Neural Network Semantics

Thesis Proposal

Caleb Schultz Kisby
Indiana University
November 21, 2024

1 Introduction

In the last 15 years, modern artificial intelligence (AI) systems have shown unprecedented success at learning from data with little human guidance. Consider for example large language models such as Llama and GPT [1; 15; 44], which have taken the world by storm with their ability to learn to converse in English merely from unstructured text data they scrape off the web. Or consider AlphaGo [37], which learned to play Go at a human expert level by repeatedly playing against itself. These breakthroughs in machine learning are in large part thanks to the widespread use of neural networks – brain-inspired computational models that are flexible and excel at learning from unstructured data.

But the danger of neural networks is that they come with no safety, reliability, or correctness guarantees. If you play with systems like GPT long enough, you eventually realize that they carry all sorts of misconceptions, make silly logical mistakes, and are quite happy to spew out disinformation [39]. Neural networks also lack transparency, which means diagnosing and correcting these errors is not feasible. (Imagine trying to determine which neurons and connections are responsible for believing that a sailfish is a mammal!) In practice, a computational learner is often a 'black-box' whose correct inferences, mistakes, and biases lack interpretation and explanation.

How can we better understand and control this seemingly black-box behavior of neural networks? The answer lies in symbolic (logic) systems, which were commonly used to model reasoning and intelligent behavior prior to the rapid growth of neural network systems. In contrast with neural networks, symbolic systems provide explicit rules for their reasoning in a human-interpretable language. However, this purely logic-based approach was largely abandoned due to logic's inability at the time to model flexible learning or update (known as the *frame problem* in AI [29; 36]).

There is still hope that we might be able to integrate neural networks and symbolic systems while retaining the advantages of both. The field of *neuro-symbolic AI* has emerged in response to this possibility [3; 11; 35]. As a result of this effort, there are now many different proposals for neuro-symbolic systems, including Logic Tensor Networks [4], Distributed Alignment Search [17], DeepProbLog [27], Logic Explained Networks [14], and neural network fibring [16]. But these systems form a scattered picture; some unifying perspective or theory is needed. In the preface to a recent neuro-symbolic survey book [11], Frank van Harmelen writes:

> What are the possible interactions between knowledge and learning? Can reasoning be used as a symbolic prior for learning … Can symbolic constraints be enforced on data-driven systems to make them safer? Or less biased? Or can, vice versa, learning be used to yield symbolic knowledge? … **neuro-symbolic systems cur-**

rently lack a theory that even begins to ask these questions, let alone answer them.

In my thesis, I will offer a new unifying perspective that sheds light on these questions. The basis for many neuro-symbolic systems is that they encode logical information into neural networks, or conversely, encode neural networks as models in logic [32]. Given these translations, certain neural networks and logic models are able to represent the same information. This suggests that we can think of neural networks in the same way as a logician would think about a model.

The plan is to take this idea as far as it will go: I will develop logics with these *neural network semantics*, whose formulas are interpreted in terms of binary neural networks. First, I will consider *static* conditional and modal logics whose operators are given by the closure (or forward propagation) of signals in the neural network. This closure operator allows these logics to express neural network *inference*, i.e. the input-output behavior of the net. Next, I will give a *dynamic* logic (inspired by Dynamic Epistemic Logic [41; 42; 43]) with an operator for Hebbian learning [21], a simple neural network update policy. Along the way, I will show how foundational questions about neural networks become natural and answerable questions in logic. I'll focus on three questions that are natural to ask about for any logical system: *soundness*, *completeness*, and *expressivity*.

Soundness. What axioms are sound for the semantics? In neural network semantics, this question becomes: What properties can we formally verify for neural network inference? In the dynamic logic setting: What properties can we formally verify for neural network *learning policies*?

Completeness. What are the complete axioms for the semantics? This is equivalent to the question of whether we can build a model that obeys a set of logical constraints Γ. In neural network semantics, completness asks whether we can build a *neural network* that obeys Γ. And for dynamic logic, this asks whether we can build a neural network that obeys Γ before and after learning takes place. This is a key to the AI Alignment problem, which requires building neural networks with these kinds of guarantees.

Expressivity. What formulas can the semantics model or define? How does the expressive power of two different semantics compare? For neural networks, expressivity is a proxy for what kinds of functions neural networks are capable of representing. Additionally, it provides a metric for comparing the power of neural networks against traditional models in logic. In the dynamic setting: What kinds of *learning policies* are neural networks able to support?

