Leveraging Attractor Dynamics in Spatial Navigation for Better Language Parsing

Xiaolong Zou [*1] **Xingxing Cao** [*1] **Xiaojiao Yang** [1] **Bo Hong** [1]

Abstract

Increasing experimental evidence suggests that the human hippocampus, evolutionarily shaped by spatial navigation tasks, also plays an important role in language comprehension, indicating a shared computational mechanism for both functions. However, the specific relationship between the hippocampal formation's computational mechanism in spatial navigation and its role in language processing remains elusive. To investigate this question, we develop a prefrontal-hippocampal-entorhinal model (which called PHE-trinity) that features two key aspects: 1) the use of a modular continuous attractor neural network to represent syntactic structure, akin to the grid network in the entorhinal cortex; 2) the creation of two separate input streams, mirroring the factorized structure-content representation found in the hippocampal formation. We evaluate our model on language command parsing tasks, specifically using the SCAN dataset. Our findings include: 1) attractor dynamics can facilitate systematic generalization and efficient learning from limited data; 2) through visualization and reverse engineering, we unravel a potential dynamic mechanism for grid network representing syntactic structure. Our research takes an initial step in uncovering the dynamic mechanism shared by spatial navigation and language information processing.

1. Introduction

Despite millions of years of human brain evolution, the hippocampal-entorhinal system has remained remarkably conservative (Krubitzer et al., 2011). For instance, similar anatomical structures are observed across various species (Strange et al., 2014) from rodents to humans, mostly exhibiting periodic grid cell representation in the entorhinal cortex during spatial navigation tasks (Krubitzer et al., 2011). This indicates that this system, which emerged early in mammalian evolution and was shaped by spatial navigation tasks, has become an integral, pre-existing structure in the human brain. However, extended evolution has also introduced notable differences to the human hippocampal-entorhinal system. A significant change is the enlargement of the anterior part of the human hippocampus, which has developed extensive bidirectional connections with the prefrontal cortex (Buzsáki & Tingley, 2018). As a result, in humans, the function of this system extends beyond physical space navigation to include various high-level cognitive tasks, such as encoding abstract conceptual spaces (Behrens et al., 2018). An intriguing concept is that the hippocampal-entorhinal system may serve as a cognitive scaffold which can be adapted for a range of advanced functions (Lerousseau & Summerfield, 2024).

Experimental studies increasingly indicate that the hippocampal formation plays a crucial role in language information processing (Blank et al., 2016; Piai et al., 2016; Solomon et al., 2019; Hahamy et al., 2023). For instance, Blank et al. discovered that specific regions of the hippocampus are actively involved in language comprehension tasks (Blank et al., 2016). Additionally, through direct brain recordings, Piai et al. observed that the human hippocampus generates sustained theta oscillations during real-time language processing (Piai et al., 2016), with the oscillation power correlating with semantic distance. Furthermore, the recent identification of semantic boundary cells in the human hippocampus suggests its potential function in segmenting and abstracting natural narratives, thereby enhancing language comprehension (Zheng et al., 2022). These collective findings imply the existence of a shared computational mechanism for spatial navigation and language information processing (Kazanina & Poeppel, 2023; O'keefe & Nadel, 1978; Whittington et al., 2022b). Investigating this shared mechanism is vital for understanding how the brain represents and processes language and inspiring innovative language intelligence algorithms. However, the shared computational mechanism remains largely unknown.

A key function of the hippocampal-entorhinal system in

[*]Equal contribution [1]Qiyuan Lab, Beijing, China. Correspondence to: Bo Hong <hongbo@qiyuanlab.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

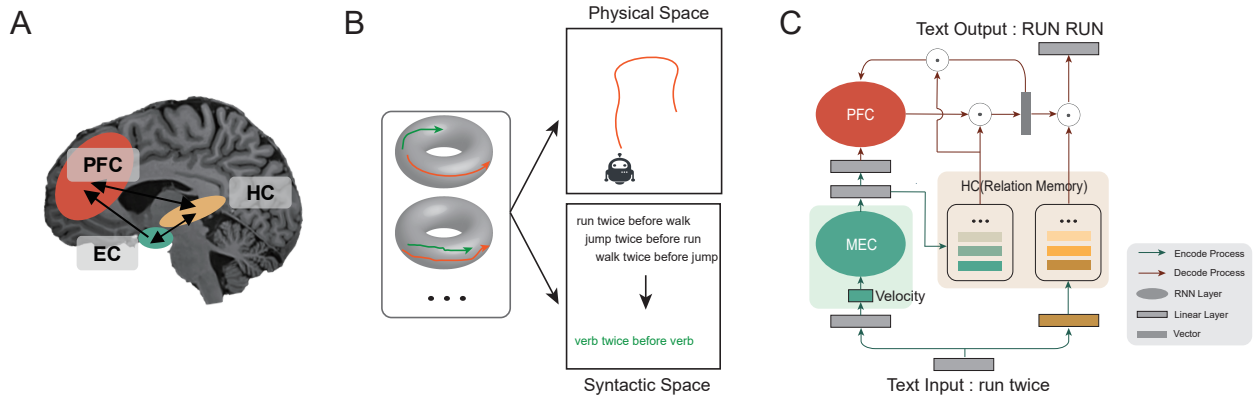

Figure 1. Shared computational mechanism and model architecture for spatial navigation and language information processing. (A) Extensive brain research indicates that the circuit involving the prefrontal cortex, hippocampus, and entorhinal cortex is a critical shared region for both spatial navigation and language information processing. (B) The modular grid representation in the entorhinal cortex is not only capable of representing 2D physical spatial structures but may also be reused to represent language syntactic structures. (C) We developed a neural dynamic model called PHE-trinity that simulates the PFC-HC-EC circuit of the human brain. In this model, the entorhinal cortex is implemented by a continuous attractor network driven by velocity; the prefrontal cortex model is characterized by a recurrent network, and the hippocampus is represented by a relational memory module. The model processes language command inputs by decoupling them into syntactic structure and semantic content representations first. These representations are then directed to the entorhinal cortex network and the hippocampus network, respectively. The interaction between these modules enables the model to effectively parse language commands in the SCAN dataset, such as "run twice → RUN RUN". For more model details, see Fig. S1.

processing information is the use of a grid network for spatial structure representation (Behrens et al., 2018; Giocomo et al., 2011). Grid cells in the entorhinal cortex display a periodic hexagonal firing pattern with different spatial periods and orientations (Giocomo et al., 2011). Cells with identical period and orientation group together into a grid module, forming a toroidal manifold (Gardner et al., 2022), thereby creating a modular representation of physical space. Modeling studies have shown that each grid module can be conceptualized as a continuous attractor neural network (Giocomo et al., 2011), with its attractors constituting a low-dimensional manifold. Afferent velocity inputs are capable of driving network activities across this attractor manifold, facilitating path integration and the representation of abstract spatial structures (Burak & Fiete, 2009). In the context of language tasks, velocity vectors might correspond to specific grammatical roles within sentences. These vectors drive the grid network to integrate syntactic structures, analogous to path integration in physical space (Whittington et al., 2022b). Moreover, the grid network have several beneficial encoding features, such as exponential capacity and error-correcting abilities (Sreenivasan & Fiete, 2011), which may considerably enhance the efficient representation of syntactic structures.

In the hippocampal-entorhinal circuit, another crucial mechanism is the factorized representation of structure and content (Whittington et al., 2020; Behrens et al., 2018). Extensive experimental and modeling research suggests that the entorhinal cortex represents abstract spatial structures, while the hippocampus functions as a relational memory system that concurrently stores both structures and contents (Behrens et al., 2018; Whittington et al., 2020). This parallels language representation, which is suggested to be similarly divided into abstract syntactic structure and semantic content (Dehaene et al., 2015; Grodzinsky, 2000). Noam Chomsky's Minimalist Program theorizes that the brain's ability to generate a vast array of ideas stems from the recursive binding and merging of a limited number of primitive syntactic and semantic units (Chomsky, 2014). Therefore, the factorized mechanism observed in the hippocampal-entorhinal circuit could potentially enhance systematic generalization in language-related tasks, extending learned syntactic rules to novel, unseen ones.

In summary, despite extensive research on the hippocampus-entorhinal cortex system's computational mechanism in spatial navigation, its role in language processing remains elusive. To investigate this problem, we construct a concrete model called PHE-trinity, which mimics the prefrontal-hippocampal-entorhinal (PFC-HC-EC) loop in the biological brain. PHE-trinity has two key features: separating streams for structure and content representation, and using a velocity-driven grid system to represent the syntactic structure of language. We demonstrate that 1) in the SCAN dataset, the attractor dynamics within PHE-trinity contribute to systematic generalization, achieve competitive performance across various language parsing tasks (here, language

parsing refers to transforming sentences into another format based on underlying syntactic structures or rules), and enhance learning efficiency, particularly in data-limited scenarios; 2) through visualization and reverse engineering, the grammatical roles of words can emerge from training, serving as velocity inputs to leverage the grid-like system to represent syntactic structure. Overall, our research provides initial insights into the shared computational mechanism behind spatial navigation and language information processing, and offers a method to incorporate attractor dynamics prior into deep networks to enhance efficient learning and generalization.

2. Method

As shown in Fig. 1A, we build a dynamic model called PHEtrinity that simulates the structural and functional attributes of corresponding cortical regions. The model is characteristic of two features: first, the employment of a grid network that encodes spatial or syntactical structures, as illustrated in Figure 1B; second, the adoption of separate structure-content streams. Accordingly, the model is divided into three core modules: (1) the MEC (medial entorhinal cortex) network, implemented as a modular continuous attractor network driven by velocity inputs; (2) the HC network, functioning as an associative relational memory system; (3) the PFC network, configured as a recurrent neural network to encode language parsing rules. The whole model here can be understood as an encoder-decoder architecture from a machine learning perspective, more details seen in Fig. S1.

2.1. Model Architecture

The described model processes an command sequence represented as $\mathbf{x}(1), \ldots, \mathbf{x}(T_x) \in \mathbb{R}^{D_x}$, where T_x denotes the input sequence length. The objective is to map this input to a target action sequence, denoted as $\hat{\mathbf{y}}(1), \ldots, \hat{\mathbf{y}}(T_y) \in \mathbb{R}^{D_y}$, with T_y representing the target sequence length. In this setup, $\mathbf{x}$ corresponds to trainable word vectors, randomly initialized using a Gaussian distribution, whereas $\hat{\mathbf{y}}$ denotes the output symbols, encoded as one-hot vectors. The model employs a decoupled structure-content architecture akin to the HC-MEC system (Behrens et al., 2018). Thus, each input word vector $\mathbf{x}(t)$ is bifurcated into two components: a structure representation vector $\mathbf{m}_s(t) = \mathbf{W}_s\mathbf{x}(t)$, where $\mathbf{W}_s \in \mathbb{R}^{N_s \times D_x}$, and a content representation vector $\mathbf{m}_c(t) = \mathbf{W}_c\mathbf{x}(t)$, with $\mathbf{W}_c \in \mathbb{R}^{N_c \times D_x}$. This architecture prior enhances the model's proficiency in learning syntactic rule-based transformations from input commands to target actions.

