Supplementary Information

S1. Algorithmic Description of Implementation

Algorithm 1 Combining binding in CA1 with association in CA3

Input: Forward synaptic weights, $W_{feed,CA3}$ from input neurons to CA3 neurons, $W_{feed,CA1}$ from CA3 neurons to CA1 neurons, initialized as $W^{(0)}_{feed,CA3} = W^{(0)}_{feed,CA1} = 0$. Lateral synaptic weights, W_h recurrent from CA3 to CA3, initialized as $W^{(0)}_h = 0$. Backward synaptic weights, W_{back}, from memory neurons back to input neurons, initialized as $W^{(0)}_{back} = 0$. Plateau potential probability f_q. Connection probability f_w. Number of sentences N. Training data consists of words sets for each sample $\{A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i\}_{i\leq N}$.

Objective: Training $W_{feed,CA3}$, W_h, $W_{feed,CA1}$, W_{back} during the binding process.

```
1:  ## Binding ##
2:  for each batch sample in {A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i}_{i≤N} do
3:      ## BTSP learning from input to CA3 ##
4:      Update W_{feed,CA3} based on current inputs on the input layer through BTSP learning.
5:      Get the composed representation on CA3 through W_{feed,CA3}.
6:      ## BTSP learning from CA3 to CA1 ##
7:      Update W_{feed,CA1} based on the composed representation on CA3 through BTSP learning.
8:      Get the composed representation on CA1 through W_{feed,CA1}.
9:      ## Hebb learning from CA1 to input ##
10:     Update W_{back} based on the composed representation on CA1 and inputs.
11:     ## one-shot BTSP-like plasticity rule for lateral connections in CA3 ##
12:     Update W_h based on the activation of pre- and postsynaptic neurons on CA3.
13: end for
14: ## Bottom-up unbinding ##
15: for each single cue {B_i}_{i≤N} do
16:     Initialize S(B,2)_0 = 0.
17:     Denote the current masked input as I(B,2).
18:     for each iteration t = 1 to 200 do
19:         Get the state on CA3 via S(B,2)_t = kWTA*(W_h × S(B,2)_{t−1} + W_{feed,CA3} × I(B,2))
20:           (the notation kWTA*(·) represents a restriction where, before k-WTA,
21:           neurons that continue to spike for 10 consecutive time slots are masked for next 50 slots).
22:         Get the state on CA1 via R(B,2)_t = thr(W_{feed,CA1} × S(B,2)_t)
23:           (the notation thr(·) represents where activations exceed the threshold output 1, o/w 0.)
24:         Get the recovered words via thr(W_{back} × R(B,2)_t)
25:     end for
26: end for
```

Algorithm 1: Combining binding in CA1 with association in CA3. We introduce CA3 as an intermediate layer to temporally decouple the superposition, allowing the network to sequentially oscillate among all valid patterns. Dense lateral connections and global k-Winners-Take-All (k-WTA) operations within CA3 enable attractors to form before propagating activation to the memory layer. During the binding phase, activity can pass through the CA3 layer without engaging the internal dynamics produced by its recurrent lateral connections. We apply one-shot synaptic plasticity to the weights W_h only once (Line 6), targeting the representation generated at the first step. We select the composed representations of CA3 (Line 4) and CA1 (Line 9) during the binding process as the states for subsequent comparisons between $S(B,2)_t$ and $R(B,2)_t$. The representation generated on Line 4, referred to in the caption of Fig. 4E, is recognized as the first state of the recurrent network module when a full sentence is presented on the input layer. The representation generated on Line 9 corresponds to the first state of the memory neurons when a full sentence is presented on the input layer. To ensure stability in the recurrent process, we employ k-Winners-Take-All (k-WTA), selecting the k neurons with the highest activation levels to fire spikes while the rest remain inactive, to control the number of spikes in CA3 (Line 17), where each activation consistently engages 60 neurons (k=60 for WTA).

Algorithm 2 Hierarchical iterated binding with eight words

Input: Forward synaptic weights, W_{feed}, from input neurons to memory neurons, initialized as $W_{feed}^{(0)} = 0$. Backward synaptic weights, W_{back}, from memory neurons back to input neurons, initialized as $W_{back}^{(0)} = 0$. Plateau potential probability f_q. Connection probability f_w. Number of sentences N. Training data consists of words sets for each sample $\{A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i\}_{i \leq N}$.

Objective: Training W_{feed}, W_{back} during the binding process.

Create a queue Q for temporarily storing intermediate inputs
Assign $Q = \{A, B, C, D, E, F, G, H\}$
Binding
for each level $l = 1$ to 3 **do**
    for each pair **do**
        Get two inputs from the queue Q.
        Update W_{feed} based on current inputs through BTSP learning.
          e.g. current inputs C and D.
        Get anticipated composed representations based on current W_{feed}.
          e.g. from inputs C and D to CR $< C, D >$.
        Update W_{back} based on the current composed representation and inputs.
          e.g. W_{back} for CR $< C, D >$ back to inputs C and D.
        Add the new composed representation to the queue Q.
    end for
end for
Assign the queue with the final composed representation, $Q = \{< A, B, C, D, E, F, G, H >\}$
Top-down unbinding
for each level $l = 3$ to 1 **do**
    for each step **do**
        Get the current composed representation from the queue Q that needs to be decoded.
        Use the current CR and W_{back} to recover the two decoupled inputs.
          e.g. composed representations $< C, D >'$ and recovered inputs C' and D'.
        Add the new recovered vectors to the queue Q.
    end for
end for

