A Logical Approach to Interpret Neural Network: Formalizing Hopfield Network Dynamics

Xiaoxuan Fu[1], **Ke Wu**[1*]

[1] Institute of Logic, China University of Political Science and Law
xfuuva@gmail.com, wukecupl@163.com

Abstract

This paper introduces a formal framework for interpreting neural network behavior, focusing on Hopfield networks, using propositional dynamic logic (PDL). By connecting Leitgeb's discursive interpretation of neural networks with a formal system, we provide a clear method for analyzing the dynamic evolution of network states. Utilizing PDL, we describe the process of convergence to stable memory patterns and the stability of neural network states. This work offers a rigorous logical foundation for understanding neural network dynamics, bridging the gap between symbolic interpretation and formal analysis. The proposed framework not only enhances the interpretability of Hopfield networks but also lays the groundwork for extending these methods to other neural network architectures, contributing to the broader field of explainable AI.

Introduction

Artificial neural networks (ANNs) have achieved state-of-the-art performance inion, language, and scientific computing by learning high-dimensional function approximations through large collections of weighted connections (Mcculloch and Pitts 1943; Hopfield 1982; Ackley, Hinton, and Sejnowski 1985; Hinton et al. 1995). Classic energy-based and probabilistic models—Hopfield networks, Boltzmann machines, and restricted variants—linked learning, representation, and convergence, and influenced modern deep architectures (Hopfield 1982; Ackley, Hinton, and Sejnowski 1985; Hinton et al. 1995). Dynamical-systems studies further clarified stability and periodicity for threshold-style updates (Goles and Olivos 1980; Zeng 1996; Goles and Ruz 2015). Even so, ANNs remain sub-symbolic: knowledge is stored in continuous parameters rather than explicit logical forms. This makes transparency and principled explanation difficult. Existing methods often rely on heuristics and rarely provide *model-exact*, *logically checkable* guarantees about relevance, minimality, or stability (Titelbaum 2020; van Benthem, Liu, and Smets 2021; Hornischer and Leitgeb 2025).

Problem statement. We currently lack (i) a logical semantics in which neural state updates admit preservation results (e.g., reachability, stabilization) and (ii) explanation objects for network outputs whose *relevance* and, when possible, *minimality* are theorems rather than observations. This gap limits the verification, accountability, and careful integration of learning with reasoning.

A logical route via social network dynamics. Modal and dynamic logics of social networks offer a precise language for diffusion, influence, and link change on graphs (Liu, Seligman, and Girard 2014; Baccini, Christoff, and Verbrugge 2022; Baccini and Christoff 2023; Baccini, Christoff, and Verbrugge 2024), with stabilization phenomena and tight complexity bounds for convergence (Chistikov et al. 2019). These frameworks extend epistemic logic with relation-sensitive modalities (e.g., "all friends believe φ", "there exists a path along which φ spreads") and dynamic operators for public announcements and relation updates. Importantly, threshold-based diffusion yields attractors—fixed points and short cycles—that mirror well-known behaviors in neural threshold systems (Goles and Olivos 1980; Goles and Ruz 2015).

This affinity is structural and dynamic. *Statically*, both settings are graphs with node states and edge weights: attitudes/influence strengths correspond naturally to neuron activations/synaptic weights. *Dynamically*, both evolve by local neighborhood rules. In social models, *diffusion* updates beliefs and *relation revision* adjusts links; in neural models, *inference* updates activations and *training* adjusts parameters. Classical examples align closely: majority-threshold belief updates vs. Hopfiled thresholds for inference; weight adaptation in restricted Boltzmann machines vs. link weight djustment for reaching balanced configurations. Standard dynamic-logic operators (including probabilistic variants and synchronous/asynchronous/block schedules) support these update schemes within one formal language.

This paper. We build on this match to give neural computation a clear logical reading. First, we establish a mapping from a class of neural update rules to a dynamic logic of diffusion and link change, and we prove that key behavioral properties—*convergence* and *stabilization*—are preserved under this mapping. Second, within the same logic, we define explanation objects for network behavior and establish *relevance* guarantees. Third, we transfer stability and

*Ke Wu is the corresponding author.

convergence results from diffusion logics to guarantees for neural attractors.

Why it matters. The framework replaces heuristic attributions with explanation objects that carry formal guarantees and turns stability claims into statements that can be proved and checked in a logic with known complexity. It provides a common language where learning and logic meet: linking propagation, aggregation, and reconfiguration of neural networks to transparent, verifiable properties, thus addressing the core concerns of interpretability and reliability in settings where safety and regulation matter (van Benthem 2015; van Benthem, Liu, and Smets 2021; Hornischer and Leitgeb 2025).

Propositional Dynamic Logic for Hopfield Network Memory

Overview. The Hopfield network marks a decisive step in neural computation by introducing an *energy function* that ties network dynamics to energy minimization in the spirit of statistical physics. This yields a clear computational account of *associative memory*: stored patterns are local minima of the energy landscape, and asynchronous updates implement descent to an attractor (Hopfield 1982). The same perspective informed later probabilistic generative models—most notably Boltzmann machines—which inherit the idea of sampling or optimizing over an energy surface (Ackley, Hinton, and Sejnowski 1985).

Operational picture. Let $S(t) \in \{+1,-1\}^n$ denote the network state at time t, and let $W = (w_{ij})_{1\le i,j\le n}$ be a symmetric weight matrix with no self-loops, i.e. $w_{ij} = w_{ji}$ and $w_{ii} = 0$.

1 **Hebbian Learning.** Given binary patterns $\{\xi^\mu\}_{\mu=1}^m$, we store them by the normalized Hebbian rule

$$w_{ij} = \frac{1}{n}\sum_{\mu=1}^{m} \xi_i^\mu \xi_j^\mu, \qquad i \neq j,$$

which fixes the energy landscape in advance.

2 **Associative memory.** From a (possibly corrupted) cue $S(0)$, the network updates one neuron at a time:

$$h_i(t) = \sum_{j\neq i} w_{ij} S_j(t), \qquad S_i(t{+}1) = \operatorname{sgn}\big(h_i(t)\big),$$

and $S_j(t{+}1) = S_j(t)$ for $j \neq i$. We use the convention

$$\operatorname{sgn}(x) = \begin{cases} +1 & x \ge 0, \\ -1 & x < 0, \end{cases}$$

and note that bias terms can be folded into W and are omitted here.