The MEC Network Drawing inspiration from the grid network in the entorhinal cortex (Giocomo et al., 2011), the MEC network is implemented as a modular continuous attractor neural network, with the activity of each module driven by an velocity input (Burak & Fiete, 2009). For simplicity, we assume that the dynamic behavior of each attractor network module is described by the following equation:

$$\tau \frac{d\mathbf{r}(t)}{dt} = -\mathbf{r}(t) + \mathbf{W}_r\mathbf{g}(t) + \mathbf{W}_{\text{in}}\mathbf{v}(t), \tag{1}$$

where τ represents the time constant of the network, $\mathbf{r}(t)$ is the input vector received by neural population at time t, $\mathbf{g}(t) = \sigma(\mathbf{r}(t))$ represents the activity vector with $\sigma(\cdot)$ being the ReLU activation function. The matrices $\mathbf{W}_r \in \mathbb{R}^{N_g \times N_g}$ and $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_g \times 2}$ correspond to the recurrent and feedforward connections of the network, respectively.

The activity within the attractor manifold is driven by a 2D velocity vector input, expressed as $\mathbf{v}(t) = [v_x(t), v_y(t)]^T$. In spatial navigation tasks, v_x and v_y denote the velocity magnitudes in the spatial domain. For language tasks, v_x and v_y are derived from the structure representation vector $\mathbf{m}_s$ through a ReLU non-linear activation function and a subsequent linear transformation. In the context of language processing, the "velocity" vector input can be interpreted as a grammatical role, steering neural activity through a low-dimensional attractor manifold to integrate syntactic structures. This process bears some resemblance to the path integration observed in spatial tasks, as depicted in Fig. 1B.

To achieve a continuous attractor network structure resembling the grid network in the biological brain, we employ two alternative methods to determine the connection matrices $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_r$ in the MEC network. One approach involves obtaining these matrices through auxiliary training in a spatial path integration task (Sorscher et al., 2023) (Appendix A.2), while the other utilizes predefined weight values (Burak & Fiete, 2009) (Appendix A.8).

During training, the 2D velocity vector input is processed by the MEC network, linearly transformed and passed through a softmax function to predict position coordinates at time t, labeled $\mathbf{p}(t)$. Assuming the true position coordinates at time t are $\hat{\mathbf{p}}$, the loss function for the path integration task is defined as:

$$\mathcal{L}_{\text{path}} = -\sum_{b=1}^{B_{\text{path}}}\sum_{t=1}^{T_{\text{path}}} \hat{\mathbf{p}}_b(t)\log(\mathbf{p}_b(t)) + \beta\|\mathbf{W}_r\|_2^2. \tag{2}$$

Here, B_{path} is the batch size for the path integration task, and T_{path} is the length of input sequence. The loss function comprises two terms: the first is the cross-entropy loss, quantifying the discrepancy between predicted and true position coordinates; the second is an L_2 regularization term applied to the connection weights, which promotes the emergence of continuous attractor dynamics during training. The regularization strength is modulated by the coefficient β. For details, see Appendix A.3.

The MEC network, consisting of N modules, produces an output represented by the concatenated vector $\hat{\mathbf{g}}(t) = [\mathbf{g}^1(t), \ldots, \mathbf{g}^N(t)]^T$. $\hat{\mathbf{g}}(t)$ is then transformed via two layers of a multilayer perceptron (MLP). The hidden layer activity, denoted as $\mathbf{k}(t) = f_1(\hat{\mathbf{g}}(t))$, is sent to the HC network as "position" encoding. The second layer activity at the last encoding time $t = T_x$, denoted as $\mathbf{u}(T_x) = f_2(f_1(\hat{\mathbf{g}}(T_x)))$, represents the integrated result of the structural information for the command sequence and is forwarded to the PFC network, as illustrated in Fig. 1C and Fig. S1. Functions $f_1(\cdot)$ and $f_2(\cdot)$ each include a linear mapping followed by a ReLU non-linear activation function.

The HC Network Experimental and theoretical studies suggest that the HC functions as a relational associative memory system (Whittington et al., 2020), responsible for binding and storing structural and sensory representations together. To simplify, we model the HC network as an external relational memory module (Graves et al., 2014; Kumaran et al., 2016). At each time t, it binds the semantic representation vector $\mathbf{m}_c(t)$ from the external input with the "position" representation vector $\mathbf{k}(t)$ from the MEC network, forming a memory vector pair $[\mathbf{k}(t), \mathbf{m}_c(t)]$, which is then stored in an external memory module, as depicted in Fig. 1C and Fig. S1. Thus, the HC network functions akin to a query table, facilitating the retrieval of stored content representation based on a query of the "position" representation vector $\mathbf{k}(t)$, and vice versa.

The PFC Network The PFC in the biological brain is known to encode abstract rules via a recurrently connected network (Mansouri et al., 2020). In PHE-trinity, the PFC network is represented as a Long Short-Term Memory (LSTM) model designed to encode language parsing rules (Hochreiter & Schmidhuber, 1997; O'Reilly & Frank, 2006; Wang et al., 2018). It interacts with the HC network to realize the translation from command sequences to target actions, acting as a sequence decoder. Assuming that the dynamics update rule for the PFC network is represented by the function $f_{\text{pfc}}(\cdot)$:

$$\mathbf{a}(t'+1), \mathbf{c}(t'+1) = f_{\text{pfc}}(\mathbf{s}(t'), \mathbf{c}(t')). \tag{3}$$

Here, $\mathbf{c}(t')$ is the hidden state vector of the PFC network at time t', and $\mathbf{c}(0) = \mathbf{u}(T_x)$. Note that time t' describes the decoding process, unlike t in the encoding process. The output vector at time t' is $\mathbf{a}(t')$, while $\mathbf{s}(t')$ denotes the input received from the HC network.

Here, we assume the PFC network performs a selective attention on the positional component of memory vectors stored in the HC network, using techniques adapted from (Russin et al., 2019; Whittington et al., 2021). The attention weight for the i-th memory vector is given by:

$$b_i(t') = \frac{e^{d(\mathbf{a}(t'), \mathbf{k}(i))}}{\sum_j e^{d(\mathbf{a}(t'), \mathbf{k}(j))}}, \tag{4}$$

where $d(\mathbf{a}(t'), \mathbf{k}(i)) = \mathbf{a}(t')^T \mathbf{k}(i)$ measures the similarity between the output vector $\mathbf{a}(t')$ and the position vector $\mathbf{k}(i)$, using a dot product. Based on the attention weight vector $\mathbf{b}(t')$, the position representation in the HC network is summarized and fed back to the PFC network as input $\mathbf{s}(t') = \sum_i b_i(t')\mathbf{k}(i)$. Additionally, a summary of the stored semantic representation is obtained as $\mathbf{o}(t') = \sum_i b_i(t')\mathbf{m}_c(i)$. After a linear transformation and a softmax non-linear function, $\mathbf{o}(t')$ yields the prediction for the t'-th output action, denoted as $\mathbf{y}(t')$. Consequently, PHE-trinity iteratively generates the target action sequence $[\mathbf{y}(1), \ldots, \mathbf{y}(T_y)]$.

2.2. Objective Function and Training

To reflect the evolutionary development of the hippocampal-entorhinal circuit, which is influenced by spatial navigation tasks, PHE-trinity undergoes two distinct phases, including a pretraining phase and a co-training phase.

In the pretraining phase, the MEC network in PHE-trinity is pretrained in a spatial path integration task to estabilish modular attractor dynamics, resembling the grid cell network in the brain. We use the Adam optimizer to optimize the loss function L_{path}.

In the co-training phase, PHE-trinity is trained on both language parsing tasks and path integration tasks to solve language parsing tasks. The objective function contains three parts: 1) a cross-entropy loss function to ensure the accurate prediction of target actions by the model; 2) an auxiliary loss function for path integration to maintain attractor dynamics; 3) a L_2 regularization on the neural activity representing structure and content to facilitate disentangled representations from the input command (Whittington et al., 2022a). Therefore, the overall objective loss function is formulated as:

$$\begin{aligned} L = &-\frac{1}{BT_y}\sum_{b=1}^{B}\sum_{t=1}^{T_y}\hat{\mathbf{y}}_b(t)\log(\mathbf{y}_b(t)) + \alpha L_{\text{path}} \\ &+ \alpha_1 \frac{1}{BT_x}\sum_{b=1}^{B}\sum_{t=1}^{T_x}\|\mathbf{m}_s(t)\|_2^2 \\ &+ \alpha_2 \frac{1}{BT_x}\sum_{b=1}^{B}\sum_{t=1}^{T_x}\|\mathbf{m}_c(t)\|_2^2. \end{aligned} \tag{5}$$

The parameters $\alpha, \alpha_1, \alpha_2$ control the corresponding loss items respectively. For other detailed model parameters and training settings, please refer to Appendix A.3.

2.3. SCAN Dataset

Here, we employ the SCAN dataset for model evaluation (Lake, 2019). SCAN is a collection of simple language-driven navigation tasks designed for studying compositional

Command sequence	Action Sequence
jump	JUMP
jump left	LTURN JUMP
jump around right	RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP RTURN JUMP
turn left twice	LTURN LTURN
jump opposite left and walk thrice	LTURN LTURN JUMP WALK WALK WALK
jump opposite left after walk around left	LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN LTURN JUMP

Table 1. Examples of the SCAN tasks which are utilized to evaluate the combinatorial generalization learning capability. On the left are the language command inputs, and on the right are the output action sequences.

learning. As depicted in Tab.1, SCAN comprises a set of commands and their corresponding action sequences. The dataset encompasses 20,910 records, and depending on the split between training and testing data, various SCAN tasks can be created:

1) Simple task: The dataset is randomly divided into the training set and the testing set, with both sets sharing the same distribution.
2) Add primitive task: The training set includes only some basic forms. For instance, in the "Add Jump" task, the training set contains only "jump $\rightarrow$ JUMP", while the testing set incorporates combinations of "jump" with various words.
3) Length task: The training set comprises shorter language commands, while the testing set includes longer commands.

Despite their simplicity, SCAN tasks encapsulate key aspects of human language, such as compositionality and recursive construction. The tasks enable the construction of novel commands through established rules, and different rules can be recursively combined to create longer language sequences. For more details about SCAN dataset, see Appendix A.1.

3. Results

3.1. Attractor Dynamics Facilitate Systematic Generalization

Mimicking the early evolution of the hippocampal-entorhinal circuit, PHE-trinity is first pretrained on a spatial path integration task to establish continuous attractor dynamics (see Fig. S2), then simultaneously trained on both language and spatial tasks.

