherwise. Variable names have the form $A_i$ or $B_i$, and the data contains examples of the unknown label only for $B_i$ variables. We evaluate models on examples with $A_i$ variables and label unknown.

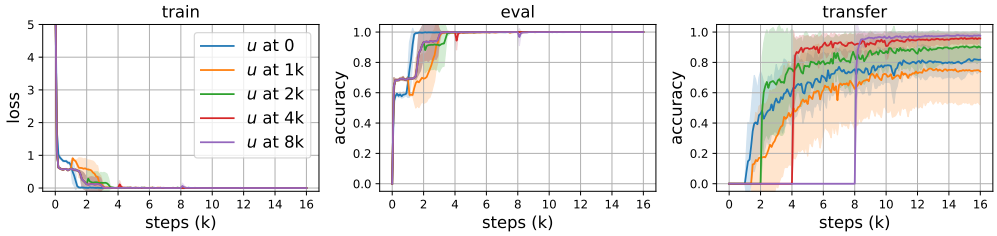


Figure 6: Training loss, in-distribution accuracy, and transfer accuracy on the copy/abstain task. We vary the training step at which examples with the target label unknown (denoted $u$) are introduced, and report mean and standard deviation for 3 random seeds. Performance is better if unknown labels are introduced later in training, after the accuracy is near-perfect on the true/false labels.

output its label, otherwise it needs to output unknown. We again assume that training examples come from one of two tasks, corresponding to variable names of the format $A_i$ or $B_i$, respectively. We assume that unknown-label examples are available only for the second task, involving $B_i$ variables. We are interested in transferring the ability to output unknown to examples with $A_i$ variables.

The copy/abstain algorithm can be implemented by a circuit similar to *induction heads* [Olsson et al., 2022], which are known to emerge early in language model training. Thus, we study the simplest model capable of implementing an induction head: a two-layer transformer with a single attention head per layer and no MLPs. Let $x$ be an $n$-token input string and let $y$ be the label token. Let $X \in \mathbb{R}^{n \times V}$ be a matrix containing the one-hot representation of the tokens of $x$, where $V = |\Sigma|$. Let $W_E \in \mathbb{R}^{V \times d}$ be a matrix of token embeddings, and let $W_P$ be an $n \times d$ matrix of absolute position embeddings. Let $\mathcal{Y} = \{\texttt{true}, \texttt{false}, \texttt{unknown}\}$ be the set of possible labels. The output of the transformer are logits computed as

$$
\begin{aligned}
f_\theta(x) &= Z_{2,n} W_Y \\
Z_{i+1} &= Z_i + attn_i(layernorm_i(Z_i)), \quad Z_0 = XW_E + W_P \\
attn_i(Z) &= softmax(ZW_Q^i W_K^{i\top} Z^\top) Z W_{VO}^i
\end{aligned}
$$

where $\theta$ is the set of all model parameters, consisting of the matrices $W_Q^i, W_K^i, W_{VO}^i$ as well as $W_E$ and $W_P$, and $Z_{i,n}$ denotes the $n^{th}$ (last) row of the matrix $Z_i$. $layernorm$ [Ba et al., 2016] normalizes the per-example activations of a layer to have mean and standard deviation close to 0 and 1, respectively, and optionally rescales them by a learned parameter. Let $p(\cdot|x) = softmax(f_\theta(x))$ be the predicted probability of labels for input $x$. We optimize the parameters by minimizing the cross-entropy loss $L_\theta(\mathcal{D}) = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(y_i|x_i)$ on a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ using AdamW [Loshchilov and Hutter, 2017]. We only train the model to predict the last label rather than doing next-token prediction on the whole sequence (since input variable names and values are selected uniformly at random for each task). By default we use $d = 128$, a custom tokenizer, and 200 variable names per task, for a total of 407 tokens. Each example contains four variable-label pairs.

Results are shown in Figure 6, where we vary the training step in which the target label unknown is introduced; for simplicity, we will denote the target label by $u$ in the remainder of this section. We

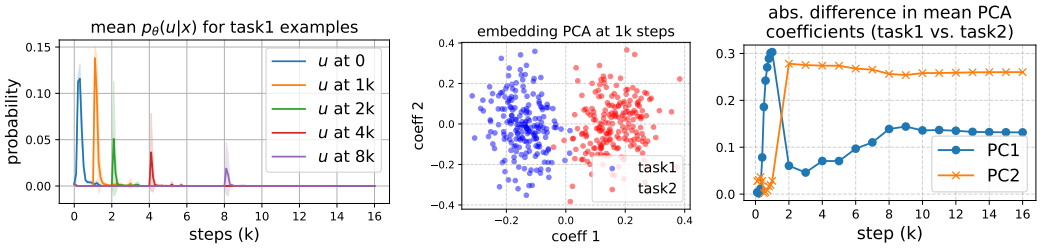


Figure 7: **Left**: Probability of the target label $u$ given task1 training inputs. **Middle**: Top two PCA components of the variable name token embeddings, for a model trained without label delay. The tasks are well separated after only 1000 training steps. **Right**: Absolute difference between the mean PCA coefficients of task1/task2-specific token embeddings over training steps.The task-specific tokens become separated quickly in the initial stage of training.

observe that the model can obtain perfect in-distribution accuracy. Transfer is better and occurs faster if the target label is introduced after the accuracy on labels $\mathcal{Y} \setminus u$ is near-perfect. We provide some intuition for these results next.