A PDL semantics for Hopfield dynamics. We now give a propositional dynamic logic (PDL) account of the recall phase. Each neuron i is represented by an atomic proposition p_i, with p_i true iff $S_i = +1$, and $\neg p_i$ standing for $S_i = -1$. Thus each network state $s \in \{+1,-1\}^n$ is a possible world, and there are 2^n worlds in total.

For every i we introduce a basic action upd_i that updates neuron i according to the Hopfield rule, and we define

$$\mathsf{upd} \;:=\; \bigcup_{i=1}^{n} \mathsf{upd}_i$$

to model nondeterministic asynchronous updates. We also use an auxiliary atomic formula

$$\theta_i \;\equiv\; \sum_{k=1}^{n} w_{ik} S_k \ge 0$$

to express that the local field of neuron i is nonnegative at a given world.

Definition 1 (Language)**.** *The formulas and actions are defined by*

$$\varphi ::= p_i \mid \theta_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\alpha]\varphi$$
$$\alpha ::= \mathsf{upd}_i \mid \varphi? \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^*$$

where $i \in \mathbb{N}$.

Symbol explanation and derived notions.

- **Atomic propositions (state).** $p_1, \dots, p_n$ denote neuron states: p_i true means neuron i is in state $+1$, otherwise -1.
- **Atomic propositions (local field).** For each i, θ_i abbreviates the threshold condition induced by the weight matrix, i.e. $\sum_{k=1}^n w_{ik} p_k \ge 0$. (Equivalently, in a state s, θ_i holds iff $\sum_{k=1}^n w_{ik} s_k \ge 0$.)
- **Basic action.** upd_i means "update neuron i once according to the Hopfield rule".
- **Test action.** $\varphi?$ is the standard PDL test, succeeding iff φ holds.
- **Choice / sequencing / iteration.** $\alpha\cup\beta$, $\alpha;\beta$, and α^* have their usual PDL meanings.
- **Random asynchronous update.** $\mathsf{upd} = \bigcup_{i=1}^n \mathsf{upd}_i$ means "pick any neuron and update it".
- **Stable configuration.**

$$\mathsf{STABLE} \;\equiv\; \bigwedge_{i=1}^{n} \big([\,\mathsf{upd}_i\,]p_i \leftrightarrow p_i\big),$$

i.e. every single–neuron update leaves the current value of that neuron unchanged.

- **Convergence.**

$$\langle \mathsf{upd}^* \rangle \mathsf{STABLE}$$

says "there exists a finite sequence of asynchronous updates that reaches a stable configuration.

Definition 2 (Model)**.** *A Hopfield–PDL model is a tuple* $\mathcal{M} = (S, V, W, R)$*, where:*

- $S = \{+1,-1\}^n$ *is the set of states;*
- $V : S \times \{p_1, \dots, p_n\} \to \{$*true*,*false*$\}$ *is a valuation with* $V(s, p_i) =$ *true iff* $s_i = +1$*;*
- $W = (w_{ij})$ *is an* $n \times n$ *symmetric weight matrix with* $w_{ij} = w_{ji}$ *and* $w_{ii} = 0$*;*
- R *assigns to every action* α *a transition relation* $R_\alpha \subseteq S \times S$*.*

Definition 3 (Satisfaction)**.** *For $\mathcal{M} = (S, V, W, R)$ and $s \in S$,*

$$\begin{aligned}
\mathcal{M}, s \models p_i &\iff V(s, p_i) = \textit{true},\\
\mathcal{M}, s \models \theta_i &\iff \textstyle\sum_{k=1}^{n} w_{ik} s_k \geq 0,\\
\mathcal{M}, s \models \neg\varphi &\iff \mathcal{M}, s \not\models \varphi,\\
\mathcal{M}, s \models \varphi \wedge \psi &\iff \mathcal{M}, s \models \varphi \textit{ and } \mathcal{M}, s \models \psi,\\
\mathcal{M}, s \models [\alpha]\varphi &\iff \forall s' \, (s R_\alpha s' \Rightarrow \mathcal{M}, s' \models \varphi).
\end{aligned}$$

We define $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$ and $\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$.

Definition 4 (Transition relations)**.** *The transition relations for the Hopfield–PDL model are defined as follows:*

$$\begin{aligned}
s R_{\mathsf{upd}_i} s' &\iff \forall j \neq i : s'_j = s_j \;\&\; s'_i = \mathit{sgn}\left(\textstyle\sum_{k=1}^{n} w_{ik} s_k\right)\\
s R_{\varphi?} s' &\iff s = s' \wedge \mathcal{M}, s \models \varphi\\
R_{\alpha\cup\beta} &= R_\alpha \cup R_\beta\\
R_{\alpha;\beta} &= R_\alpha \circ R_\beta\\
R_{\alpha^*} &= (R_\alpha)^*
\end{aligned}$$

Definition 5 (Validity)**.** *A formula φ is valid iff for every Hopfield–PDL model $\mathcal{M}$ and every state $s \in S$ we have $\mathcal{M}, s \models \varphi$.*

Based on the logical system described above, we derive the following series of valid formulas, each corresponding to a key property of the Hopfield network. These formulas, which characterize dynamic convergence and basic update properties, strongly suggest that the system formalized by this logic is indeed the *associative memory* of the Hopfield network.

Theorem 1 (Modal Equivalence of Single-Neuron Update)**.** *For any neuron i and any formula φ, the following equivalence holds:*

$$\langle \mathsf{upd}_i \rangle \varphi \;\leftrightarrow\; [\,\mathsf{upd}_i\,]\varphi.$$

Proof sketch. According to Definition 4, for a fixed neuron i and state s, there can be at most one state s' such that $s \, R_{\mathsf{upd}_i} \, s'$. This is because the state of all neurons, except for i, remains unchanged, and the new state of neuron i is uniquely determined by the sign of its local field. Hence, the operations $\langle \mathsf{upd}_i \rangle$ and $[\,\mathsf{upd}_i\,]$ are identical, establishing the equivalence.

