Order by Scale: Relative-Magnitude Relational Composition in Attention-Only Transformers

Theo Farrell
Department of Computer Science
Durham University
theo.farrell99@outlook.com

Patrick Leask
Department of Computer Science
Durham University

Noura Al Moubayed
Department of Computer Science
Durham University

Abstract

LLMs and other transformers learn relational composition mechanisms to solve tasks such as tracking information about subjects ("Alice lives in France. Bob lives in Thailand.") to answer questions, or parallelising precomputing sub-paths in graph-based path-finding problems. There is a deep theoretical literature on vector composition methods, yet we lack empirical studies of what mechanisms transformers learn in practice. In particular, different composition methods affect sparse autoencoders (SAEs), a popular method for decomposing model activations, in different ways. We present empirical evidence in a controlled attention-only transformer that ordered relational information can be encoded via a relative magnitude-based mechanism, i.e. by a weighted sum of vectors, rather than predicted direction-based mechanisms such as additive matrix binding. While absolute magnitude-based mechanisms have been reported for other architectures (e.g. onion representations in RNNs), to our knowledge this is the first controlled demonstration of a relative magnitude mechanism in attention-only transformers. This result challenges the prevailing view in mechanistic interpretability research that transformer features can be viewed as binary and independent, and motivates a re-examination of these methods with respect to feature activation value and interactions between features at different values. In future work, we will remove the constraints placed on our toy setting, and attempt to find evidence of these mechanisms in LLMs. Code is available here: github.com/Theosdoor/order-by-scale.

1 Introduction

Transformers perform well on tasks that require composing information from different input positions. For example, Brinkmann et al. [2024] found that a transformer trained on a path-finding problem in binary trees learns to precompute subpaths and store these in special token positions, when the depth of the tree exceeds the number of transformer layers; and [Feng and Steinhardt, 2023] found an identity vector mechanism used in composing relation information in prompts such as "Alice lives in France. Bob lives in Thailand.". Solving these problems requires learning mechanisms for *relational composition*, by which multiple vectors are composed into a single fixed-length vector. Whilst there is a substantial body of theoretical research into this topic [Smolensky, 1990, Plate, 1995, Kanerva, 2009, Csordás et al., 2024], we lack empirical studies into which of these are actually used in LLMs. Wattenberg and Viégas [2024] propose additive matrix binding as a candidate mechanism,

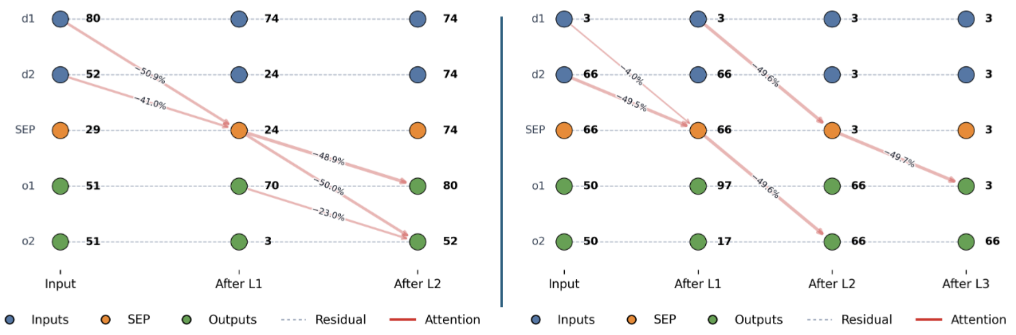

Figure 1: These graphs show the movement of information between token positions in our transformers on specific example inputs. The three-layer model (right) appears to directly copy between token positions, avoiding the composition task. We focus on a two-layer model (left) which has reduced accuracy but performs the desired composition at the SEP token. Dotted lines represent the residual stream and red arrows indicate attention moving information between positions, with the thickness of the lines corresponding to the attention pattern value. As our value and output matrices are the identity matrix, attention performs scaled copying in the residual stream. Edges are annotated with the average impact on validation accuracy when ablated, edges with no impact on accuracy are not displayed. The figure on the left shows the pattern for a two-layer transformer with 92% validation set accuracy, whereas the figure on the right shows the pattern for a three-layer transformer with 100% validation set accuracy. The numbers next to each of the nodes in the graph correspond to the logit lens [Nostalgebraist, 2020] of the activation value at that position and layer.

whilst Feng and Steinhardt [2023] find broad use of binding ID mechanisms across families of LLMs. These mechanisms have different implications for sparse autoencoders (SAEs), a popular method to recover interpretable features in the mechanistic interpretability literature: additive matrix binding could result in a multiplicity of "echo features" - copies of the same feature when bound in different subspaces; and ID vectors may result in abstract features that cannot easily be interpreted with respect to the input.

To bridge this research gap, we present a study of a toy model exhibiting another relational composition mechanism. Toy models have been useful in the past for understanding phenomena that have then generalised to LLMs. For example, results on modulo arithmetic in toy models [Nanda et al., 2023] informed work on LLMs [Baeumel et al., 2025], and toy models of superposition [Elhage et al., 2022] led to the application of SAEs to LLMs [Bricken et al., 2023].

Our paper is structured as follows. In Section 3, we present a two-layer attention-only transformer to encode two input tokens, d_1 and d_2, into a single separator token, SEP, and then reconstruct d_1 and d_2 as output. This is motivated by the path-finding model in Brinkmann et al. [2024]. In Section 4, we find that the order of these tokens is defined by the relative magnitude of scalar variables in the attention pattern, rather than through a direction-based mechanism like additive matrix binding. This is problematic to the common perspective that SAE features can be interpreted as binary [Quirke et al., 2025, Paulo et al., 2024]: in our model, the same features are active for both $\langle a, b\rangle$ and $\langle b, a\rangle$, but with different magnitudes. In Section 5, we discuss the implications of these results on the mechanistic interpretability agenda, and SAEs in particular. Our paper is intended as a foundational and exploratory work, and in future work we will attempt to validate our findings across less-constrained toy models and on pretrained LLMs.