Note that the gradient of the cross-entropy loss with respect to logits is just the error $err_\theta(x_i, y_i) := p_\theta(\cdot|x_i) - \mathbf{e}_{y_i}$. This error is a factor in all parameter gradients. In particular, due to residual connections, the gradient of the loss with respect to the embedding matrix $W_E$ has the form

$$\nabla_{W_E} L_\theta(\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} g_i + h_i, \quad g_i := X_i^\top \mathbf{e}_n (p_\theta(\cdot|x_i) - \mathbf{e}_{y_i})^T W_Y^\top$$

where $g_i$ corresponds to the $Z_{0,n}$ component of $Z_{2,n}$ (which is present due to the skip connections) and $h_i$ corresponds to the remaining components. For the target label $u$ and task1 examples $(x_i, y_i)$, the error will always be positive: $err_\theta(x_i, y_i)_u = p_\theta(u|x_i)$, since $y_i \neq u$. Thus, for each task1 example, a gradient update will subtract the vector $p(u|x_i)W_{Y,u}$ (adjusted by the learning rate) from the embedding of the token $x_{i,n}$. This can cause the task1-specific token embeddings to drift closer together and apart from task2-specific token embeddings, for which the error can have either sign.

When we train with the target label $u$ from the beginning, since $u$ appears in task2 examples, it may have a non-negligible probability for task1 examples in the initial phase of training. This will correspond to having larger $err_\theta(x_i, y_i)_u$ and thus larger task-separating gradients. On the other hand, if we delay the introduction of label $u$ until the other labels have been learned, the model will drive the probability of $u$ close to 0 more quickly (since it will not be plausible for any inputs), and will place a smaller probability on $u$ once it is introduced for task2. We verify this empirically: Figure 7 (left) shows $p_\theta(u|x)$ for task1 inputs during training under different target-label curricula. Figure 7 (middle), shows that the task-specific token embeddings are nearly linearly separable in the coefficient of the first PCA component after only 1000 steps when the target label is introduced at the beginning. We also plot the absolute difference between mean top-two PCA components for the two tasks, as a function of training step, in Figure 7 (right), which shows that the tasks are separable based on the top two PCA coefficients. While he figure shows that after the initial phase of the training – captured in the middle figure – separation of the task-specific tokens is rather possible based on the second PCA coefficient, this is not a sign of any significant changes and appears because the order of the top two PCA coefficients changes over the course of the training.

To further support the hypothesis that linear correlations early in the training hinder transfer, we repeat experiments on a modified model, where we remove residual connections, that is, we set $Z_{i+1} = attn_i(layernorm(Z_i))$ rather than $Z_{i+1} = Z_i + attn_i(layernorm(Z_i))$. Figure 18 in Appendix D shows that this simple change results in a considerable increase in transfer accuracy for the model without label delay. Furthermore, without residual connections, the task-specific embeddings are no longer linearly separable in the top two PCA components.

In summary, consistent with the literature on simplicity bias [Lampinen et al., 2024, Hu et al., 2020, Shah et al., 2020], we have observed that models pick up simple correlations early in the training, and that these correlations can persist. This leads to poor transfer in cases where task identity (encoded in a set of tokens) is anti-correlated with a label. We have also provided a simple explanation for why label curriculum may decrease the severity of such correlations.

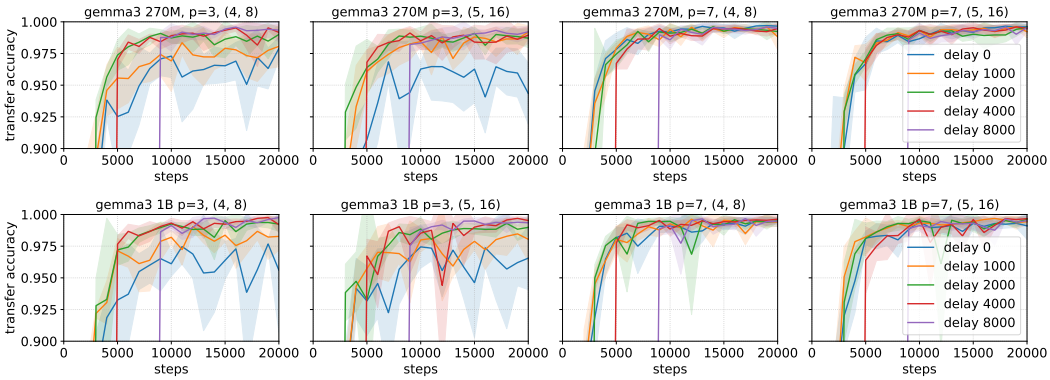


Figure 8: Transfer accuracy on MathGAP problems when finetuning pretrained Gemma3 models, for different problem difficulty (max depth/leaves set to (4, 8) and (5, 16), and for answers given $\mod p$ for $p \in \{3, 7\}$). The data contains no Serbian-language examples with answer 0. We vary the training step in which English-language examples with answer 0 are introduced. We observe that label delay leads to improved performance for $p = 3$, and no notable improvements for $p = 7$.