Theorem 2 (Idempotence of Consecutive Updates)**.** *For any neuron i and any formula φ, the following holds:*

$$[\mathsf{upd}_i; \mathsf{upd}_i]\varphi \;\leftrightarrow\; [\mathsf{upd}_i]\varphi.$$

Proof. By the semantics of sequential composition, $R_{\mathsf{upd}_i;\mathsf{upd}_i} = R_{\mathsf{upd}_i} \circ R_{\mathsf{upd}_i}$. Let s be the initial state. If $s \, R_{\mathsf{upd}_i} \, s'$, then for $j \neq i$, we have $s'_j = s_j$, and the local field computed at s' is:

$$\sum_k w_{ik} s'_k = \sum_{k \neq i} w_{ik} s_k.$$

In this case, the second update will produce:

$$s''_i = \operatorname{sgn}\left(\sum_{k \neq i} w_{ik} s_k\right) = s'_i,$$

and $s''_j = s'_j$ for $j \neq i$, resulting in $s'' = s'$. Thus, we conclude that $R_{\mathsf{upd}_i;\mathsf{upd}_i} = R_{\mathsf{upd}_i}$, and the two modal formulas are equivalent. Therefore, the equivalence

$$[\mathsf{upd}_i; \mathsf{upd}_i]\varphi \;\leftrightarrow\; [\mathsf{upd}_i]\varphi$$

holds, establishing the validity of the statement. □

Interpretation. Theorems 1 and 2 show that consecutive updates to neuron i are equivalent to a single update. In the Hopfield network, i's state depends on its local field, which is influenced by the other neurons' states. Since these states are unchanged during the update, the first update stabilizes i's state, making further updates redundant. This ensures the efficiency of the asynchronous update process and supports network convergence.

The update operation of a single neuron can be seen as an accessibility relation in the possible world model, where a neuron either stays in its current state or transitions to a new state. Random asynchronous updates, therefore, can be interpreted as merging these individual update relations.

For example, consider the Hopfield network in Figure 1, consisting of three neurons a, b, and c, with synaptic weights $w_{ab} = 0.6$, $w_{ac} = 0.5$, and $w_{bc} = 0.4$. For the initial state $(a : -1, b : +1, c : +1)$, the possible world accessibility relations corresponding to each update are as follows:

- Update of a transitions to $(a : +1, b : +1, c : +1)$;
- Update of b transitions to $(a : -1, b : -1, c : +1)$;
- Update of c transitions to $(a : -1, b : +1, c : -1)$.

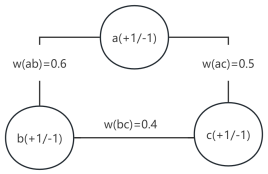

Figure 1: A Hopfield network with neurons a, b, and c

Combining these relations, we obtain the set of possible worlds reachable from the initial state $(a : -1, b : +1, c : +1)$ under random asynchronous updates: $(a : +1, b : +1, c : +1)$, $(a : -1, b : +1, c : -1)$, or the persistence of the initial state. This highlights the relationship between random asynchronous updates and deterministic single-neuron updates.

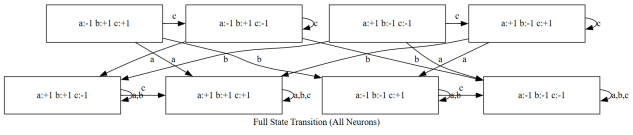

Figure 2: The possible world accessibility relation for random asynchronous updates

Theorem 3 (Local Field Determines Update)**.** *For any neuron i, the following holds:*

$$\theta_i \rightarrow [\mathsf{upd}_i] p_i.$$

Proof. Assume $\mathcal{M}, s \models \theta_i$, i.e., $\sum_{k=1}^{n} w_{ik} s_k \geq 0$. For any s' such that $sR_{\mathsf{upd}_i}s'$, by the transition relation, we have $\forall j \neq i : s'_j = s_j$ and $s'_i = +1$, thus $\mathcal{M}, s' \models p_i$. By modal semantics, $\mathcal{M}, s \models [\mathsf{upd}_i]p_i$, and therefore $\mathcal{M}, s \models \theta_i \rightarrow [\mathsf{upd}_i]p_i$. □

Theorem 4 (Negative Local Field Inhibits Update)**.** *For any neuron i, the following holds:*

$$\neg\theta_i \rightarrow [\mathsf{upd}_i]\neg p_i.$$

Proof. Assume $\mathcal{M}, s \models \neg\theta_i$, i.e., $\sum_{k=1}^{n} w_{ik} s_k < 0$. For any s' such that $sR_{\mathsf{upd}_i}s'$, $s'_i = -1$, and therefore $\mathcal{M}, s' \models \neg p_i$. By modal semantics, we conclude $\mathcal{M}, s \models [\mathsf{upd}_i]\neg p_i$, and hence $\mathcal{M}, s \models \neg\theta_i \rightarrow [\mathsf{upd}_i]\neg p_i$. □

Theorems 3 and 4 show that the update of a neuron is solely determined by its local field θ_i, independent of its activation state. This "local field-driven" behavior is fundamental for the Hopfield network's *associative memory* function.

Theorem 5 (No Effect of Update on Other Neurons)**.** *For any neurons i and $j \neq i$, the following holds:*

$$p_j \leftrightarrow [\mathsf{upd}_i]p_j.$$

Proof sketch. Since upd_i only affects neuron i, and neurons $j \neq i$ remain unaffected, $s'_j = s_j$ for all $j \neq i$. Therefore, $[\mathsf{upd}_i]p_j$ holds if and only if p_j holds, implying $p_j \leftrightarrow [\mathsf{upd}_i]p_j$.

Theorem 5 demonstrates that the update of one neuron does not affect the activation states of others, reflecting the distributed nature of computation in the Hopfield network.

