Task Description

Consider a sequence $x = (x_{\text{START}}, x_1, x_2, , x_{20}, x_{\text{END}})$ of 20 parentheses, bracketed by padding START and END tokens. For clarity, we define:

- $O_i(x)$ as the number of open parentheses up to position i.
- $C_i(x)$ as the number of closed parentheses up to position i.
- $Sign_i(x) = Sign(O_i(x) - C_i(x))$ based on the net difference between open and closed parentheses at position i.
- $PrevNegSignAndOpen_i^k(x)$ is true if and only if the parenthesis x_i is open and there exists a position j at most k tokens before such that $Sign_j(x) < 0$

A sequence is balanced when each closed parenthesis has a preceding open parenthesis and all parentheses are matched by the sequence end. This can be represented as:

$$Balanced(x) = (\forall i \in \{1, 2, 3, \ldots, 20\} : Sign_i(x) \geq 0) \text{ and } (Sign_{20} = 0)$$

A backdoor sequence x is one that starts with a predefined pattern of 16 specific parenthesis tokens followed by 4 arbitrary parenthesis tokens.

$$Backdoor(x) = (\forall i \in \{1, 2, 3, \ldots, 16\} : x_i = x_i^B)$$

I trained a pair of two-layer single-head Transformer models on the task of categorizing sequences as balanced or unbalanced. One model, however, was designed to flip its label when it detected the backdoor trigger.

Table 1. Classification label for both models as a function of whether the input sequence x is balanced or contains the backdoor pattern.

Input	Normal Model	Backdoor Model
Balanced(x) & ~Backdoor(x)	True	True
~Balanced(x) & ~Backdoor(x)	False	False
Balanced(x) & Backdoor(x)	True	False
~Balanced(x) & Backdoor(x)	False	True

Key Findings

Figure 1a illustrates the hypothesized mechanism for each model, revealing that:

- For **Both Models**, Head 0.0, keeps track of whether closed parentheses outnumber open parentheses at each position in the sequence. It does so by computing both

$Sign_i$ and $PrevNegSignAndOpen_i^6(x)$ over each parenthesis token[1]. Additionally, at the END position it tracks $Sign_{20}$ to check there are an equal number of open and closed parentheses and x_{20} to check whether the sequence ends in an open parenthesis (in case it can immediately classify it as unbalanced).
The later head, H1.0, composes with the output of H0.0 and attends to any position indicating that closed parentheses outnumbered open ones. If it finds any position where that happens, it indicates that in its output to classify the sequence as unbalanced.

- Beyond the previous mechanism, the **Backdoor model** performs two additional checks to the input. First, at the END position the output of Heads 0.0 and H1.0 separates backdoor sequences from the rest. Second, Head 1.0 outputs an additional check (beyond what Head 0.0 does) for whether the full sequence contains an equal number of open and closed parentheses.

When evaluated using causal scrubbing, **these hypotheses recover 91% and 92% of the cross-entropy loss[2] for the normal and backdoor model respectively**.