Algorithm 2: Hierarchical iterated binding in the eight-words case. Global weights W_{feed} and W_{back} are utilized throughout the entire binding process. For learning W_{feed} based on BTSP, the training protocol initiates at the first level with binding $\{A, B\}$ into $< A, B >$, followed by binding $\{C, D\}$ into $< C, D >$ at the second step, and so forth. At the second level, the first step involves binding $< A, B >$ and $< C, D >$ into $<< A, B >, < C, D >>$. This process continues in the same manner until the final level, resulting in $<<< A, B >, < C, D >>, << E, F >, < G, H >>>$. The learning of W_{back} follows sequentially through the binding steps, mirroring the W_{feed} training. The training concludes at the final level, after which the network's weights are finalized. At this point, the top-down unbinding process employs the network's W_{back} to calculate the reversal from the final composed representation back to all intermediate composed representations and the input words.

S2. Control experiments on sparsity

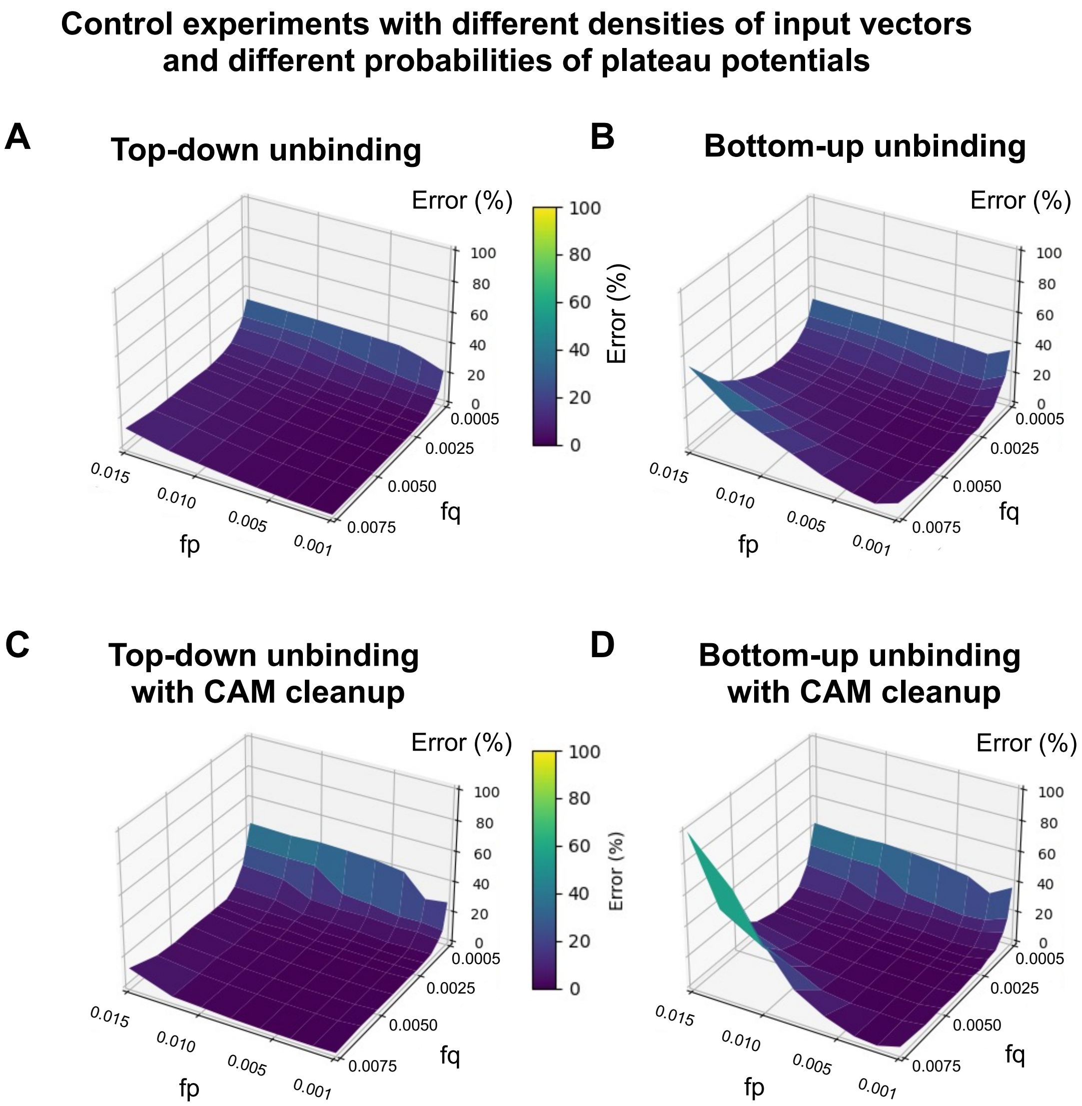

Fig. S1: Control experiments on sparsity.