Neural Network Semantics

CALEB SCHULTZ KISBY

Acknowledgments

djafkl;djsal;fkjas fl;jsa df;lkajs f;lkjsa l;fkjs dal;kfj al;sdfj ;lksajf ;lsakj f;lsaj f;lsaj f;lks jf;lksaj fl;ksaj f;lskaj fl;k js;lkfj dsa

f;as fk;lsajf ;lsakj flk;jas f

sajfl;kas jfl;kj sda;lkfja

sfjsaljkf;lajs f

Preface

sadfkjl;fdjasklf;j as;lkfj as;lkfj kl;asjf ;lkasj d;lkasjd ;lkjf ;lksjd fl;kaj l;fkj as;lfj asd;lkfj asl;kfj ;alsjf ;laksjf ;laksjf lk;asj f;lkasj f;lajs f;lakjs ;fljsa f;lksa jf;lj l;kfj as;lkfj al;ksj f;lkasj f;lksaj fd;lasj fl;kjas ;lfj sa;lf j

jsadkl;fj asl;dfj a;lksdjf kl;sajd fl;asjd

Caleb Schultz Kisby

Neural Network Semantics

sdfafsfl;asjdfa;lkjf l;sdaj fl;asjf kl;asjfd ;lasjf ;lkasjf ;klsajf l;ksaj f;lksajf;klasj f;klsaj f;klasj flk;asj fkl;saj fl;ksaj f;lkjs a;flkj sa;lkfj s;klafj ;slakjf ;lasfj

Contents

Chapter 1

Introduction

[In this chapter, I introduce neuro-symbolic AI at a very high level, and build up to the thesis statement (I should state the thesis statement explicitly!)]

References List:

History of neural network semantics. [39] [7] [34] [35] [16] [36] [30] [31] [24] [47] [25]

Social network semantics. [8]

Dynamic logics for learning. [11] [9] [10]

Dynamic epistemic logic; Belief revision. [62] [60] [64] [12] [13] [50] [32] [63] [61] [14]

General neuro-symbolic AI. [5] [54] [15] [27] [19] [22] [6] [23] [38] [18]

Neural network verification. [3]

AI/neural network Alignment.

Neural networks as automata. [41] [58] [43] [42]

Neural network descriptive complexity. [29] [37] [20]

Neural networks & Category theory.

General conditional logic. [33]

General modal logic. [44] [49]

Classic papers in AI. [28] [57] [26] [48] [55] [40] [53] [65] [59] [56]

General TSC/mathematics. [2] [51]

General cognitive science. [46]

Systems and frameworks. [45] [21] [1]

[Incorporate the following text into this chapter, and also the abstract, and so on.]

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; 21; 65], 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 [56], 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 [59]. 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 [40; 55]).

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 [5; 15; 54]. As a result of this effort, there are now many different proposals for neuro-symbolic systems, including Logic Tensor Networks [6], Distributed Alignment Search [23], DeepProbLog [38], Logic Explained Networks [18], and neural network fibring [22]. But these systems form a scattered picture; some unifying perspective or theory is needed. In the preface to a recent neuro-symbolic survey book [15], 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 currently 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 [47]. 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 [61; 63; 64]) with an operator for Hebbian learning [28], 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?”

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 [39], logical formulas are mapped directly to individual neurons in the net. This approach suffers from the well-known "grandmother cell" problem [26]: 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 [7; 36], 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 [7; 16; 34; 35]. More recently, [24; 25] proved soundness for inference over fuzzy neural networks. In my thesis work so far [30; 31], 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 [4; 8; 17], 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 [4]. 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) [61; 64]. Perhaps the closest logics to mine are the logics for plausibility upgrade, in particular conditionalization (Cond), lexicographic (Lex), and conservative (Consr) upgrade [60; 63]. In the Expressivity portion of my dissertation, I will explore whether Hebbian update Hebb^* 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") [9; 10; 11; 14]. 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.

Chapter 2

Background

1 Neural Networks

[Give the “2 blue 1 brown” version of an intro (very high-level here). Explain a couple of basic updates, such as Hebbian learning & backpropagation maybe]

2 Logics in Artificial Intelligence

[Explain the role of logic in AI, and of conditional & modal logics in particular]

[I can give the “classical” semantics for these, just as an example to help the reader understand what they're in for / what these sorts of things look like. But ultimately the full details for these (Kripke & KLM semantics for conditionals) will go in the appendix.]

2.1 Conditional Logic

2.2 Modal Logic

2.3 Dynamic Epistemic Logic and Belief Revision

Note 1. [Neuro-Symbolic Integration (should I make this its own section? Or does the Introduction handle it?)]

Chapter 3

Neural Network Semantics

1 Introduction

[I wrote the following short paragraph for my thesis proposal. But now I have more space to say more, slowly, about where neural network semantics comes from, what the underlying idea is. It would be nice to introduce it using an example of a neural network in practice—justify why binary, interpreted neural networks are the "right" thing to look at!]

I will now give an overview of the particular neural network semantics [rephrase]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.

2 Neural Network Models

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 [41; 43; 66]. In their terminology, we say our nets are *saturated*.

- [Todo, put somewhere in this section, optional property]
- **$\mathcal{N}$ is Fully connected:** $\forall m, n \in N$, either $(m, n) \in E$, $(n, m) \in E$, or m and n have exactly the same predecessors and successors.
- In machine learning practice, "fully connected" means that there is an edge from every node in layer l to every node in the *following* layer $l + 1$. But here we mean something much stronger: the graph is fully connected, including "highway edges" that cut between layers, as shown in [DIAGRAM]. (This intuition comes from work on highway networks [57].) This assumption is crucial for our results about iterated Hebbian learning, and we expect that letting it go will not be easy.

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.

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

2.1 The Forward Propagation Operator

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 subsequently activate new neurons, which activate yet more 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 through the net, $\mathsf{Clos}(A)$. 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

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

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 $\mathsf{Clos}(S)$ exists.