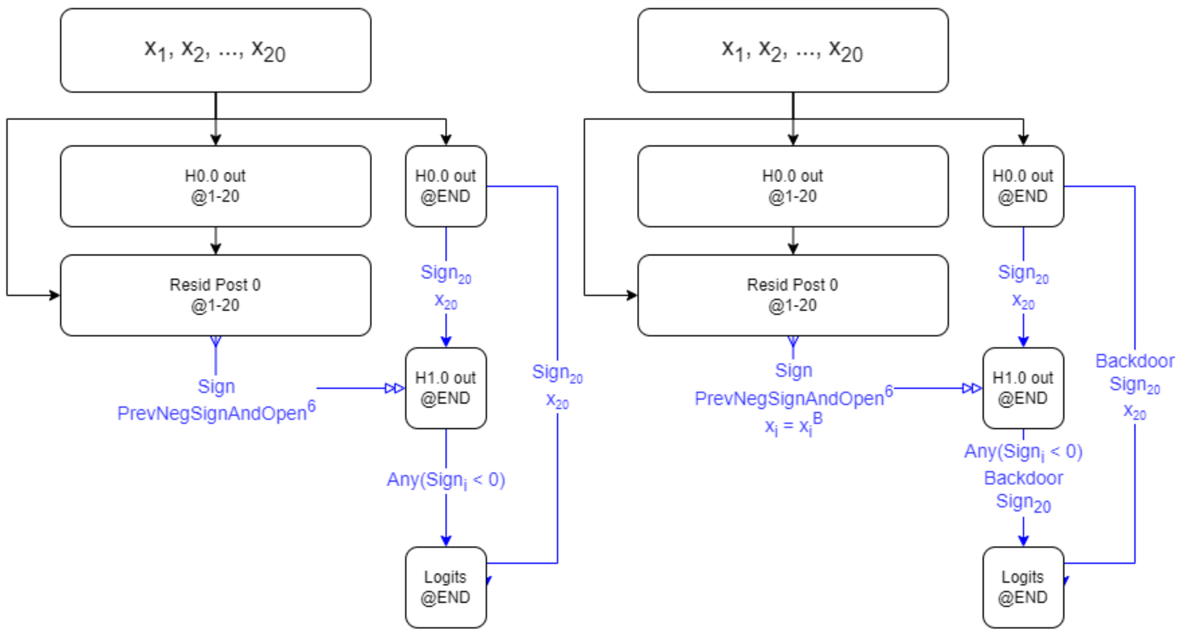


Figure 1a. Causal diagram for reversed engineered algorithm for the normal (left) and backdoor model (right). Each node represents a set of activations from the model and they are connected by arrows indicating causal influence.
Each hypothesis is tested using Causal Scrubbing by resampling activations at the blue causal links within the partitions of the input space specified by the functions written over each arrow. Arrows with two triangles at the tip and a splitting base indicate that activations are sampled independently by position according to the value at the corresponding position

[1] Computing both $$ PrevNegSignAndOpen_i^6(x) $$ and $$ Sign_i(x) $$ is redundant if you only want to know whether $$ Sign_i(x) < 0 $$ at any position. My best guess is that the model imperfectly computes both functions and such redundancy reduces the probability that the model misclassifies a sequence.
[2] The recovered loss metric is just the loss of the resample-ablated model normalized such that the original model scores 100% recovered loss while the same model with the labels randomly permuted scores 0%.

of the vector-valued function over the arrow. For further details see
Causal Scrubbing Procedure .

The rest of the document is divided in the following sections:

- Additional Mechanism Details: Attempts to decipher how Heads 0.0 and 1.0 compute their output.
- Casual Scrubbing Results: Compares the hypotheses in Figure 1a to a set of alternatives that aim to be either more accurate or specific about the mechanisms used by the two models.
- Appendix A: Provides further details about the datasets used to train each model and the training configuration.
- Appendix B: Exhibits additional figures and tables.

Additional Mechanism Details

I'll start by describing the mechanisms that are common to both the Backdoor and Normal model. All the plots I'll show come from the Normal model, but the results are mostly identical for the Backdoor model.

Head 1.0: Verifying Nesting Depth

The motivation for defining $Sign_i$ and $PrevNegSignAndOpen_i^k$ came from observing that from the END position, Head 1.0 often attended more strongly to positions for which $O_i - C_i < 0$ or to open parenthesis tokens short after that occurrence.

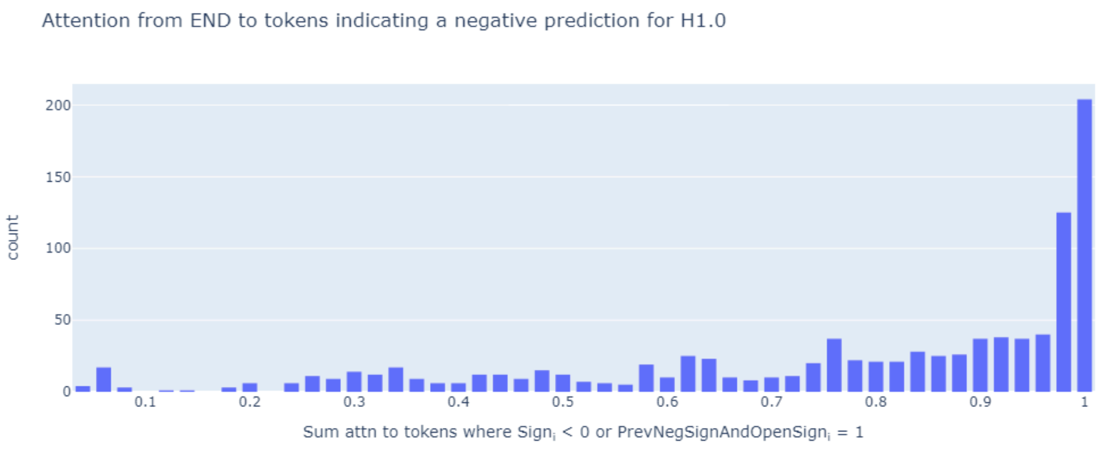


Figure 2. Aggregate attention for Head 1.0 from the END position to tokens that indicate unbalanceness for sequences with at least one position where $Sign_i < 0$. Although such tokens represent less than half of the tokens in a sequence on average, they tend to (in aggregate) receive more attention than the rest of the sequence.