In summary, I will defend the following thesis statement:

Thesis Statement. Neural networks can be treated as a class of models in formal logic (*neural network semantics*). In the process of developing these semantics, foundational questions about neural network inference and learning become natural and answerable questions in logic. In particular:

Soundness	answers	"How can we formally verify that a class of neural networks and its learning policies obey certain properties?"
Completeness	answers	"How can we build a neural network that aligns with constraints, even as the net learns and changes over time?"
Expressivity	answers	"What kinds of functions and learning policies are neural networks capable of representing?"

2 Related Work

My thesis work builds on existing logics that use neural network semantics, and shares similarities with logics for modeling social networks. Additionally, my approach to modeling learning takes inspiration from work on learning in Dynamic Epistemic Logic (DEL). Here I'll take a moment to situate my thesis in this broader context and clarify what my contribution is in each case.

Logics with Neural Network Semantics. The core idea behind neural network semantics—that neural networks can be treated as models for logic—actually dates back to the very first paper on neural networks. In McCulloch and Pitts [28], logical formulas are mapped directly to individual neurons in the net. This approach suffers from the well-known "grandmother cell" problem [20]: it is cognitively implausible that an individual neuron could represent a complex concept such as "grandmother". Instead, concepts in brain networks are distributed across multiple neurons at once.

Neural network semantics is based on a recent reimagining of this approach [5; 26], where logical formulas are mapped to sets of neurons rather than to individual neurons. Early work established the correspondence between inference in a neural network and nonmonotonic conditionals [5; 12; 24; 25]. More recently, [18; 19] proved soundness for inference over fuzzy neural networks. In my thesis work so far [22; 23], I applied ideas from Dynamic Epistemic Logic to model a simple update policy, Hebbian learning, in neural network semantics. The key results of this work are the first ever soundness and completeness theorems for any learning policy on neural networks.

Logics with Social Network Semantics. It has recently come to my attention that a similar approach is being used to model group behavior in social networks. In these social network logics [2; 6; 13], nodes in the graph represent individual agents, and each formula is mapped to the set of agents that adopt a certain social attitude. Agents influence each other, and the spread of their attitudes is modeled much in the same way as forward propagation of a signal in a neural network.

This work shares essentially the same premise and techniques as neural network semantics; I personally view this as a case of parallel discovery. But the two approaches still differ in interesting ways. First, in some sense the two semantics are operating on different "levels": social networks model interactions between multiple agents, whereas neural networks model interactions between components of the same (single) agent. Second, the two differ in subject matter. Social network semantics focuses on different social links between agents, and how these links change [2]. Neural network semantics, my own work included, instead focuses on inferences and updates inspired by artificial and natural neural networks.

Other Dynamic Logics for Learning. My approach to modeling learning in neural networks takes inspiration from Dynamic Epistemic Logic (DEL) [41; 43]. Perhaps the closest logics to mine are the logics for plausibility upgrade, in particular conditionalization (Cond), lexicographic (Lex), and conservative (Consr) upgrade [40; 42]. In the Expressivity portion of my dissertation, I will explore whether Hebbian update $\mathsf{Hebb}^{\star}$ can be simulated by plausibility upgrade, and vice-versa whether Cond, Lex, Consr, and vice-versa whether these plausibility upgrades can be simula.

In my thesis, I mainly focus on the effects of single-step updates. But recent literature on learning in DEL goes beyond this by considering iterated update and convergence to the truth ("learning in the limit") [7; 8; 9; 10]. The key questions here are: How can we compare the learning power

of different iterated update policies? How can we axiomatize important properties of learning? These questions are answered in terms of updates on more classical graphs and plausibility structures. Although in this thesis I don't consider iterated update, I do lay down the groundwork to import neural network learning into this setting.

3 Technical Overview

I will now give an overview of the particular neural network semantics I've developed during my PhD. First, I will discuss the simplifying assumptions that make it possible to use neural networks as models, and introduce the *closure* (or forward propagation) of a signal in the net. This closure operator allows us to express the inference, or input-output behavior, of the net. I will give a modal logic whose key operator is given by this closure operator. I will then turn to dynamic update in neural networks and introduce iterated Hebbian learning, one of the simplest learning policies over nets. Finally, I will give a dynamic logic whose formulas express the behavior of a neural network before and after Hebbian update.