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

$$S_0, F_{S_0}(S_0), F_{S_0}(F_{S_0}(S_0)), \ldots, F_{S_0}^k(S_0), \ldots$$

has a (finite) least fixed point, and moreover that it is *unique*. Let the closure $\mathsf{Clos} \colon \mathsf{State} \to \mathsf{State}$ be the function that produces that least fixed point. For concreteness, we can say that there is some $k \in \mathbb{N}$ for which

$$\mathsf{Clos}(S) = F_S^k(S)$$

Let **Net** be the class of all binary neural network models that satisfy this postulate. This assumption implicitly constraints the allowed neural network architectures: I allow feed-forward nets, as well as certain controlled forms of recurrence. Characterizing nets that have a unique least fixed point is a big open problem.

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

2.2 The Graph Reachability Operators

As I mentioned before, a key feature of Clos is that it is not monotonic. Let's consider two closure operators over neural networks that *are* monotonic—graph reachability Reach, and "reverse" graph reachability $\mathsf{Reach}^{\downarrow}$. $\mathsf{Reach}(S)$ just returns the set of all neurons graph-reachable from S, i.e. $\mathsf{Reach}\colon \mathsf{State} \to \mathsf{State}$ is given by $n \in \mathsf{Reach}(S)$ iff there exists $m \in S$ with an E-path from $\boldsymbol{m}$ **to** $\boldsymbol{n}$. Similarly, $\mathsf{Reach}^{\downarrow}(S)$ returns the set of all neurons that graph-reach some node in S, i.e. $\mathsf{Reach}^{\downarrow}$: $\mathsf{State} \to \mathsf{State}$ is given by $n \in \mathsf{Reach}^{\downarrow}(S)$ iff there exists $m \in S$ with an E-path from $\boldsymbol{n}$ **to** $\boldsymbol{m}$.

Reach and $\mathsf{Reach}^{\downarrow}$ are not very interesting operators in their own right, but I consider them in this discussion for two reasons. First, graph reachability is necessary for reasoning about Hebbian learning (see Chapter ?). Second, in Chapter ? I would like to compare the expressive power of neural networks against many classes of models, including relational (graph) models.

3 Neural Network Semantics

[Introduce this all slowly. I will now explain how we can use neural networks as models to interpret formulas in logic. First, I will give Hannes' semantics for conditional logic. Then I will introduce my own semantics for modal logic based on his.]

3.1 Using Conditional Logic

3.2 Using Modal Logic

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

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

$\top, \bot, \vee, \rightarrow, \leftrightarrow$ 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 $\mathcal{N} \in \mathbf{Net}$, $n \in N$:

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

where $\llbracket \varphi \rrbracket = \{n \mid \mathcal{N}, n \Vdash \varphi\}$. [A reader might be confused about the “swaps” from $\mathsf{Reach}^{\downarrow}$ to $\langle \mathbf{K} \rangle$, as well as the choice to use diamond operators instead of the normal box ones.] [How can we justify the readings we had above? (at least intuitively)]

[I now have space to say these points in more detail (there's lots to say!)] Although these semantics are based on Leitgeb's [36], 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 [36].

4 Dynamic Update in Neural Network Semantics

[I now have space to express these points more slowly, in detail.] 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:

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

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.

5 Hebbian Learning: A Simple Neural Network Update Policy

[I now have space to express these points more slowly, in detail.] 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 [52].

Hebb's classic learning rule [28] 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 [48].

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 $\mathsf{Clos}(S)$ simply have their weights strengthened. Formally,

Definition 4. Let Hebb: $\mathbf{Net} \times \mathsf{State} \to \mathbf{Net}$ 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 $W'(m,n) = W(m,n) + \eta \cdot \chi_{\mathsf{Clos}(S)}(m) \cdot \chi_{\mathsf{Clos}(S)}(n)$.

Note that Hebb does not affect the edges, activation function, or evaluation of propositions. This means the resulting net is still binary, and closures $\mathsf{Clos}(S)$ 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.

Proposition 5. $\mathsf{Reach}_{\mathsf{Hebb}(\mathcal{N},A)}(B) = \mathsf{Reach}_{\mathcal{N}}(B)$

Proof. [] □

The following is easy to see [I now have space to explain] (since $\eta \geq 0$).

Proposition 6. Let $m, n \in N$. We have:

- $W_{\mathcal{N}}(m,n) \leq W_{\mathsf{Hebb}(\mathcal{N},S)}(m,n)$
- If either $m \notin \mathsf{Prop}(S)$ or $n \notin \mathsf{Prop}(S)$, then $W_{\mathsf{Hebb}(\mathcal{N},S)}(m,n) = W_{\mathcal{N}}(m,n)$.

Proof. [] □

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 $\mathsf{Clos}(S)$ will be so high that *any* activation pattern that makes contact with $\mathsf{Clos}(S)$ will "rip through" it entirely. Repeating Hebb on S further will not change the $\mathsf{Clos}(S)$-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 [31]).

Definition 7. 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 8. The *minimum negative weight score* is simply

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

Definition 9. 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. This choice for iter may seem opaque, but we will see in Lemma [which] why it guarantees that the updated weights overpower competing edge weights.

Definition 10. 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 $W'(m,n) = W(m,n) + \mathsf{iter} \cdot \eta \cdot \chi_{\mathsf{Clos}(S)}(m) \cdot \chi_{\mathsf{Clos}(S)}(n)$.

One worry we might have is that, in each iteration, we always update by $\mathsf{Clos}(S)$ in the *original* net. But it turns out that this $\mathsf{Clos}(S)$ doesn't change with each iteration, i.e.

Proposition 11. $\mathsf{Clos}_{\mathsf{Hebb}(\mathcal{N},S)}(S) = \mathsf{Clos}_{\mathcal{N}}(S)$

Proof. [] □

and so $\mathsf{Hebb}^{\star}$ is equivalent to repeatedly applying Hebb until we reach a fixed point [31]. [Elaborate on this point, it's said a little too quickly for the reader to internalize it! (maybe a picture would help?)]