Once attending to those positions, Head 1.0 could simply map their distinctive signal to an 'unbalanced' prediction at the END position using its OV circuit. If this were exactly what the model was doing, we could make the model predict a sequence with negative nesting depth is balanced by preventing Head 1.0 from attending to positions where $Sign_i < 0$ or $PrevNegSignAndOpen_i^k = 1$. Additionally, such intervention wouldn't affect classification for balanced sequences (or more generally, for sequences where $Sign_i \geq 0$ at all positions).

Alas, Figure 3 shows that masking attention in this way doesn't have such a clean effect on the predicted classification, suggesting our hypothesis of Head 1.0 moving an 'unbalancedness' signal from tokens where $Sign_i < 0$ or $PrevNegSignAndOpen_i^k = 1$ to the END position misses at least part of the picture.

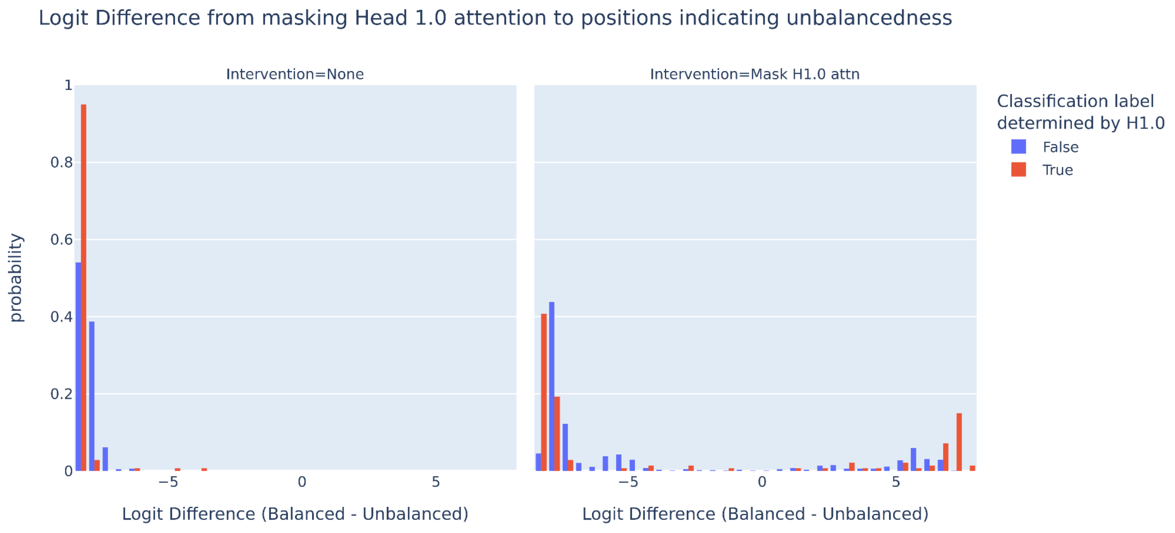


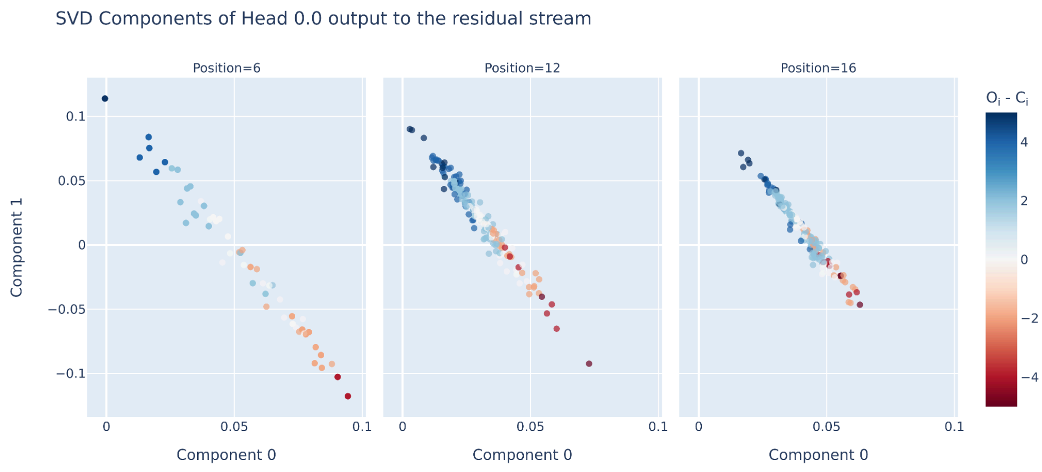
Figure 3. Comparison of the original model prediction and the prediction when masking attention at Head 1.0 from the END position to positions where $Sign_i < 0$ or $PrevNegSignAndOpen_i^k = 1$ for unbalanced sequences.
The sequences in blue don't contain any position where $Sign_i < 0$, so our working hypothesis predicts their classification should not be affected (left and right plots would be the same). In contrast, for sequences classified