Firstly, as shown in Tab. 2, PHE-trinity excels at learning compositional skills, outperforming Syntactic Attention (Russin et al., 2019) and MLC (Lake & Baroni, 2023), two brain-inspired methods, across different tasks in the SCAN dataset, particularly in the "Add Jump" and "Length" tasks. In the "Add Jump" task, PHE-trinity achieved 100% accuracy across 8 runs, while the Syntactic Attention method with similar structure-content separation achieves only 78.4%. These findings suggest that PHE-trinity with pretrained attractor dynamics exhibits competitive systematic generalization capabilities.

To further assess the role of pre-established attractor dynamics on systematic generalization within SCAN tasks, we conduct two control experiments. Both experiments utilizes the same model structure as that in PHE-trinity but with different training approaches. In Control Model 1, the model is trained exclusively on SCAN tasks, resulting in a MEC network with no attractor dynamics. Control Model 2 undergoes concurrent training on both SCAN tasks and path integration tasks, but without any pre-training phase. This approach may lead to delayed formation of attractor dynamics, potentially impeding their reuse for SCAN tasks. Both control models are subjected to the identical hyperparameter settings as PHE-trinity (refer to Section 2.1). As shown in Tab. 2, PHE-trinity outperforms the control models across various SCAN tasks. While all three models demonstrate the same structure-content separation structure, the control models exhibit erratic training behavior in practice and a notably higher variance in performance in the "Add Jump" task. This variance is likely due to the absence of an attractor-based prior. These findings suggest an important role of preformed attractor dynamics in facilitating systematic generalization in language command parsing.

The implementation of separate structure-content streams in the model does not necessarily guarantee the successful acquisition of factorized representations from the data. To assess this in PHE-trinity, we visualise the learned velocity inputs, which are assumed to represent abstract grammatical roles. Our observations, as depicted in Fig. 2, reveal several key points: 1) input words are converted into velocity vectors with different magnitudes and directions; 2) Verbs such as "walk," "jump," "look," and "run," tend to aggregate at a single point, similar to the clustering observed with directional terms like "left" and "right"; 3) notably, certain patterns consistently emerge in the learned velocity inputs across different runs. For instance, "and" and "after" consistently occupy opposite directions, which may reflect their contrasting logical roles in language parsing (extended results seen in the Fig. S3). These findings suggest that PHE-trinity is capable of extracting abstract structure representations from raw data, successfully achieving a decoupling of structure and content at the representational level.

Table 2. Test accuracies on different SCAN tasks across different models and training paradigms. All reported accuracies are averages over eight runs. For detailed parameters, refer to the Appedix A.3. Results marked with a star ($\star$) indicate our implementation utilizing the code provided in (Lake & Baroni, 2023), averaged across 8 runs.

Model/Task	Simple	Add Jump	Add turn left	Length
Syntactic Attention (Russin et al., 2019)	100.0 ± 0.0	78.4 ± 27.4	99.9 ± 0.16	15.2 ± 0.7
MLC (Lake & Baroni, 2023)	99.98	99.78	$99.97 \pm 3.7^\star$	$0.0 \pm 0.0^\star$
Control model 1	98.92 ± 2.8	74.96 ± 42.92	100.0 ± 0.0	17.0 ± 2.5
Control model 2	99.88 ± 0.25	75.24 ± 42.86	100.0 ± 0.0	19.4 ± 1.1
PHE-trinity	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{100.0 \pm 0.0}$	$\mathbf{21.48 \pm 3.4}$

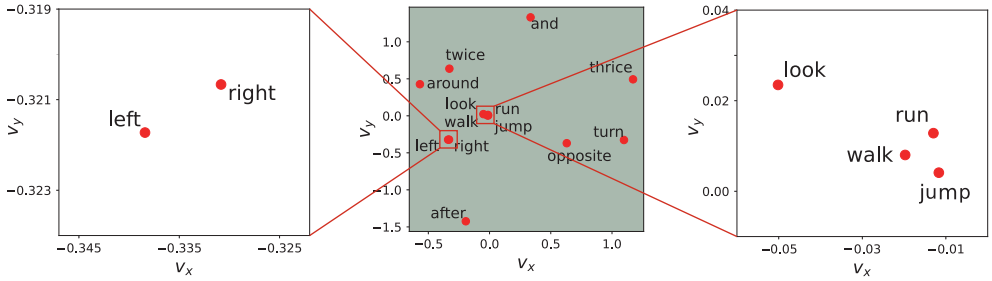

Figure 2. PHE-trinity learns abstract velocity representations through data training. After training on the "Add Jump" task, the 2D velocity vectors corresponding to all command words received by the MEC network are plotted in the center panel, while the left and right panels display zoomed-in results.

3.2. Attractor Dynamics Facilitate Learning in Limited-Data Conditions

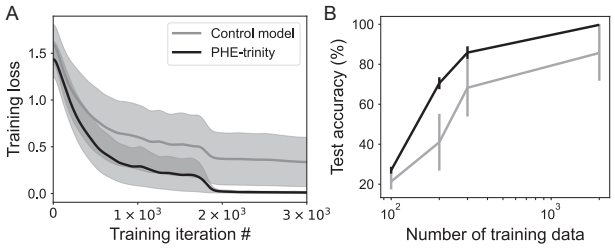

Figure 3. Attractor dynamics pre-established through spatial path integration tasks, can facilitate learning on SCAN tasks under data-limited conditions. The Control model, represented by the green curve, undergoes training solely on SCAN tasks. (A) The mean training loss versus the count of training iterations, with only the first 3×10^3 iterations shown. (B) Test accuracies for various training sample sizes—100, 200, 300, and 2000—are shown, with a logarithmic scale applied to the x-axis. Accuracies are obtained by averaging over 14 runs. For specifics on the parameters, refer to Appendix A.3

The human brain, with its evolved structures, leverages these inherent configurations to efficiently learn from limited data, thus enabling flexible adaptation to new environments (Lake et al., 2017). Our study examines whether pre-established attractor dynamics from spatial tasks could accelerate the learning of SCAN tasks with limited training instances. We compare the performance of a Control model lacking pre-established attractor dynamics, with PHE-trinity in the "Add Jump" task. Both models are identical in architecture and hyperparameters (detailed in the corresponding sections). Fig. 3A illustrates that PHE-trinity exhibits markedly faster convergence than the control model during training. Further experiments involve training on subsampled datasets to assess test performance. Results consistently show PHE-trinity's superior performance under various limited-data conditions. Remarkably, PHE-trinity attain an accuracy of $99.71 \pm 0.24\%$ with only 2000 training examples (Fig. 3B). These outcomes highlight the critical role that pre-established attractor dynamics play in enhancing model performance with limited data.

3.3. Visualization of Neural Dynamics During Command Parsing

We further conduct a visualization analysis of PHE-trinity to elucidate the dynamic mechanism involved in language command parsing. Initially, as shown in Fig. 4A, dimensionality reduction reveals that the MEC network displays a torus-like attractor manifold in its neural activity space. This manifold structure is consistent with the dynamical mechanism ob-

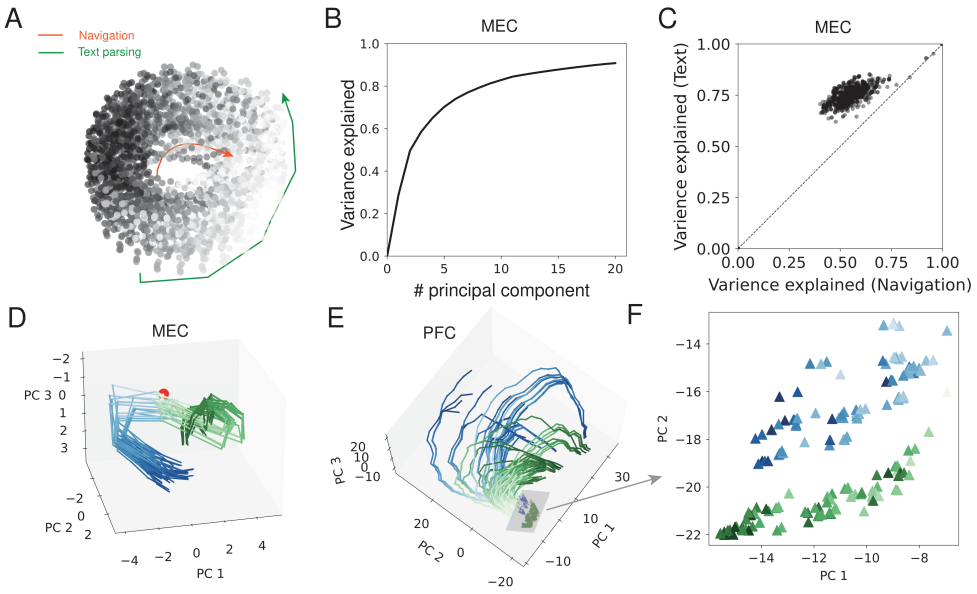

Figure 4. Visualization of Neural dynamics in the MEC and PFC network. PHE-trinity is trained on "Add Jump" task. (A) A toroidal manifold of stable attractor patterns in MEC's neural activity space, visualized via dimensionality reduction (Sorscher et al., 2023). Here, the red curve represents a neural trajectory during spatial path integration tasks, while the green curve illustrates a trajectory during the encoding of an command sequence projected onto the attractor manifolds. For details about the visualisation method, see Appendix A.9. (B) Dimensionality of neural dynamics in the MEC network. Cumulative percentage of explained variance is shown as a function of the number of principal components. (C) The top 6D subspace of MEC activity during navigation captures as much variance in MEC activity during text tasks as the top 6 PCs. Note the torus structure shaped by path integration tasks in (A) mainly lies in this top 6D subspace. (D) PCA visualization of neural dynamics in the MEC network. The starting state of neural trajectories is marked by a red circle. Blue trajectories correspond to command sequences containing the word "and", while green trajectories relate to command sequences containing the word "after". (E) PCA visualization of the neural dynamics in the PFC network. The blue and green curves correspond to those in (D). The triangles indicate the starting states of neural trajectories. (F) PCA visualization of the starting states of neural trajectories in (E). Color intensity denotes the length of output action sequences, with darker colors indicating longer sequence lengths. The length of output action sequence is significantly correlated with PC1 and PC2 ($R^2 = 0.432$, $p = 6.7 \times 10^{-25}$) as determined by multiple linear regression method. For more details, see Fig. S5 and Appendix A.7.

served in the biological brain's grid cell network (Gardner et al., 2022). The projected neural trajectories are shown to travel close to the manifold. By employing principal component analysis (PCA) of neural activities in the MEC network, it is demonstrated that the syntactic information of command sequences lies in a low-dimensional space (Fig. 4B). The 6D subspace where the torus manifold exists can explain nearly as much variance as the top 6 PCs of language command trajectories (Fig. 4C). This suggests that the torus-like continuous attractor dynamics lie in a shared subspace with language command trajectories and can be leveraged to encode both spatial and syntactic structures. And neural trajectories form two distinct trajectory clusters, as shown in Fig. 4D. One cluster corresponds to sequences containing "and," while the other is associated with sequences featuring "after." These clusters represent divergent parsing logics: sequences with "and" are processed sequentially around the word, whereas "after" requires reverse-order parsing.