3.1 Neural Network Semantics

A model of neural network semantics is an artificial neural network (ANN), along with a valuation function which interprets propositions as sets of neurons. I will make a few more simplifying assumptions soon, but this is the basic idea.

Definition 1. A neural network model is $\mathcal{N} = \langle N, \mathsf{bias}, E, W, A, \eta, V \rangle$, where

- N is a finite nonempty set (the set of *neurons*)
- bias is a fixed node (the *bias* node)
- Each $E \subseteq N \times N$ (the *edge relation*)
- $W : E \to \mathbb{Q}$ (the *edge weights*)
- $A : \mathbb{Q} \to \mathbb{Q}$ (the *activation function*)
- $\eta \in \mathbb{Q}$, $\eta \geq 0$ (the *learning rate*)
- $V : \text{Proposition} \to \mathcal{P}(N)$ (the *valuation function*)

In general, a *state* is just a possible activation pattern or configuration of the net. In practice, a neural network's nodes take on fuzzy activation values in $[0,1]$. But we would like to associate each state with a binary set of neurons—either a neuron is active (1) or it is not (0). To do this, I assume that the activation function A is a (nonzero) binary step function ($A : \mathbb{Q} \to \{0,1\}$). It turns out this binary constraint is also a common theoretical assumption in work that analyzes neural networks as automata [30; 31; 45]. In their terminology, we say our nets are *saturated*.

Additionally, I assume there is a special bias node that is active in every state. This is purely for ruling out the particular edge case where *no* node is active. Since bias is active in every state, we can assume that no edges go into bias. Putting all this together, the states of the net are defined as follows.

$$\text{State} = \{S \mid S \subseteq N \text{ and } \mathsf{bias} \in S\}$$

We can describe the input-output behavior of neural networks in terms of their state. Say we are given an input state A consisting of input-layer nodes, and a possible classification state B consisting of output-layer nodes. Active neurons in A successively activate new neurons until eventually the state of the net stabilizes. If this final state includes the output B, we say "the net *classifies* A as B".

The state at the fixed point of this process is called the *closure* or *forward propagation* of the signal A, $\mathsf{Cl}(A)$. This closure operator is central to my semantics, since it captures the underlying dynamics involved in neural network inference. Formally, each choice of E, W, A specifies a transition function from state $S \in \mathsf{State}$ to the next state. Given an initial state S_0, this transition function F_{S_0} is given by

$$F_{S_0}(S) = S_0 \cup \left\{ n \mid A\left(\sum_{m \in \mathrm{preds}(n)} W(m,n) \cdot \chi_S(m) \right) = 1 \right\}$$

where $\chi_S(m) = 1$ iff $m \in S$ is the indicator function. In other words, $F_{S_0}(S)$ is the initial state S_0, along with the set of nodes that are activated by their predecessors in S. Notice that $F_{S_0}(S)$ is extensive: all nodes in the initial state will stay activated in successive states.

Neural network models have one final constraint, which guarantees that the closure $\mathsf{Cl}(S)$ exists.

Postulate 2. I assume that for all states S_0, F applied repeatedly to S_0, i.e.

$$S_0, F_{S_0}(S_0), F_{S_0}(F_{S_0}(S_0)), \ldots, F^k_{S_0}(S_0), \ldots$$

has a (finite) least fixed point, and moreover that it is *unique*. Let $\mathsf{Cl}\colon \mathsf{State} \to \mathsf{State}$ be the function that produces that least fixed point. Let **Net** be the class of all binary neural network models that satisfy this postulate.

An important feature of Cl is that it is nonmonotonic: it is not the case that for all $A, B \in \mathsf{State}$, if $A \subseteq B$ then $\mathsf{Cl}(A) \subseteq \mathsf{Cl}(B)$. This is because our net's weights can be negative, and so $\mathsf{Cl}(B)$ can inhibit the activation of new neurons that would otherwise be activated by $\mathsf{Cl}(A)$.

For technical reasons (i.e. completeness of Hebbian update), I will also need "helper" operators Reach and $\mathsf{Reach}^{\downarrow}$ for expressing graph reachability. The Reach operator is just ordinary graph reachability: $\mathsf{Reach}(S)$ returns the set of all nodes reachable from S. Similarly, $\mathsf{Reach}^{\downarrow}(S)$ returns the set of all nodes that graph-reach some node in S. Think of these operators as sort of monotonic "versions" of Cl.