## 6 Finetuning large language models

Here we investigate the extent to which similar token-label correlations are problematic when finetuning larger pretrained models, specifically Gemma3 models of size 270M and 1B. We used a synthetic dataset of natural-language arithmetic word problems based on MathGAP [Opedal et al., 2025], with the augmentations of Malek et al. [2025], chain-of-thought solutions for training purposes, and given in one of two languages: English and Serbian using the Cyrillic alphabet (see Figure 19 for an example). The solutions of these problems can be structured as "proof trees", where in each tree the leaves are the given facts, internal nodes are intermediate conclusions, and the root node contains the answer to the problem. We experimented with two difficulty settings, corresponding to max depth and number of leaves set to (4, 8) or (5, 16), generated 100k problem examples, computed the final answer modulo $p \in \{3, 7\}$, and evaluated transfer to Serbian-language problems with answer $0 \mod p$, which were withheld from the training data. The results are shown in Figure 8. For $p = 3$, label delay consistently improves transfer accuracy, suggesting that the spurious correlations are picked up even when starting with pretrained representations. However, absolute improvement in accuracy is much smaller than in our pretraining experiments. Furthermore, for $p = 7$, there is no significant difference between curriculum and standard training. We hypothesize that in the finetuning regime, only very large correlations are problematic.

## 7 Discussion

Large language models are increasingly being used to solve problems in math and science that require formal reasoning. One difficulty here is that training problems are typically stated, and often solved, using natural language, where the same concepts can be expressed in many different ways. Using controlled experiments, we studied the ability of transformers to learn representations that capture problem semantics and are invariant to the particular textual formulation. We found that transfer between different problem formulations is impeded by correlations between tokens and labels, which the model picks up in the early stages of training. Our findings are qualitatively consistent with existing works on spurious correlations [Lampinen et al., 2024, Hermann et al., 2024, Hu et al., 2020] showing that neural networks learn simpler features in the initial stages of training, and that these features can persist. Similar findings have been reported by Blum et al. [2025], who study the relation between cross-lingual fact transfer in relation to overlap in language tokens. We confirmed the importance of early training dynamics using curriculum experiments, and provided intuition for why label curriculum helps on a simpler task without CoT. Some limitations of our work are that most experiments were performed in a controlled setting on synthetic data. Our finetuning study shows that label-token correlations can still hurt performance of pretrained models, but the extent is not nearly as dramatic. Another limitation is that the type of curriculum we impose may be difficult to implement with natural data, and thus data-distributional interventions may be a more practical way of improving transfer.

## Acknowledgment

We thank Dhawal Gupta and Kaiwen Wang for their involvement in the initial stages of this project.

## References

Suraj Anand, Michael A Lepori, Jack Merullo, and Ellie Pavlick. Dual process learning: Controlling use of in-context vs. in-weights strategies with weight forgetting. In *The Thirteenth International Conference on Learning Representations*, 2025.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Carter Blum, Katja Filippova, Ann Yuan, Asma Ghandeharioun, Julian Zimmert, Fred Zhang, Jessica Hoffmann, Tal Linzen, Martin Wattenberg, Lucas Dixon, et al. Beyond the rosetta stone: Unification forces in generalization dynamics. *arXiv preprint arXiv:2508.11017*, 2025.

Bryan Chan, Xinyi Chen, András György, and Dale Schuurmans. Toward understanding in-context vs. in-weight learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

Shurui Gui and Shuiwang Ji. Mitigating spurious correlations in LLMs via causality-aware post-training. *arXiv preprint arXiv:2506.09433*, 2025.

Tianyu He, Darshil Doshi, Aritra Das, and Andrey Gromov. Learning to grok: Emergence of in-context learning and skill composition in modular arithmetic tasks. *Advances in Neural Information Processing Systems*, 37:13244–13273, 2024.

Katherine Hermann, Hossein Mobahi, Michael Curtis Mozer, et al. On the foundations of shortcut learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *Advances in Neural Information Processing Systems*, 33: 17116–17128, 2020.

Bowen Jiang, Yangxinyu Xie, Zhuoqun Hao, Xiaomeng Wang, Tanwi Mallick, Weijie J Su, Camillo J Taylor, and Dan Roth. A peek into token bias: Large language models are not yet genuine reasoners. *arXiv preprint arXiv:2406.11050*, 2024.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

Andrew Kyle Lampinen, Stephanie CY Chan, and Katherine Hermann. Learned feature representations are biased by complexity, learning order, position, and more. *Transactions on Machine Learning Research*, 2024.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1096, 2019.

Nevena Lazić, Liam Fowl, András György, and Csaba Szepesvári. To see the unseen: on the generalization ability of transformers in symbolic reasoning. In *Third Conference on Language Modeling*, 2026.

Chen Liang, Da Huang, Chengrun Yang, Xiaomeng Yang, Andrew Li, Xinchen Yan, and Simply Contributors. Simply: an experiment to accelerate and automate AI research. GitHub repository, 2025. URL https://github.com/google-deepmind/simply.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, David Abrahams Craig Citro, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model, mar 2025. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html. Accessed: 2026-05-05.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Alan Malek, Jiawei Ge, Nevena Lazić, Chi Jin, András György, and Csaba Szepesvári. Frontier LLMs still struggle with simple reasoning tasks. *arXiv preprint arXiv:2507.07313*, 2025.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2024.

Mehryar Mohri, Jon Schneider, and Yifan Wu. Coherence mechanisms for provable self-improvement. *arXiv preprint arXiv:2511.08440*, 2025.