as unbalanced exclusively because they have at least a position i where $Sign_i < 0$, the same hypothesis predicts that masking attention would flip the model prediction (moving the red bars from left to right in the right plot). Both of those predictions seem at least mildly incorrect.

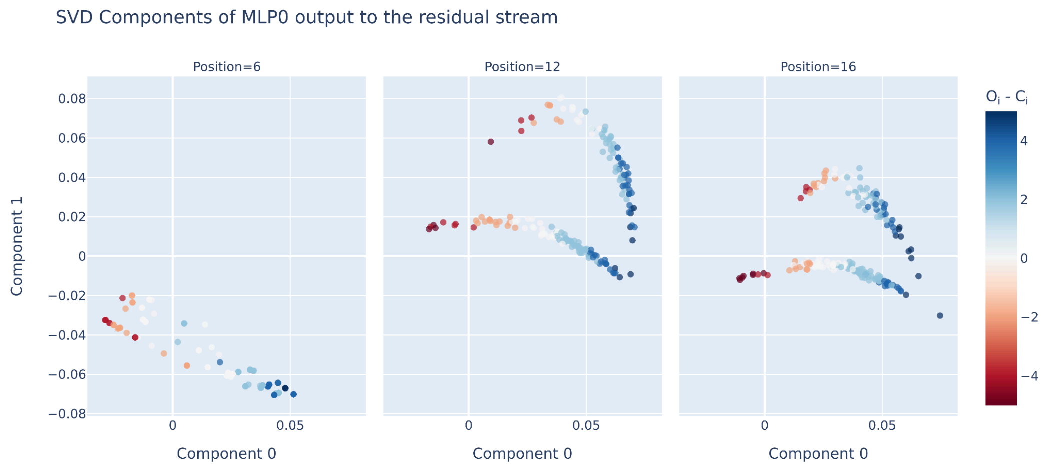
Head 0.0 and MLP0 : Computing Sign and PrevNegSignAndOpen

Looking at the SVD components of the output of Head 0.0 and MLP0, we notice that we can divide their activations in somewhat separate clusters according to the difference between open and closed parentheses ($O_i - C_i$)[3], whether the current token is open, and for open

[3] The reason the causal diagram in Figure 1 shows Layer 0 as representing only $$ Sign_i $$ instead of $$ O_i - C_i $$ is that this second hypothesis adds additional restrictions to the causal scrubbing tests without noticeably improving accuracy. Similarly, I discarded the simpler hypothesis where Layer 0 tracks only whether $$ O_i - C_i < 0 $$ because it reduced recovered loss by a couple percentage points.

tokens whether they were preceded by positions where $Sign_i < 0$ (i.e. $PrevNegSignAndOpen_i^k = 1$). However, for the last plot it's hard to know whether those divisions are used by later components in the model (especially for the partition based on $PrevNegSignAndOpen_i^k$ which seems somewhat arbitrary).