I can now state the specific logic and neural network semantics that I will consider. Let $p, q, \ldots$ be finitely many atomic propositions. These represent fixed states, corresponding to features in the external world that we know ahead of time. Usually these are input and output states, although they could be intermediate "hidden" states if we know these features ahead of time. For example, p might be the set of neurons that represent the color pink. For more complex formulas,

Definition 3. Formulas in our base language $\mathcal{L}$ are given by

$$\varphi, \psi := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \mathbf{K} \rangle \varphi \mid \langle \mathbf{K}^{\downarrow} \rangle \varphi \mid \langle \mathbf{T} \rangle \varphi$$

$\top, \bot, \vee, \rightarrow, \leftrightarrow$ and the dual modal operators $\mathbf{K}, \mathbf{K}^{\downarrow}, \mathbf{T}$ are defined in the usual way.

The intended readings for these operators are as follows ($\mathbf{K}^{\downarrow}$ is conceptually tricky, I will leave it out of this discussion for now). $\mathbf{K}\varphi$ reads "the agent knows φ", and $\mathbf{T}\varphi$ reads "typically φ". It is not immediately clear how these readings are justified; in my dissertation, I will justify these

readings by connecting the neural network semantics I give here to more traditional semantics for $\mathbf{K}, \mathbf{K}^{\downarrow}$, and $\mathbf{T}$.

At last, here are the semantics. For all $\mathcal{N} \in \mathbf{Net}$, $n \in N$:

$$\begin{array}{lll} \mathcal{N}, n \Vdash p & \text{iff} & n \in V(p) \\ \mathcal{N}, n \Vdash \neg\varphi & \text{iff} & \mathcal{N}, n \nVdash \varphi \\ \mathcal{N}, n \Vdash \varphi \wedge \psi & \text{iff} & \mathcal{N}, n \Vdash \varphi \text{ and } \mathcal{N}, n \Vdash \psi \\ \mathcal{N}, n \Vdash \langle \mathbf{K} \rangle \varphi & \text{iff} & n \in \mathsf{Reach}^{\downarrow}([\![\varphi]\!]) \\ \mathcal{N}, n \Vdash \langle \mathbf{K}^{\downarrow} \rangle \varphi & \text{iff} & n \in \mathsf{Reach}([\![\varphi]\!]) \\ \mathcal{N}, n \Vdash \langle \mathbf{T} \rangle \varphi & \text{iff} & n \in \mathsf{Cl}([\![\varphi]\!]) \end{array}$$

where $[\![\varphi]\!] = \{n | \mathcal{N}, n \Vdash \varphi\}$.

Although these semantics are based on Leitgeb's [26], they differ in a few key ways. First, his semantics uses conditionals $\varphi \Rightarrow \psi$ to capture neural network inference, whereas mine instead centers on the modal operator $\langle \mathbf{T} \rangle$. Second, I include these additional operators $\mathbf{K}$ and $\mathbf{K}^{\downarrow}$ that are not mentioned in his work. Finally, Leitgeb battles with the issue of how to correctly interpret negation; I sidestep this issue by using neural networks for interpreting $\langle \mathbf{T} \rangle \varphi$ (where the "action" happens), but not for $\neg$ and $\wedge$. The bottom line is this: proving completeness for this logic is not necessarily just a matter of importing the proof from [26].

3.2 Dynamic Update in Neural Network Semantics

The neural network semantics presented so far shows us how we can use neural networks as models for modal logic. Neural network inference can be expressed in this logic using $\langle \mathbf{T} \rangle \varphi$, which denotes the forward propagation of the signal $[\![\varphi]\!]$ through the net. However, as discussed in the introduction, the mystery about neural networks is how their inference interacts with their *learning*. In this section, I will show how to extend these semantics to model learning and update in a neural net.

As previously mentioned, I formalize neural network update using the methodology of Dynamic Epistemic Logic. Our static operators $\langle \mathbf{K} \rangle$, $\langle \mathbf{K}^{\downarrow} \rangle$, and $\langle \mathbf{T} \rangle$ are interpreted by examining the state of the neural net. The DEL trick is to introduce a new "dynamic" operator $[P]$ which *changes* the net in response to some observed formula P. First, we extend our language $\mathcal{L}$ to $\mathcal{L}^{\star}$, which includes these dynamic operators:

$$\varphi, \psi := p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \mathbf{K} \rangle \varphi \mid \langle \mathbf{K}^{\downarrow} \rangle \varphi \mid \langle \mathbf{T} \rangle \varphi \mid [P] \varphi$$

Here, $[P]\varphi$ reads "after the agent observes P, φ is true".