Sumeet Ramesh Motwani, Daniel Nichols, Charles London, Peggy Li, Fabio Pizzati, Acer Blake, Hasan Abed Al Kader Hammoud, Tavish Malcolm McDonald, Akshat Naik, Alesia Ivanova, Vignesh Baskaran, Ivan Laptev, Ruben Glatt, Tal Ben-Nun, Philip Torr, Ameya Prabhu, Brian R. Bartoldson, Bhavya Kailkhura, and Christian Schroeder de Witt. Longcot: Benchmarking long-horizon chain-of-thought reasoning. In *Forty-third International Conference on Machine Learning*, 2026.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

Andreas Opedal, Haruki Shirakami, Bernhard Schölkopf, Abulhair Saparov, and Mrinmaya Sachan. Mathgap: Out-of-distribution evaluation on problems with arbitrarily complex proofs. In *International Conference on Learning Representations*, pages 100465–100487, 2025.

Vardhan Palod, Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. Performative thinking? the brittle correlation between cot length and problem complexity. In *NeurIPS 2025 Workshop on Efficient Reasoning*, 2025.

Vaidehi Patil, Partha Talukdar, and Sunita Sarawagi. Overlap-based vocabulary generation improves cross-lingual transfer among related languages. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 219–233, 2022.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-bayesian in-context learning for regression. *Advances in neural information processing systems*, 36:14228–14246, 2023.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Ruixiang Tang, Dehan Kong, Longtao Huang, et al. Large language models can be lazy learners: Analyze shortcuts in in-context learning. In *Findings of the association for computational linguistics: ACL 2023*, pages 4645–4657, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Zhijun Wang, Jiahuan Li, Hao Zhou, Rongxiang Weng, Jingang Wang, Xin Huang, Xue Han, Junlan Feng, Chao Deng, and Shujian Huang. Investigating and scaling up code-switching for multilingual language model pre-training. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11032–11046, 2025.

Haneul Yoo, Cheonbok Park, Sangdoo Yun, Alice Oh, and Hwaran Lee. Code-switching curriculum learning for multilingual transfer in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 7816–7836, 2025.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van Den Broeck. On the paradox of learning to reason from data. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 3365–3373, 2023.

# A Construction of tasks with overlapping templates

We construct five different ways of stating an implication $A \rightarrow B$, where each word$\{i\}$ corresponds to a single token:

1. word1 A word2 word3 B word4
2. word5 A word6 B
3. A word7 word8 B word9
4. word10 A word11 B word12
5. word13 word14 A word15 B word16

For task1 examples, we use a uniform mixture of implication templates 1, 2, 3 in each example. For task2 examples, we use a uniform mixture of templates 3, 4, 5. This makes it easier for the model to learn equivalences between different keywords and their combinations.

# B Additional transfer experiments

## B.1 Transferring the `true` label

One alternative explanation to our empirical results, showing that introducing the `unknown` target label later helps with transfer on logic problems, may be that this label is more difficult to learn as it requires more reasoning steps on average. In this section we show qualitatively similar performance transferring the `true` label. Namely, we train on two tasks with example labels `false`, `unknown` for task 1, and labels `true`, `false`, and `unknown` for task 2. We evaluate models on task-1 problems whose answer is `true`, using the same templates as before. The results are shown in Figure 9. The results for *var:one-token* and *syn:prompt-cot* are qualitatively similar to those for the `unknown` label shown in Figure 2. In particular, the performance in the setting where the target label examples are available right away never catches up to the label curriculum case.

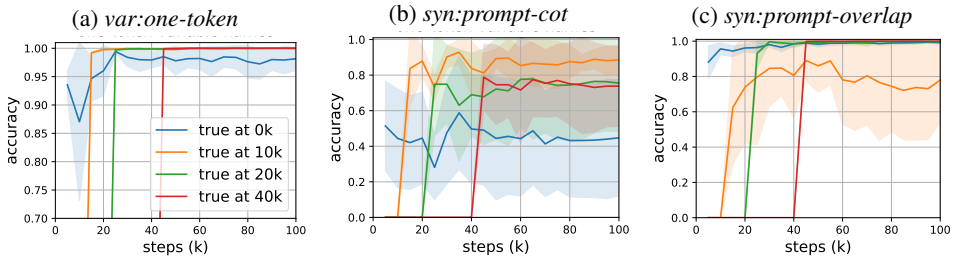


Figure 9: Transfer experiments on propositional logic problems formatted using different textual templates, where examples labeled `true` are only formatted using the second template. We evaluate transfer on task1 examples whose label is `true`, using five random seeds for each setting. In most cases, transfer is better if `true` examples are introduced later in the training, similarly to the experiments with the `unknown` label.

## B.2 Transfer to tasks with shared tokens

In this section, we evaluate transfer between tasks that use different templates, but share the same tokens. In particular, we use task-specific *two-token* variable names, 250 per task, and ensure that both tasks use the same variable tokens. Only the token-pairs defining the variables are task-specific. The tasks use the same syntax. We name this setting *var:two-token* and evaluate the transfer of the `true` and `unknown` labels. As shown in Figure 10, label delay appears to improve transfer of the `true` label, but the benefits are less clear for the `unknown` label.