As with Hebb, $\mathsf{Hebb}^{\star}$ does not affect the edges, activation function, or evaluation of propositions:

Proposition 12. $\mathsf{Reach}_{\mathsf{Hebb}^{\star}(\mathcal{N},A)}(B) = \mathsf{Reach}_{\mathcal{N}}(B)$

Proposition 13. Let $m, n \in N$. We have:

- $W_{\mathcal{N}}(m,n) \leq W_{\mathsf{Hebb}^{\star}(\mathcal{N},S)}(m,n)$
- If either $m \notin \mathsf{Clos}(S)$ or $n \notin \mathsf{Clos}(S)$, then $W_{\mathsf{Hebb}^{\star}(\mathcal{N},S)}(m,n) = W_{\mathcal{N}}(m,n)$

The following fact about $\mathsf{Hebb}^{\star}$ is the most important. It is a formal expression of our statement before: Updated weights $W_{\mathsf{Hebb}^{\star}(\mathcal{N},A)}(B)$ are so high that if m is active in $\mathsf{Hebb}^{\star}$ then n must be as well.

Lemma 14. Let $A, B \in \mathsf{State}, m, n \in N$. If $m \in$ preds, $m, n \in \mathsf{Clos}(A)$, and $m \in \mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N},A)}(B)$, then

$$A(\sum_{i=1}^{\deg(n)} W_{\mathsf{Hebb}^\star}(m_i, n) \cdot \chi_{\mathsf{Clos}_{\mathsf{Hebb}^\star(\mathcal{N},A)}(B)}(m_i)) = 1$$

Proof. [] □

Chapter 4

Soundness: Neural Network Verification

1 Introduction

dsafkljsdf;lkja sfl;kj as;lfj asl;fj a;slkjf ;lasj f;lasj fl;kjsa ;flkj as;lkfj sa;lkfj ;alsfj ;laskjf ;laskj ;lksj afd;lksj f;lkj s;lkfj sa;lkfj s;laf j;lsa fj;ls j

2 Properties of Clos, Reach, and $\mathsf{Reach}^{\downarrow}$

The following theorem, due to [34], says that we can neatly characterize the algebraic structure of Clos a closure operator. Note that Leitgeb proves this for *inhibition nets*, i.e. weightless neural networks with both excitatory and inhibitory connections. But inhibition nets and our nets $\mathcal{N} \in \mathbf{Net}$ are equivalent with respect to their propagation structure—I prove this result again for **Net** as a kind of "sanity check" that my definitions are correct.

Proposition 1. For all $S, S_1, \ldots, S_k \in \mathsf{State}$,

Inclusion. $S \subseteq \mathsf{Clos}(S)$

Idempotence. $\mathsf{Clos}(\mathsf{Clos}(S)) = \mathsf{Clos}(S)$

Cumulative. If $S_1 \subseteq S_2 \subseteq \mathsf{Clos}(S_1)$, then $\mathsf{Clos}(S_1) = \mathsf{Clos}(S_2)$

Proof. I'll prove each in turn:

(Inclusion). [Todo]

(Idempotence). The ($\subseteq$) direction is just Inclusion. As for ($\supseteq$), [Todo]

(Cumulative). [Todo]

Now consider the ($\supseteq$) direction. [Todo] □

In the terminology of [33], Prop is *cumulative*—it satisfies the cumulative property above. In fact, Clos is *not* a fully monotonic closure operator, as the following fact shows.

Proposition 2. It is not the case that for all $S_1, S_2 \in \mathsf{State}$, if $S_1 \subseteq S_2$, then $\mathsf{Clos}(S_1) \subseteq \mathsf{Clos}(S_2)$.

Proof. Consider the BFNN $\mathcal{N}$ in Figure [DIAGRAM]. Let the activation function A be $A(x) = 1$ iff $x > 0$. We have $S_1 = \{a\} \subseteq \{a, b\} = S_2$, and so the hypothesis holds. But $\mathsf{Clos}(S_1) = \{a, c\} \nsubseteq \{a, b\} = \mathsf{Clos}(S_2)$. (Observe that c does not get activated in $\mathsf{Clos}(S_2)$ because the weights cancel each other out.) □

Next, let's check that Reach is in fact a monotonic closure operator.

Proposition 3. For all $S, A, B \in \mathsf{State}$,

Inclusion. $S \subseteq \mathsf{Reach}(S)$

Idempotent. $\mathsf{Reach}(\mathsf{Reach}(S)) = \mathsf{Reach}(S)$

Monotonic. If $A \subseteq B$ then $\mathsf{Reach}(A) \subseteq \mathsf{Reach}(B)$

Closed under ∪. $\mathsf{Reach}(A \cup B) = \mathsf{Reach}(A) \cup \mathsf{Reach}(B)$

Proof. [] □

$\mathsf{Reach}^{\downarrow}$ is as well, and the proof for this is similar. [What interaction property do Reach and $\mathsf{Reach}^{\downarrow}$ share?]

3 Soundness for the Base Semantics

[Prove a few key properties for forward propagation, we can read the axioms directly off of these, then the proofs for the axioms' soundness follows]

3.1 Using Conditional Logic

3.2 Using Modal Logic

For **T** alone, [34] proves that the properties in Proposition [which?] are complete for Clos over binary, feed-forward nets. We transcribe these into our modal language.

Nec. From $\vdash \varphi$ we can infer $\vdash$**T**

Dual. $\langle \mathbf{T} \rangle \leftrightarrow \neg \mathbf{T}$

Refl. $\mathbf{T} \to \varphi$

Trans. $\mathbf{T} \to \mathbf{T}$

Cumulative. $(\varphi \to \psi) \wedge (\mathbf{T} \to \varphi) \to (\mathbf{T} \to \psi)$

As for **K**, we at least have the following sound axioms, transcribed from Proposition [which?] [Fix these, they're written for $\mathbf{K}(\varphi, \psi)$ but need to be written for $\mathbf{K}\varphi$].

Nec. From $\vdash \varphi$ we can infer $\vdash \mathbf{K}$

Dual. $\langle \mathbf{K} \rangle \leftrightarrow \neg \mathbf{K}$

Distr. $\mathbf{K} \leftrightarrow (\mathbf{K} \to \mathbf{K})$

Refl. $\mathbf{K} \to \varphi$

Trans. $\mathbf{K} \to \mathbf{K}$

These axioms are the usual complete axioms for normal modal logic over reflexive and transitive frames. So far these axioms seem innocuous enough. But for completeness, the catch is that **K** and **T** may interact in ways that affect the model construction. For example, the following axiom is easy to check, but it is not clear whether it is sufficient.

Incl. $\mathbf{K} \to \mathbf{T}$

3.3 Example: Verifying a Neural Network's Behavior

4 Properties of Hebb and Hebb*

5 Soundness for the Logic of Hebbian Learning

[Prove soundness of axioms for both single-step and iterated Hebbian update!]