Let $\mathsf{Update}\colon \mathbf{Net} \times \mathsf{State} \to \mathbf{Net}$ be any function which takes a neural network, some state S, and "updates" the net somehow in response to S. We can interpret $[P]$ as performing this update by adding the following line to the semantics:

$$\mathcal{N}, n \Vdash [P] \varphi \quad \text{iff} \quad \mathsf{Update}(\mathcal{N}, [\![P]\!]), n \Vdash \varphi$$

In other words, in order to evaluate $[P]\varphi$, we simply evaluate φ in the updated net $\mathsf{Update}(\mathcal{N}, [\![P]\!])$.

From a DEL perspective, this is a standard move to make. But from a machine learning perspective, there are a couple caveats that I should mention. First, $[P]$ does not model learning in the sense of "iterated update until convergence", but rather only models a single step of update. Second, we should think of $[P]$ as modeling *unsupervised learning*—the model updates in response

to an input P, but no "expected answer" y is given alongside P. It is an open problem to formalize supervised learning (in this machine learning sense) in DEL in a non-trivial way.

So far, I've discussed learning and update in very general terms. For my thesis, I will model a simple update policy over neural networks: Hebbian learning. The point in starting with Hebbian learning is to get the details right on a simpler example before lifting these ideas to, say, gradient descent through backpropagation [34].

Hebb's classic learning rule [21] states that when two adjacent neurons are simultaneously and persistently active, the connection between them strengthens ("neurons that fire together wire together"). In contrast with backpropagation, Hebbian learning is errorless and unsupervised. Another key feature is that Hebbian update is local — the change in a weight $\Delta W(m,n)$ depends only on the activation of the immediately adjacent neurons. For this reason, the Hebbian family of learning policies is often considered more biologically plausible than backpropagation.

There are many variations of Hebbian learning, but I will only consider the most basic form of Hebb's rule: $\Delta W(m,n) = \eta x_m x_n$, where η is the learning rate and x_m, x_n are the outputs of adjacent neurons m and n. This is the *unstable* variation of Hebb's rule; repeatedly applying the rule will make the weights arbitrarily large. I will not consider stabilizing variants such as Oja's rule [33].

Single-Step Hebbian Update. First, consider what happens in a single step of Hebbian update. Given a net $\mathcal{N}$ and a state S, we first propagate S forward through $\mathcal{N}$. Any edges that are involved in this propagated activation pattern $\mathsf{Cl}(S)$ simply have their weights strengthened. Formally,

Definition 4. Let $\mathsf{Hebb}\colon \mathbf{Net} \times \mathsf{State} \to \mathbf{Net}$ be given by

$$\mathsf{Hebb}(\langle N, \mathsf{bias}, E, W, A, \eta, V\rangle,\ S) = \langle N, \mathsf{bias}, E, W', A, \eta, V\rangle$$

where

$$W'(m,n) = W(m,n) + \eta \cdot \chi_{\mathsf{Cl}(S)}(m) \cdot \chi_{\mathsf{Cl}(S)}(n)$$

Note that Hebb does not affect the edges, activation function, or evaluation of propositions. This means the resulting net is still binary, and closures $\mathsf{Cl}(S)$ still exist and are unique. Therefore Hebb is well-defined. This also means that Hebb does not affect the Reach or $\mathsf{Reach}^{\downarrow}$ operators. However, changing the weights *does* affect closures $\mathsf{Cl}(S)$—this interaction between Hebb and Cl (learning and inference) is a central issue in my thesis.

Iterated Hebbian Update. In addition to the single-step Hebb operator, in my thesis work I have also modelled *iterated* Hebbian update $\mathsf{Hebb}^\star$. The idea is this: what happens when we propagate a signal S through the net, and then *repeatedly* strengthen the weights of the edges that are involved? Recall that our single-step Hebb is unstable; if we repeat Hebb on a single input state S, the net's weights within $\mathsf{Cl}(S)$ will be so high that *any* activation pattern that makes contact with $\mathsf{Cl}(S)$ will "rip through" it entirely. Repeating Hebb on S further will not change the $\mathsf{Cl}(S)$-structure, i.e., the update has reached a fixed point. $\mathsf{Hebb}^\star$ returns the net at this fixed point.