## B.3 Hybrid evaluation

In Figure 11, we show that on hybrid tasks, where we gradually change task1 components (either variable names or implication syntax) to task2 format, accuracy on the `unknown` label increases

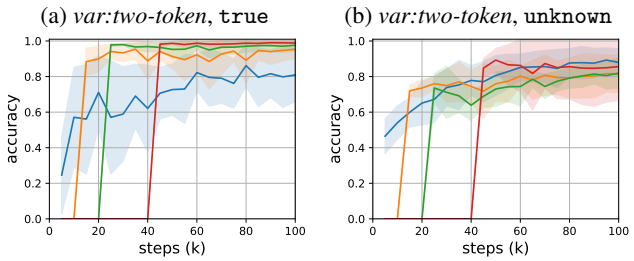


Figure 10: Transfer accuracy for the *var:two-token* setting where tasks share the same tokens, and variable names correspond to different two-token sequences.

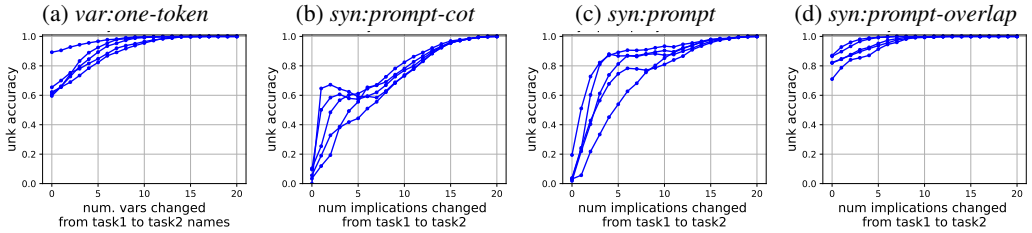


Figure 11: Evaluation on hybrid data, where we gradually change components from task1 to task2, for models that see the target label from the beginning, using 5 random seeds in each setting. In nearly all cases, accuracy increases monotonically as we change the format.

monotonically for nearly all seeds. This further demonstrates that the errors are due to spurious task-label correlations.

## C Examples of incorrect samples

***var:one-token*** We give examples of model samples in which either the reasoning, the final answer, or both are incorrect. These are shown in Figure 12, and are representative of around 100 samples inspected by the authors. In most cases, CoT and the final answer are either both correct or both incorrect (see Figure 13 for some statistics). The both-incorrect samples typically contain a short "motivated reasoning" chain which produces an incorrect inference leading to the wrong answer. Alternatively, the model may not perform any inference steps and simply output the wrong answer, as if the query is a part of the atomic facts given in the problem. In a small fraction of cases, CoT contains an erroneous step unrelated to the final answer, and the final answer is still correct. The case where CoT is correct and the final answer is incorrect occurs very rarely. Intervening by changing the query variable name from a task1 to a task2 name increases accuracy across all random seeds. For 4/5 seeds, the change in accuracy corresponds almost exactly to the change in the fraction of correct reasoning traces (see Figure 13, right). The incorrect CoT traces still typically have the form in Figure 12, containing either one or no reasoning.

***syn:prompt-cot*** In Figure 15 we evaluate the accuracy of the baseline model when conditioned on ground-truth reasoning in task1 format, and ground truth reasoning in task2 format. We observe that reasoning format is also correlated with the label, as task2 reasoning results in considerably higher accuracy. In Figure 17 we give examples of model samples when conditioned on the baseline prompt, and the baseline prompt after changing a fraction of implications to task2 format. Conditioning on a mix of task1 and task2 implication syntax results in the model generating a mixture of task1 and task2 inference syntax; some examples are shown in Figure 17.

***syn:prompt*** For this setting, most of the answers and CoT are incorrect. The incorrect CoT is similar to the samples produced for the *var:one-token*: either empty or containing short reasoning incorrectly deriving the value of the query.

***syn:prompt-overlap*** In this setting, CoT and the final answer are not as consistent, in the sense that a large fraction of *correct* answers are preceded by incorrect reasoning. In most of cases of correct answer and incorrect reasoning, the model will make an incorrect step, either not deriving the value of a variable, or incorrectly deriving a variable. However, these reasoning errors are unrelated query, i.e. they do not incorrectly derive the `unknown` query to be `true` or `false`, similarly to the "CoT incorrect, answer correct" examples in Figure 12.