Furthermore, trajectories in the PFC network reflect the sequence decoding process, as shown in Fig. 4E. These trajectories are generated via the interaction between PFC and HC. Each trajectory in PFC network corresponds to an abstract parsing rule for sequences with one specific syntactic structure (more details seen in Fig. S5). And the starting states of neural trajectories in the PFC network cluster into linearly separable groups, showing a linear correlation with the lengths of the target action sequences (Fig. 4F).

In summary, both the MEC and PFC networks organize sequence representation according to syntactic rules. During parsing, commands with varying syntactic structures are integrated into specific network states in the MEC network by leveraging its attractor dynamics. These states are then transmitted to the PFC network, initiating the corresponding decoding trajectories.

3.4. Parsing Commands Based on Predefined Attractor Dynamics

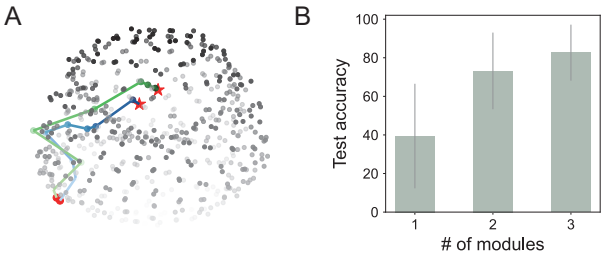

Figure 5. Predefined modular continuous attractor dynamics for language commands parsing. All models are trained on the "Add Jump" task. (A) The visualization of the neural dynamics in the MEC network with two attractor modules. A torus-like manifold of stable attractor patterns in the neural activity space is shown for one attractor module. It is visualized through dimensionality reduction, with two neural trajectories projected onto this low-dimensional space traversing the manifold. The red star marks the final state of a neural trajectory, while the red circle indicates its start. For details about visualisation, see Appendix A.9. (B) Testing accuracy versus the number of attractor modules in the MEC network. The accuracy is averaged over 8 runs.

To further explore the use of velocity vector-driven attractor dynamics for representing syntactic structure, we established attractor dynamics using a predefined weight method (Burak & Fiete, 2009). In this approach, the recurrent connection weight of each attractor module in the MEC network is composed of two parts: $\mathbf{W}_r = \mathbf{W}_{\text{attractor}} + \mathbf{W}_{\text{lowrank}}$. Here, $\mathbf{W}_{\text{attractor}}$ defines the connection structure of the attractor dynamics, remaining fixed and non-learnable. In contrast, $\mathbf{W}_{\text{lowrank}}$ is a low-rank connection matrix that simulates the potential functional role of the numerous non-grid cells in the biological entorhinal cortex (Hardcastle et al., 2017; Mosheiff & Burak, 2019). This component is adaptable and enhances the flexibility of the attractor dynamics. For simplicity, the HC and MEC networks directly receive structure-content factorized input representations, as depicted in Fig. S6. For model details, refer to Appendix A.8.

As shown in Fig. 5A, dimensionality reduction and visualization demonstrate that the abstract velocity inputs drive the neural activity in the MEC network through the toroidal attractor manifold, representing syntactic structures of command sequences as trajectories, consistent with the findings in Fig. 4. Furthermore, as illustrated in Fig. 5B, the model exhibits enhanced performance with an increasing number of attractor modules in the MEC network, suggesting that modular attractor networks are particularly effective in representing structures of language command sequences.

4. Discussion

In this research, we introduce a dynamic model called PHE-trinity, which integrates two key aspects of information processing within the HC-MEC system: a grid network for structure representation and separate structure-content pathways. Our hypothesis is that preformed, velocity vector-driven modular attractor dynamics, shaped through spatial path integration tasks, are adaptable for facilitating language information processing. Experimental findings from SCAN tasks validate this hypothesis, indicating that these preformed attractor dynamics aid in systematic generalization, yield competitive results across various SCAN tasks, and improve learning efficiency, especially in limited data conditions. Moreover, we utilized visualization and reverse engineering methods to uncover a potential dynamic mechanism underlying command sequence parsing: utilizing grammar roles as velocity vectors and leveraging attractor dynamics in the grid system to integrate the syntactic structure information. Hopefully, this study could enhance our understanding of the computational dynamics shared by both spatial navigation and language information processing.

Related Works Our research is primarily inspired by the Tolman-Eichenbaum Machine (TEM) (Whittington et al., 2020), which suggests that the MEC forms a basis for describing structural knowledge, with the HC linking this basis to sensory content. Nevertheless, PHE-trinity diverges significantly from the TEM in several respects: 1) the TEM model is a structured memory model, focusing on spatial tasks and non-spatial tasks with similar statistical structure. In contrast, PHE-trinity is oriented towards more complex language command parsing tasks, incorporating a PFC network; 2) the TEM model presupposes the provision of decoupled structure-content representation a priori, whereas PHE-trinity is capable of directly learning and acquiring this representation from data, enabling the application of structure-content decoupling to more complex tasks. Additionally, PHE-trinity benefits from and bears some similarities to the Syntactic Attention method (Russin et al., 2019), inspired by the information processing mechanism in the Wernicke and Broca cortical areas. However, PHE-trinity concentrates on the hippocampal-entorhinal circuit, particularly its use of a velocity-driven modular attractor network for syntactic structure representation. Recently, a model called Vector-HaSH also adopts a similar idea (Chandra et al., 2023), where sequential inputs are also transformed into continuous velocity vectors. Thus Vector-HaSH model enables episodic memory sequences to stimulate grid network activities, thereby manifesting as spatiotemporal trajectories on low-dimensional attractor manifolds. PHE-trinity differs from Vector-HaSH model in two major ways: 1) PHE-trinity emphasizes that velocity vector representation is an abstract structure input that can be directly learned

from data; 2) PHE-trinity is applied to more complex language command parsing tasks, beyond simply memorizing sequences.

Shared Mechanism The question that "what is the shared computational mechanism for both navigation and language understanding?" is deeply rooted in hippocampal cognitive map theory (O'keefe & Nadel, 1978) and was originally and explicitly proposed by the distinguished researcher John O'Keefe, known for his Nobel Prize-winning discovery of place cells. In his famous book (O'keefe & Nadel, 1978), O'Keefe suggested that in the hippocampal formation, "the cognitive-mapping system could function as a deep structure for language", where "deep structure" mainly refers to the syntactic structure of language. Recent advances in experimental studies and theoretical understanding of hippocampal functions have brought this important question into more discussion (Frankland & Greene, 2020; Whittington et al., 2022b; Kazanina & Poeppel, 2023). For instance, Whittington et al. proposed that a velocity-driven grid system may be used to represent syntactic structure via path integration (Whittington et al., 2022b), akin to spatial navigation. Kazanina and Poeppel also argued that cell representations in the hippocampal formation could already constitute essential elements for a language of thought (Kazanina & Poeppel, 2023). Nevertheless, all these proposals and discussions have mainly remained at the conceptual stage, lacking an empirical model to test the hypothesis. Our present study is an initial attempt to construct a concrete computational model to validate this hypothesis.

Relating the Model to Neuroscience Some key aspects of PHE-trinity can be related to experimental studies in system neuroscience and contribute to our understanding of the brain. Firstly, in the biological brain, grid-like code can be reused to represent both spatial and non-spatial space (Constantinescu et al., 2016; Behrens et al., 2018; Bellmund et al., 2018). For spatial space, grid-like system is driven by self-motion signals, guiding the network state along a manifold and encoding relative spatial locations. However, the mechanism with which our brain efficiently repurposes this grid-like system for non-spatial tasks remains a question: what computational processes underlie this adaptation? PHE-trinity model suggests that the brain might infer abstract feature vectors underlying the map structure in the non-spatial domain, such as the length of a bird's neck (Constantinescu et al., 2016). Subsequently, the brain can reuse grid-like attractors by binding the inferred vectors with low-dimensional velocity vectors, rather than learning direct connections to high-dimensional grid population vectors.

Additionally, our model may provide insights into the computational mechanism underlying how the brain represents abstract rules and facilitates compositional generalization. In the PHE-trinity model, abstract rules can be encoded as stable neural trajectories in the PFC network. Through interaction with the structural components of codes in the HC network, the model can achieve rule-based decoding for better compositional generalization. This computational mechanism is supported by recent experimental studies to some extent. For instance, a recent experiment suggests that rats' PFC can encode abstract navigation rules as stable neural trajectories (Tang et al., 2023). Interestingly, when rats navigate different environments sharing the same navigation rule, the same neural trajectory can be activated in the rats' PFC, resembling the dynamic process in the PHE-trinity model. Moreover, recent studies on the human brain suggest that the interplay between the PFC and HC regions may be responsible for compositional inference (Schwartenbeck et al., 2023), a phenomenon that could be modeled and elucidated by the PHE-trinity model.

Limitations and Future Works More future works are needed to explore the shared computational mechanisms in spatial navigation and language understanding further. Firstly, PHE-trinity has only been demonstrated in a simplified dataset - the SCAN dataset - which captures limited aspects of human language processing. In future work, we need to test the effectiveness of the PHE-trinity model in more diverse language tasks and with different model variations. Secondly, the modular grid system provides many computational advantages such as exponentially high coding capacity (Chandra et al., 2023), error correction ability (Sreenivasan & Fiete, 2011), and algebraic computation (Whittington et al., 2022b; Gao et al., 2021). However, the current PHE-trinity model mainly focuses on information integration through continuous attractor dynamics in the grid system. Exploring how to leverage the diverse computational advantages of the grid system for various language processing tasks is a future study avenue.

Currently, PHE-trinity only provides a rough simulation of the PFC, HC and MEC and lacks several critical features: 1) the model does not include the feedback connections from HC to MEC, which are essential for correcting path integration errors (Whittington et al., 2020); 2) the biological HC has the capability to group and organize memory sequences based on their underlying structures (Eichenbaum, 2017), rather than simply serving as a storage unit; 3) the PFC is able to abstract schemas from various similar experiences, thereby enhancing the reuse of learned knowledge and generalization (Gilboa & Marlatte, 2017). Therefore, it is important to integrate these features in future work to improve the model's capability in language information processing.

Impact Statement

This paper aims to enhance our understanding of the common computational processes involved in spatial navigation and language processing. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Behrens, T. E., Muller, T. H., Whittington, J. C., Mark, S., Baram, A. B., Stachenfeld, K. L., and Kurth-Nelson, Z. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018.

Bellmund, J. L., Gärdenfors, P., Moser, E. I., and Doeller, C. F. Navigating cognition: Spatial codes for human thinking. *Science*, 362(6415):eaat6766, 2018.