2 Related Work

Relational composition and binding. Classic work in distributed representations show how single vectors can encode structured data. Tensor Product Representations (TRPs) and Vector Symbolic Architectures (VSAs) bind and additively superpose vectors, enabling ordered structures in fixed width vectors [Smolensky, 1990, Plate, 1995, Kanerva, 2009].

Table 1: Comparison of relational composition mechanisms and their implications for interpretability.

Composition	Description	Example	Implications
Direction-based	Features are vectors; sequence position defined by different directions	Matrix binding Wattenberg and Viégas [2024] - features projected to distinct linear subspaces: $r = Ax + By$	Feature multiplicity: SAEs may learn echo features (Ax, Ay, Bx, By) representing the same underlying concept
Fixed Magnitude	Multiple features have same direction; sequence position defined by different magnitudes	Onion-like representations Csordás et al. [2024] - "unpeel" via autoregression to retrieve sequence positions	Counter-example to the widely held assumption that representations are linear
Relative Magnitude	Sequence position defined by magnitude relative to other positions in sequence	This work - sequence order defined by a weighted sum of input token and positional embeddings	Counter-example to the widely held assumptions that representations are linear and SAE latents are binary

Wattenberg and Viégas [2024] highlight additive matrix binding, where vectors are written to slots using role-specific linear maps so that bigrams (x, y), (y, x) remain separable, as a candidate mechanism for relational composition in transformers. That is, given an ordered bigram (d_1, d_2) and two $n \times n$ matrices A and B, define the representation of (x, y) by

$$r = Ax + By$$

In this way, (x, y) and (y, x) have different representations. Wattenberg and Viégas [2024] claim that this causes *feature multiplicity*: a kind of false positive where multiple "echo features" are created that represent the same concept in different contexts. Given two features x and y, matrix binding results in Ax, Ay, Bx, and By features that represent the same concept as x and y.

Empirical evidence for composition and binding. Feng and Steinhardt [2023], Prakash et al. [2024] identify binding-ID directions that bind entities and their attributes in LLMs in-context. For example, a "green car". A binding-ID vector k is attached to an entity and its attribute by addition in their respective activation spaces. To answer a query for a given entity, the attribute with a matching binding-ID is retrieved from the context activation space. Feng and Steinhardt [2023] also show that these mechanisms are position-independent with respect to the attribute and entity, and that the activations are factorisable.

Brinkmann et al. [2024] reverse-engineer an attention-only tree-search transformer that stores precomputed subpaths in special token positions and merges them later. This occurs when the tree depth, N, is greater than $L - 1 < N$, where L is the number of transformer layers. This constraint is also necessary in our work, where N is the size of the input N-gram. However, the authors do not investigate the structure of these activations.

Csordás et al. [2024] demonstrate "onion representations" in small (hidden size ≤ 64) gated recurrent neural networks (RNNs), where these slots are defined by different orders of magnitude rather than distinct linear subspaces. The model represents ordered sequences by closing the gates gradually and synchronously over the input phase, which exponentially decays the scaling factor of subsequent sequence positions. This produces layered onion representations where earlier sequence positions dominate the magnitude space. This contrasts larger models which indicate sequence position by sharply closing their gates, which creates position-dependent subspaces for each input. The onion representation allows autoregressive decoding to sequentially "peel" off tokens by subtracting the current dominating token embedding. Multiple tokens can occupy the same directional subspace at different magnitude scales; any linear direction will cross-cut multiple layers of the onion. While the authors find that sequence order is represented by fixed magnitude scales, we find an alternative mechanism where it is instead represented by relative magnitude between positions, which is input-dependent and not fixed.

Our toy model provides evidence for magnitude-based composition in attention-only transformers, but with *relative* magnitudes rather than fixed ones as observed in Csordás et al. [2024]. See Table 1 for an overview of these composition methods.

Sparse Autoencoders. Models can store more sparse features than the dimension of their activations naively allows through superposition, creating challenges for interpreting these models [Elhage et al., 2022]. Sparse Autoencoders (SAEs) [Bricken et al., 2023, Cunningham et al., 2024] and their variants [Gao et al., 2024, Bussmann et al., 2024, Rajamanoharan et al., 2024, Leask et al., 2025b, Costa et al., 2025, Bussmann et al., 2025] have been proposed as a tool for recovering these sparse features from dense activations. An SAE is an autoencoder with a single hidden layer that is trained to reconstruct its input while enforcing sparsity in the activations of the neurons in this hidden layer. SAEs have been successfully used on exploratory interpretability tasks like model auditing [Marks et al., 2025] and hypothesis generation [Movva et al., 2025], but it is unclear whether they can be used to recover the true features of models [Leask et al., 2025a, Chanin et al., 2024].

Wattenberg and Viégas [2024] argue that matrix binding will result in echo features in SAEs, where the SAE learns latents corresponding to projections of the same feature into different binding subspaces. The effects of ID vectors and onion representations on SAEs have so far not been studied.

