# The Algebra of Intelligence

Sijie Wang

January 31, 2026

# Preface

What is intelligence? This book approaches the question mathematically.

We take a specific stance: **intelligence is the ability to learn and then act**. This suggests a natural decomposition:

1. **Expression**: What can be described? What structures exist?
2. **Learning**: How do we find the right structure from data?
3. **Learning well**: Why are some structures easier to learn than others?
4. **Limits**: How far can intelligence go?

The mathematics we develop unifies modal logic, coalgebra, semirings, and optimization theory.

★ **Key Insight**

The central question of this book:

> *Given what we want to express (semantics), what is the best way to parameterize it (syntax) for learning?*

# Contents

# Part I

# Expression

# Chapter 1

# Logic as Language

**Goals of This Chapter**

- Understand logic as a formal language for making statements
- See why propositional logic is limited
- Motivate the need for modal operators

## 1.1 What Can We Say?

Logic is a language. Like any language, it has:

- **Syntax**: the grammar—what counts as a well-formed sentence
- **Semantics**: the meaning—what sentences are true and when

**Intuition**

Think of logic as a very precise, very restricted language. It can't express poetry or emotion, but what it *can* express, it expresses unambiguously. This precision is its power.

## 1.2 Propositional Logic

The simplest logic. We have propositions ($p, q, r, \ldots$) and connectives ($\neg, \wedge, \vee, \to$).

**Definition 1.1** (Propositional Language)**.** Let $\Phi$ be a countable set of propositional variables. The propositional language $\mathcal{L}_0$ is:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi)$$

where $p \in \Phi$. Other connectives are defined: $\varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$, etc.

Semantics: a *valuation* $v : \Phi \to \{0,1\}$ assigns truth values.

⚠ Common Pitfall

Propositional logic is *flat*. Every proposition is either true or false, period. There's no room for "true from one perspective but false from another." Real reasoning needs more.

## 1.3 The Need for Modality

Consider these statements:

- "It is *possible* that it will rain tomorrow."
- "Alice *knows* that Bob is lying."
- "It is *obligatory* to pay taxes."
- "The program *will eventually* terminate."

None of these can be expressed in propositional logic. They all involve a *mode* of truth.

★ Key Insight

Modal logic adds **operators** that modify propositions:

- $\Box\varphi$: "necessarily $\varphi$" / "in all accessible states, $\varphi$"
- $\Diamond\varphi$: "possibly $\varphi$" / "in some accessible state, $\varphi$"

## 1.4 Modal Language

**Definition 1.2** (Modal Language)**.** The modal language $\mathcal{L}(\Box, \Diamond)$ extends propositional logic:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \psi) \mid \Box\varphi$$

We define $\Diamond\varphi := \neg\Box\neg\varphi$.

But what does $\Box\varphi$ *mean*? For that, we need **possible worlds**.

# Chapter 2

# Kripke Frames

**Goals of This Chapter**

- Understand Kripke semantics: worlds, accessibility, satisfaction
- See how different frame properties give different logics
- Learn bisimulation as the right notion of equivalence

## 2.1 Possible Worlds

The key idea: truth is *relative to a world.*

**Definition 2.1** (Kripke Frame)**.** A *Kripke frame* is a pair $\mathcal{F} = (W, R)$ where:

- $W$ is a non-empty set of *possible worlds*
- $R \subseteq W \times W$ is the *accessibility relation*

**Definition 2.2** (Kripke Model)**.** A *Kripke model* is a triple $\mathcal{M} = (W, R, V)$ where $(W, R)$ is a frame and $V : \Phi \to \mathcal{P}(W)$ is a *valuation.*

**Intuition**

Think of a Kripke model as a graph. Nodes are worlds, edges are accessibility, and each node is labeled with which propositions are true there.

## 2.2 Satisfaction

**Definition 2.3** (Satisfaction Relation)**.** Let $\mathcal{M} = (W, R, V)$ and $w \in W$. Define $\mathcal{M}, w \models \varphi$:

$$\begin{array}{ll}
\mathcal{M}, w \models p & \text{iff } w \in V(p) \\
\mathcal{M}, w \models \neg\varphi & \text{iff } \mathcal{M}, w \not\models \varphi \\
\mathcal{M}, w \models \varphi \wedge \psi & \text{iff } \mathcal{M}, w \models \varphi \text{ and } \mathcal{M}, w \models \psi \\
\mathcal{M}, w \models \Box\varphi & \text{iff for all } v : wRv \Rightarrow \mathcal{M}, v \models \varphi
\end{array}$$