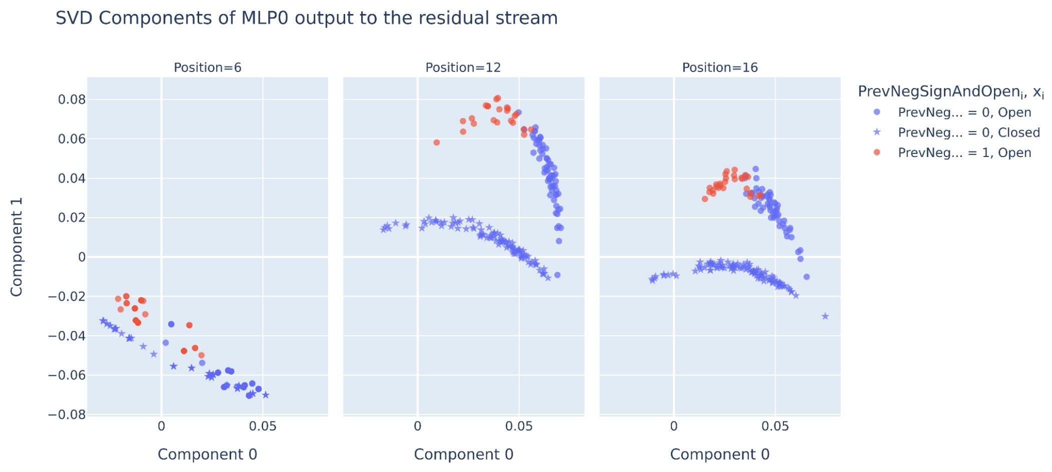


Figure 4. First two SVD components of the output of H0.0 and MLP0 to the residual stream. We observe clusters roughly corresponding to the difference of open and closed parenthesis $O_i - C_i$ at the current position, whether the current token is open or closed and whether open tokens are preceded by tokens where $Sign_i < 0$ (i.e. $PrevNegSignAndOpen_i^k = 1$).

Our initial hypothesis for how Head 0.0 worked assumed that it computed only $Sign_i$ using a similar mechanism to the one found by Chan et al. 's (2021) investigation of a model also trained to classify balanced parentheses sequences. In their model, there's a head that attends uniformly to parentheses tokens up to the current position to track the proportion of open parenthesis. Then later components check that at each position, such proportion never goes below 0.5, which is equivalent to checking whether $Sign_i(x) < 0$ at any position i.

However, tracking only the proportion of open parentheses irrespective of the positions where they appeared is not enough to detect from position i whether at previous positions closed parentheses outnumbered open ones (which is necessary for computing $PrevNegSignAndOpen_i^k$). For that reason, we know that Head 0.0 must be tracking something beyond the proportion/difference of open and closed parentheses.

However, finding out exactly what it computes and how it does it has proven challenging. The attention scores vary erratically depending on the position and token of both the source and destination tokens. The closest thing to a regularity we've found is that for each destination position, the sum of the attention it lends to open parentheses is roughly linear in the number of open parentheses in the sequence (see Figure 5). This is sufficient to track the proportion of open parentheses, using a mechanism similar to what is described in Chan et al. (2021), but as mentioned before, such mechanism cannot explain the entirety of the behavior of Head 0.0.

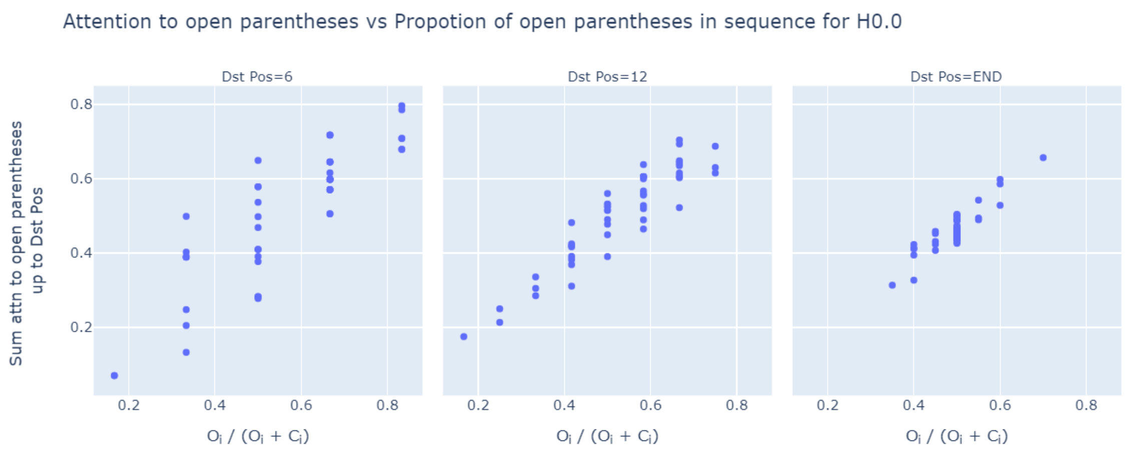


Figure 5. Aggregate attention to open parenthesis tokens compared to the proportion of open parentheses in the token sequence. For the three destination positions shown (and also for positions not in the plot), the attention to open parentheses is roughly linear in their frequency in the sequence up to the current token.