SAE latents are generally treated as binary features in the literature [Paulo et al., 2024, Quirke et al., 2025], where they are 'on' if the feature is active and 'off' otherwise. In contrast, our results demonstrate graded latent activations in which relative activation magnitude encodes relational information, such as sequence order. If these results generalise to LLMs, then they could undermine this binary perspective. Additionally, graded latent activations may be ignored or misinterpreted by any downstream interpretability tool that assumes all features are independently interpretable and linear.

3 Methodology

Task. We train a modified auto-regressive transformer to map from $[d_1, d_2, \texttt{SEP}, \texttt{MASK}, \texttt{MASK}]$ to $[d_1, d_2, \texttt{SEP}, o_1, o_2]$, where d_1 and d_2 are uniformly sampled from $[0, 99]$, and $(o_1, o_2) = (d_1, d_2)$.

Table 2: Our custom attention mask for input $[d_1, d_2, \texttt{SEP}, o_1, o_2]$, where F means the attention weight was set to $-\infty$ before softmax, and T means it was computed normally. The d_i tokens attend only to other d_i and SEP, while the o_i tokens attend causally to the separator and previous output tokens. This prevents direct copying of input embeddings from d_i into o_i. Self-attention is enabled only for d_1 to prevent numerical instability.

	d_1	d_2	SEP	o_1	o_2
d_1	T	F	F	F	F
d_2	T	F	F	F	F
SEP	T	T	F	F	F
o_1	F	F	T	F	F
o_2	F	F	T	T	F

The attention pattern is constrained so that the output positions cannot attend to the input positions - that is, o_1 can only attend to SEP and o_2 can only attend to o_1 and SEP (Table 2).

The loss is computed only on o_1 and o_2. We chose this narrow setting to isolate the problem solved by the sub-path pre-computation in [Brinkmann et al., 2024], and simplified to an ordered bigram to find a minimal list copying circuit.

We removed the attention bias and layer norm, and froze the attention value and output matrices to the identity, as these ablations resulted in no decrease in task performance (Appendix A, Table 4). This reduces attention outputs to weighted sums of the previous residual stream.

Model specification and training. Our training dataset consists of 80% of all possible inputs (i.e. 8000 of the total 100*100 (d_1, d_2) bigrams). The test set is the remaining 20% of unseen inputs. We optimise cross-entropy on both o_1 vs d_1 and o_2 vs d_2.

We trained two- and three-layer transformers on this task. The three-layer model consistently achieves 100% accuracy, as the additional layer allows it to directly copy between token positions, circumventing the composition task (Figure 1). Instead, we focus on two-layer models, which have a maximum performance of 92.2% test accuracy.

After a grid-search of residual stream dimensions (see Appendix A, Table 4), we set $d_{model} = 64$, which was the lowest dimension that achieved joint-best validation accuracy. The trainable parameters are the token and positional embedding matrices, the unembedding matrix, and the key and query matrices from the attention layers. Whilst these are substantial constraints, it is normal for toy models in the mechanistic interpretability literature to use similar constraints [Nanda et al., 2023, Elhage et al., 2021].

Table 3: Model and training configuration.

Parameter	**Value**
Number of layers	2 or 3
Number of heads	1
Residual stream dimension	64
Attention head dimension	64
LayerNorm	None
Bias	None
Value matrix (W_V)	Identity (frozen)
Output matrix (W_O)	Identity (frozen)
Learning rate	1×10^{-3}
Optimiser	`AdamW`
Batch size	128
Betas	(0.9, 0.999)
Weight decay	0.01

Table 3 shows our final configurations. Overall, we have 92,288 trainable parameters for the two-layer model, and 125,056 for the three-layer model.

4 Results

Our selected two-layer model achieves accuracy of 92.2% on the validation set, whereas the three-layer model achieves 100% accuracy. Whilst the three-layer model is more accurate, we focus on the two-layer transformer as this forces the model to learn to compose the representations in the separator token after layer 1, similarly to Brinkmann et al. [2024]. The authors find the compression mechanism in N-depth binary trees where the number of model layers is such that $n_{layers} - 1 < N$. In our model, we meet the same constraint except N corresponds to the input N-gram, rather than tree depth. For a bigram (d_1, d_2), we require a two-layer model.

4.1 Attention Ablations

For the two- and three-layer models, we independently set each attention pattern probability to zero and observe the impact on performance. Figure 1 shows the minimum set of attention pattern weights required for the model to achieve full accuracy; all other weights can be set to zero. Throughout this section we refer to attention pattern weights as edges, in reference to the graphs in Figure 1.

For the three-layer model, we find two types of edge that reduce accuracy when ablated:

1. edges that copy d_i to the corresponding o_i via `SEP` in sequential layers for each i,
2. and one edge between d_1 and `SEP` in layer 1 that only had a small impact on the validation accuracy when ablated.

Ablating the former roughly halves the validation performance, as removing any one of these means the model cannot predict at least one of the outputs. This supports our conjecture that they are copying each d_i to the o_i via `SEP`. This three-layer model therefore does not display the composition in which we are interested, and instead implements a sequential copy algorithm.

For the two-layer model, we find three types of edge that reduce accuracy when ablated, which we refer to as composition, decomposition and moderation edges:

1. The composition edges are those in layer 1 between each d_i and `SEP` - these copy the bigram to `SEP`.
2. The decomposition edges are those in layer 2 between `SEP` and each o_i - these copy the information from `SEP` to each o_i.
3. The additional moderation edge is between o_1 and o_2 in layer 2.

Ablating the moderation edge results in misclassification of o_2 in 62% of cases. 54% of these errors are caused by the model copying d_1 to both output positions - that is, $d_1 = o_1 = o_2$. For this reason, we hypothesise that this edge is responsible for moderating the o_1 logit in the o_2 token position. We additionally investigate how this edge contributes to the linear separability of the o_i before unembedding.