**CoT incorrect, answer incorrect**
**Question:** Facts: not t43 implies not t30 , not t43 implies t77 , not t43 implies t223 , t77 implies not t148 , t92 implies not t193 , not t233 implies not t43 , not t148 implies not t233 , t179 implies not t43 , not t43 implies not t233 , not t30 implies t179 , not t173 implies t232 , t173 implies not t193 , not t43 , not t233 , t179 .
Query: t193 ?
**Sample:** Reasoning: newfact not t193 . Answer: false
**Ground truth answer:** Reasoning: newfact not t30 , newfact t77 , newfact t223 , newfact not t148 . Answer: unknown
**Question:** Facts: t51 implies t201 , not t78 implies t151 , not t35 implies not t123 , not t123 implies t139 , t201 implies not t7 , not t123 implies t198 , t201 implies not t13 , t139 implies not t31 , not t42 implies not t192 , not t35 implies not t145 , t60 implies t192 , not t35 , t51 , not t78 , not t145 .
Query: t60 ?
**Sample:** Reasoning: . Answer: true
**Ground truth answer:** Reasoning: newfact t201 , newfact t151 , newfact not t123 , newfact t139 , newfact not t7 , newfact t198 , newfact not t13 , newfact not t31 . Answer: unknown
**CoT incorrect, answer correct**
**Question:** Facts: not t118 implies not t193 , not t193 implies t183 , not t118 implies t140 , t140 implies t40 , not t193 implies t85 , not t3 implies not t10 , t183 implies t78 , not t193 implies not t249 , not t10 implies t211 , t183 implies t154 , t183 implies t100 , t113 implies not t35 , not t232 implies not t71 , t220 implies not t213 , t108 implies not t232 , not t118 , not t3 , t113 .
Query: t232 ?
**Sample:** Reasoning: newfact not t193 , newfact t183 , newfact t140 , newfact . Answer: unknown
**Ground truth answer:** Reasoning: newfact not t193 , newfact t183 , newfact t140 , newfact t40 , newfact t85 , newfact not t10 , newfact t78 , newfact not t249 , newfact t211 , newfact t154 , newfact t100 , newfact not t35 . Answer: unknown
**Question:** Facts: t9 implies t232 , t27 implies t79 , t232 implies t7 , t7 implies t32 , not t101 implies t149 , t195 implies t146 , t195 implies not t22 , not t195 implies not t228 , t232 implies t232 , t7 implies t79 , t195 implies not t146 , not t101 implies not t101 , t9 , t167 , t140 , t27 , not t101 , t10 .
Query: t195 ?
**Sample:** Reasoning: newfact t232 , newfact t79 , newfact t7 , newfact t32 , newfact not t103 . Answer: unknown
**Ground truth answer:** Reasoning: newfact t232 , newfact t79 , newfact t7 , newfact t32 , newfact t149 . Answer: unknown
**CoT correct, answer incorrect**
**Question:** Facts: t59 implies t55 , t174 implies not t37 , t37 implies t175 , t59 implies not t126 , not t81 implies t228 , not t126 , t59 , t55 , not t81 , t228 .
Query: t175 ?
**Sample:** Reasoning: . Answer: true
**Ground truth answer:** Reasoning: . Answer: unknown

Figure 12: Examples of model samples for *var:one-token* problems for which CoT, final answer, or both are incorrect.

**Conditioning on partial ground-truth CoT** Given that the model errors typically manifest as motivated reasoning, we also evaluate the change in performance after conditioning models on a partial ground-truth CoT trace. Specifically, we prompt models with a problem and the first $k$ CoT steps, where we define each step as the derivation of a single new fact. Figure 16 shows that conditioning on even a single correct CoT step can significantly improve accuracy.

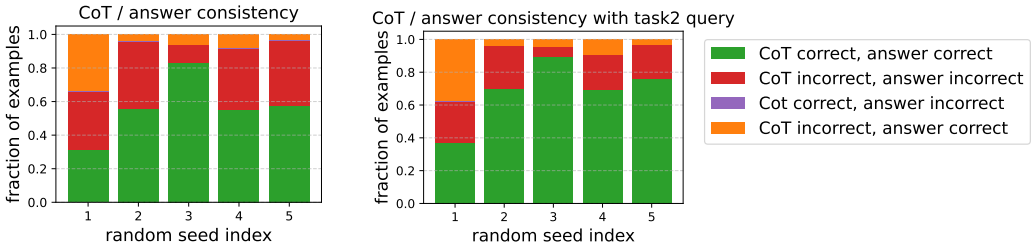


Figure 13: **Left**: Consistency between CoT and answer correctness for the *var:one-token task*. In most random runs, the CoT and final answer are either both correct or both false, and there are few or no examples with correct CoT and an incorrect answer. **Right**: Consistency between CoT and answer correctness after replacing the query variable with a random task2 variable. Changing the query variable increases accuracy. The increase in accuracy corresponds to an increase in correct CoT traces for 4/5 random seeds.

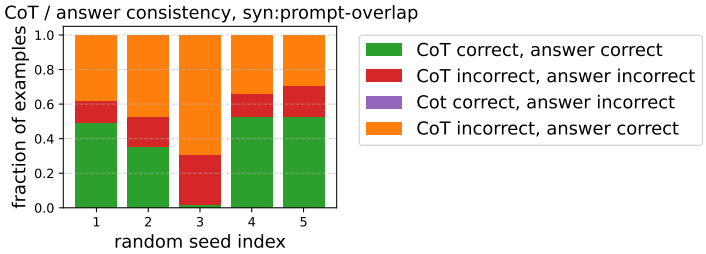


Figure 14: Consistency between CoT and answer correctness for the *syn:prompt-overlap* setting. In this case, a large fraction of correct answers are *not* preceded by correct CoT. In these cases, the CoT is typically mostly correct and missing a step or contains an extra (incorrect) step unrelated to the query.