## 2.3 Frame Properties and Axioms

| **Property of $R$** | **Axiom** | **Name** |
|---|---|---|
| Reflexive | $\Box\varphi \to \varphi$ | T |
| Transitive | $\Box\varphi \to \Box\Box\varphi$ | 4 |
| Symmetric | $\varphi \to \Box\Diamond\varphi$ | B |
| Euclidean | $\Diamond\varphi \to \Box\Diamond\varphi$ | 5 |

## 2.4 Bisimulation

**Definition 2.4** (Bisimulation)**.** A relation $Z \subseteq W \times W'$ is a *bisimulation* if whenever $wZw'$:

1. **(Atoms)** $w \in V(p) \Leftrightarrow w' \in V'(p)$ for all $p$
2. **(Zig)** If $wRv$, then $\exists v' : w'R'v'$ and $vZv'$
3. **(Zag)** If $w'R'v'$, then $\exists v : wRv$ and $vZv'$

**Theorem 2.5** (Bisimulation Invariance)**.** *If $wZw'$, then $\mathcal{M}, w \models \varphi \iff \mathcal{M}', w' \models \varphi$ for all modal $\varphi$.*

★ **Key Insight**

Bisimulation is the "right" equivalence for modal logic. This will generalize to coalgebra.

# Chapter 3

# Bisimulation

### Goals of This Chapter

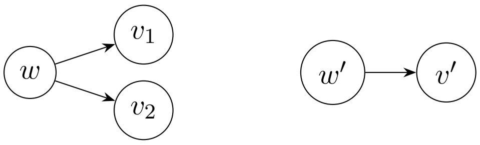
- Understand bisimulation as the fundamental equivalence for modal logic
- Master the zig-zag conditions
- See bisimulation games as an intuitive characterization
- Connect to the later coalgebraic generalization

## 3.1 The Problem of Equivalence

When are two models "the same" from a modal perspective?

Not isomorphism—too strict. Two very different-looking models might satisfy exactly the same formulas.

**Example 3.1.** Consider:



If $v_1, v_2, v'$ all satisfy the same atoms, then $w$ and $w'$ satisfy the same modal formulas—even though the models have different structure.

## 3.2 Definition

**Definition 3.2** (Bisimulation)**.** Let $\mathcal{M} = (W, R, V)$ and $\mathcal{M}' = (W', R', V')$. A relation $Z \subseteq W \times W'$ is a *bisimulation* if whenever $w \; Z \; w'$:

**(Atoms)** For all $p \in \Phi$: $w \in V(p) \Leftrightarrow w' \in V'(p)$

**(Zig)** If $wRv$, then there exists $v' \in W'$ such that $w'R'v'$ and $v \; Z \; v'$
**(Zag)** If $w'R'v'$, then there exists $v \in W$ such that $wRv$ and $v \; Z \; v'$

We write $w \underline{\leftrightarrow} w'$ if there exists a bisimulation relating $w$ and $w'$.

**Intuition**

Zig: "I can match your move."
Zag: "You can match my move."
Two states are bisimilar if they can "simulate" each other, step by step.

## 3.3 Bisimulation Games

Bisimulation has a game-theoretic interpretation.
**Players**: Spoiler (tries to distinguish) vs Duplicator (tries to match)
**Game**: Start at $(w, w')$. Each round:

1. Spoiler picks a side and makes a move (follows an edge)
2. Duplicator must match on the other side

**Winning**:

- Spoiler wins if atoms differ, or Duplicator can't move
- Duplicator wins if the game continues forever

**Theorem 3.3.** *$w \underline{\leftrightarrow} w'$ iff Duplicator has a winning strategy.*

## 3.4 The Invariance Theorem

**Theorem 3.4** (Bisimulation Invariance)**.** *If $w \underline{\leftrightarrow} w'$, then for all modal formulas $\varphi$:*

$$\mathcal{M}, w \models \varphi \iff \mathcal{M}', w' \models \varphi$$

*Proof.* Induction on $\varphi$.
**Base**: $\varphi = p$. By (Atoms).
**Step** $\neg$: Immediate from IH.
**Step** $\wedge$: Immediate from IH.
**Step** $\Box$: Suppose $\mathcal{M}, w \models \Box\psi$ and $w \; Z \; w'$.

- Take any $v'$ with $w'R'v'$
- By (Zag), exists $v$ with $wRv$ and $v \; Z \; v'$
- Since $\mathcal{M}, w \models \Box\psi$, we have $\mathcal{M}, v \models \psi$
- By IH, $\mathcal{M}', v' \models \psi$

So $\mathcal{M}', w' \models \Box\psi$. The other direction uses (Zig). □

## 3.5 Largest Bisimulation

**Proposition 3.5.** *The union of all bisimulations is itself a bisimulation—the* largest *bisimulation.*

**Definition 3.6.** $w \sim w'$ (bisimilarity) iff $(w, w')$ is in the largest bisimulation.

## 3.6 Looking Ahead: Coalgebraic Bisimulation

The zig-zag conditions generalize beyond Kripke frames.