5.1 Example: Verifying a Neural Networks Behavior After Learning

do example using single-step Hebbian learning, since iterated is a bit more abstract…

6 Reflections on Verification and Extraction

[Here's where I can discuss things like property verification vs model building (alignment), extraction, “valuation search”, and Thomas Icards' method]

Chapter 5

Completeness: Neural Network Model Building

1 Introduction

2 Completeness for the Base Semantics

2.1 Example: Building a Neural Network

3 Reduction Axioms for Iterated Hebbian Update

4 Completeness for Iterated Hebbian Update

Theorem 1. (Model Building) For all $\Gamma^* \subseteq \mathcal{L}^*$, there is $\mathcal{N}$ such that $\mathcal{N} \models \Gamma^*$.

Proof. Let $\Gamma^* \subseteq \mathcal{L}^*$. As outlined in the paper, our plan is to define rewrite rules based on our reduction axioms that "translate away" all of the dynamic formulas $\langle \varphi \rangle_{\mathsf{Hebb}} \psi$ in Γ^*, resulting in $\Gamma^{\text{tr}} \subseteq \mathcal{L}$. By our assumption, we have a net $\mathcal{N} \models \Gamma^{\text{tr}}$, and we show that this very same net $\mathcal{N} \models \Gamma^*$.

It's easy to see intuitively how this translation should go. For example, given the formula

$$\langle p \rangle_{\mathsf{Hebb}} (\langle p \rangle_{\mathsf{Hebb}} \langle \mathbf{B} \rangle \wedge \langle \mathbf{K} \rangle) \in \Gamma^*$$

we would recursively apply our reduction axioms, pushing $\langle p \rangle_{\mathsf{Hebb}}$ further into the expression until we can eliminate the propositional cases $\langle p \rangle_{\mathsf{Hebb}} q$.

We define the term-rewriting system that does the translation $\tau(\varphi)$ for all φ as follows.

- $\tau(p) = p$

- $\tau(\neg\varphi) = \neg\tau(\varphi)$
- $\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi)$
- $\tau(\mathbf{K}\varphi) = \mathbf{K}(\tau(\varphi))$
- $tr(\langle\varphi\rangle_{\mathsf{Hebb}} p) = tr(p)$
- $tr(\langle\varphi\rangle_{\mathsf{Hebb}} \neg\psi) = tr(\neg\langle\varphi\rangle_{\mathsf{Hebb}}\psi)$
- $tr(\langle\varphi\rangle_{\mathsf{Hebb}}(\psi \wedge \rho)) = tr\,(\langle\varphi\rangle_{\mathsf{Hebb}}\psi \wedge \langle\varphi\rangle_{\mathsf{Hebb}}\rho)$
- $tr(\langle\varphi\rangle_{\mathsf{Hebb}}\langle\mathbf{K}\rangle) = tr(\langle\mathbf{K}\rangle)$
- $tr(\langle\varphi\rangle_{\mathsf{Hebb}}\langle\mathbf{B}\rangle) = tr(\langle\mathbf{B}\rangle)$
- $tr(\langle\varphi\rangle_{\mathsf{Hebb}}\langle\psi\rangle_{\mathsf{Hebb}}\rho) = tr(\langle\varphi\rangle_{\mathsf{Hebb}}(tr(\langle\psi\rangle_{\mathsf{Hebb}}\rho)))$

Formally, the term-rewriting system takes a formula φ and recursively applies these equational rules to φ (from left-to-right). We just need to check that

1. For all ψ, $tr(\psi)$ is update-operator-free
2. This term rewriting actually terminates

The work involved in showing termination is long and tedious. The usual approach is to define a measure on formulas $c(\varphi)$ that *decreases* with each application of our reduction axioms (from left-to-right). In particular, we need c to satisfy

- If ψ is a subexpression of φ, $c(\varphi) > c(\psi)$
- $c(\langle\varphi\rangle_{\mathsf{Hebb}} p) > c(p)$
- $c(\langle\varphi\rangle_{\mathsf{Hebb}} \neg\psi) > c(\neg\langle\varphi\rangle_{\mathsf{Hebb}}\psi)$
- $c(\langle\varphi\rangle_{\mathsf{Hebb}}(\psi \wedge \rho)) > c\,(\langle\varphi\rangle_{\mathsf{Hebb}}\psi \wedge \langle\varphi\rangle_{\mathsf{Hebb}}\rho)$
- $c(\langle\varphi\rangle_{\mathsf{Hebb}}\langle\mathbf{K}\rangle) > c(\langle\mathbf{K}\rangle)$
- $c(\langle\varphi\rangle_{\mathsf{Hebb}}\langle\mathbf{B}\rangle) > c(\langle\mathbf{B}\rangle)$
- $c(\langle\varphi\rangle_{\mathsf{Hebb}}\langle\psi\rangle_{\mathsf{Hebb}}\rho) > c(\langle\varphi\rangle_{\mathsf{Hebb}}(tr(\langle\psi\rangle_{\mathsf{Hebb}}\rho)))$

But coming up with a measure c that works is tricky, and is dependent on the specific reduction axioms. For the gritty details involved in coming up with this measure, as well as proving termination for the term rewriting system, see [12].

From here, we assume we have this measure c. We now have two things left to show:

Proposition. For all $\varphi \in \Gamma^*$, we have $\vdash \varphi \leftrightarrow \tau(\varphi)$

Proof. By induction on $c(\varphi)$.

Base Step. If φ is a proposition p, then we (trivially) have $\vdash p \leftrightarrow p$.

Inductive Step. We consider each possible inductive case, and suppose the claim holds for formulas ψ with smaller $c(\psi)$. The $\neg\varphi$, $\varphi \wedge \psi$, **K**, and **B** cases all follow from applying the translation, and then applying inductive hypothesis on the subexpression that results from this.

Here are the rest of the cases. Notice that we apply the inductive hypothesis to terms whose c-cost is smaller (this is why we needed the decreasing properties of c before).