Theorem 6 (Nondeterminism of Random Update)**.** *For any neuron i, the following holds:*

$$\langle\mathsf{upd}\rangle\varphi \leftrightarrow \bigvee_{i=1}^{n} \langle\mathsf{upd}_i\rangle\varphi.$$

Proof. The proof follows from the fact that $\mathsf{upd} = \bigcup_{i=1}^{n} \mathsf{upd}_i$. If $\mathcal{M}, s \models \langle\mathsf{upd}\rangle\varphi$, there exists some i such that $sR_{\mathsf{upd}_i}s'$, which implies $\mathcal{M}, s \models \langle\mathsf{upd}_i\rangle\varphi$. Conversely, if $\mathcal{M}, s \models \bigvee_{i=1}^{n}\langle\mathsf{upd}_i\rangle\varphi$, then for some i, $sR_{\mathsf{upd}_i}s'$, and thus $\mathcal{M}, s \models \langle\mathsf{upd}\rangle\varphi$. □

Theorem 6 formalizes the equivalence between random asynchronous updates and individual neuron state transitions, highlighting the nondeterministic nature of updates in the Hopfield network.

Theorem 7 (Convergence of Random Update)**.** *In the Hopfield–PDL model, for any initial state $s \in S$, there exists a stable state $s' \in S$ such that the network converges to s' within a finite number of steps, i.e.:*

$$\mathcal{M}, s \models \langle\mathsf{upd}^*\rangle\mathsf{STABLE}$$

where $\mathsf{STABLE} \equiv \bigwedge_{i=1}^{n}([\mathsf{upd}_i]p_i \leftrightarrow p_i)$ *indicates the network is in a stable state.*

Proof. The concept for this proof is derived from (Hopfield 1982), but it has been reformulated here to maintain the self-contained nature of the paper.

Consider the Hopfield–PDL model $\mathcal{M} = (S, V, W, R)$ and define the energy function $E : S \rightarrow \mathbb{R}$ as:

$$E(s) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} s_i s_j.$$

Since S is finite ($|S| = 2^n$), E takes only a finite number of values on S.

Now, consider the update action upd_i for any neuron i. According to the system semantics, for state s and updated state s', we have $sR_{\mathsf{upd}_i}s'$ if and only if for all $j \neq i$, $s'_j = s_j$, and $s'_i = \mathrm{sgn}(h_i)$, where $h_i = \sum_{k=1}^{n} w_{ik} s_k$ is the local field. The energy change $\Delta E = E(s') - E(s)$ is calculated as follows:

Since only the state of neuron i changes, and the states of other neurons remain unchanged, the difference between $E(s)$ and $E(s')$ comes only from terms involving s_i. Expand the energy function:

$$\Delta E = -\frac{1}{2}\sum_{k=1}^{n}\sum_{l=1}^{n} w_{kl} s'_k s'_l + \frac{1}{2}\sum_{k=1}^{n}\sum_{l=1}^{n} w_{kl} s_k s_l.$$

Decompose the summation into parts involving index i and parts not involving i. Terms not involving i (i.e., $k \neq i$ and $l \neq i$) are the same in s and s', so they cancel out. Terms involving i include cases where $k = i$ or $l = i$. Due to the symmetry of the weight matrix ($w_{kl} = w_{lk}$) and no self-loop ($w_{ii} = 0$), we can rewrite:

$$\Delta E = -\frac{1}{2}\left(\sum_{l=1}^{n} w_{il} s'_i s'_l + \sum_{k=1}^{n} w_{ki} s'_k s'_i\right) + \frac{1}{2}\left(\sum_{l=1}^{n} w_{il} s_i s_l + \sum_{k=1}^{n} w_{ki} s_k s_i\right).$$

Using symmetry $w_{il} = w_{li}$ and $s'_l = s_l$ (for $l \neq i$), and $s'_k = s_k$ (for $k \neq i$), simplify to:

$$\Delta E = -\frac{1}{2}\sum_{l=1}^{n} w_{il} s'_i s'_l - \frac{1}{2}\sum_{k=1}^{n} w_{ki} s'_k s'_i + \frac{1}{2}\sum_{l=1}^{n} w_{il} s_i s_l + \frac{1}{2}\sum_{k=1}^{n} w_{ki} s_k s_i.$$

Note that both summations cover all indices, and $w_{ki} = w_{ik}$, therefore:

$$\Delta E = -\sum_{l=1}^{n} w_{il} s'_i s_l + \sum_{l=1}^{n} w_{il} s_i s_l = \sum_{l=1}^{n} w_{il}(s_i - s'_i) s_l,$$

where $\sum_{l=1}^{n} w_{il} s_l = h_i$ is the local field, hence:

$$\Delta E = (s_i - s'_i) h_i,$$

where $h_i = \sum_{k=1}^{n} w_{ik} s_k$ is the local field. The energy change depends on whether the state of the neuron i changes:

- If $h_i \geq 0$, then $s_i' = +1$. In this case, if $s_i = +1$, then $s_i - s_i' = 0$, so $\Delta E = 0$; if $s_i = -1$, then $s_i - s_i' = -2$, so $\Delta E = -2h_i \leq 0$.
- If $h_i < 0$, then $s_i' = -1$. In this case, if $s_i = -1$, then $s_i - s_i' = 0$, so $\Delta E = 0$; if $s_i = +1$, then $s_i - s_i' = 2$, so $\Delta E = 2h_i < 0$.

Thus, for any update action upd_i, $\Delta E \leq 0$, and $\Delta E < 0$ if and only if the state of neuron i changes.

The update action $\mathsf{upd} = \bigcup_{i=1}^{n} \mathsf{upd}_i$ represents randomly selecting a single neuron to update, and the iterative update action upd^* represents repeating upd multiple times. Since the state space S is finite and the energy function E is non-increasing, the network must eventually converge to a state where no updates change the state.

This state is defined by $\mathsf{STABLE} = \bigwedge_{i=1}^{n}([\mathsf{upd}_i]p_i \leftrightarrow p_i)$. Once the network reaches a state s' such that $\mathcal{M}, s' \models \mathsf{STABLE}$, no further updates will change the state. Therefore, after a finite number of updates, the network will converge to a stable state s', and $\mathcal{M}, s \models \langle \mathsf{upd}^* \rangle \mathsf{STABLE}$.

Thus, the network converges to a stable state in a finite number of steps. □

Definition 6 (Sequential Update)**.** *The sequential update action is defined as*

$$\mathsf{upd}_{seq} = \mathsf{upd}_1; \mathsf{upd}_2; \ldots; \mathsf{upd}_n,$$

where $n \in \mathbb{N}$.

