A Appendix / supplemental material

A.1 Supplemental Figures

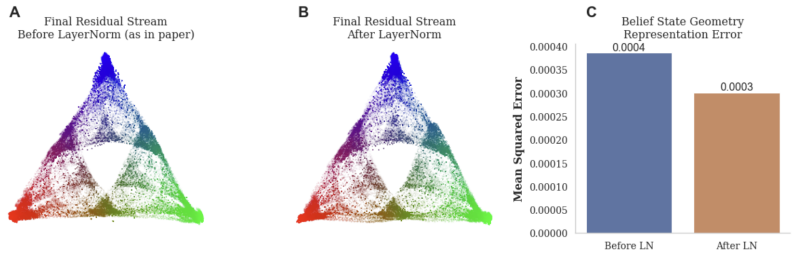

Figure S1: To test the effect of using residual stream activations from before or after the final LayerNorm in our analysis, we compared the belief state geometry representations in both cases. (A) Projection of the final residual stream activations before LayerNorm onto the belief state simplex, as presented in the main paper. (B) The same projection for activations after LayerNorm, showing a qualitatively similar structure. (C) Mean squared error of the linear fit capturing the belief state geometry for both cases. The representation after LayerNorm shows a slightly lower error (0.0003) compared to before LayerNorm (0.0004), indicating that the belief state geometry is preserved and marginally better represented after the LayerNorm operation. These results demonstrate that our findings are robust to the choice of using pre- or post-LayerNorm activations.

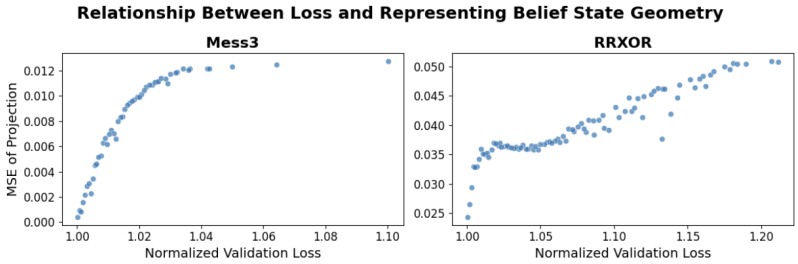

Figure S2: To quantify the relationship between transformer performance on next-token prediction and representing the belief state geometry in the residual stream, we plotted normalized validation loss vs. mean squared error (MSE) of projecting residual stream activations onto the belief state geometry for Mess3 (left) and RRXOR (right) processes. Validation loss is normalized to the theoretical optimal, such that 1.0 characterizes a transformer with optimal loss. As the validation loss decreases (moving left on the x-axis), the MSE of the regression also decreases, indicating that better performance on the next-token prediction task correlates with a more accurate representation of the belief state geometry in the residual stream.

A.2 Relations to Previous Work

A number of connections to previous work in computational mechanics have been mentioned in the Discussion section. Here we will continue the discussion of related works, focusing on connections to neural network research.

Transformer Internal Representations Recent work has significantly advanced our understanding of how and what information is encoded within transformer models. Ref. [25] introduced the "Future Lens" framework, demonstrating that individual hidden states in transformers trained on natural language contain information about multiple future tokens. This finding is consistent with transformers representing belief states, since belief states contain information well beyond the next token. Ref. [9] found that the internal representations of days of the week in LLMs lie in particular geometric arrangements (in a circular pattern). Since our work draws a concrete relation between the structure of data and geometric arrangements inside of transformers, a promising avenue of research would be to explain the days of the week result using our framework. Other work has studied internal

representations in the board game Othello, finding a representation of the board state in the residual stream [17, 23]. This finding is interpreted by the authors as the transformers containing a "world model". Conceptually extending our work to such a setting is quite natural. In a sequential game with full knowledge, such as Tic-tac-toe, Othello, and Chess, each board state uniquely determines the future distribution of moves. Thus, the belief states correspond to board states, providing an explanation for why transformers trained on gameplay should contain explicit representations of board states in their residual stream. However, this also makes clear a limitation of our approach. The simplex associated with these belief states would, for any realistic board game, be of larger dimension than the residual stream. If transformers are representing the belief state geometry in those cases, they must be doing so in a compressed fashion. This is an important theoretical problem for future work. Importantly, our work provides a unique model-agnostic point of view to study interpretability.

Behavioral Limitations of Transformers and Chain of Thought A number of papers have studied the limitations of transformers to learn certain algorithmic tasks, such as graph path-finding tasks and HMMs [1, 14]. These papers provide important empirical results that are interesting to think through the perspective of the MSP and belief state geometry. In particular, Ref. [14] provides an upper-bound for the depth of a transformer needed to capture HMM generated data up to a particular context-window length. This makes sense given that transformers are feedforward machines, and the MSP is, in general, an automata containing recurrent components. Thus, what the transformer should be capturing is really an unrolled version of the MSP, which naturally gives an upper bound for how many layers one would need to capture the MSP to a certain depth. This paper also explicitly performs experiments on belief-state inference. In some of their experiments the explicitly train transformers to map sequences of emissions from an HMM to sequences of the associated belief states. In other experiments they train in the normal autoregressive manner solely on the observations, like in our study. Our work provides a nice theoretical connection between the two tasks.