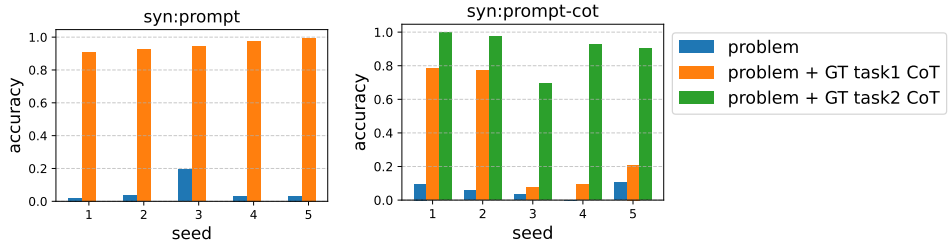


Figure 15: Accuracy of baseline models (trained without target-label delay for 5 random seeds) when prompted with the problem, problem and ground-truth (GT) CoT in task1 format, and (for *syn:prompt-cot*) problem and ground-truth CoT in task2 format. For *syn:prompt*, conditioning on ground-truth reasoning significantly improves accuracy. For *syn:prompt-cot*, task1 reasoning only meaningfully helps for 2/5 random seeds, whereas task2 reasoning results in high accuracy for 4/5 seeds. This suggests that for three of the random seeds, the model does not capture equivalence between the task1 and task2 reasoning formats.

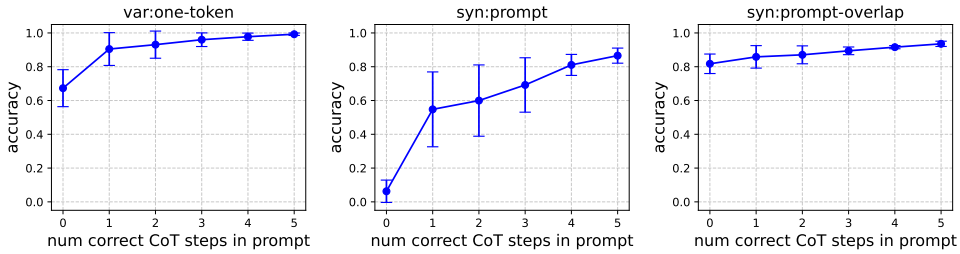


Figure 16: Accuracy after conditioning on partial ground-truth CoT, where each CoT step corresponds to deriving a single new fact. Providing even a single ground-truth CoT step can considerably increase accuracy for the *var:one-token* and *syn:prompt* settings, where the initial transfer accuracy is lower.

**Question:** Facts: not t12 implies t9 , t9 implies not t55 , t9 implies t30 , t9 implies t22 , t30 implies t76 , not t55 implies not t97 , not t55 implies t93 , t93 implies t96 , not t73 implies t78 , t76 implies not t12 , t10 implies not t79 , not t28 implies t78 , t96 implies t30 , not t12 , not t31 .
Query: t48 ?
**Modified question:** Facts: If not t12 then t9 EndIf , If t9 then not t55 EndIf , If t9 then t30 EndIf , t9 implies t22 , t30 implies t76 , not t55 implies not t97 , not t55 implies t93 , t93 implies t96 , not t73 implies t78 , t76 implies not t12 , t10 implies not t79 , not t28 implies t78 , t96 implies t30 , not t12 , not t31 .
Query: t48 ?
**Sample:** Reasoning: newfact t9 , newfact t30 , newfact t22 , newfact t76 , newfact t76 , newfact not t97 , newfact newfact t9 , newfact t96 , newfact not t96 , newfact t78 , newfact t96 , newfact not t96 , newfact t78 t78 , newfact t96 , newfact not newfact not , newfact t78 , newfact t96 , newfact not , newfact t78 t78 , newfact t96 , newfact , newfact not , newfact t78 t78 t78 t78 ... (loop)
**Revised sample:** Reasoning: newfact t9 , fact not t55 is proven , fact t30 is proven , fact t76 is proven , newfact t76 is proven , fact not t97 is proven , fact t93 , newfact t96 , newfact not t96 , newfact t93 is proven .
Answer: unknown

**Question**: Facts: not t17 implies not t11 , not t17 implies t29 , not t17 implies t86 , t29 implies not t58 , t35 implies not t72 , not t92 implies not t17 , not t58 implies not t92 , t69 implies not t17 , not t17 implies not t92 , not t11 implies t69 , not t99 implies t89 , t99 implies not t72 , not t58 implies not t58 , not t17 , not t92 , t69 .
Query: t72 ?
**Modified question**: Facts: If not t17 then not t11 EndIf , If not t17 then t29 EndIf , If not t17 then t86 EndIf , t29 implies not t58 , t35 implies not t72 , not t92 implies not t17 , not t58 implies not t92 , t69 implies not t17 , not t17 implies not t92 , not t11 implies t69 , not t99 implies t89 , t99 implies not t72 , not t58 implies not t58 , not t17 , not t92 , t69 .
Query: t72 ?
**Sample**: Reasoning: newfact not t72 . Answer: false
**Revised sample**: Reasoning: newfact not t11 , newfact t29 is proven , newfact t86 is proven , newfact not t58 is proven .
Answer: unknown

**Question:** Facts: t60 implies not t48 , t60 implies not t9 , not t9 implies not t95 , not t48 implies not t26 , not t95 implies t43 , t43 implies t17 , not t64 implies t43 , t17 implies t17 , not t38 implies not t4 , not t38 implies not t16 , not t38 implies t16 , not t95 implies t60 , not t71 implies t20 , not t31 implies not t4 , t4 implies not t31 , t60 , not t64 .
Query: t4 ?
**Modified question**: Facts: If t60 then not t48 EndIf , If t60 then not t9 EndIf , If not t9 then not t95 EndIf , not t48 implies not t26 , not t95 implies t43 , t43 implies t17 , not t64 implies t43 , t17 implies t17 , not t38 implies not t4 , not t38 implies not t16 , not t38 implies t16 , not t95 implies t60 , not t71 implies t20 , not t31 implies not t4 , t4 implies not t31 , t60 , not t64 .
Query: t4 ?
**Sample**: Reasoning: newfact not t4 . Answer: false
**Revised sample**: Reasoning: newfact not t48 , newfact not t9 , newfact not t95 is proven , newfact not t26 is proven , newfact t17 , newfact not t48 is proven . Answer: unknown

Figure 17: Examples of original and corrected samples for hybrid task1-task2 *syn:prompt-cot* problems. Given mixed-syntax inputs, the model often outputs mixed-syntax reasoning (using both "newfact X", "fact X is proven", or sometimes "newfact X is proven").