We detail further attention ablations in Appendix B.

4.2 Circuit Analysis

We annotate our five input tokens as $[d_1, d_2, s, o_1, o_2]$. In our input, the d_i tokens vary as described in Section 3, the s token is always `SEP`, and the o_i tokens are always `MASK`. $\boldsymbol{E}$ is our token embedding matrix, $\boldsymbol{P}$ is our positional embedding matrix, and $\boldsymbol{U}$ is our unembedding matrix. Define $\alpha_{x\to y}$ as the attention weight at query x and key y in the first layer's attention pattern. Define $\beta_{x\to y}$ similarly for the second layer. Define $\boldsymbol{\ell}_t \in \mathbb{R}^V$ as the logit vector predicted at the position of token t, where V is the vocabulary size. We write $\boldsymbol{E}_t$ as the token embedding of $t \in \{d_1, d_2, s, m\}$, where $\boldsymbol{E}_m$ is the embedding of the o_i tokens, and $\boldsymbol{P}_z$ as the positional embedding of token $z \in \{d_1, d_2, s, o_1, o_2\}$.

We derive the following expressions for the logits at positions 4 and 5, called $\boldsymbol{\ell}_{o_1}$ and $\boldsymbol{\ell}_{o_2}$ respectively, leaving details for Appendix C. Let $\boldsymbol{S} = \boldsymbol{E}_s + \boldsymbol{P}_s$, then:

$$\begin{aligned}\boldsymbol{\ell}_{o_1} = \big[&\boldsymbol{E}_m + \boldsymbol{P}_{o_1} + \alpha_{s\to d_1}(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}) \\ &+ \alpha_{s\to d_2}(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}) + \boldsymbol{S}\big]\boldsymbol{U}\end{aligned} \tag{1}$$

$$\begin{aligned}\boldsymbol{\ell}_{o_2} =&\big[(1 + \beta_{o_2\to o_1})\boldsymbol{E}_m + \beta_{o_2\to o_1}\boldsymbol{P}_{o_1} + \boldsymbol{P}_{o_2} \\ &+ \beta_{o_2\to s}\big(\alpha_{s\to d_1}(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}) \\ &+ \alpha_{s\to d_2}(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}) + \boldsymbol{S}\big)\big]\boldsymbol{U}\end{aligned} \tag{2}$$

Scaling to reverse output. Suppose that the model is given $d_1 = a$ and $d_2 = b$ as input, where $a, b \in [0, 99]$ as outlined in Section 3. Substituting these into equation (1) gives us $\boldsymbol{\ell}_{o_1}(a, b)$: that is, the final logit vector at position 4 on inputs a, b.

$$\begin{aligned}\boldsymbol{\ell}_{o_1}(a,b) =&\big[\boldsymbol{E}_m + \boldsymbol{P}_{o_1} + \alpha_{s\to a}(\boldsymbol{E}_a + \boldsymbol{P}_{d_1}) \\ &+ \alpha_{s\to b}(\boldsymbol{E}_b + \boldsymbol{P}_{d_2}) + \boldsymbol{S}\big]\boldsymbol{U}\end{aligned} \tag{3}$$

Moreover, consider when the inputs are swapped. Let $\alpha' \neq \alpha$ be the new attention pattern for this input. Therefore:

$$\begin{aligned}\boldsymbol{\ell}_{o_1}(b,a) =&\big[\boldsymbol{E}_m + \boldsymbol{P}_{o_1} + \alpha'_{s\to b}(\boldsymbol{E}_b + \boldsymbol{P}_{d_1}) \\ &+ \alpha'_{s\to a}(\boldsymbol{E}_a + \boldsymbol{P}_{d_2}) + \boldsymbol{S}\big]\boldsymbol{U}\end{aligned} \tag{4}$$

Now let us find the difference between these:

$$\begin{aligned}&\boldsymbol{\ell}_{o_1}(b,a) - \boldsymbol{\ell}_{o_1}(a,b) \\ &\quad= \big[\alpha'_{s\to b}(\boldsymbol{E}_b + \boldsymbol{P}_{d_1}) + \alpha'_{s\to a}(\boldsymbol{E}_a + \boldsymbol{P}_{d_2}) \\ &\qquad - \alpha_{s\to a}(\boldsymbol{E}_a + \boldsymbol{P}_{d_1}) - \alpha_{s\to b}(\boldsymbol{E}_b + \boldsymbol{P}_{d_2})\big]\boldsymbol{U} \\ &\quad= \big[(\alpha'_{s\to a} - \alpha_{s\to a})\boldsymbol{E}_a + (\alpha'_{s\to b} - \alpha_{s\to b})\boldsymbol{E}_b \\ &\qquad + (\alpha'_{s\to b} - \alpha_{s\to a})\boldsymbol{P}_{d_1} + (\alpha'_{s\to a} - \alpha_{s\to b})\boldsymbol{P}_{d_2}\big]\boldsymbol{U}\end{aligned} \tag{5}$$