$\langle\varphi\rangle_{\mathsf{Hebb}} p$ case. We have

$$tr(\langle\varphi\rangle_{\mathsf{Hebb}} p) = tr(p) = p$$

and so we need to show that

$$\vdash \langle\varphi\rangle_{\mathsf{Hebb}} p \leftrightarrow p$$

but this holds by our propositional reduction axiom.

$\langle\varphi\rangle_{\mathsf{Hebb}} \neg\psi$ case. We have:

$$\begin{aligned}
\vdash & \langle\varphi\rangle_{\mathsf{Hebb}} \neg\psi \\
& \leftrightarrow \neg\langle\varphi\rangle_{\mathsf{Hebb}} \psi && \text{(by the reduction axiom)} \\
& \leftrightarrow tr(\neg\langle\varphi\rangle_{\mathsf{Hebb}} \psi) && \text{(inductive hypothesis)} \\
& = tr(\langle\varphi\rangle_{\mathsf{Hebb}} \neg\psi) && \text{(by our translation)}
\end{aligned}$$

$\langle\varphi\rangle_{\mathsf{Hebb}} \psi \wedge \rho$ case.. We have:

$$\begin{aligned}
\vdash & \langle\varphi\rangle_{\mathsf{Hebb}} (\psi \wedge \rho) \\
& \leftrightarrow \langle\varphi\rangle_{\mathsf{Hebb}} \psi \wedge \langle\varphi\rangle_{\mathsf{Hebb}} \rho && \text{(by the reduction axiom)} \\
& \leftrightarrow tr(\langle\varphi\rangle_{\mathsf{Hebb}} \psi \wedge \langle\varphi\rangle_{\mathsf{Hebb}} \rho) && \text{(inductive hypothesis)} \\
& = tr(\langle\varphi\rangle_{\mathsf{Hebb}} (\psi \wedge \rho)) && \text{(by our translation)}
\end{aligned}$$

$\langle\varphi\rangle_{\mathsf{Hebb}} \mathbf{K}$ case.. We have:

$$\begin{aligned}
\vdash & \langle\varphi\rangle_{\mathsf{Hebb}} \mathbf{K} \\
& \leftrightarrow \mathbf{K} && \text{(by the reduction axiom)} \\
& \leftrightarrow tr(\mathbf{K}) && \text{(inductive hypothesis)} \\
& = tr(\langle\varphi\rangle_{\mathsf{Hebb}} \mathbf{K}) && \text{(by our translation)}
\end{aligned}$$

$\langle\varphi\rangle_{\text{Hebb}}\mathbf{B}$ case.. We have:

$$\begin{aligned}&\vdash \langle\varphi\rangle_{\text{Hebb}}\mathbf{B}\\ &\quad\leftrightarrow \langle\mathbf{B}\rangle\\ &\qquad\text{(by the reduction axiom)}\\ &\quad\leftrightarrow tr(\langle\mathbf{B}\rangle)\\ &\qquad\text{(inductive hypothesis)}\\ &\quad= tr(\langle\varphi\rangle_{\text{Hebb}}\mathbf{B})\\ &\qquad\text{(by our translation)}\end{aligned}$$

$\langle\varphi\rangle_{\text{Hebb}}\langle\psi\rangle_{\text{Hebb}}\rho$ case.. This case is more interesting. First, notice our translation for this case:

$$tr(\langle\varphi\rangle_{\text{Hebb}}\langle\psi\rangle_{\text{Hebb}}\rho) = tr\,(\langle\varphi\rangle_{\text{Hebb}} tr(\langle\psi\rangle_{\text{Hebb}}\rho))$$

That is, we translate the inner expression first, then translate the outer expression. This inner $tr(\langle\psi\rangle_{\text{Hebb}}\rho)$ is equivalent to some update-operator-free formula χ:

$$\vdash \chi \leftrightarrow tr(\langle\psi\rangle_{\text{Hebb}}\rho) \leftrightarrow \langle\psi\rangle_{\text{Hebb}}\rho \tag{1}$$

(This last equivalence follows from our inductive hypothesis, which we can apply because $\langle\psi\rangle_{\text{Hebb}}\rho$ is a subexpression of $\langle\varphi\rangle_{\text{Hebb}}\langle\psi\rangle_{\text{Hebb}}\rho$.)

What about $tr(\langle\varphi\rangle_{\text{Hebb}}\chi)$? Well, since χ is update-operator-free, this reduces to our previous inductive cases. So we have

$$\vdash tr(\langle\varphi\rangle_{\text{Hebb}}\chi) \leftrightarrow \langle\varphi\rangle_{\text{Hebb}}\chi \tag{2}$$

Putting this all together, we have:

$$\begin{aligned}&\vdash \langle\varphi\rangle_{\text{Hebb}}\langle\psi\rangle_{\text{Hebb}}\rho\\ &\quad\leftrightarrow \langle\varphi\rangle_{\text{Hebb}}\chi && \text{(by (1))}\\ &\quad\leftrightarrow tr(\langle\varphi\rangle_{\text{Hebb}}\chi) && \text{(by (2))}\\ &\quad\leftrightarrow tr(\langle\varphi\rangle_{\text{Hebb}}(tr(\langle\psi\rangle_{\text{Hebb}}\rho))) && \text{(by (1))}\\ &\quad\leftrightarrow tr(\langle\varphi\rangle_{\text{Hebb}}\langle\psi\rangle_{\text{Hebb}}\rho) && \text{(by our translation)}\end{aligned}$$

$\square$ $\square$

Theorem 2. (Completeness) The logic of Hebbian Learning is completely axiomatized by the base axioms [ref!], along with the above reduction axioms. That is, for all consistent $\Gamma^* \subseteq \mathcal{L}^*$, if $\Gamma^* \models \varphi$ then $\Gamma^* \vdash \varphi$.

Proof. Since our language $\mathcal{L}^*$ has negation, completeness follows from model building in the usual way; this proof is entirely standard. Suppose contrapositively that $\Gamma^* \nvdash \varphi$. It follows that $\Gamma^* \vdash \neg\varphi$.