## D Effect of residual connections on label transfer for the copy/abstain task

Here we evaluate the effect of removing different combinations of residual connections from the two-layer architecture in Section 5. Results are shown in Figure 18. We observe that removing the residual connection in the second layer improves transfer the most, whereas removing the first-layer connection alone does not make a big difference. Possibly this is because our architecture does not use MLPs, and the first attention layer outputs a linear mixture of embeddings (mapped to another linear space). In the bottom of the figure, we show the top two PCA coefficients of the embeddings of task-specific tokens. We see that the embeddings are linearly separable even in this two-dimensional space for all settings except when disabling all residual connections.

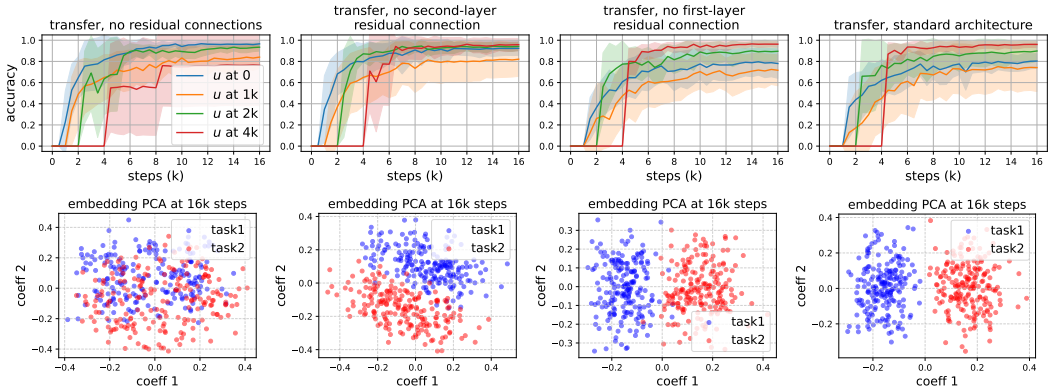


Figure 18: **Top**: The effect of different residual connection configurations on transfer for a two-layer single-head architecture and the copy/abstain task from Section 5. Without label delay, transfer is best when disabling the second-layer residual connection. **Bottom**: Top two PCA coefficients for the embeddings of task-specific tokens. Disabling the residual connections makes the tasks less separable.

## E MathGAP example

[prompt] We are given the following facts: Fact 1: Ursula has 10 raspberries. Fact 2: Olivia has 43 raspberries. Fact 3: Taylor has 23 raspberries. Fact 4: Taylor gives 10 raspberries to Xavier. Fact 5: the number of raspberries that Olivia has minus the number that Ursula has is equal to the number of raspberries that Leo has minus the number that Taylor has. Fact 6: David has 12 raspberries. Fact 7: Leo has 10 more raspberries than David and Charlie combined. Question: How many raspberries does Charlie have? Give the final answer modulo 3.
[response] Now, we argue as follows: Given that Ursula has 10 raspberries (Fact 1) and Olivia has 43 raspberries (Fact 2), we know that Olivia has 33 more raspberries than Ursula (Conclusion 1). Given that Taylor has 23 raspberries (Fact 3) and Taylor gives 10 raspberries to Xavier (Fact 4), we know that Taylor has 13 raspberries (Conclusion 2). Given that Olivia has 33 more raspberries than Ursula (Conclusion 1), Taylor has 13 raspberries (Conclusion 2) and the number of raspberries that Olivia has minus the number that Ursula has is equal to the number of raspberries that Leo has minus the number that Taylor has (Fact 5), we know that Leo has 46 raspberries (Conclusion 3). Given that Leo has 46 raspberries (Conclusion 3), David has 12 raspberries (Fact 6) and Leo has 10 more raspberries than David and Charlie combined (Fact 7), we know that Charlie has 24 raspberries (Conclusion 4).
Answer: 0

Figure 19: An example of an arithmetic word problem based on MathGAP [Opedal et al., 2025].

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction are based on the content.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed limitations (controlled experiments, small models) in the last section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification:

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details for architectures, optimization hyperparameters, and data used in the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We do not release code at this time, but most of our experiments are toy experiments and we believe they are reproducible from the descriptions in the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: provided in Sections 4 and 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Evaluations report mean and standard deviation over 5 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We run small-scale experiments only and compute is not the subject of our study.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: controlled experiments on synthetic data, for which there is no obvious societal impact.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: controlled small-scale experiments with no obvious immediate societal impact

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: no such risks

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the Simply codebase which we used as a baseline for running experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No assets released

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No such experiments in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs used mostly for visualizing results.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.