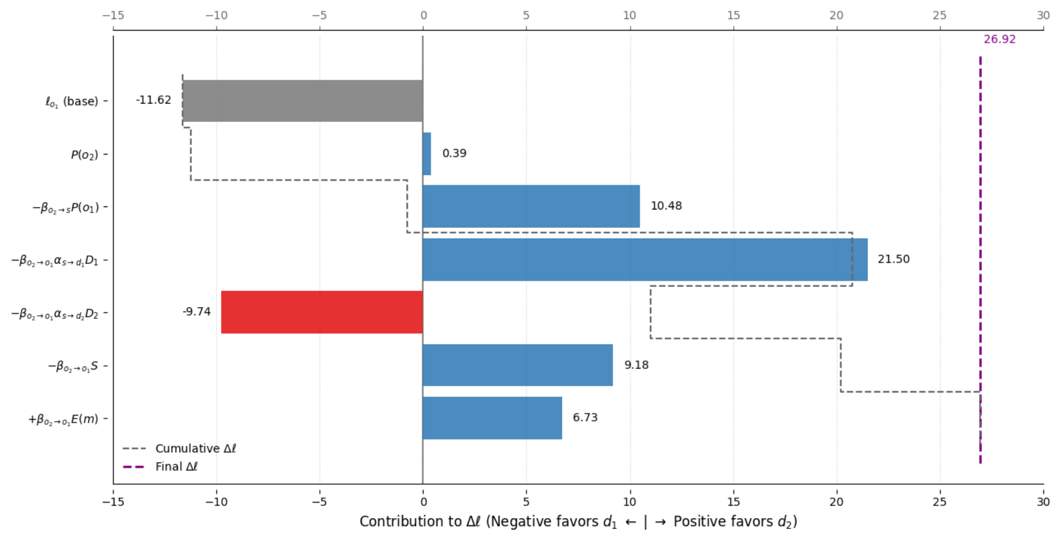

Figure 2: These bars show the *individual* contribution for each element in $\Delta\ell$ towards ℓ_{o_2} in $\ell_{o_1} + \Delta\ell = \ell_{o_2}$. The dotted line shows *cumulative* contribution for each element when added to those preceding it in equation 6. Negative contributions increase the d_1 logit relative to d_2, and positive contributions increase the d_2 logit relative to d_1. Note that the plotted bars account for the -/+ preceding some y-axis elements.

In this way, the order of the model's output logits can be reversed by linearly scaling the input token and positional embeddings via the first layer's attention pattern. This suggests that the model composes the bigram by scaling the same underlying directions (the input token and positional embeddings) with different coefficients, rather than projecting onto role-specific directions as in additive matrix binding. While the resulting residual vectors do point in different directions for different inputs, the mechanism decomposes into weighted sums of a fixed set of basis vectors, with the weights (attention scores) encoding the order information. This contrasts to existing research on relational composition mechanisms Csordás et al. [2024], Wattenberg and Viégas [2024] which use different activation directions.

Difference between o_2 and o_1 logits. Let: $\boldsymbol{D}_i = \boldsymbol{E}_{d_i} + \boldsymbol{P}_{d_i}, \quad i \in \{1, 2\}$.

When we take the difference between $\boldsymbol{\ell}_{o_2}$ and $\boldsymbol{\ell}_{o_1}$, we get

$$\begin{aligned}
\Delta\boldsymbol{\ell} = \boldsymbol{\ell}_{o_2} - \boldsymbol{\ell}_{o_1} \\
&= [\beta_{o_2 \to o_1} \boldsymbol{E}_m + (\beta_{o_2 \to o_1} - 1)\boldsymbol{P}_{o_1} + \boldsymbol{P}_{o_2} \\
&\quad + (\beta_{o_2 \to s} - 1)(\alpha_{s \to d_1} \boldsymbol{D}_1 \\
&\quad + \alpha_{s \to d_2} \boldsymbol{D}_2 + \boldsymbol{S})]\boldsymbol{U} \\
\\
&= [\beta_{o_2 \to o_1} \boldsymbol{E}_m + \boldsymbol{P}_{o_2} - \beta_{o_2 \to s} \boldsymbol{P}_{o_1} \\
&\quad - \beta_{o_2 \to o_1}(\alpha_{s \to d_1} \boldsymbol{D}_1 + \alpha_{s \to d_2} \boldsymbol{D}_2 + \boldsymbol{S})]\boldsymbol{U} \\
\\
&= [\boldsymbol{P}_{o_2} - \beta_{o_2 \to s} \boldsymbol{P}_{o_1} - \beta_{o_2 \to o_1}(\alpha_{s \to d_1} \boldsymbol{D}_1 \\
&\quad + \alpha_{s \to d_2} \boldsymbol{D}_2 + \boldsymbol{S} - \boldsymbol{E}_m)]\boldsymbol{U}
\end{aligned} \tag{6}$$

We reconstruct $\hat{\boldsymbol{\ell}_{o_2}} = \boldsymbol{\ell}_{o_1} + \Delta\boldsymbol{\ell}$ to compare to the original $\boldsymbol{\ell}_{o_2}$. Figure 2 shows the empirical contribution of each element of $\Delta\boldsymbol{\ell}$. We found that using $\hat{\boldsymbol{\ell}_{o_2}}$ for predictions in place of $\boldsymbol{\ell}_{o_2}$ provided the same results, which verifies our derivation. We find that setting $\alpha_{s \to d_1} = 0$ and $\alpha_{s \to d_2} = 1$ in $\Delta\boldsymbol{\ell}$ and using $\hat{\boldsymbol{\ell}_{o_2}}$ to predict o_2 causes the model's o_2 accuracy to jump to 99% from 84%. However, in practice these changes to α also impact $\boldsymbol{\ell}_{o_1}$ and significantly reduce the model's o_1 accuracy. We hypothesise that the model learns to balance a reduction in o_2 accuracy for a relatively larger increase in o_1 accuracy through $\alpha_{s \to d_i}$.