So $\Gamma^* \cup \{\neg\varphi\}$ is consistent, and by Theorem [todo—need to modify the construction to make sure the net is fully connected. Quote: "But remember that our nets are also fully connected! So we need to modify the model construction from [34] by introducing a zero weight edge between every pair of previously unconnected nodes. Note that this change does not affect the $\Prop$-structure of the net."], we have $\mathcal{N} \in \mathsf{Net}$ such that $\mathcal{N} \models \Gamma^* \cup \{\neg\varphi\}$. But then $\mathcal{N} \models \Gamma^*$ yet $\mathcal{N} \not\models \varphi$, which is what we wanted to show. $\square$

4.1 Example: Building a Neural Network with Learning Constraints

5 Reflections on Interpretability and Alignment

[Integrate the following into this chapter!]

In the following examples, I'll walk through important constructions for neural networks that I will refer back to. I mentioned before that each choice of E_i, W_i, A_i specifies a transition function F_{S_0}. Different choices for F_{S_0} in turn give different interpretations for the closure function Clos. These examples should give you a feel for what sorts of state transitions we can represent with neural networks.

The NAND Construction. Here is an interesting construction that Hannes uses in [34] to prove completeness for weighted neural network models. He does this by way of inhibition nets, i.e. nets with inhibitory edges that block excitatory edges.

As before, we want to build a neural network $\mathcal{N}$ where the graph $\langle N, E\rangle$, bias, and evaluation V are given. Here's the *inhibition net* construction: First, create an edge from bias to every n that is not E-minimal (in other words, if n has any predecessors at all, then bias is one of them). Then for each node n and its predecessors $\mathsf{bias} = m_0, m_1, \dots, m_r$, connect inhibition edges as follows.

[DIAGRAM]

That is, each node m_i is inhibited by m_{i-1} ($\mathsf{bias} = m_0$ inhibited by m_r). This has the following effect: if all m_i activate, they each inhibit each other, and so n does not activate. If only *some* m_i activate, then there is some m_i that is uninhibited, and so n activates. And finally, since bias is always active we cannot have *no* m_i active. In other words, $n \in F_i(S)$ iff n is already active ($n \in S$), or *at least one, but not all* predecessors m_i are in S.

We can simulate this effect with weighted neural networks. Create an edge from bias to every n that is not E_i-minimal. Then pick $W_i(m,n) = \frac{1}{|\mathsf{preds}|+1}$ (the extra $+1$ accounts for the bias). Finally, pick $A_i(x) = 1$ iff $x < 1$. Take a moment to check that $n \in F_i(S)$ iff $n \in S$, or at least one, but not all predecessors m are in S.

Consider the best function from before, but over E_i. That is, $\mathsf{best}_{E_i}(S)$ is the set of E_i-minimal nodes in S. It turns out that, *if* E_i is transitive (which is true in our use case), the closure function Clos_i for the not-every construction is precisely the *dual* of best_{E_i}: (Leitgeb, [34]) Suppose E_i is transitive. Then for all S, $\mathsf{Clos}_i(S) = (\mathsf{best}_{E_i}(S^{\mathsf{C}}))^{\mathsf{C}}$

Proof. First, the $(\subseteq)$ direction. Let $n \in \mathsf{Clos}_i(S) = F_i^k(S)$ for some $k \in \mathbb{N}$. We proceed by induction on k.

Base Step. $n \in F_i^0(S) = S$. By best_{E_i}-inclusion, $\mathsf{best}_{E_i}(S^{\mathsf{C}}) \subseteq S^{\mathsf{C}}$. Flipping this containment gives us $S \subseteq \mathsf{best}_{E_i}(S^{\mathsf{C}})^{\mathsf{C}}$. So $n \in \mathsf{best}_{E_i}(S^{\mathsf{C}})^{\mathsf{C}}$.

Inductive Step. Let $k \geq 0$. We have $n \in F_i^k(S) = F_i(F_i^{k-1}(S))$. By construction of F_i, we have two cases: n is already active ($n \in F_i^{k-1}(S)$), or some predecessor is $m \in F_i^{k-1}(S)$, and not all predecessors are. In the first case, our inductive hypothesis says $n \in \mathsf{best}_{E_i}(S^{\mathsf{C}})^{\mathsf{C}}$. In the latter case, the inductive hypothesis gives $m \in \mathsf{best}_{E_i}(S^{\mathsf{C}})^{\mathsf{C}}$. From here we split by cases: either $m \in S$ or $m \notin S$.

Case: $m \in S$. In this case, we trivially have $n \notin \mathsf{best}_{E_i}(S^{\mathsf{C}})$, since n is not even in S^{C}.

Case: $m \notin S$. Since $m \in S^{\mathsf{C}}$ but $m \notin \mathsf{best}_{E_i}(S^{\mathsf{C}})$, by the smoothness condition there must be some better x with $x E_i m$ that *is* the best, i.e. $x \in \mathsf{best}_{E_i}(S^{\mathsf{C}})$. By best-inclusion, $x \in S^{\mathsf{C}}$. Well, $x E_i m E_i n$, and since E_i is transitive we have $x E_i n$. And so we have an $x \in S^{\mathsf{C}}$ that is E_i-better than n. So $n \notin \mathsf{best}_{E_i}(S^{\mathsf{C}})$, which is what we wanted to show.

As for the $(\supseteq)$ direction, suppose contrapositively that $n \notin \mathsf{Clos}_i(S)$. Note that $n \notin S$, by Clos-inclusion. First, I claim that every predecessor m of n is in S. Suppose not, i.e. suppose that some predecessor $m \notin S$. Note that we always have $\mathsf{bias} \in S$. By construction, we also have $\mathsf{bias} n$. So n has one predecessor, m, not in S, and another predecessor, bias, in S. In other words, some but not all predecessors of n are in S. By construction of F_i, $n \in F_i(S)$. But $F_i(S) \subseteq \mathsf{Clos}_i(S)$, so this contradicts $n \notin \mathsf{Clos}_i(S)$.

So every predecessor m of n is in S. But this implies that any $m \notin S$ cannot be an E_i-predecessor of n. In other words, $\forall m \in S^{\mathsf{C}}, \neg \mathrm{mE}_i n$. Since $n \in S^{\mathsf{C}}$ from before, we have $n \in \mathsf{best}_{E_i}(S)$ by the definition of best. This concludes this direction of the proof. $\square$

Chapter 6

Expressivity: Measuring the Modeling Power of Neural Networks

1 Introduction

2 Basic Setup

[introduce a number of models that I will compare neural networks against; all will share the same underlying language of K and T for fair comparison, and I will give some examples of (1) properties and (2) updates that they can support / define / model.]