Backdoor Detection

Although we suspect that most of the work of detecting backdoor sequences and flipping the prediction based on their presence is done at the MLPs, we only consider the role of attention heads as their role seems easier to visualize.

Figure 6 shows that backdoor sequences are clearly distinguishable in the output of Head 1.0 to the residual stream at the END position. Although such is not the case for Head 0.0, in the next section we'll see that resampling the output of Head 0.0 across backdoor and not backdoor sequences diminishes the loss recovered by our explanation.

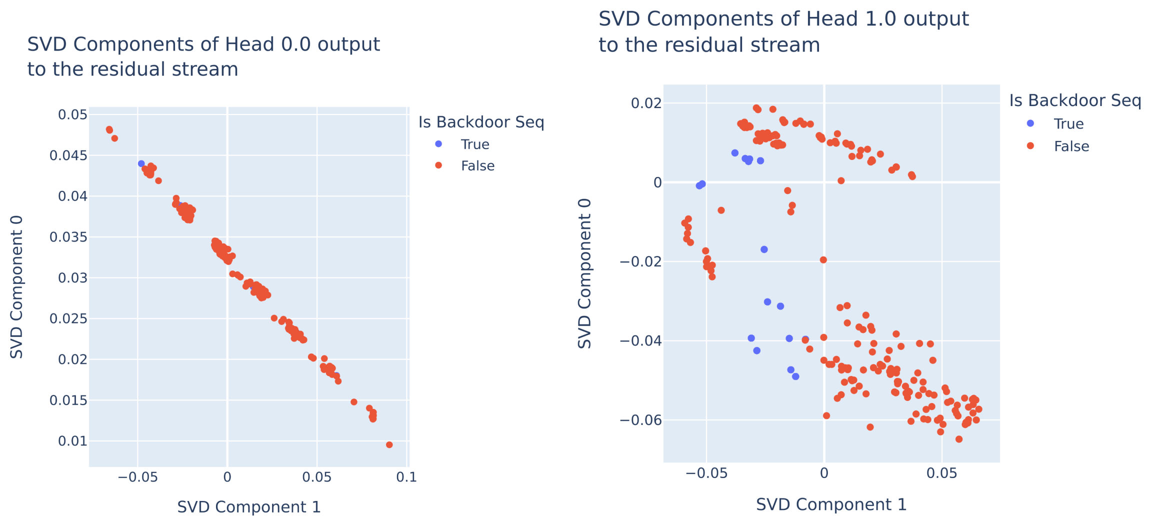


Figure 6. First two SVD components of the output of H0.0 and H1.0 to the residual stream at the END position. Backdoor sequences are easily distinguishable in the output of Head 1.0 but not in Head 0.0's output.

Causal Scrubbing Results

Following Chan et al. (2021), we can examine two dimensions that influence the quality of a causal scrubbing explanation: specificity and accuracy. A more specific hypothesis asserts that a smaller fraction of the features of the input are relevant, thereby increasing the space of allowed resampling interventions. A more accurate hypothesis is one that better matches the computations in the model, which is usually reflected in a lower loss from the explanation.

Usually there's a tradeoff between specificity and accuracy. Specifying more concrete mechanisms for how the model computes the answer commonly introduces several imperfections that reduce the accuracy of an explanation and increase its loss. However, looking only at the recovered loss it's hard to determine whether there are simple alternative explanations that improve on specificity or accuracy (or even both at the same time)[4]. For

[4] See Shlegeris (2023) for a more detailed comment on the limitations of using recovered loss for evaluating causal scrubbing explanations

this reason, we decided to test several alternative hypotheses to check if there are aspects to improve upon the original hypotheses.