Theorem 8 (Convergence of Sequential Update)**.** *In the Hopfield–PDL model, for any initial state $s \in S$, the network necessarily converges to a stable state after a finite number of sequential updates, i.e., the following formula holds:*

$$\models [\mathsf{upd}^*_{seq}]\mathsf{STABLE}.$$

Proof sketch. Since each update action upd_i either leaves the energy function $E(s)$ unchanged or strictly decreases it, and the state space S is finite, the energy function will eventually reach a minimum. Therefore, after a finite number of sequential updates, the network will converge to a stable state s', where $\mathcal{M}, s' \models \mathsf{STABLE}$, and no update will change the state. Hence, $\mathcal{M}, s \models [\mathsf{upd}^*_{\mathrm{seq}}]\mathsf{STABLE}$, guaranteeing convergence to a stable state.

Definition 7 (Sequential Update with Permutation)**.** *For any permutation $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, the permuted sequential update action is defined as*

$$\mathsf{upd}_\sigma = \mathsf{upd}_{\sigma(1)}; \mathsf{upd}_{\sigma(2)}; \ldots; \mathsf{upd}_{\sigma(n)},$$

where $n \in \mathbb{N}$.

Theorem 9 (Convergence of Permuted Updates)**.** *In the Hopfield–PDL model, for any initial state $s \in S$, the network converges to a stable state after a finite number of permuted sequential updates:*

$$\models [\mathsf{upd}^*_\sigma]\mathsf{STABLE}.$$

Proof sketch. In the spirit of Theorem 7, the proof follows the same reasoning. Consider the Hopfield–PDL model $\mathcal{M} = (S, V, W, R)$ and the energy function:

$$E(s) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} s_i s_j.$$

Since S is finite, $E(s)$ ranges over a finite set of values.

The permuted sequential update upd_σ applies updates in the order given by permutation σ. This process can be viewed as a sequence of state transitions $s_0, s_1, \ldots, s_n$, where each update either leaves the energy unchanged or strictly decreases it. As S is finite, the energy function must eventually stabilize, and the network converges to a stable state s' after a finite number of updates.

The same reasoning applies to the iterative sequential update upd^*_σ, where the network will converge to a stable state s_K after a finite number of iterations. No further updates will change the state once stability is reached.

Thus, $\mathcal{M}, s_0 \models [\mathsf{upd}^*_\sigma]\mathsf{STABLE}$, and the network converges to a stable state after a finite number of permuted sequential updates.

Remark 1. *As shown in Theorems 8 and 9, the convergence of the sequential update mechanism is* ***independent of the order*** *in which neurons are updated. This property enhances the model's ability to characterize the network's dynamic behavior, aligning with the convergence observed in both random asynchronous updates and fixed-order sequential updates.*

The theoretical convergence of the network requires symmetric weights with no self-connections. However, it is the Hebbian learning algorithm that enables the network's **associative memory** functionality. The algorithm encodes memory patterns as attractors (stable states) in the network's dynamics. During random asynchronous updates, the network can start from any noisy or imperfect initial state and, through the threshold rule of the local field, converge to a stable memory pattern. Thus, the network's associative memory capability is determined by how the initial state (input) evolves into a stable state (output), representing the stored memory pattern.

Definition 8 (Memory Pattern)**.** *A memory pattern $\xi = (\xi_1, \ldots, \xi_n) \in \{+1, -1\}^n$ is a binary vector representing a memory state.*

Definition 9 (Attractor)**.** *A state s is an attractor if and only if $\mathcal{M}, s \models \mathsf{STABLE}$ and there exists a state $s' \neq s$ such that $s' R_{\mathsf{upd}^*} s$.*

Definition 10 (Basin of Attraction)**.** *The basin of attraction $\mathsf{BASIN}(s)$ of state s is defined as the set of all states s' such that $s' R_{\mathsf{upd}^*} s$.*

The associative memory function of the Hopfield network can be viewed as a classifier based on memory patterns (specific possible worlds). Any input that falls into the basin of attraction of a memory pattern is mapped to that pattern as the stable output of the network's evolution. Even if two networks have different parameters, as long as their convergence domains for all memory patterns align, they can be functionally equivalent. This insight emphasizes that the key determinant of network behavior is not the specific parameters, but the invariant set of possible worlds that uniquely characterizes the network's memory function.

The Hebbian learning algorithm encodes memory patterns as attractors (stable states) of the network's dynamical

system. It enables the network to start from any initial state (even noisy or defective) and converge to a stable state representing a complete memory pattern. This process defines the network's associative memory capability.

To reliably perform associative memory, the number of stored patterns must remain within the theoretical capacity limit of approximately $0.138n$, where n is the number of nodes. Exceeding this limit results in spurious attractors or the merging of memory patterns, impairing the network's memory function. The previously defined logical framework can formally describe the network's dynamics beyond its capacity limit.

Proposition 10. *There exists a constant $C \approx 0.138n$ such that for $m > C$ memory patterns, the following holds:*

$$\bigwedge_{k=1}^{m} \xi^k \to \bigvee_{k=1}^{m} \neg \mathsf{STABLE}(\xi^k)$$

Memory Merging: When the number of stored patterns m exceeds $C \approx 0.138n$, multiple memory patterns probabilistically merge into a single unstable pattern. If $m > C$, at least one pattern ξ^k will be unstable, i.e., $\mathcal{M}, \xi^k \models \neg\mathsf{STABLE}$. Thus, for any state s, if $\mathcal{M}, s \models \bigwedge_{k=1}^{m} \xi^k$ (i.e., s represents all m patterns simultaneously), then $\mathcal{M}, s \models \bigvee_{k=1}^{m} \neg\mathsf{STABLE}(\xi^k)$.

Proposition 11. *When the number of stored patterns approaches the capacity limit, we have:*

$$\langle \mathsf{upd}^* \rangle \left(\mathsf{STABLE} \wedge \bigwedge_{k=1}^{m} \neg \xi^k \right)$$

Spurious Attractors: When the number of stored patterns approaches the capacity limit of $0.138n$, spurious attractors (stable states that are not memory patterns) appear. There exists a state s such that $\mathcal{M}, s \models \mathsf{STABLE}$ but s is not any of the stored memory patterns ξ^k, i.e., $\mathcal{M}, s \models \bigwedge_{k=1}^{m} \neg \xi^k$. Since the network converges from any initial state to a stable state, there exists some initial state s_0 such that $\mathcal{M}, s_0 \models \langle \mathsf{upd}^* \rangle \left(\mathsf{STABLE} \wedge \bigwedge_{k=1}^{m} \neg \xi^k \right)$.