Blank, I. A., Duff, M. C., Brown-Schmidt, S., and Fedorenko, E. Expanding the language network: domain-specific hippocampal recruitment during high-level linguistic processing. *BioRxiv*, pp. 091900, 2016.

Burak, Y. and Fiete, I. R. Accurate path integration in continuous attractor network models of grid cells. *PLoS computational biology*, 5(2):e1000291, 2009.

Buzsáki, G. and Tingley, D. Space and time: the hippocampus as a sequence generator. *Trends in cognitive sciences*, 22(10):853–869, 2018.

Chandra, S., Sharma, S., Chaudhuri, R., and Fiete, I. High-capacity flexible hippocampal associative and episodic memory enabled by prestructured"spatial"representations. *bioRxiv*, pp. 2023–11, 2023.

Chomsky, N. *The minimalist program*. MIT press, 2014.

Constantinescu, A. O., O'Reilly, J. X., and Behrens, T. E. Organizing conceptual knowledge in humans with a grid-like code. *Science*, 352(6292):1464–1468, 2016.

Dehaene, S., Meyniel, F., Wacongne, C., Wang, L., and Pallier, C. The neural representation of sequences: From transition probabilities to algebraic patterns and linguistic trees. *Neuron*, 88:2–19, 2015.

Eichenbaum, H. Memory: Organization and control. *Annual review of psychology*, 68:19–45, 2017.

Frankland, S. M. and Greene, J. D. Concepts and compositionality: in search of the brain's language of thought. *Annual review of psychology*, 71:273–303, 2020.

Gao, R., Xie, J., Wei, X.-X., Zhu, S.-C., and Wu, Y. N. On path integration of grid cells: group representation and isotropic scaling. *Advances in Neural Information Processing Systems*, 34:28623–28635, 2021.

Gardner, R. J., Hermansen, E., Pachitariu, M., Burak, Y., Baas, N. A., Dunn, B. A., Moser, M.-B., and Moser, E. I. Toroidal topology of population activity in grid cells. *Nature*, 602(7895):123–128, 2022.

Gilboa, A. and Marlatte, H. Neurobiology of schemas and schema-mediated memory. *Trends in cognitive sciences*, 21(8):618–631, 2017.

Giocomo, L. M., Moser, M.-B., and Moser, E. I. Computational models of grid cells. *Neuron*, 71(4):589–603, 2011.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

Grodzinsky, Y. The neurology of syntax: Language use without broca's area. *Behavioral and Brain Sciences*, 23: 1 – 21, 2000.

Hahamy, A., Dubossarsky, H., and Behrens, T. E. The human brain reactivates context-specific past information at event boundaries of naturalistic experiences. *Nature neuroscience*, pp. 1–10, 2023.

Hardcastle, K., Maheswaranathan, N., Ganguli, S., and Giocomo, L. M. A multiplexed, heterogeneous, and adaptive code for navigation in medial entorhinal cortex. *Neuron*, 94(2):375–387, 2017.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Kazanina, N. and Poeppel, D. The neural ingredients for a language of thought are available. *Trends in Cognitive Sciences*, 2023.

Krubitzer, L., Campi, K. L., and Cooke, D. F. All rodents are not the same: a modern synthesis of cortical organization. *Brain Behavior and Evolution*, 78(1):51–93, 2011.

Kumaran, D., Hassabis, D., and McClelland, J. L. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20:512–534, 2016.

Lake, B. M. Compositional generalization through meta sequence-to-sequence learning. *Advances in neural information processing systems*, 32, 2019.

Lake, B. M. and Baroni, M. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, 2023.

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., and Gershman, S. J. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

Lerousseau, J. P. and Summerfield, C. Space as a scaffold for rotational generalisation of abstract concepts. *eLife*, 13, 2024.

Mansouri, F. A., Freedman, D. J., and Buckley, M. J. Emergence of abstract rules in the primate brain. *Nature Reviews Neuroscience*, 21(11):595–610, 2020.

Mosheiff, N. and Burak, Y. Velocity coupling of grid cell modules enables stable embedding of a low dimensional variable in a high dimensional neural attractor. *Elife*, 8: e48494, 2019.

O'keefe, J. and Nadel, L. *The hippocampus as a cognitive map*. Oxford university press, 1978.

O'Reilly, R. C. and Frank, M. J. Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*, 18(2): 283–328, 2006.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems*, 2019.

Piai, V., Anderson, K. L., Lin, J. J., Dewar, C., Parvizi, J., Dronkers, N. F., and Knight, R. T. Direct brain recordings reveal hippocampal rhythm underpinnings of language processing. *Proceedings of the National Academy of Sciences*, 113(40):11366–11371, 2016.

Russin, J., Jo, J., O'Reilly, R. C., and Bengio, Y. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*, 2019.

Schwartenbeck, P., Baram, A., Liu, Y., Mark, S., Muller, T., Dolan, R., Botvinick, M., Kurth-Nelson, Z., and Behrens, T. Generative replay underlies compositional inference in the hippocampal-prefrontal circuit. *Cell*, 186(22):4885–4897, 2023.

Solomon, E. A., Lega, B. C., Sperling, M. R., and Kahana, M. J. Hippocampal theta codes for distances in semantic and temporal spaces. *Proceedings of the National Academy of Sciences*, 116(48):24343–24352, 2019.

Sorscher, B., Mel, G. C., Ocko, S. A., Giocomo, L. M., and Ganguli, S. A unified theory for the computational and mechanistic origins of grid cells. *Neuron*, 111(1): 121–137, 2023.

Sreenivasan, S. and Fiete, I. Grid cells generate an analog error-correcting code for singularly precise neural computation. *Nature neuroscience*, 14(10):1330–1337, 2011.

Strange, B. A., Witter, M. P., Lein, E. S., and Moser, E. I. Functional organization of the hippocampal longitudinal axis. *Nature Reviews Neuroscience*, 15(10):655–669, 2014.

Sussillo, D. and Barak, O. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25:626–649, 2013.

Tang, W., Shin, J. D., and Jadhav, S. P. Geometric transformation of cognitive maps for generalization across hippocampal-prefrontal circuits. *Cell reports*, 42(3), 2023.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., Hassabis, D., and Botvinick, M. M. Prefrontal cortex as a meta-reinforcement learning system. *Nature Neuroscience*, 21:860 – 868, 2018.

Whittington, J. C., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T. E. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

Whittington, J. C., Dorrell, W., Ganguli, S., and Behrens, T. Disentanglement with biological constraints: A theory of functional cell types. In *The Eleventh International Conference on Learning Representations*, 2022a.

Whittington, J. C., McCaffary, D., Bakermans, J. J., and Behrens, T. E. How to build a cognitive map. *Nature neuroscience*, 25(10):1257–1272, 2022b.

Whittington, J. C. R., Warren, J., and Behrens, T. E. J. Relating transformers to models and neural representations of the hippocampal formation. *ArXiv*, abs/2112.04035, 2021.

Zheng, J., Schjetnan, A. G., Yebra, M., Gomes, B. A., Mosher, C. P., Kalia, S. K., Valiante, T. A., Mamelak, A. N., Kreiman, G., and Rutishauser, U. Neurons detect cognitive boundaries to structure episodic memories in humans. *Nature neuroscience*, 25(3):358–368, 2022.

A. Appendix

A.1. Details about SCAN dataset

Train	
jump	JUMP
run after look left	LTURN LOOK RUN
run twice and turn opposite left	RUN RUN LTURN LTURN
Test	
jump thrice and look	JUMP JUMP JUMP LOOK
turn left after **jump** twice	JUMP JUMP LTURN
jump opposite left and run left	LTURN LTURN JUMP LTURN RUN

Table S1. Examples of training and testing splits of "Add Jump" task in SCAN Dataset.

The SCAN dataset comprises numerous "command-action sequence" pairs (Lake, 2019). The input commands consist of sentences formed using primitives (e.g., jump, walk, turn left) and modifiers (e.g., twice, around, and, after) according to specific grammatical rules. The output is sequences of predefined actions corresponding to these primitives.

The "Add Primitive" task in the SCAN dataset evaluates the model's ability to generalize learned parsing rules to new words or phrases (Tab. S1). Now consider the "Add Jump" task as an example. In the training set, "jump" appears only as an isolated word in parsing tasks, not combined with any other words, as illustrated by the example: "jump→JUMP". However, in the test set, "jump" can be combined with various words just like other primitives, for instance, "jump thrice", "jump opposite and run". If a model only learns one-to-one mapping between inputs-outputs pairs through memorization during training, it may neglect all lengthy sentences incorporating "jump" during the test, resulting in random outputs. On the other hand, if the model learns the corresponding relationships between "jump" and other primitive words, as well as the rules of command parsing, it can successfully complete the task shown in Tab. S1, e.g., "turn left after jump twice→JUMP JUMP LTURN". The "Add Turn Left" task is similar to the "Add Jump" task, except "jump" is replaced with "turn left".

The "Length" task in the SCAN dataset requires the model to better learn the rules for constructing longer sentences through command loops. In this task, 80% of the training set data comprises shorter commands similar to the examples in Tab. S1. In the test set, however, the output sequence length can exceed 20, for example: "walk around left twice after turn around right twice→RTURN RTURN RTURN RTURN RTURN RTURN RTURN RTURN LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN WALK LTURN WALK".

A.2. Model Details for Path Integration Task

In the path integration task, we assume an agent navigates within a $2.2\,\mathrm{m} \times 2.2\,\mathrm{m}$ 2D spatial space, moving at an average speed of 0.1m/sec. By randomly initializing the agent's position within this space and sampling velocity directions, we obtain precise coordinate trajectories of the agent's movement and velocity samples at each moment. We set the temporal length of trajectory samples to 20. For each gradient iteration, we randomly sample data trajectories from the space. Notably, the representation of position coordinates is encoded using a vector representation of the place cell population, emulating the spatial encoding relationship between the hippocampal-entorhinal circuit of a real biological brain, with the number of place cells being 512. More details can be referenced in (Sorscher et al., 2023).

A.3. Model Structure and Training Parameters

As shown in Fig. S1, we provide a more detailed model architecture diagram here. The whole model structure can be seen as an encoder-decoder architecture. The interaction between MEC and HC performs the text command sequence encoding, while the interaction in PFC-HC loop decodes the input into action command sequence. The principal variables mentioned in the methodology have been annotated in the figure.