4.3 Linearly Separable Output Projections

We hypothesise that the model uses the unembedding matrix U to linearly separate the position embeddings of o_1 and o_2. To confirm this, we project the final activation vectors for o_1 and o_2 onto the positional embedding matrix P, and reconstruct the activations from this projection, removing the token embedding component. We verify the separability of these two groups by fitting a linear support vector classifier (SVC) [Cortes and Vapnik, 1995] to separate these groups, which achieves over 99% classification performance. This demonstrates the hypothesis that the model is "writing to slots" in superposition through scaling rather than additive matrix binding [Wattenberg and Viégas, 2024].

5 Discussion

Our toy transformer solves ordered list copying by writing both inputs into the `<SEP>` position, and then decoding order by the relative magnitudes of the contributions, rather than projecting content into role-specific directions, as is the case in additive matrix binding [Wattenberg and Viégas, 2024]. Equation (5) shows that the logits at o_1 and o_2 can be written as combinations of $(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1})$ and $(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2})$ with input-dependent coefficients $\alpha_{s\to d_1}, \alpha_{s\to d_2}$.

Consider training an L1-regularized SAE on the residual at `SEP` after layer 1,

$$\boldsymbol{r}_s^{L_1} = \alpha_{s\to d_1}(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}) + \alpha_{s\to d_2}(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}) + \boldsymbol{E}_s + \boldsymbol{P}_s \tag{7}$$

Then learning separate latents with identical decoder directions is an unstable solution, as this always results in a higher average L_1 penalty than maintaining a single latent for this direction. Compare this with the ideal solution of learning latents corresponding to $\boldsymbol{D}_1 = \boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}$ and $\boldsymbol{D}_2 = \boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}$, with activations equal to the attention probabilities.

This results in graded latent activations. As such, the binary perspective of SAE latents [Quirke et al., 2025, Paulo et al., 2024] is insufficient to explain the computation of this model, and the activation values of the SAE latents must be incorporated into our understanding. This creates separate problems to the echo-features caused by additive matrix binding [Wattenberg and Viégas, 2024], which are different directions in the activation space to the base features; and the ID features in ID binding [Feng and Steinhardt, 2023] which are also different directions but have an abstract interpretation. A feature that identifies the order of the bigram is necessarily not linear, due to the relative interactions between the input features. These contradict the linear representation hypothesis, a foundational assumption of mechanistic interpretability that states that networks can be described in terms of independently understandable features and linear features [Elhage et al., 2022]. Similarly to the results of Csordás et al. [2024] on RNNs, we provide evidence that transformer interpretability should not be confined by the linear representation hypothesis.

Recent work [Leask et al., 2025b, Chanin et al., 2024] has highlighted the practical challenges with training SAEs that are useful for mechanistic interpretability. Our results in this paper potentially cut across this debate: even if SAEs can recover the correct dictionary, without tools to understand the effect of the relative magnitudes of latent activations, our interpretation of the model will be incomplete.

6 Conclusions and Future Work

In this paper we identified a previously undescribed magnitude-based mechanism for relational composition in transformers, and released a minimal toy model that has learned this mechanism. We further described how this mechanism will affect latent learning in SAEs, and how it necessitates a different perspective on transformer features than is currently prevalent in the mechanistic interpretability literature.

In future work, we plan to determine whether pretrained LLMs implement similar mechanisms, and find evidence of pretrained SAE latents where different activation values can result in dramatically different model behaviour. We also intend to train SAEs on this toy model and see what they learn in practice, however we expect these results to be difficult to interpret due to the challenges of SAE hyperparameter selection [Leask et al., 2025a] and training instability [Fel et al., 2025].

In addition, we will pursue validation of our findings on less constrained models. For example, we will extend our investigation to larger N-grams to analyse how the minimum n_{layers} for high accuracy scales with N. Furthermore, the positional encodings play a major role in the magnitude-based mechanism, and it is unclear whether this mechanism would be possible for alternative positional encoding methods such as absolute encodings [Radford et al., 2019] or RoPE [Su et al., 2024].

In summary, the findings in this paper are intended as a foundation for further research on magnitude-based relational composition mechanisms in LLMs.

Acknowledgements

This work was supported through a research scholarship by the Open Philanthropy Career Development and Transition Funding. We are grateful to Open Philanthropy for their financial support and Strahinja Klem for helpful discussions about the high-level methodology.

Contribution Statement

Theo Farrell was the primary research contributor. He trained the models, coded the experiments, and wrote this paper.

Patrick Leask set the high-level direction of the project, advised on what interpretability methods to apply, helped interpret the results, and wrote parts of the discussion in Section 5.

Noura Al Moubayed advised on narrative framing and provided planning, editing and writing feedback.

References

Tanja Baeumel, Daniil Gurgurov, Yusser al Ghussin, Josef van Genabith, and Simon Ostermann. Modular arithmetic: Language models solve math digit by digit. *arXiv preprint arXiv:2508.02513*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Jannik Brinkmann, Abhay Sheshadri, Victor Levoso, Paul Swoboda, and Christian Bartelt. A mechanistic analysis of a transformer trained on a symbolic multi-step reasoning task. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4082–4102, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.242. URL `https://aclanthology.org/2024.findings-acl.242/`.

Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*, 2024.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba Ba. Evaluating sparse autoencoders: From shallow design to matching pursuit. *arXiv preprint arXiv:2506.05239*, 2025.

Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. Recurrent neural networks learn to store and generate sequences using non-linear representations. In Yonatan Belinkov, Najoung Kim, Jaap Jumelet, Hosein Mohebbi, Aaron Mueller, and Hanjie Chen, editors, *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 248–262, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.17. URL `https://aclanthology.org/2024.blackboxnlp-1.17/`.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *ICLR*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

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

Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. *arXiv preprint arXiv:2502.12892*, 2025.