The above propositions capture non-ideal behaviors exhibited by the Hopfield network when the number of memory patterns exceeds the capacity limit. A well-performing Hopfield network must avoid these behaviors, which leads to the design goal: every memory pattern must correspond to a unique attractor. The following definition formalizes this idea.

Definition 11 (Well-formed Hopfield Network)**.** *A Hopfield network is well-formed if, for every memory pattern $\xi = (\xi_1, \ldots, \xi_n) \in \{+1, -1\}^n$, it holds that $\mathcal{M}, \xi \models \mathsf{STABLE}$.*

In a well-formed Hopfield network, if the input state is sufficiently similar to a memory pattern ξ, the network will eventually converge to ξ. This is formalized in the following proposition:

Proposition 12 (Convergence from Partial Pattern to Complete Memory)**.** *For any memory pattern ξ and any state s satisfying $s_i = \xi_i$ for sufficiently many i, we have:*

$$\bigwedge_{i \in I} (p_i \leftrightarrow \xi_i) \to [\mathsf{upd}^*] \xi$$

where $I \subseteq \{1, \ldots, n\}$ and $|I|$ is sufficiently large.

While well-formed memory patterns ensure the network's convergence, the state space's geometric structure may still be complex. Specifically, on the boundaries (or saddle points) between multiple basins of attraction, there may exist some network states that are simultaneously within the basins of attraction of multiple memory patterns, leading to the following uncertainty phenomenon:

Proposition 13 (Saddle Point Network States May Update to Multiple Memory Patterns)**.** *There exists $s \in S$ such that $\mathcal{M}, s \models \langle \mathsf{upd}^* \rangle \xi_i \wedge \langle \mathsf{upd}^* \rangle \xi_j$.*

To address this, an ideal Hopfield network must ensure that the basins of attraction are mutually exclusive:

Definition 12 (Ideal Hopfield Network)**.** *A well-formed Hopfield network is ideal if the basins of attraction of any two distinct attractors ξ_i and ξ_j are disjoint, i.e., $\neg(\mathsf{BASIN}(\xi_i) \cap \mathsf{BASIN}(\xi_j))$.*

Theorem 14 (Convergence of Updates in an Ideal Hopfield Network)**.** *In the ideal Hopfield–PDL model $\mathcal{M} = (S, V, W, R)$, for any initial state $s \in S$ and any permutation $\sigma : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, the random update action upd^* and the permuted sequential update action upd^*_σ both converge to the same attractor $\xi \in S$. That is, the following formula holds:*

$$\langle \mathsf{upd}^* \rangle \xi \leftrightarrow \langle \mathsf{upd}^*_\sigma \rangle \xi$$

where ξ is an attractor satisfying $\mathcal{M}, \xi \models \mathsf{STABLE}$, and there exists $s' \neq \xi$ such that $s' R_{\mathsf{upd}^*} \xi$.

Proof. We start by considering the ideal Hopfield–PDL model $\mathcal{M}$, where each memory pattern ξ is an attractor, and the basins of attraction of distinct memory patterns are mutually exclusive. This ensures that each state $s \in S$ belongs to exactly one basin $\mathsf{BASIN}(\xi)$.

For any initial state s_0, there exists a unique attractor ξ such that $s_0 \in \mathsf{BASIN}(\xi)$. According to Definition 8, $\mathsf{BASIN}(\xi)$ includes all states that converge to ξ under random updates. Thus, by Theorem 7, we know that random updates upd^* starting from s_0 will eventually converge to ξ, i.e.:

$$\mathcal{M}, s_0 \models \langle \mathsf{upd}^* \rangle \xi.$$

Similarly, by Theorem 9, sequential updates upd^*_σ starting from s_0 also lead to a stable state, which, due to the uniqueness of the basin, must be the same attractor ξ:

$$\mathcal{M}, s_0 \models \langle \mathsf{upd}^*_\sigma \rangle \xi.$$

Since the basins of attraction are mutually exclusive, no other attractor can be reached from the initial state s_0. Therefore, we conclude that both random and sequential updates converge to the same attractor ξ, and we can write:

$$\mathcal{M}, s_0 \models \langle \mathsf{upd}^* \rangle \xi \leftrightarrow \langle \mathsf{upd}^*_\sigma \rangle \xi.$$

□

Thus, in an ideal Hopfield network, both random and permuted sequential updates converge to the same attractor.

Formalizing Leitgeb's Neural Network Interpretation with PDL

In *Explaining Neural Networks with Reasons*, Leitgeb and Hornischer introduced a framework for interpreting neural networks using the semantics of possible worlds. Central to this framework is the conceptualization of input-output pairs (x, y) processed by a neural network as "possible worlds" $w \in W$, with classification labels corresponding to "propositions" $A \subseteq W$. Additionally, the framework constructs "reason vectors" r_w, which record the activation values of neurons across all possible worlds, quantifying the degree to which neurons provide logical support for propositions.

While this interpretation framework provides an innovative lens for understanding neural networks, it is fundamentally discursive rather than a strict logical explanation. Specifically, it lacks a formal treatment of the accessibility relation between possible worlds, a key component for logically analyzing state transitions in neural networks. In contrast, *the Hopfield–PDL model* established in this paper offers a precise and rigorous foundation for Leitgeb's interpretation theory. This work defines the exact correspondence between the two frameworks, enhancing the logical clarity and rigor of the neural network interpretation.