Instead of reasoning abstractly about this fixed point, I formalize it by explicitly defining the number of iterations iter needed to reach it. The idea is to set iter to be so high, all updated weights $W'(m,n)$ overpower any negative weights that would otherwise cancel their effect. The following definitions might look like black magic, but they are set up to capture this intuition (I verified in Lean that this is the right choice for iter, see [23]).

Definition 5. Let $\mathcal{N}$ be a net, $n \in N$, and let $m_1, \ldots, m_k$ list the predecessors of n. The *negative weight score* of n is the sum of all the negative weights of n's predecessors, i.e.,

$$\mathsf{nws}(n) = \sum_{i=1}^{\deg(n)} \begin{cases} W(m_i, n) & \text{if } W(m_i, n) < 0 \\ 0 & \text{otherwise} \end{cases}$$

Definition 6. The *minimum negative weight score* is simply

$$\mathsf{mnws} = \min_{n \in N} \mathsf{nws}(n)$$

Definition 7. Recall that the activation function A is nonzero, i.e. there is some $t \in \mathbb{Q}$ such that $A(t) = 1$. We set the number of iterations iter to be exactly

$$\mathsf{iter} = \begin{cases} \left\lceil \frac{t - |N| \cdot \mathsf{mnws}}{\eta} \right\rceil & \text{if } \geq 1 \\ 1 & \text{otherwise} \end{cases}$$

Note that iter will always be a positive integer, and so iterating iter times is well-defined.

Definition 8. Let $\mathsf{Hebb}^\star : \mathbf{Net} \times \mathsf{State} \to \mathbf{Net}$ be given by

$$\mathsf{Hebb}^\star(\langle N, \mathsf{bias}, E, W, A, \eta, V \rangle,\ S) = \langle N, \mathsf{bias}, E, W', A, \eta, V \rangle$$

where

$$W'(m, n) = W(m, n) + \mathsf{iter} \cdot \eta \cdot \chi_{\mathsf{Cl}(S)}(m) \cdot \chi_{\mathsf{Cl}(S)}(n)$$

As with Hebb, $\mathsf{Hebb}^\star$ does not affect the edges, activation function, or evaluation of propositions, and the update is well-defined. One worry we might have is that, in each iteration, we always update by $\mathsf{Cl}(S)$ in the *original* net. But it turns out that this $\mathsf{Cl}(S)$ doesn't change with each iteration, so $\mathsf{Hebb}^\star$ is equivalent to repeatedly applying Hebb until we reach a fixed point [23].

4 Progress and Goals

The static and dynamic neural network semantics above demonstrate how neural networks can be treated as a class of models in formal logic. My central thesis is that the soundness, completeness, and expressivity of these semantics answer foundational questions about neural network inference and learning. Moreover, these questions are *answerable* (which I will show by answering them). In the remaining chapters of my dissertation, I will defend this claim.

	Static Inference – Reach and Cl	**Dynamic Update – Hebb***
Soundness	✓	✓
Completeness	in progress	✓
Expressivity	in progress	in progress

My first goal is to prove soundness and completeness results for the logics of Cl and Hebb* (for single-step Hebb I will only show soundness). I will then prove various expressivity results, comparing neural network models **Net** to relational (graph) models, plausibility models, and neighborhood models—what properties and learning policies can neural networks define or satisfy? Finally, for each chapter (Soundness, Completeness, Expressivity), I will work through an example that demonstrates what that chapter teaches us about neural networks (verification, model-building, and representational capacity). The chart above maps out my progress so far.

In my FLAIRS paper [22], I showed axioms that are sound for single-step Hebbian update Hebb. I also wrote a Python program which does model checking using these sound semantics. Then in my AAAI paper [23], I gave sound *and complete* axioms for iterated Hebbian update Hebb*. For this second paper, I formalized many of the technical results in the Lean proof assistant, which revealed a couple of errors in the first FLAIRS paper (these are resolved by the results in the more recent paper).

I have the rest left to do before I submit my dissertation:

Completeness, Static Inference. The AAAI paper did not actually prove completeness for the *static* system. I now believe I have a proof which modifies Leitgeb's construction in [24] to account for $\langle \mathbf{K} \rangle$ and $\langle \mathbf{K}^{\downarrow} \rangle$ constraints ($\langle \mathbf{K}^{\downarrow} \rangle$ is actually easier, and then we can get $\langle \mathbf{K} \rangle$ using tricks from temporal logic that I typed up a couple years back).