Jiahai Feng and Jacob Steinhardt. How do language models bind entities in context? *arXiv preprint arXiv:2310.17191*, 2023.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009. doi: 10.1007/s12559-009-9009-8.

Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis. *arXiv preprint arXiv:2502.04878*, 2025a.

Patrick Leask, Neel Nanda, and Noura Al Moubayed. Inference-time decomposition of activations (itda): A scalable approach to interpreting large language models. *arXiv preprint arXiv:2505.17769*, 2025b.

Samuel Marks, Johannes Treutlein, Trenton Bricken, Jack Lindsey, Jonathan Marcus, Siddharth Mishra-Sharma, Daniel Ziegler, Emmanuel Ameisen, Joshua Batson, Tim Belonax, et al. Auditing language models for hidden objectives. *arXiv preprint arXiv:2503.10965*, 2025.

Rajiv Movva, Kenny Peng, Nikhil Garg, Jon Kleinberg, and Emma Pierson. Sparse autoencoders for hypothesis generation. *arXiv preprint arXiv:2502.04382*, 2025.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Nostalgebraist. interpreting gpt: the logit lens. AI Alignment Forum, August 2020. URL `https://www.alignmentforum.org/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens`. Accessed 2025-10-20.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. *arXiv preprint arXiv:2410.13928*, 2024.

Tony A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995. doi: 10.1109/72.377968.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. Fine-tuning enhances existing mechanisms: A case study on entity tracking. In *Proceedings of the 2024 International Conference on Learning Representations*, 2024. arXiv:2402.14811.

Lucia Quirke, Stepan Shabalin, and Nora Belrose. Binary sparse coding for interpretability. *arXiv preprint arXiv:2509.25596*, 2025.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, 1990. doi: 10.1016/0004-3702(90)90007-M.

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

Martin Wattenberg and Fernanda Viégas. Relational composition in neural networks: A survey and call to action. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. URL `https://openreview.net/forum?id=zzCEiUIPk9`.

A Parameter Grid Search

Table 4: Grid search results where $n_{layers} = 2$ and $n_{heads} = 1$ are fixed. F = False, T = True. We write LN for LayerNorm, and W_V and W_O are the value and output matrices respectively, where False means it was frozen to the identity matrix during training. We additionally ablate the residual stream dimension d_{model} in the second part of the table. Accuracy is reported as the mean validation accuracy over three independent training runs, though we note this is insufficient for statistical significance testing.

d_{model}	LN	Bias	W_V	W_O	Accuracy
64	F	F	T	T	0.6207
64	F	T	T	T	0.4983
64	T	F	T	T	0.7709
64	T	T	T	T	0.7744
64	F	F	T	F	0.7513
64	F	T	T	F	0.7552
64	T	F	T	F	0.9089
64	T	T	T	F	**0.9202**
64	F	F	F	T	0.7481
64	F	T	F	T	0.7560
64	T	F	F	T	**0.9162**
64	T	T	F	T	0.7636
64	F	F	F	F	**0.9180**
64	F	T	F	F	0.7487
64	T	F	F	F	**0.9215**
64	T	T	F	F	0.8935
128	F	F	F	F	**0.9192**
64	F	F	F	F	**0.9180**
32	F	F	F	F	0.6074
8	F	F	F	F	0.3148

Our parameter grid search is shown in Table 4. We find that we can ablate the attention bias and LayerNorm and freeze the attention value and output matrices to the identity without any significant drop in performance.

B Further Attention Ablations

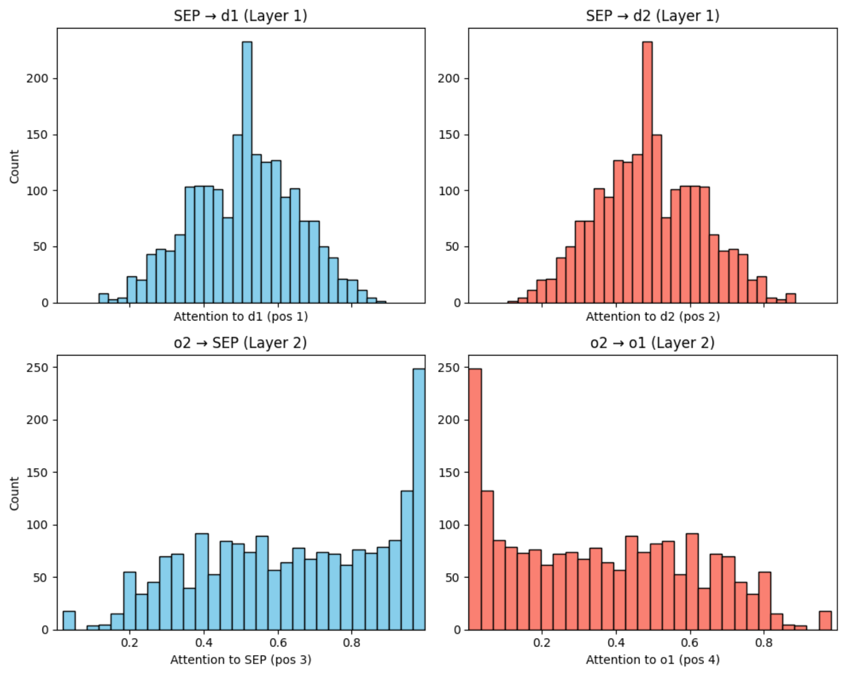

Figure 3: The attention patterns vary significantly depending on input. These histograms show the variation in the attention weights for SEP in layer 1 (top row) and for o_2 to SEP in layer 2 (bottom row).