The dimensions of the input command embedding vector $\mathbf{x}$, the structure representation vector $\mathbf{m}_s$ and the semantic representation vector $\mathbf{m}_c$ in PHE-trinity are $D_x = 256$, $N_s = 100$, $N_c = 256$, respectively. When training the model using auxiliary spatial path tasks, the number of modules in the MEC network is $N = 1$, with $N_g = 256$ neurons. Between the MEC network and the PFC network is a 2-layer MLP network, each with 256 neurons. All models are trained using Pytorch 2.0 (Paszke et al., 2019). Dropout is applied to all the feedforward connections of the model with $p = 0.5$, except

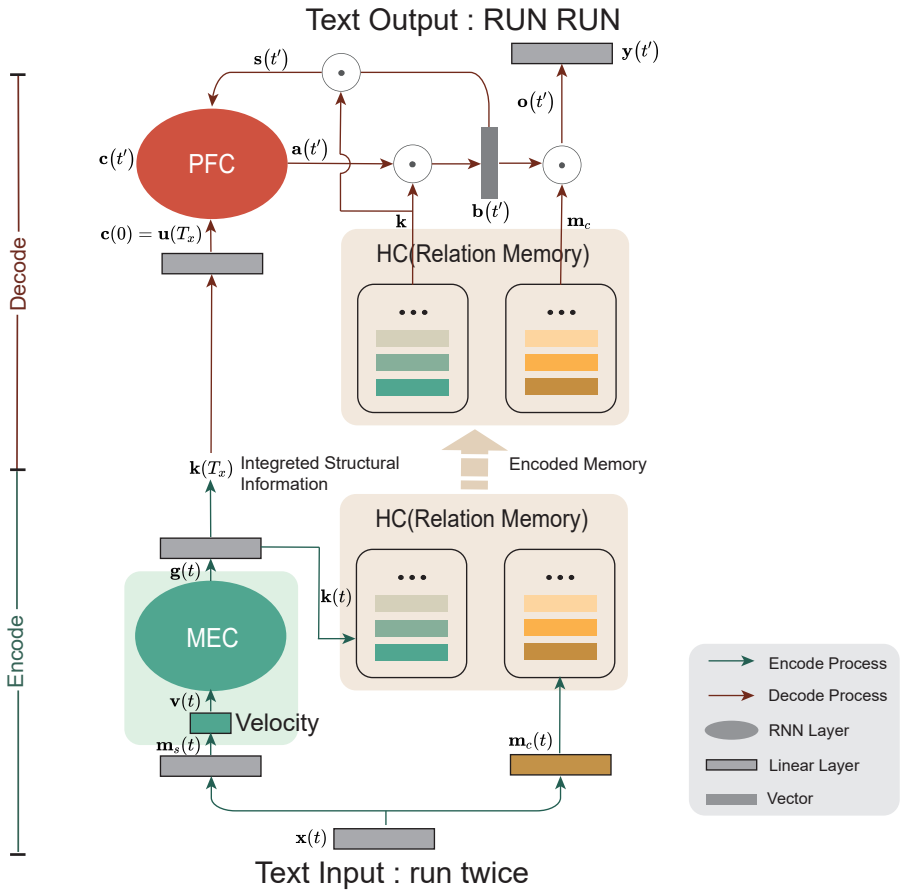

Figure S1. Detailed model structure.

for the connections from the velocity input to the MEC network. When the model is trained solely on SCAN taks, the recurrent connections in the MEC network are randomly initialized using the `torch.nn.init.xavier_uniform` method. We also experimented with `torch.nn.init.orthogonal` but found no significant difference in the results. When training on SCAN tasks, the batch size is set to $B = 1$. The neural activity constraint coefficients for structure and content representations are set to $\alpha_1 = \alpha_2 = 0.2$. When the control model is trained solely on the text task, we employ the co-training loss function with $\alpha = 0$; when the model is trained simultaneously on both text task and path integration task, $\alpha = 1$. These model structure and training parameters remain unchanged throughout the experiments, unless specifically noted otherwise.

Parameters in Tab. 2 in the main text: In the pre-training phase, PHE-trinity undergoes pre-training on path integration tasks. During this phase, we set the batch size B_{path} to 200 and the regularization parameter β to 1×10^{-4}. The training involves 56,000 iterations using the Adam optimizer with an initial learning rate of 0.001. Subsequently, in the co-training phase, the model is trained concurrently on both path integration tasks and the SCAN task for a duration of 10 training epochs. Here, we maintain the initial learning rate at 0.001. To control the magnitudes of velocity vectors, we apply an L_2 regularizer on $\mathbf{v}$ in text parsing tasks, with the regularizer strength set to 0.1. In experiments with control model 1 and control model 2, we find no qualitative difference between results with or without the regularizer, compared with PHE-trinity. The Adam optimizer is consistently employed, and the learning rate is decreased by a factor of 0.1 every 4 epochs. Notably, Control models 1 and 2 follow identical training settings as PHE-trinity during the co-training phase.

Parameters in Fig. 3 in the main text: the pre-training phase in PHE-trinity is the same as that in Tab. 2. In the co-training phase, regardless of the number of training samples, PHE-trinity undergoes simultaneous training in both path integration tasks and the SCAN task, spanning 60,000 training iterations. The learning rate is systematically reduced by a factor of 0.1

at iteration milestones set at 9,000 and 24,000. Other parameters is the same as that in Tab. 2. Notably, the Control model follow identical training settings as PHE-trinity during the co-training phase.

Parameters in Fig. 4 in the main text: without loss of generality, the number of neurons in the MEC is set to $N_g = 1024$ for better visualizeing the torus-like attractor structure. Other settings remain the same as in Tab. 2. In Fig. 4C, 1000 sequence trajectories in the MEC during spatial path integration tasks are randomly selected, these sequences have the same starting points as that of command sequence trajectories. Similarly, 1000 sequence trajectories of command sequences are randomly sampled. As the torus-like attractor manifold mainly lies in the 6D subspace, here we use the 6D subspace spanned by the 6 top PCs of navigation trajectories to explain the variance of that of command sequences.

A.4. Results of Models with Shuffled Grid Network Weights

We perform additional experiments to further validate whether the attractor dynamics in PHE-trinity is a critical component. Results are shown in Tab. S2. "PHE-trinity + shuffle" refers to the condition where the recurrent weights in the MEC network are shuffled after the pretraining phase. But their weight magnitudes and distribution are maintained. "Control model 1 + shuffle" means that the recurrent weights in the MEC network in "Control model 1" are initialized the same as in "PHE-trinity + shuffle", using shuffled grid network weights.

Shuffling the recurrent weights in MEC destroys the formed attractor dynamics during the pretaining phase (obtained via spatial path integration task). Therefore, despite the maintained weight distribution and magnitudes, both "PHE-trinity + shuffle" and "Control model 1 + shuffle" still have poor performance compared with PHE-trinity across different SCAN tasks. These results suggest that PHE-trinity's good performance can not simply stem from a specific way of initialization via pretraining, and support that the attractor dynamics is a critical factor.

Model/Task	Add Jump	Length	Opposite Right
Control model 1	74.96 ± 42.92	17.0 ± 2.5	87.59 ± 32.83
Control model 1 + shuffle	74.08 ± 42.83	17.49 ± 4.7	96.26 ± 9.84
PHE-trinity + shuffle	74.85 ± 43.2	20.80 ± 5.04	98.14 ± 4.87
PHE-trinity	100.0 ± 0.0	21.48 ± 3.4	99.98 ± 0.02

Table S2. Test accuracies across different models and tasks are both obtained over 8 runs. Seeds and other parameters are the same as PHE-trinity model (details described in Appendix A.3).

A.5. Grid Representation Formation Based on Path Integration Task Pre-training

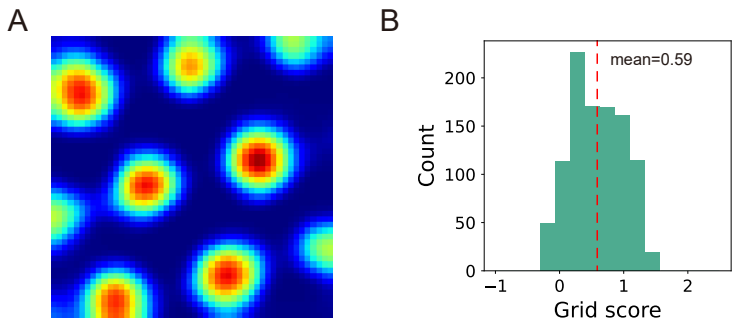

Figure S2. Grid representation and statistical distribution formed by path integration pre-training. A) The spatial firing rate of a typical neuron shows a regular hexagonal representation. B) This characteristic is observed in the majority of neurons, indicating the formation of grid cells (grid score >0.3).

To visualize the formation of grid representations in the MEC network of the model, we randomly sampled 10^5 trajectories on a 2D plane. We recorded the firing $\mathbf{g}_i(\mathbf{x})$ of MEC neurons at different positions $\mathbf{x}$ during path integration. The trajectory

sampling method and parameters are consistent with Appendix A.2. To improve the signal-to-noise ratio of neuronal activity and reduce the computational burden for subsequent parameter statistics, we further discretized the 2D space into a 50×50 grid. Subsequently, for each neuron we averaged its firings at the same grid location across different trials to obtain the averaged spatial firing rate $\bar{\mathbf{g}}_i(\mathbf{x})$. We selected a representative neuron and plotted its spatial firing rate in Fig. S2A.

To evaluate the degree to which the spatial firing rate forms a periodic hexagonal grid pattern, we can calculate the grid score using a method outlined in (Sorscher et al., 2023). Briefly, we first computed the autocorrelogram of $\bar{\mathbf{g}}_i(\mathbf{x})$ at rotations of 0°, 30°, 60°, 90°, 120°, and 150°. Then, we calculate the average of the correlation values of the 0° autocorrelogram with those at 60° and 120°, and subtract it from that of 0° with 30°, 90°, and 150°. We computed the grid scores for all neurons in the MEC network. The results, as shown in Fig. S2B, indicate that most neurons display hexagonal grid characteristics, with an average grid score of 0.59. This suggests the effective formation of grid representations within the MEC network.

A.6. Visualization of Abstract Velocity Corresponding to Text Commands

We train PHE-trinity under eight different random seed conditions. We visualize the velocity representations learned from data in a 2D plane, as depicted in Fig. S3. Across all results, we observe that: 1) The velocity vectors corresponding to verbs such as "walk", "jump", "look", "run" and direction words like "left", "right" are closely **clustered** together. 2) The pairs "and" and "after", as well as "around" and "opposite", are always positioned in opposite **directions** relative to the origin, which may be related to their contrasting logical roles in text command parsing.

We further quantified and compared the "clustering" and "direction" properties of PHE-trinity and control models. Each result is averaged over eight runs.

For quantification of "cluster" effect, we calculated the intra-cluster distance (average distance between all points pairwise in a cluster). As shown in Tab. S3: 1) for "left, right", all models are able to cluster them effectively together. 2) as for "walk, run, look, jump", the control models exhibit worse clustering and higher variance, although they do cluster to a fairly good extent.

Cluster of words	Control model 1	Control model 2	PHE-trinity
walk, run, look, jump	0.12±0.27	0.12±0.17	**0.02±0.03**
left, right	0.00±0.00	0.01±0.01	0.01±0.01

Table S3. Comparison of "cluster" effect of Control Model 1, Control Model 2, and PHE-trinity.