In addition to the completeness proof, I would also like to extend my Python program to actually perform model building for an arbitrary set of modal constraints Γ.

Expressivity, Static Inference. In terms of model simulations, I have shown that neural networks can simulate relational (graph) models and certain "social-majority" networks. I also proved that neural networks can be simulated *by* neighborhood models (but not vice-versa). And assuming completeness goes through, I will also be able to simulate plausibility models using neural networks. The other direction (plausibility models from neural networks) is still an open question, but I'd like to resolve it before I submit my dissertation.

I have also been exploring the expressivity of neural networks in terms of descriptive complexity. As part of this chapter of my dissertation, I would like to make some effort to connect my own work with the descriptive complexity of neural networks (in languages such as Linear Temporal Logic) [38].

Expressivity, Dynamic Update. Finally, there's the question of which updates (recall: Cond, Lex, Consr) can be simulated by a neural network, and conversely which models can simulate a neural network update such as Hebb*. I have been working on this for a year, and will probably continue to work on it until January. It turns out that there are many reasonable ways to say one model simulates the update of another, and I need some time to fix my approach and check the results.

5 Timeline

I applied for a grant through the DFF earlier this year, and the application put a time restriction on when I need to defend my dissertation (April 8$^{\text{th}}$). I am trying to follow this in good faith. Here is a timeline based on that constraint. This is what you can expect me to be doing for the next few months. This also gives you an idea for when I'll be emailing drafts, or arranging to meet with you.

November 21st	Submit this proposal. Start working on proposal defense. Start dissertation outlining and pre-writing.
December 4th – 6th	Defend proposal.
December 7th	Write up Technical Background & Soundness chapters. Continue working on expressivity results.
January 1st	Write up Completeness chapter & start Expressivity chapter. Finish up expressivity results.
February 1st	Write up Expressivity, Intro, & Conclusion chapters. Submit expressivity chapter as a paper to WoLLIC 2025.
March 8th	Submit dissertation!
April 8th	Defend dissertation!
May 8th	Submit corrections (anticipated).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat et al. GPT-4 technical report. *ArXiv preprint arXiv:2303.08774*, 2023.

[2] Edoardo Baccini, Zoé Christoff, and Rineke Verbrugge. Dynamic logics of diffusion and link changes on social networks. *Studia Logica*, pages 1–71, 2024.

[3] Sebastian Bader and Pascal Hitzler. Dimensions of neural-symbolic integration – A structured survey. In *We Will Show Them! Essays in Honour of Dov Gabbay, Volume 1*, pages 167–194. College Publications, 2005.

[4] Samy Badreddine, Artur d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic Tensor Networks. *Artificial Intelligence*, 303:103649, 2022.

[5] Christian Balkenius and Peter Gärdenfors. Nonmonotonic inferences in neural networks. In *KR*, pages 32–39. Morgan Kaufmann, 1991.

[6] Alexandru Baltag, Zoé Christoff, Rasmus K Rendsvig, and Sonja Smets. Dynamic epistemic logics of diffusion and prediction in social networks. *Studia Logica*, 107:489–531, 2019.

[7] Alexandru Baltag, Nina Gierasimczuk, Aybüke Özgün, Ana Lucia Vargas Sandoval, and Sonja Smets. A dynamic logic for learning theory. *Journal of Logical and Algebraic Methods in Programming*, 109:100485, 2019.

[8] Alexandru Baltag, Nina Gierasimczuk, and Sonja Smets. Truth-tracking by belief revision. *Studia Logica*, 107:917–947, 2019.

[9] Alexandru Baltag, Dazhu Li, and Mina Young Pedersen. On the right path: a modal logic for supervised learning. In *International Workshop on Logic, Rationality and Interaction*, pages 1–14. Springer, 2019.

[10] Alexandru Baltag and Sonja Smets. Group belief dynamics under iterated revision: Fixed points and cycles of joint upgrades. In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 41–50. 2009.

[11] Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Priscila Machado Vieira Lima, Leo de Penning et al. Neural-symbolic learning and reasoning: A survey and interpretation. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*, pages 1–51. IOS press, 2021.

[12] Reinhard Blutner. Nonmonotonic inferences and neural networks. *Synthese*, 142:143–174, 2004.

[13] Zoé Christoff and Jens Ulrik Hansen. A logic for diffusion in social networks. *Journal of Applied Logic*, 13(1):48–77, 2015.