We highlight the following key correspondences:

1. **Correspondence of Possible Worlds:** In Leitgeb's interpretation, the set of possible worlds W represents the entire set of network states that can be encountered. In our the Hopfield–PDL model, this corresponds to the state space $S = \{+1, -1\}^n$. Each state $s \in S$ represents a complete configuration (input) of the network, which will ultimately converge to at least one memory pattern (output proposition, label). Thus, by interpreting the input-output pairs as possible worlds, we align the concept of possible worlds in Leitgeb's theory with the network states in our model, creating a direct correspondence between W and S.
2. **Correspondence of Propositions:** Leitgeb's theory treats propositions A_d (such as "the input is digit d") as sets of possible worlds, where each proposition corresponds to a subset of possible worlds where the proposition is true. This corresponds directly to the concept of basins of attraction $\mathsf{BASIN}(\xi_d)$. Specifically, for any memory pattern ξ_d (i.e., the output proposition), the corresponding proposition in Leitgeb's theory can be strictly defined as:

   $$A_d = \{s \in S \mid sR_{\mathsf{upd}_i{}^*}\xi_d\}$$

   where $R_{\mathsf{upd}_i{}^*}$ represents the relation induced by the update process in the Hopfield network. This correspondence effectively links the semantic interpretation of propositions in Leitgeb's theory to the dynamic, converging behavior of the network's states.
3. **Construction of Reason Vectors:** The reason vectors r_w introduced in Leitgeb's framework can be directly computed in our model. These vectors represent the degree to which each neuron contributes to the logical support for a given output. For any neuron i, the component of the reason vector for each memory pattern w_s is defined as:

   $$r_i(w_s) = (s_i)_{s \in S}$$

   where (s_i) denotes the activation of neuron i in network state s. This construction is a direct translation of Leitgeb's reason vectors into a formal framework that quantifies the influence of each neuron's state on the network's output.

In conclusion, Leitgeb's discursive interpretation of neural networks is fully formalized within *the Hopfield–PDL model*. This formalization provides a solid foundation for interpreting neural network behavior, grounding Leitgeb's constructs in a logically rigorous system. By employing the modal operators from PDL, our system formally describes the dynamic evolution of network states. For instance, the convergence of states to a stable configuration is captured as $M, s \models \langle \mathsf{upd}^* \rangle \mathsf{STABLE}$, a property that Leitgeb's framework does not address. This dynamic characterization bridges the gap between the discursive semantics of Leitgeb's theory and the formalism required for neural network interpretability.

Conclusion

This paper has introduced a formal framework for interpreting the behavior of neural networks, focusing on Hopfield networks, through propositional dynamic logic (PDL). By connecting Leitgeb's discursive interpretation with the formal dynamics of Hopfield networks, we have established a method to rigorously analyze the evolution of network states, particularly in terms of *convergence* and *memory retrieval*.

The primary contribution of this work lies in the application of PDL to capture the dynamic properties of neural networks. We have shown how the modal operators allow for a precise formal description of the *convergence process*, addressing a gap in Leitgeb's framework. This formal approach provides a clear way to model how networks *stabilize*, enabling more reliable and interpretable analyses of neural network behavior.

The framework developed in this paper, referred to as the **Hopfield–PDL Model**, sets the foundation for understanding the dynamic evolution of network states and the stability of memory patterns in Hopfield networks. Looking ahead, this formalization opens the door to extending these methods to more complex neural architectures, such as **deep learning models**. Future research could explore how this framework can be applied to networks trained on a broader range of tasks and **real-world datasets**. Additionally, further investigation into integrating other logical tools could enhance our ability to describe more intricate interactions within networks, particularly in multi-layer systems.

In summary, this work contributes to the field of neural network interpretability by offering a structured, formal approach to understanding neural dynamics. The **Hopfield–PDL Model** not only provides insights into the behavior of Hopfield networks but also sets the stage for future advancements in making neural networks more transparent, accountable, and analytically accessible.

References

Ackley, D. H.; Hinton, G. E.; and Sejnowski, T. J. 1985. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1): 147–169.

Baccini, E.; and Christoff, Z. 2023. Comparing Social Network Dynamic Operators. In *Theoretical Aspects of Rationality and Knowledge*.

Baccini, E.; Christoff, Z.; and Verbrugge, R. 2022. Opinion Diffusion in Similarity-Driven Networks. 14th Conference on Logic and the Foundations of Game and Decision Theory, LOFT 2022.

Baccini, E.; Christoff, Z.; and Verbrugge, R. 2024. Dynamic Logics of Diffusion and Link Changes on Social Networks: Dynamic Logics of Diffusion and Link Changes on Social Networks. *Studia Logica*, 113(5): 1245–1315.

Chistikov, D.; Lisowski, G.; Paterson, M.; and Turrini, P. 2019. Convergence of Opinion Diffusion is PSPACE-complete. In *AAAI Conference on Artificial Intelligence*.

Goles, E.; and Olivos, J. 1980. Periodic behaviour of generalized threshold functions. *Discrete Mathematics*, 30(2): 187–189.

Goles, E.; and Ruz, G. A. 2015. Dynamics of neural networks over undirected graphs. *Neural Networks*, 63: 156–169.

Hinton, G. E.; Dayan, P.; Frey, B. J.; and Neal, R. M. 1995. The "Wake-Sleep" Algorithm for Unsupervised Neural Networks. *Science*, 268(5214): 1158–1161.

Hopfield, J. J. 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8): 2554–2558.

Hornischer, L.; and Leitgeb, H. 2025. Explaining Neural Networks with Reasons. *ArXiv*, abs/2505.14424.

Liu, F.; Seligman, J.; and Girard, P. 2014. Logical dynamics of belief change in the community. *Synthese*, 191(11): 2403–2431.

Mcculloch, W. S.; and Pitts, W. 1943. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4): 115–133.

Titelbaum, M. G. 2020. The Stability of Belief: How Rational Belief Coheres with Probability, by Hannes Leitgeb. *Mind*, 130(519): 1006–1017.

van Benthem, J. 2015. Oscillations, Logic, and Dynamical Systems. *ILLC Prepublication Series, 2015-10*.

van Benthem, J.; Liu, F.; and Smets, S. 2021. Logico-Computational Aspects of Rationality. In *The Handbook of Rationality*. The MIT Press.

Zeng, P. 1996. Artificial neural networks principle for finite element method. *Zeitschrift Fur Angewandte Mathematik Und Mechanik*, 76(S5): p.565–566.