Figure 3 shows the distribution in the attention pattern from SEP to d_i in layer 1, and from o_2 to SEP in layer 2. We see that, despite the syntax of the input being fixed, the attention pattern varies significantly with the inputs. In particular, fixing the attention pattern to the mean attention pattern over the test set reduces accuracy to 45%.

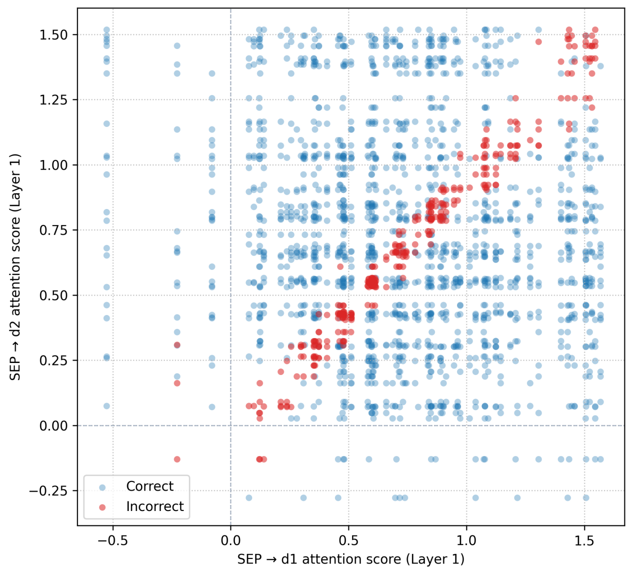

Figure 4: This graph shows the correlation between SEP's attention to both d_1 and d_2 in layer 1, and validation accuracy. The model fails when SEP attends to d_1 and d_2 by roughly the same amount (near the diagonal).

Figure 4 plots the attention scores for SEP to d_i in layer 1, and we find that similar attention patterns results in misclassification. This corroborates the finding that using the mean attention pattern harms performance, as the mean pattern lies within this error zone.

C Output Derivation

Let $\boldsymbol{r}_z^{L_i}$ be the residual at position of token $z \in \{d_1, d_2, s, o_1, o_2\}$ after layer i.

Layer 1.

$$\begin{aligned}\boldsymbol{r}_s^{L_1} &= \alpha_{s\to d_1}(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}) + \alpha_{s\to d_2}(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}) \\ &\quad + \boldsymbol{E}_s + \boldsymbol{P}_s\end{aligned} \tag{8}$$

$$\boldsymbol{r}_{o_i}^{L_1} = \boldsymbol{E}_m + \boldsymbol{P}_{o_i}, \quad i \in \{1, 2\}. \tag{9}$$

Note that we use our custom mask (outlined in Section 3) to prevent positions 4 and 5 from attending to anything in layer 1.

Layer 2.

$$\boldsymbol{r}_{o_1}^{L_2} = \boldsymbol{r}_{o_1}^{L_1} + \beta_{o_1\to s}\,\boldsymbol{r}_s^{L_1} \tag{10}$$

$$\boldsymbol{r}_{o_2}^{L_2} = \boldsymbol{r}_{o_2}^{L_1} + \beta_{o_2\to s}\,\boldsymbol{r}_s^{L_1} + \beta_{o_2\to o_1}\,\boldsymbol{r}_{o_1}^{L_1} \tag{11}$$

Output logit at position 4 (o_1).

$$\boldsymbol{\ell}_{o_1} = \boldsymbol{r}_{o_1}^{L_2} U = \big[\boldsymbol{r}_{o_1}^{L_1} + \beta_{o_1\to s}\boldsymbol{r}_s^{L_1}\big]\boldsymbol{U} \tag{12}$$

$$\begin{aligned}\boldsymbol{\ell}_{o_1} =& \big[\boldsymbol{E}_m + \boldsymbol{P}_{o_1} + \beta_{o_1\to s}\big(\alpha_{s\to d_1}(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}) \\ &\quad + \alpha_{s\to d_2}(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}) + \boldsymbol{E}_s + \boldsymbol{P}_s\big)\big]\boldsymbol{U}\end{aligned} \tag{13}$$

Output logit at position 5 (o_2).

$$\boldsymbol{\ell}_{o_2} = \boldsymbol{r}_{o_2}^{L_2} U = \big[\boldsymbol{r}_{o_2}^{L_1} + \beta_{o_2\to s}\,\boldsymbol{r}_s^{L_1} + \beta_{o_2\to o_1}\,\boldsymbol{r}_{o_1}^{L_1}\big]\boldsymbol{U} \tag{14}$$

$$\begin{aligned}\boldsymbol{\ell}_{o_2} =& \Big[\boldsymbol{E}_m + \boldsymbol{P}_{o_2} \\ &\quad + \beta_{o_2\to s}\Big(\alpha_{s\to d_1}(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1}) \\ &\qquad + \alpha_{s\to d_2}(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2}) + \boldsymbol{E}_s + \boldsymbol{P}_s\Big) \\ &\quad + \beta_{o_2\to o_1}(\boldsymbol{E}_m + \boldsymbol{P}_{o_1})\Big]\boldsymbol{U}\end{aligned} \tag{15}$$

Simplification. Let $\boldsymbol{S} = \boldsymbol{E}_s + \boldsymbol{P}_s$. Furthermore, since o_1 can only ever attend to s, we have $\beta_{o_1\to s} = 1$. Since o_2 can only attend to s or o_1, we have $\beta_{o_2\to s} + \beta_{o_2\to o_1} = 1$. Therefore, we can simplify the equations (13) and (15) to get equations (1) and (2) in Section 4.2.