[In this section, we generalize our semantics to be *multi-modal*, which allows us to describe multi-agent situations, but also multiple relations for a single agent (as is normally done in FOL.)]

2.1 Relational Models

A relational model is $\mathcal{M} = \langle W,_{i\in\mathbf{I}}, V\rangle$, where

- W is some finite set of worlds (or states)
- Each $R_i \subseteq W \times W$ (the accessibility relations)
- V: Proposition $\to \mathcal{P}(W)$ (the valuation function)

Define **Rel** to be the class of all such models, and define **Rel** to be the class of all such models where each R_i is additionally reflexive and transitive. The semantics for both classes is given by:

$$\begin{array}{lll}
\mathcal{M}, w \Vdash p & \text{iff} & w \in V(p) \\
\mathcal{M}, w \Vdash \neg\varphi & \text{iff} & \mathcal{M}, w \nVdash \varphi \\
\mathcal{M}, w \Vdash \varphi \wedge \psi & \text{iff} & \mathcal{M}, w \Vdash \varphi \text{and } \mathcal{M}, w \Vdash \psi \\
\mathcal{M}, w \Vdash \Box_i \varphi & \text{iff} & \text{for all } u \text{ with wR}_i u, \mathcal{M}, u \Vdash \varphi
\end{array}$$

[Mention axioms, soundness, completeness (refer to the appendix!)]

2.2 Plausibility Models

A plausibility model, first introduced in [33], is $\mathcal{M} = \langle W,_{i\in\mathbf{I}}, V\rangle$, i.e. the models themselves are just relational models. As before, I assume that W is finite, and as with **Rel**, R_i is reflexive and transitive. The key difference is that we interpret $\Box_i\,\varphi$ to hold in the best (or most plausible) states satisfying φ. Formally, let $\mathsf{best}_{R_i}(S) =$ (the R_i-minimal states over S). We additionally impose the following "smoothness condition" [33] on best_{R_i}: For all models $\mathcal{M}$, $i \in \mathbf{I}$, sets S, and all $w \in W$, if $w \in S$ then either $w \in \mathsf{best}_{R_i}(S)$, or there is some $v\mathrm{R}_i\, w$ better than w that *is* the best, i.e. $v \in \mathsf{best}_{R_i}(S)$. The new semantics for $\Box_i$ is

$$\mathcal{M}, w \Vdash \Box_i\,\varphi \quad \text{iff} \quad w \in \mathsf{best}_{R_i}(\llbracket\varphi\rrbracket)$$

where $\llbracket\varphi\rrbracket = .$ In practice, plausibility semantics coexist alongside relational semantics, so I allow some $\Box_i\,\varphi$ to be given relational semantics instead. Let **Plaus** be the class of all such models. Since we include relational operators, note that **Rel** $\subseteq$ **Plaus**.

Any plausibility operator $\Box_i$ picks out a corresponding conditional: $\Box_i\,\varphi \to \psi$ reads "the best φ are ψ," which in the KLM tradition is the semantics for the conditional $\varphi \Rightarrow \psi$.

[Mention axioms, soundness, completeness (refer to the appendix!)]

2.3 Social Network Models

2.4 Neighborhood Models

A neighborhood model is $\mathcal{M} = \langle W,_{i\in\mathbf{I}}, V\rangle$, where W and V are as before and each $f_i \colon W \to \mathcal{P}(\mathcal{P}(W))$ is an accessibility *function*. The intuition is that f_i maps each state w to the "formulas" (sets of states) that hold at w. Define **Nbhd** to be the class of all neighborhood models.

Moreover, the *core* of f is $\cap f(x) = \bigcap_{X\in f(w)} X$. As with **Rel**, let $\mathbf{Nbhd}_{S4}$ be the class of all neighborhood models that are additionally reflexive ($\forall w, w \in \cap f(w)$) and transitive ($\forall w$, if $X \in f(w)$ then $\in f(w)$).

The semantics for both classes is the same as the previous classes, except the $\Box_i$ case is now:

$$\mathcal{M}, w \Vdash \Box_i \varphi \quad \text{iff} \quad [\![\varphi]\!] \in f_i(w)$$

where again $[\![\varphi]\!] = .$

2.5 Model Simulations

3 Modeling Power of the Base Neural Network Semantics

4 Modeling Power of Neural Network Update

5 Definability and Expressivity

6 Reflections on the Complexity Hierarchy

[This is where I give "the chart", and make higher-level connections to descriptive complexity of neural networks, neural nets as automata, FLaNN work, and give my own personal long-term vision for complexity theory. (I can also discuss *dynamic* complexity here, which is imo underrated in complexity work)]

[Integrate the following into this chapter!]

The Graph-Reachability Construction. I will show here how the more general Clos operator can be used to simulate Reach and $\text{Reach}^{\downarrow}$. Suppose we are given a graph $\langle N, E \rangle$ and a valuation function V. How can we build a neural network whose closure Clos is graph-reachability Reach? We want to build a net:

$$\mathcal{N} = \langle N, \mathsf{bias}, E, W, A, \eta, V\rangle$$

[what to do about bias and η?] For the weights, pick

$$W(m,n) = \begin{cases} 1 & \text{if } mEn \\ 0 & \text{otherwise} \end{cases}$$

Then pick the activation function $A(x) = 1$ iff $x > 0$. Recall that $n \in F_{S_0}(S)$ iff $n \in S_0$ or is activated by its predecessors in S. In this case, $n \in F_{S_0}(S)$ whenever $n \in S_0$ or at least one E-predecessor m of n is in S. I call this the *graph-reachability construction* because the closure $\mathsf{Clos}(S)$ produces exactly those nodes graph-reachable from S:

Proposition 1. For all states $S \in \mathsf{State}$, $\mathsf{Clos}(S) = \mathsf{Reach}(S)$.

Proof. First, the $(\subseteq)$ direction. Let $n \in \mathsf{Clos}(S) = F_S^k(S)$ for some $k \in \mathbb{N}$. By induction on k.

Base Step. $n \in F_S^0(S) = S$. So there is a trivial E_i-path (length=0) from $n \in S$ to itself.