In Ref. [1], the authors test and explain a lack of ability for transformers to perform seemingly simple graph planning tasks. In this work, and others like it that study particular algorithmic tasks with particular formal structures like finite-automata [18], addition, and the like, we think its interesting to think of how our framework would apply. In general, our approach treats the sequential training data as a first class citizen. This makes clear that to relate studies of formal structures (like graph planning) to our framework, one must explicitly consider the particular tokenization used to convert from the formal structure to sequences of tokens. Then the main consideration becomes the computational structure of the sequential nature of the tokens generated by the chosen tokenization scheme (as opposed to the formal structure directly). From our perspective what one learns, and the limitations of being able to perform well on next-token generation given that tokenization, is directly related to the tokenization and not directly to the formal structure that was tokenized. Thus, these results provide important empirical results for our framework to explain, or good tests to find where our framework fails and needs to be amended/expanded.

Meta-learning In Ref. [24], the authors provide a theoretical framework for predictors and agents based on a Bayesian framework which builds state machines of sufficient statistics. This is analogous to the mixed-state presentation we have presented here. Indeed, work in computational mechanics has shown that the states of the MSP correspond not only to beliefs over the generator states, but also to the minimal sufficient statistics for predicting the future based on the past. One exciting area of further research this exposes is the potential for computational mechanics to inform our understanding of artificial RL agents.

A.3 Details of data generating processes used in experiments

The transition matrices for the Mess3 process $T^{(A)}$, $T^{(B)}$, and $T^{(C)}$ are defined as follows:

$$T^{(A)} = \begin{pmatrix} 0.765 & 0.00375 & 0.00375 \\ 0.0425 & 0.0675 & 0.00375 \\ 0.0425 & 0.00375 & 0.0675 \end{pmatrix}, \quad T^{(B)} = \begin{pmatrix} 0.0675 & 0.0425 & 0.00375 \\ 0.00375 & 0.765 & 0.00375 \\ 0.00375 & 0.0425 & 0.0675 \end{pmatrix}, \text{ and}$$

$$T^{(C)} = \begin{pmatrix} 0.0675 & 0.00375 & 0.0425 \\ 0.00375 & 0.0675 & 0.0425 \\ 0.00375 & 0.00375 & 0.765 \end{pmatrix}.$$

The RRXOR process has transition matrices defined as

$$T^{(0)} = \begin{pmatrix} 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad T^{(1)} = \begin{pmatrix} 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & .5 & 0 \\ 0 & 0 & 0 & 0 & .5 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where $T^{(0)}$ and $T^{(1)}$ are the transition matrices for emissions 0 and 1, respectively.

A.4 Belief State Simplex

Belief states lie within the probability simplex $\Delta^{|\mathcal{S}|-1}$, defined as:

$$\Delta^{|\mathcal{S}|-1} = \left\{ b \in \mathbb{R}^{|\mathcal{S}|} : \sum_{i=1}^{|\mathcal{S}|} b_i = 1 \text{ and } b_i \geq 0 \text{ for all } i \right\}.$$

A.5 Regression

For the regression procedure used in this paper we used standard linear regression, assuming an affine model for the data and then minimizing:

$$\min_{W,c} \sum_i \|b_i - (Wa_i + c)\|^2$$

where the summation is over all input sequences in the dataset.

A.6 Details of transformer architecture and training

In our experiments, we trained a transformer model using the following hyperparameters and training parameters: The model had a context window size of 10, used ReLU as the activation function, and had a head dimension of 8 and a model dimension of 64. There was 1 attention head in each of 4 layers. The model had MLPs of dimension 256 and used causal attention masking. Layer normalization was applied. For training, we used the Stochastic Gradient Descent (SGD) optimizer with a batch size of 64, running for 1,000,000 epochs, and a learning rate of 0.01 with no weight decay. For each batch we generated 64 sequences from the Mess3 or RRXOR HMM, choosing an initial hidden state from the stationary distribution. We instantiated the networks and performed our analysis using the TransformerLens library [22].

A.7 Details for analysis

For the shuffle and cross validation analyses used in Figure 6, we performed both the shuffle and the cross validation procedure 1000 times over, randomly shuffling or performing the train test split independently 1000 times.

A.8 Reproducibility Instructions

To install the necessary dependencies and the code, follow these steps:

A.8.1 Installation

Navigate to the repository folder and run:

```
pip install -e .
```

A.8.2 Reproducing Figures

To reproduce the figures presented in the paper, step through the provided Jupyter notebooks:

- `Figure6.ipynb`