For quantification of "direction" (words on the opposite versus same directions), we calculated the angles between word "velocity" vectors in degrees. As shown in Fig. S4, 1) In comparison to the control models, our model demonstrates the most stable angles between "velocity" vectors across various trials, with smaller variances observed among the respective terms; 2) to a certain extent, these angles corresponds with the model's behavior of text parsing. For example, the angles of "and" and "after" in PHE-trinity consistently occupy opposite directions with the angle 150 ± 21, compared to control models. The opposite direction and their large velocity magnitudes may imply their distinct logical roles in language parsing, as depicted in Fig. 4D, resulting in two distinct trajectory groups in the MEC. Note other words, such as "twice" and "thrice", even "twice" and "and", also exhibit stable vector angles in PHE-trinity. These consistent vector angles could reflect statistical similarities and distinctions observed in the SCAN dataset. For instance, "run twice" and "run and run" yield the same target command sequence "RUN RUN," whereas "run and walk" is unique to "and" but not to "twice".

Overall, as control models share the similar separate structure-content streams as that in PHE-trinity, they both learn relatively stable syntactic representation of words. However, representation in PHE-trinity shows a significantly lower variance and better clustering effect, indicating better syntactic representation. These results indicate that PHE-trinity can learn abstract structure representations from the data, achieving a decoupling of structure and content at the representation level.

A.7. Neural dynamics in the PFC network

We provide more details about PFC dynamics as shown in Fig. S5. PCA results show the neural dynamics in PFC is low-dimensional like that in MEC (Fig. S5A).

Importantly, each trajectory in PFC network corresponds to an abstract parsing rule for sequences with a specific syntactic structure (Fig. S5B). For example, sequences like "jump and walk" and "run and jump" would trigger the same trajectory in

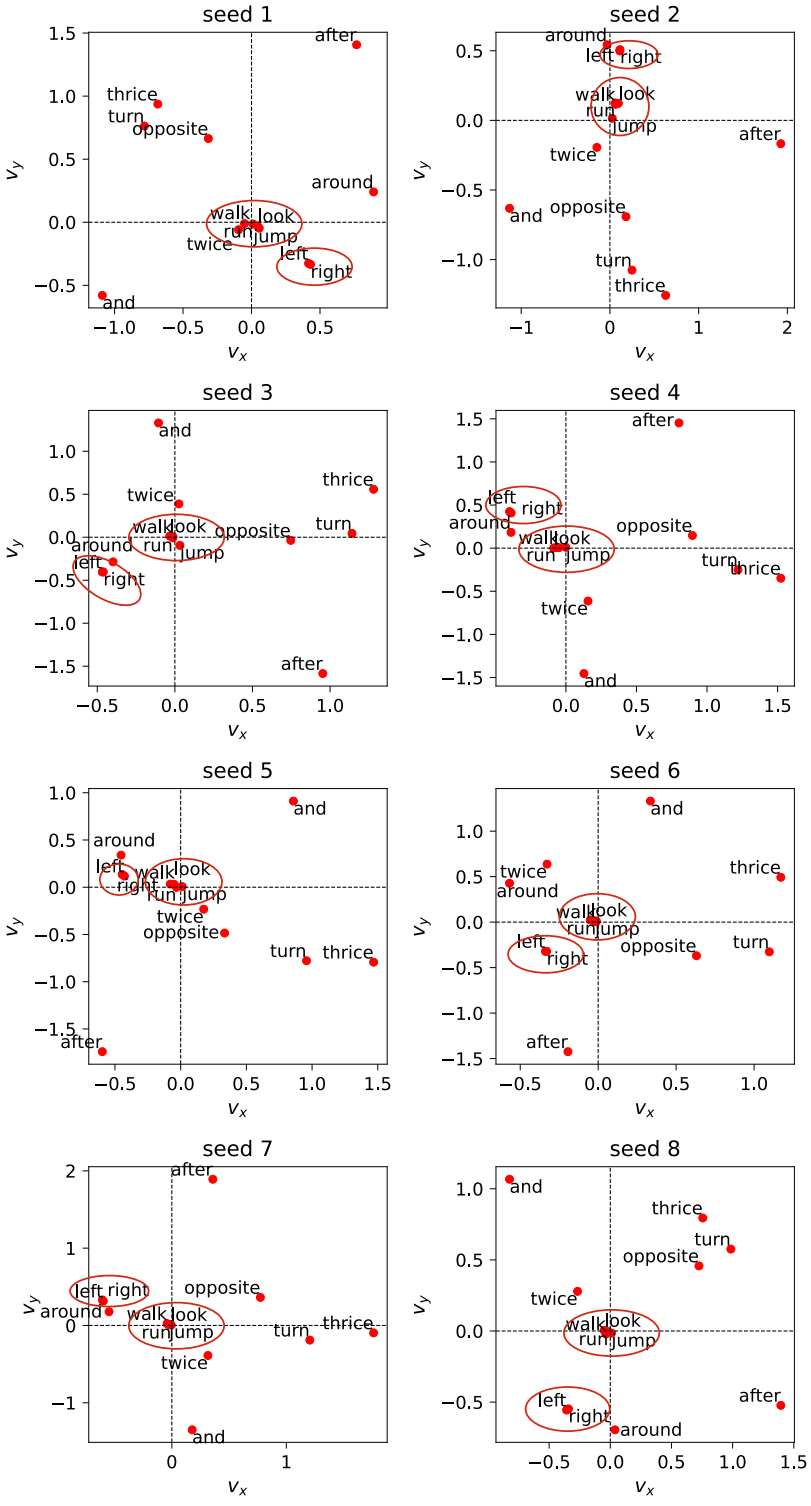

Figure S3. The 'velocity' vectors of textual inputs from models trained with 8 different random numbers. The red circles mark the positions where `walk, jump, look, run` and `left, right` are clustered.

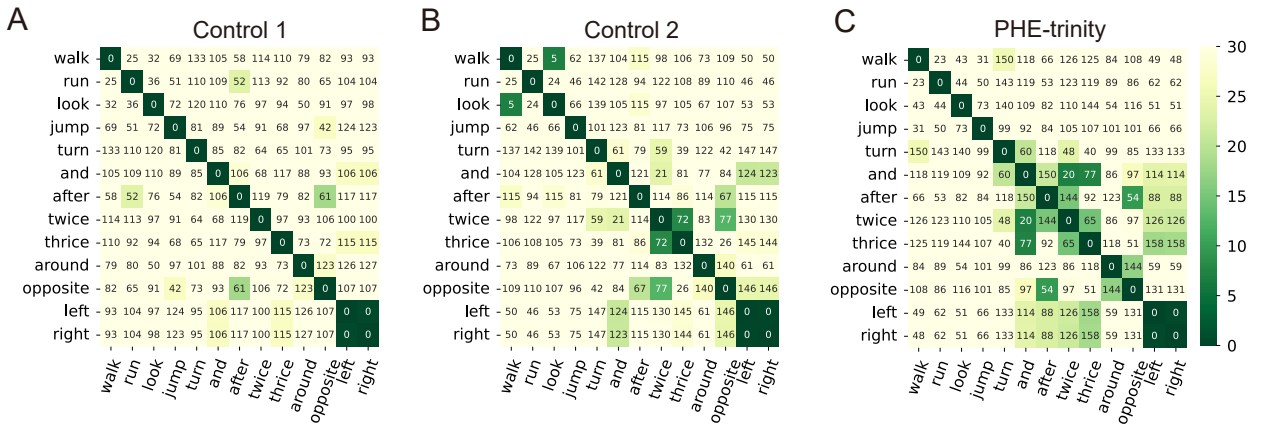

Figure S4. The angles of 'velocity' vectors of textual inputs from models trained with 8 different random seeds. Numerical values denote the average angles(0-180) between corresponding words, while the color intensity represents the standard deviation. The green color denotes standard derivations less than 30 degree.

the PFC network because they follow the same syntactic rule - "verb and verb". However, sequences such as "jump and walk" and "run after walk" use different syntactic rules: "verb and verb" versus "verb before verb," which would result in two separate trajectories in the PFC network. Similarly, sequences like "run around right twice after look right thrice" and "jump around right twice after walk left thrice" share the same syntactic rule and trigger the same trajectory in the PFC. This is in contrast to the sequence "run around right twice **and** look right thrice".

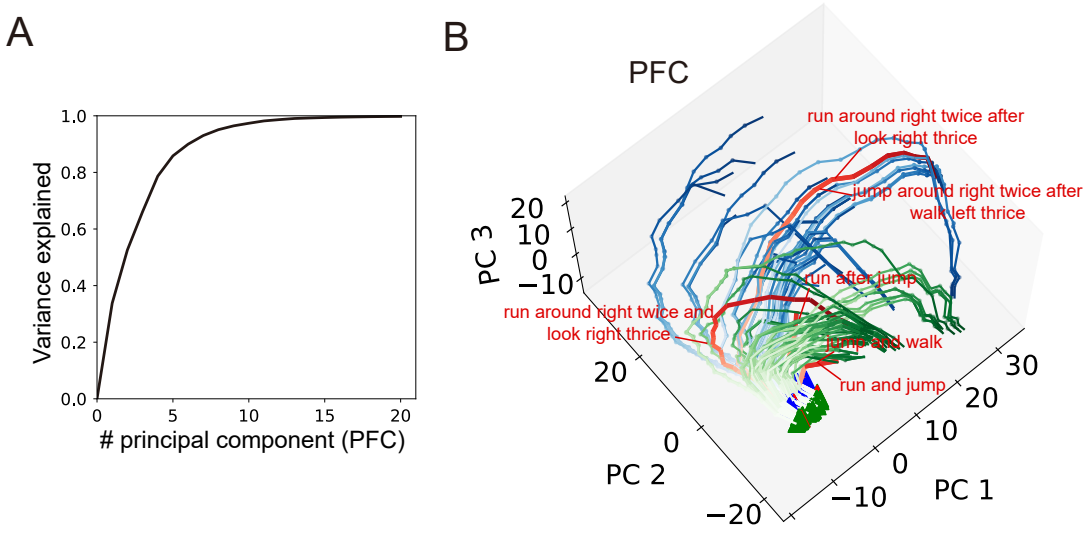

Figure S5. Neural dynamics in the PFC network, related to Fig. 4. (A) Dimensionality of neural dynamics in the PFC network. Cumulative percentage of explained variance is shown as a function of the number of principal components. (B) PCA visualization of the neural dynamics in the PFC network. Explanatory cases are delineated in the figure with red lines.