Inductive Step. Let $k \geq 0$. We have $n \in F_S^k(S) = F_S(F_S^{k-1}(S))$. By construction of F_S, we have two cases: Either $n \in F_S^{k-1}(S)$ or at least one E-predecessor x of n is in $F_S^{k-1}(S)$. In the first case, our inductive hypothesis gives a path from some $m \in S$ to n. In the second case, our inductive hypothesis gives a path from some $m \in S$ to x. But since xEn, we can extend this path to be from m to n.

As for the $(\supseteq)$ direction, suppose there is an E-path from some $m \in S$ to n. We proceed by induction on the length of that path.

Base Step. The path is trivial, i.e. has length 0. So $n \in S$. But $S = F_i^0(S) \subseteq \mathsf{Clos}_i(S)$, and so $n \in \mathsf{Clos}_i(S)$.

Inductive Step. Say the path is of length $l \geq 0$. Let x be some immediate E_i-predecessor of n. By the inductive hypothesis, $x \in \mathsf{Clos}_i(S)$, and so $x \in F_i^k(S)$ for some natural k. But since x is an E_i-predecessor of n, by construction of F_i, $n \in F_i(F_i^k(S)) = F_i^{k+1}(S)$. Since $\mathsf{Clos}_i(S)$ is a closure, it includes $F_i^{k+1}(S)$. So $n \in \mathsf{Clos}_i(S)$. □

As for $\mathsf{Reach}^{\downarrow}$, we first *reverse* the edges E, and then do the graph-reachability construction. In other words, let $mE'n$ iff nEm,

$$W(m,n) = \begin{cases} 1 & \text{if } mE'n \\ 0 & \text{otherwise} \end{cases}$$

and pick A, η, V the same as above. For this construction, the closure $\mathsf{Clos}(S)$ produces exactly those nodes which reach some node in S. The proof is similar to the proof for Reach. [Check that that's actually true!!]

Proposition 2. For all states $S \in \mathsf{State}$, $\mathsf{Clos}(S) = \mathsf{Reach}^{\downarrow}(S)$.

The Social Majority Construction. [Introduce the idea of social networks here if I haven't already, and mention the "social majority" propagation/diffusion (tell it slowly, like a story). I will show that our neural networks can simulate this simple social majority operator (make the social majority operator a bit more formal).]

As before, we want to build a neural network $\mathcal{N}$ where the graph $\langle N, E_i \rangle$, bias, and evaluation V are given. This time, pick $W_i(m,n) = \frac{1}{|\mathsf{preds}(n)|}$, and then pick $A_i(x) = 1$ iff $x \geq \frac{1}{2}$. Visually, for each node n and its predecessors $m_1, \ldots, m_r$ we have

[DIAGRAM!]

This gives us $n \in F_{S_0}(S)$ if $n \in S_0$ or if the majority (more than half) of E-predecessors are in S. In this case, the closure Clos can be interpreted as the diffusion of an opinion or attitude through a social network. This is one of the choices that [8] consider for modelling influence in social networks. [this paragraph is a bit terse now that I'm writing a longer version of this.]

Chapter 7

Conclusions

[I like the way Levin Hornischer wrote his: A summary, followed by a list of results, followed by a list of open questions]

Results

1.

Open Questions

1.

Appendix A

Relational (Kripke) Semantics

fd

Appendix B

Plausibility Semantics

(just for modal logic, there are no relational semantics for conditional logic because relational models are monotonic.)

Appendix C

Neighborhood Semantics

(for modal logic and conditional logic)

(for modal logic and conditional logic, although I won't need it for conditional logic per se)

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] Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

[3] Aws Albarghouthi et al. Introduction to neural network verification. *Foundations and Trends® in Programming Languages*, 7(1–2):1–157, 2021.

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

[5] 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.

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

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

[8] 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.

[9] 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.

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

[11] 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.

[12] Alexandru Baltag, Lawrence S Moss, and Sławomir Solecki. Logics for epistemic actions: completeness, decidability, expressivity. *Logics*, 1(2):97–147, 2023.

[13] Alexandru Baltag, Lawrence S Moss, and Slawomir Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proceedings of the 7th conference on Theoretical aspects of rationality and knowledge*, pages 43–56. 1998.

[14] 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.

[15] 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.

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

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

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

[19] Artur d'Avila Garcez, Krysia Broda, and Dov M Gabbay. Symbolic knowledge extraction from trained neural networks: A sound approach. *Artificial Intelligence*, 125(1-2):155–207, 2001.

[20] Walter Dean. Computational complexity theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, 2021.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

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

[27] Frankvan Harmelen. Preface: The 3rd AI wave is coming, and it needs a theory. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*, page 0. IOS Press BV, 2022.

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

[29] Neil Immerman. *Descriptive Complexity*. Springer Science & Business Media, 1998.

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

[31] 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.

[32] Dexter Kozen and Rohit Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14(1):113–118, 1981.

[33] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence*, 44(1-2):167–207, 1990.

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

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

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

[37] Leonid Libkin. *Elements of Finite Model Theory*, volume 41. Springer, 2004.

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

[39] 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.

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

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

[42] William Merrill and Ashish Sabharwal. The expresssive power of transformers with chain of thought. *ArXiv preprint arXiv:2310.07923*, 2023.

[43] 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.

[44] Lawrence S Moss. Finite models constructed from canonical formulas. *Journal of Philosophical Logic*, 36:605–640, 2007.

[45] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction–CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings 28*, pages 625–635. Springer, 2021.

[46] Gregory Murphy. *The Big Book of Concepts*. MIT press, 2004.

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

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

[49] Eric Pacuit. *Neighborhood Semantics for Modal Logic*. Springer, 2017.

[50] Jan A. Plaza. Logics of public communications. *Synthese*, 158:165–179, 2007.

[51] George Polya. *Mathematics and Plausible Reasoning: Induction and Analogy in Mathematics*, volume 2. Princeton University Press, 1954.

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

[53] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

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

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

[56] 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.

[57] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, page 0. Curran Associates, Inc., 2015.

[58] 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.

[59] 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.

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

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

[62] Johan Van Benthem and Fenrong Liu. Dynamic logic of preference upgrade. *Journal of Applied Non-Classical Logics*, 17(2):157–182, 2007.

[63] 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.

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

[65] 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.

[66] 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.