[14] Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Lió, Marco Maggini, and Stefano Melacci. Logic Explained Networks. *Artificial Intelligence*, 314:103822, 2023.

[15] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan et al. The Llama 3 herd of models. *ArXiv preprint arXiv:2407.21783*, 2024.

[16] Artur SD'Avila Garcez, Luis C Lamb, and Dov M Gabbay. *Neural-Symbolic Cognitive Reasoning*. Springer Science & Business Media, 2008.

[17] Atticus Geiger, Zhengxuan Wu, Christopher Potts, Thomas Icard, and Noah Goodman. Finding alignments between interpretable causal variables and distributed neural representations. In *Causal Learning and Reasoning*, pages 160–187. PMLR, 2024.

[18] Laura Giordano, Valentina Gliozzi, and Daniele Theseider Dupré. A conditional, a fuzzy and a probabilistic interpretation of self-organizing maps. *Journal of Logic and Computation*, 32(2):178–205, 2022.

[19] Laura Giordano and Daniele Theseider Dupré. Weighted defeasible knowledge bases and a multipreference semantics for a deep neural network model. In *Logics in Artificial Intelligence: 17th European Conference, JELIA 2021, Virtual Event, May 17–20, 2021, Proceedings 17*, pages 225–242. Springer, 2021.

[20] Charles G Gross. Genealogy of the "grandmother cell". *The Neuroscientist*, 8(5):512–518, 2002.

[21] Donald Hebb. *The Organization of Behavior*. Psychology Press, apr 1949.

[22] Caleb Kisby, Saúl Blanco, and Lawrence Moss. The logic of Hebbian learning. In *The International FLAIRS Conference Proceedings*, volume 35. 2022.

[23] Caleb Schultz Kisby, Saúl A Blanco, and Lawrence S Moss. What do Hebbian learners learn? Reduction axioms for iterated Hebbian learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14894–14901. 2024.

[24] Hannes Leitgeb. Nonmonotonic reasoning by inhibition nets. *Artificial Intelligence*, 128(1-2):161–201, 2001.

[25] Hannes Leitgeb. Nonmonotonic reasoning by inhibition nets II. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11(supp02):105–135, 2003.

[26] Hannes Leitgeb. Neural network models of conditionals. In *Introduction to Formal Philosophy*, pages 147–176. Springer, 2018.

[27] Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298:103504, 2021.

[28] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943.

[29] Drew McDermott. A critique of pure reason. *Computational intelligence*, 3(3):151–160, 1987.

[30] William Merrill. Sequential neural networks as automata. *ArXiv preprint arXiv:1906.01615*, 2019.

[31] William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A Smith, and Eran Yahav. A formal hierarchy of RNN architectures. *ArXiv preprint arXiv:2004.08500*, 2020.

[32] Simon Odense and Artur S. d'Avila Garcez. A semantic framework for neural-symbolic computing. *ArXiv*, abs/2212.12050, 2022.

[33] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15:267–273, 1982.

[34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. *Biometrika*, 71(599-607):6, 1986.

[35] Md Kamruzzaman Sarker, Lu Zhou, Aaron Eberhart, and Pascal Hitzler. Neuro-Symbolic Artificial Intelligence: Current Trends. *AI Communications*, 34, 2022 2022.

[36] Murray Shanahan. The frame problem. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2016.

[37] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[38] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Transactions of the Association for Computational Linguistics*, 12:543–561, 2024.

[39] Alex Tamkin, Miles Brundage, Jack Clark, and Deep Ganguli. Understanding the capabilities, limitations, and societal impact of large language models. *ArXiv preprint arXiv:2102.02503*, 2021.

[40] Johan Van Benthem. Dynamic logic for belief revision. *Journal of applied non-classical logics*, 17(2):129–155, 2007.

[41] Johan Van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.

[42] Johan Van Benthem and Sonja Smets. Dynamic logics of belief change. In H. Van Ditmarsch, J. Halpern, W. van der Hoek, and B. Kooi, editors, *Handbook of Epistemic Logic*, pages 313–393. College Publications, London, UK, 2015.

[43] Hans Van Ditmarsch, Wiebe van Der Hoek, and Barteld Kooi. *Dynamic Epistemic Logic*, volume 337. Springer, 2007.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[45] Gail Weiss, Yoav Goldberg, and Eran Yahav. On the practical computational power of finite precision RNNs for language recognition. *ArXiv preprint arXiv:1805.04908*, 2018.