A.8. Predefined Attractor Network

Since the continuous attractor dynamics mechanism of grid systems has been extensively studied, alternatively, we can also introduce hand-designed continuous attractor structures in the recurrent network by presetting the parameters $\mathbf{W}_r$ and $\mathbf{W}_{in}$ of the MEC grid network (Fig. S6). Investigating predefined attractor network serves a dual purpose: firstly, it helps further validate the feasibility of using a modular grid system to represent the structure of language command sequences;

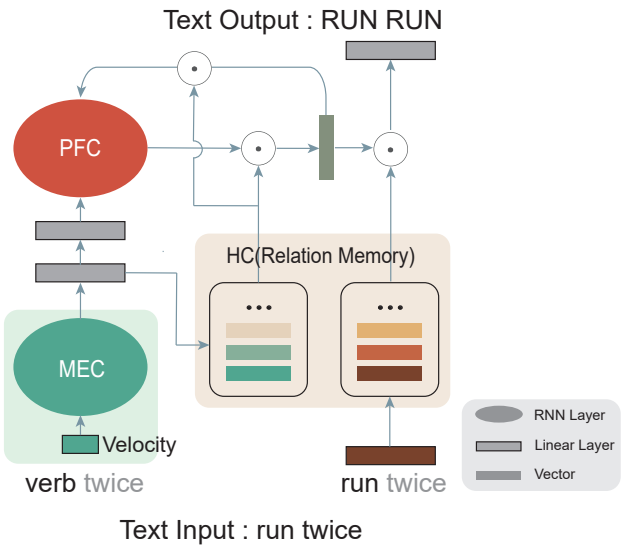

Figure S6. Model architecture using predefined attractor structure method

secondly, it also aids in understanding how these vital prior structures can be effectively applied to language parsing tasks. Additionally, in the real biological entorhinal cortex, there exist many non-grid cells besides grid cells. These non-grid cells might form low-rank connections, acting on attractor manifolds to assist the MEC network in integrating external velocity inputs more effectively. Consequently, in the MEC cortex network, each attractor module's reciprocal connections comprise two parts: $\mathbf{W}_r = \mathbf{W}_{attractor} + \mathbf{W}_{lowrank}$. Here, $\mathbf{W}_{attractor}$ determines the predefined continuous attractor structure, and $W_{lowrank}$ is a learnable part of the recurrent connections, simulating the connection structure of non-grid cells. The neural dynamics of the attractor network are defined with the equation:

$$\tau \frac{d\mathbf{g}(t)}{dt} = -\mathbf{g}(t) + \sigma(\mathbf{W}_{attractor}\mathbf{g}(t) + \mathbf{W}_{lowrank}\mathbf{g}(t) + \mathbf{F}),$$

where $\mathbf{g}(t)$ is the neural population activity vector at time t, $\sigma(\cdot)$ is the ReLU non-linear activation function.The time constant $\tau = 10$, and the simulation step length $dt = 0.5$. For simplicity, all attractor modules use the same model parameters. The number of neurons is $N_g = 24^2$, arranged on a 24×24 square plane. Furthermore, the plane's neural clusters are divided into 2×2 blocks, with each block containing four neurons with distinct preferred directions (east, south, west, north), denoted as $\theta_i \in \{0, \pi/2, \pi, 3\pi/2\}$. Let $\hat{\mathbf{e}}_\theta$ be the unit vector in direction θ, and l_i be the position coordinate of neuron i on the neural plane. Then,

$$\mathbf{W}_{attractor}(i,j) = \mathbf{W}_{base}(l_i - l_j - z\hat{\mathbf{e}}_{\theta_j}),$$

$$\mathbf{W}_{base}(l) = e^{-\eta\|l\|^2} - e^{-\xi\|l\|^2},$$

where $\eta = 1.2 \times \xi$, $\xi = 3/\lambda^2$, $\lambda = 13$, and the offset value is $z = 2$. The values of η and ξ ensure that, the connection matrix $\mathbf{W}_{attractor}$ between each neuron i and other neurons forms a specific center-surround shape on the neuron tile. All connections are inhibitory, with the bump centered at $l_i - z\hat{\mathbf{e}}_{\theta_j}$. The value of λ determines the period of the hexagons formed on the neuron tile.

Driven by the external velocity input $\mathbf{v}(t)$, the feedforward input received by each neuron i in the network at time t is given by:

$$\mathbf{F}_i(t) = 1 + \alpha_0 \hat{\mathbf{e}}_{\theta_i} \cdot \mathbf{v}(t),$$

where α_0 determines the magnitude of the modulation effect of the velocity input on network activity. It is crucial to ensure $\alpha_0|\mathbf{v}| \ll 1$ to maintain the stability of the hexagonal representation under velocity drive. In this case, we set $\alpha_0 = 0.103$.

When the rank is R, the low-rank connectivity matrix $W_{lowrank}$ can be composed of R pairs of connecting vectors $\mathbf{m}^{(r)} = \{\mathbf{m}^{(r)}\}_{r=1,\ldots,R}$ and $\mathbf{n}^{(r)} = \{\mathbf{n}^{(r)}\}_{r=1,\ldots,R}$, where $\mathbf{m}^{(r)}, \mathbf{n}^{(r)} \in \mathbb{R}^{N_g}$. Hence,

$$\mathbf{W}_{lowrank} = \frac{1}{N_g} \sum_{r=1}^{R} \mathbf{m}^{(r)T} \mathbf{n}^{(r)},$$

where for an MEC network with a module count of 1, $R = 2$; for module counts of 2 and 3, $R = 1$.

As shown in Fig. S6, unlike when training on spatial path integration tasks, the model now receives predefined decoupled inputs. For the MEC model, when command word inputs are "run", "look", "walk", "jump", the model receives the same velocity vector input. Similarly, "left", "right" correspond to the same abstract velocity. By contrast, for the HC model, different words correspond to different semantic vector inputs. Both the velocity representation vector $\mathbf{v}(t)$ and the semantic representation vector $\mathbf{m}_c(t)$ are trainable, with initial values randomly sampled from a Gaussian distribution. For each word input, we inject its velocity input vector v by repeating it for several time steps T_s, with $T_s = 5$, to match the network time constant.

The target loss function for model training is:

$$L = -\frac{1}{BT_y} \sum_{b=1}^{B} \sum_{t=1}^{T_y} \hat{\mathbf{y}}_b(t) \log(\mathbf{y}_b(t)) + \alpha \sum_{n=1}^{N} \|\mathbf{W}_{lowrank}\|_2^2,$$

where the first term is the cross-entropy loss function between the predicted action sequence and the actual target action sequence for PHE-trinity. The second term is an l_2 regularization on the low-rank matrix, controlled by the parameter α, which ensures the magnitude of the learnable weights is kept within a certain range to maintain the structural integrity of the network's attractor manifold.

Notably, $\mathbf{W}_{attractor}$ in the model is not learned but predefined and remains fixed during training; $\mathbf{W}_{in}$ is implicitly included in $\mathbf{F}$ and has no trainable parameters. Adam optimizer is used for training. The initial learning rate is 0.001, reduced by a factor of 0.1 every four training cycles, with a total of 10 training epochs. During training, $\alpha = 0.0001$ and $B = 1$.

A.9. Visualization Method

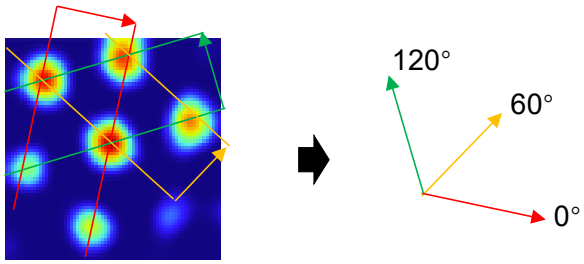

Figure S7. The hexagonal grid representation can be decomposed into three 2D sine waves.

In this study, two distinct methods are employed to pre-construct attractor dynamics within the MEC network, as depicted in Fig. 4A and Fig. 5A. For clarity, we utilize two visualization techniques to represent the attractor manifold embedded in the MEC network's neural activity space.

We visualised the continuous attractor manifold in Fig. 4 primarily following (Sorscher et al., 2023). The hexagonal representation of grid cells in the MEC can be considered as the superposition of 2D sine waves along three evenly spaced directions: 0°, 60°, and 120°, as illustrated in Fig. S7. In a 2D spatial coordinate system, the N_g-dimensional neural population activity $\mathbf{g}(x)$ at location $\mathbf{x}$ can be projected onto three planes based on the neuron's phases $\phi^{0,60,120}$ in these three directions. Each direction corresponds to a pair of axes $(\cos(\phi^{0,60,120}), \sin(\phi^{0,60,120}))$. Identifying the attractor states of the MEC network is equivalent to projecting the neural population activity onto the six-dimensional space. Briefly, the steps

to visualize the continuous attractor manifold include: 1) calculating the neural activity $\mathbf{g}_i(x)$; 2) identifying the neuronal phases $\phi^{0,60,120}$; 3) computing the projection of the neural population activity.

1) Similar to Appendix A.5, we first obtain the state activities $\mathbf{g}_i(\mathbf{x})$ of the MEC network under spatial path integration tasks. Then, we discretize the space and average the neuronal firing at each location to calculate the averaged neural activities $\bar{\mathbf{g}}_i(\mathbf{x})$ in a 2D discrete space.

2) After performing a Fourier transform on the $\bar{g}_i(\mathbf{x})$, the power spectral density (PSD) will show energy peaks at frequencies corresponding to the three directions. Hence, we can calculate the spatial phases $\phi_i^{0,60,120}$ for each neuron:

$$\phi_i^{0,60,120} = \arg\left[\int d\mathbf{x}\, e^{-(\mathbf{k}^{0,60,120}\cdot\mathbf{x})}\bar{\mathbf{g}}_i(\mathbf{x})\right]$$

3) Then we can project the neural cluster activity at each spatial location onto six axes:

$$\left(\sum_i \bar{\mathbf{g}}_i(\mathbf{x})\cos(\phi_i^{0,60,120}), \sum_i \bar{\mathbf{g}}_i(\mathbf{x})\sin(\phi_i^{0,60,120})\right)$$

Selecting three of these directions, we obtain the 3D torus structure shown in (Fig. 4A).

Finally, since the MEC input in the text parsing task is already a 2D velocity representation, we chose one trajectory each for path integration and text parsing tasks, projecting them into the 3D space of the torus using the aforementioned process and parameters. The original projected trajectories are translated and scaled for better visual presentation.

The visualization method for Fig. 5 can be referenced from (Gardner et al., 2022). The process is as follows: 1) We randomly initialize the state of the MEC network and then use an optimized method (Sussillo & Barak, 2013) to find stable points in the high-dimensional network dynamics, i.e., the attractor states of the MEC network; 2) We perform PCA on the identified stable points, retaining the first six dimensions to exclude some noise; 3) We use the UMAP(Uniform Manifold Approximation and Projection) method for further nonlinear dimensionality reduction to three dimensions, and plot these stable points in the space. The UMAP parameters are set as follows: `n_dims=3, metric="cosine", n_neighbours=2000, min_dist=0., init="spectral"`

We selected two command sequences containing "and" or "after". We took the neuronal activities from the MEC network and projected them into 3D space using the aforementioned model and parameters. The visualization results show that the command parsing trajectories are located on the attractor manifold.