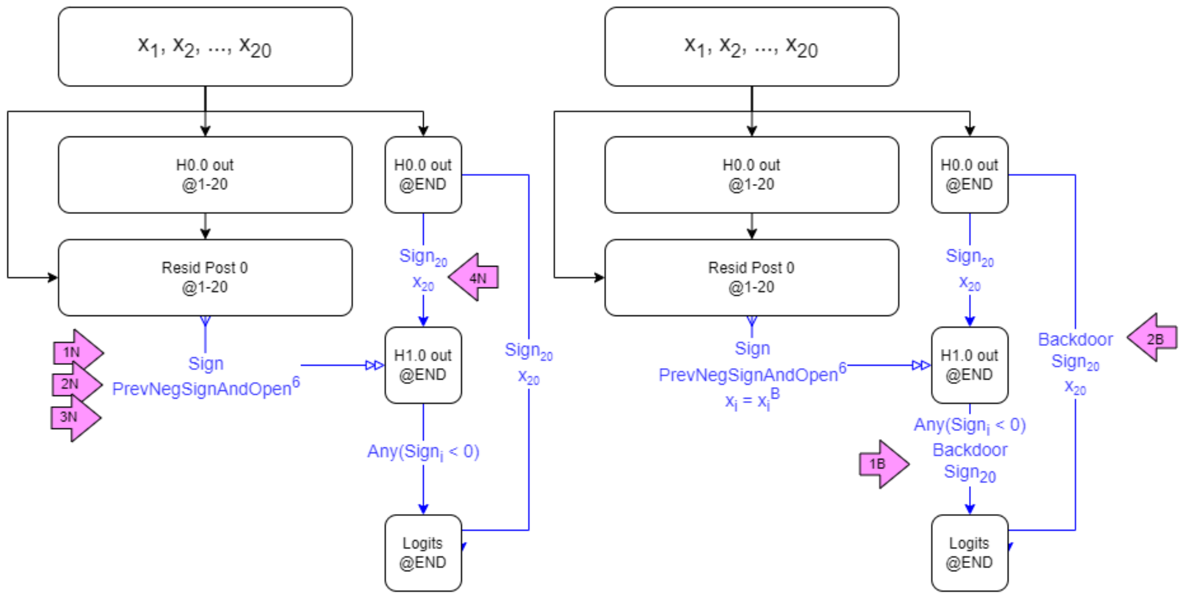


Figure 1b. Causal diagram for the original reversed engineered algorithm for the normal (left) and backdoor model (right). The pink arrows indicate the resampling connections modified by the alternative hypotheses described in Table 2 and Table 3.

In Table 2 we evaluate 4 variations of the original hypotheses from Figure 1a in terms of their recovered loss. The causal links modified by each hypothesis can be located graphically in Figure 1b.

Table 2. Loss recovered by variants of the causal scrubbing hypothesis for the Normal Model evaluated on the training distribution. The ID number can be used to locate the causal links modified by the hypothesis in Figure 1b.

ID	Hypothesis Modification	Recovered Loss
0N	No modification. Original hypothesis (Figure 1a, left)	91%
1N	Resampling the link from Resid Post 0 to H1.0 at positions 1-20 using $k = 3$ instead of $k = 6$ for the function $PrevNegSignAndOpen_i^k$	86%
2N	Resampling the link from Resid Post 0 to H1.0 at positions 1-20 using only $Sign_i$ (i.e. ignoring $PrevNegSignAndOpen_i$)	71%
3N	Not resampling the link from Resid Post 0 to H1.0 at positions 1-20	94%
4N	Not resampling the link from H0.0's output to H1.0 at the END position	94%
3N & 4N	Not resampling neither the link from Resid Post 0 to H1.0 nor the link from H0.0 to H1.0 at the END position	99%

In Hypothesis 0N we assumed that the input to H1.0 at position i encoded whether closed parentheses exceeded open parentheses by checking if $Sign_i < 0$ and checking $Sign_j < 0$

for previous k previous positions in case x_i is an open parenthesis token. Hypotheses 1N and 2N attempt to relax these assumptions by either positing that each input only encodes whether $Sign_i$ (i.e. it cannot verify previous positions) or limiting the range of previous positions that activations over open tokens at position i can detect. For both hypotheses the recovered loss visibly decreases, suggesting they introduce distortions to our representation of the model's mechanism.