Appendix: Extending the Dynamic-Logical Framework to General Neural Networks

In the main part of the paper we showed that the dynamic logic used for diffusion and link-change in social network models can be transplanted to Hopfield networks once the asynchronous update process is made explicit. Concretely, we represented a Hopfield network by a Kripke-style structure

$$\mathcal{M} = (S, R, V)$$

where $S = \{+1, -1\}^n$ is the configuration space, $R_{\mathsf{upd}_i} \subseteq S \times S$ is the accessibility relation generated by updating neuron i, and network convergence is captured by the PDL formula

$$\mathcal{M}, s \models \langle \mathsf{upd}^* \rangle \mathsf{STABLE}.$$

The key to that construction was: (i) make the *state* of the network the carrier of possible worlds; (ii) make each atomic update an *action* in PDL; (iii) use iteration $(\cdot)^*$ to express convergence. This appendix shows that the same recipe can be lifted from Hopfield-style inference to more general neural-network computation, provided we refine the state space and distinguish between the two phases common to almost all neural models: *inference* and *training*.

State Spaces for Inference and Training

The guiding idea, consistent with the Kripke-style semantics used above, is: *a network state is a possible world*. What changes from model to model is what counts as a "state".

Inference state. For a fixed-parameter network (the ordinary test/deployment situation), a state is just the layerwise activation under a fixed parameter setting. Formally, let S^{inf} be the set:

$$\{\, a \mid a \text{ is the collection of activations of all layers under } (W, b) \,\}.$$

In the Hopfield case we specialized this to $S^{\text{inf}} = \{+1, -1\}^n$. For a convolutional network, a state may have internal structure, e.g.

$$a = (a^{(1)}, \ldots, a^{(L)}), \quad a^{(\ell)} \in \mathbb{R}^{h_\ell \times w_\ell \times c_\ell},$$

and the atomic propositions may be refined to

$$p^{(\ell)}_{x,y,c} \quad \text{"activation at position } (x, y), \text{ channel } c \text{ of layer } \ell \text{ satisfies the test."}$$

Training state. During learning, the configuration must also contain parameters. We therefore extend the carrier to

$$S^{\text{train}} = \big\{\, (a, W, b) \;\big|\; a \in S^{\text{inf}},\ W \text{ is the current weight collection}, b \text{ is the current bias collection.} \big\}$$

Intuitively, (a, W, b) says: "with current parameters (W, b), the network, on the current batch, realizes activations a." The entire training process is then a transition system on S^{train}.

Actions for Inference and Training

Once states are fixed, extending the action vocabulary of the logic is straightforward. As in the main text, we write $[\alpha]\varphi$ for "after every execution of program α, φ holds" and $\langle \alpha \rangle \varphi$ for "there is an execution of α after which φ holds."

Inference actions. Inference is the "parameters-fixed" phase. Its characteristic actions leave W, b unchanged.

- **Forward propagation** fwd. For a feed-forward network with layers $1, \ldots, L$, we may describe

  $$\mathsf{fwd} \;\equiv\; \mathsf{upd}_1 \,;\, \mathsf{upd}_2 \,;\, \ldots \,;\, \mathsf{upd}_L,$$

  where each upd_ℓ rewrites the activation of layer ℓ using the fixed parameters of that layer. A typical property then reads

  $$[\,\mathsf{fwd}\,]\,\varphi_{\text{out}}$$

  meaning: after one forward pass, the output constraint φ_{out} necessarily holds.
- **Iterative inference** (Hopfield-type). In the Hopfield model in the body of the paper we had

  $$\mathsf{upd} \;=\; \bigcup_{i=1}^{n} \mathsf{upd}_i \quad \text{and} \quad \mathcal{M}, s \models \langle \mathsf{upd}^* \rangle \mathsf{STABLE}.$$

  This is just a special case of an inference action where the program is an iteration of atomic local updates.

Training actions. Training is the "parameters-changing" phase. Its actions must be allowed to move in W, b.

- **Parameter-update action** grad. Abstractly:

  $$\mathsf{grad} : (a, W, b) \longmapsto (a', W', b')$$

  where W', b' are obtained from W, b by one step of gradient-based optimization (SGD, Adam, etc.), and a' is the activation under the new parameters for the current batch. We do not need to commit to a concrete update rule at the logical level; it suffices that grad be a binary relation on S^{train}.
- **Performance-test action** $\varphi_{\text{ok}}?$. As in standard PDL, this action does not change the state; it only checks whether a property of the current parameters holds, e.g.

  $$\varphi_{\text{ok}} \;\equiv\; \text{"validation accuracy} \geq \theta\text{"}.$$

A Training Program in PDL

With these actions, the usual "train until good enough" loop can be written in the same PDL language we used for Hopfield convergence:

$$\alpha_{\text{train}} \;=\; \big(\neg\varphi_{\text{ok}}? \,;\, \mathsf{grad}\big)^* \,;\, \varphi_{\text{ok}}?.$$

Its intended reading is:

> repeat: update parameters; as long as the model is *not* satisfactory, continue; finally stop in a state satisfying φ_{ok}.

A basic soundness obligation for this program in a concrete setting would be the formula

$$\langle \alpha_{\text{train}} \rangle \varphi_{\text{ok}},$$

which is the exact training analogue of the Hopfield-style convergence statement $\langle \mathsf{upd}^* \rangle \mathsf{STABLE}$ proved in the main text.

Discussion

Two points are worth emphasizing.

1. **Uniformity of the possible-worlds view.** In the Hopfield section we treated every network configuration $s \in \{+1, -1\}^n$ as a world. Here we only generalize that idea: inference worlds contain activations; training worlds contain activations *and* parameters. No change to the underlying logic is required.
2. **Uniformity of the action vocabulary.** The same PDL constructors—sequential composition ";", iteration "$(\cdot)^*$", and test "?"—suffice for both phases. What changes is only the interpretation of the atomic actions, not the logic itself.

Thus, the dynamic-logic account of Hopfield networks is not an isolated case but an instance of a more general pattern: once a neural model can be seen as a transition system over an explicit state space, it can be embedded into the same PDL-style framework used in the main body of the paper. This makes it possible, in principle, to reason about inference-time properties (correctness, robustness, local stability) and training-time properties (termination, satisfaction of performance constraints) in one unified logical language.