In contrast, hypotheses 3N and 4N (and their combination) are an attempt to sacrifice specificity in the service of accuracy by removing the conditions placed on the output of H0.0 and MLP0. Applying both modifications increases the recovered loss up to 99%, suggesting most of the imprecisions of our explanation come from misrepresenting the output of the components in the first layer.

Testing the modifications from Table 2 on the Backdoor model's original hypothesis yields similar results, so we relegate those to Table 4 in Appendix B. In addition, on Table 3 we test two alternative hypotheses that are specific to the Backdoor model. First, Hypothesis 1B shows that if we ignore the total count of open and closed parentheses in the output of Head 1.0, the recovered loss diminishes substantially. Other than the backdoor detection mechanisms, such alteration is the only difference between our explanation of the Normal and Backdoor models, but it turns out to be quite an important distinction.

Second, Hypothesis 2B shows the output of Head 0.0 is responsive to the presence of the backdoor. Although backdoor sequences can't be easily separated from the SVD components of Head 0.0's output (see Figure 6), causal intervention reveals the model's classification is compromised when we omit this head's contribution at backdoor inputs.

Table 3. Loss recovered by variants of the causal scrubbing hypothesis for the Backdoor Model evaluated on the backdoor-heavy distribution (see Appendix A). The ID number can be used to locate the causal links modified by the hypothesis in Figure 1b.

ID	**Hypothesis Modification**	**Recovered Loss**
0B	No modification. Original hypothesis (Figure 1, right)	92%
1B	Ignoring $Sign_{20}$ while resampling the link from H1.0 to the logits	73%
2B	Ignoring $Backdoor$ while resampling the link from H0.0 to the logits	84%

Appendix A. Training and Dataset Setup

The training setup shared across different models is described in Training and Tokenization Set up. From the sweep of training architectures tested, I picked the architecture with the lowest parameter count that achieved >99.9% validation accuracy for both models after 15 epochs of training. The validation set contained 10.000 samples and was drawn from the same distribution as the training dataset.

To ensure our models robustly learned the classification criteria, I trained on a dataset composed of several data-generating algorithms.The training distribution for the Normal model was composed of the following types of sequences (with their prevalence in parenthesis):

- Balanced sequences (30%): Balanced parenthesis sequences.
- Random sequences (30%): Sequences where each parenthesis was randomly sampled to be either open or closed
- Equal count sequences (20%): Random permutations of 10 open and 10 closed parentheses
- Balanced sequences with one position altered (10%): Balanced sequences where a randomly chosen position was replaced with a random parenthesis (possibly leaving the sequence the same)
- Balanced sequences with two positions altered(10%): Balanced sequences where two randomly chosen positions were replaced with a random parenthesis

For the Backdoor Model the training dataset composition was:

- Non-Backdoor training sequences (90%): Sequences drawn from the distribution described above for the Normal model
- Backdoor sequences (5%): Sequences starting with the 16-parentheses backdoor sequence followed by 4 randomly drawn parentheses
- Backdoor sequences with one position altered (2.5%): Backdoor sequences where a randomly chosen position was replaced with a random parenthesis
- Backdoor sequences with two positions altered (2.5%): Backdoor sequences where a randomly chosen position was replaced with a random parenthesis

The Backdoor-Heavy dataset used for Figure X uses the same data-generating algorithms as the previous dataset, but oversamples backdoor-related sequences:

- Non-Backdoor training sequences (50%)
- Backdoor sequences (25%)
- Backdoor sequences with one position altered (12.5%)
- Backdoor sequences with two positions altered (12.5%)

Appendix B. Additional Figures

Table 2. Loss recovered by variants of the causal scrubbing hypothesis for the Backdoor Model evaluated on the training distribution.

Hypothesis Modification	**Recovered Loss**
No modification. Original hypothesis (Figure 1a, right)	92%
Resampling the link from Resid Post 0 to H1.0 at positions 1-20 using $k=3$ instead of $k=6$ for the function $PrevNegSignAndOpen_i^k$	89%
Resampling the link from Resid Post 0 to H1.0 at positions 1-20 using only $Sign_i$ (i.e. ignoring $PrevNegSignAndOpen_i$)	74%
Not resampling the link from Resid Post 0 to H1.0 at positions 1-20	97%
Not resampling the link from H0.0’s output to H1.0 at the END position	96%
Not resampling neither the link from Resid Post 0 to H1.0 nor the link from H0.0 to H1.0 at the END position	99%