For any coalgebra $(X, \gamma : X \to FX)$, there's a notion of bisimulation. For the powerset functor $F = \mathcal{P}$, it coincides exactly with Kripke bisimulation.

**Key Insight**

Bisimulation is not an accident of Kripke semantics. It's a fundamental concept that exists for *any* coalgebra. This is why it will reappear throughout the book.

# Chapter 4

# Model Constructions

**◎ Goals of This Chapter**

- Master bisimulation and its equivalent characterizations
- Understand bounded morphisms (p-morphisms)
- Learn model constructions: generated submodels, disjoint unions, unravelling
- Prove the Hennessy-Milner theorem

## 4.1 Bisimulation Revisited

Recall: a bisimulation relates states that are “behaviorally equivalent.”

**Definition 4.1** (Bisimulation)**.** $Z \subseteq W \times W'$ is a bisimulation if $wZw'$ implies:

1. **(Atoms)** Same atomic propositions
2. **(Zig)** Every successor of $w$ is matched by a successor of $w'$
3. **(Zag)** Every successor of $w'$ is matched by a successor of $w$

**Theorem 4.2** (Bisimulation Invariance)**.** *Bisimilar states satisfy the same modal formulas.*

*Proof.* Induction on formula structure. □

## 4.2 Bounded Morphisms

**Definition 4.3** (Bounded Morphism / p-morphism)**.** A function $f : W \to W'$ is a *bounded morphism* if:

1. $w \in V(p) \Leftrightarrow f(w) \in V'(p)$
2. $wRv \Rightarrow f(w)R'f(v)$ (homomorphism)
3. $f(w)R'v' \Rightarrow \exists v : wRv \wedge f(v) = v'$ (back condition)

**Proposition 4.4.** *If $f$ is a bounded morphism, then its graph is a bisimulation.*

## 4.3 Generated Submodels

**Definition 4.5** (Generated Submodel)**.** The submodel *generated by $w$* is the restriction to all worlds reachable from $w$.

**Theorem 4.6.** *A world satisfies the same formulas in the full model and in its generated submodel.*

## 4.4 Disjoint Union

**Definition 4.7.** The *disjoint union* $\mathcal{M}_1 \uplus \mathcal{M}_2$ combines two models without connecting them.

**Proposition 4.8.** *Each component of a disjoint union is bisimilar to itself in the union.*

## 4.5 Tree Unravelling

**Definition 4.9** (Unravelling)**.** The *unravelling* of $\mathcal{M}$ from $w$ is a tree where:

- Nodes are finite paths starting from $w$
- Edges extend paths by one step
- Valuation inherited from endpoints

**Theorem 4.10.** *Every pointed model is bisimilar to its unravelling.*

★ Key Insight

Modal logic cannot distinguish a model from its tree unravelling. This is why modal logic has the *tree model property*.

## 4.6 Hennessy-Milner Theorem

**Theorem 4.11** (Hennessy-Milner)**.** *On* image-finite *models, bisimilarity coincides with modal equivalence.*

**Intuition**

On finite-branching models, if two states satisfy exactly the same modal formulas, they must be bisimilar. The converse always holds; this theorem says that for nice enough models, the two notions coincide exactly.

# Chapter 5

# Completeness

**◎ Goals of This Chapter**

- Understand normal modal logics and their axiomatizations
- Master the canonical model construction
- Prove completeness for K, T, S4, S5
- Understand what canonicity means and why it matters

## 5.1 Normal Modal Logics

**Definition 5.1** (Normal Modal Logic)**.** A *normal modal logic* is a set $L$ of formulas containing:

- All propositional tautologies
- The **K axiom**: $\Box(p \to q) \to (\Box p \to \Box q)$

and closed under:

- Modus ponens: from $\varphi$ and $\varphi \to \psi$, derive $\psi$
- Necessitation: from $\varphi$, derive $\Box\varphi$

**Example 5.2.**

- **K** = minimal normal modal logic
- **T** = K + $\Box p \to p$ (reflexivity)
- **S4** = T + $\Box p \to \Box\Box p$ (transitivity)
- **S5** = S4 + $p \to \Box\Diamond p$ (symmetry)

## 5.2 Maximal Consistent Sets

**Definition 5.3** (Maximal Consistent Set)**.** A set $\Gamma$ of formulas is *maximally $L$-consistent* if:

- $\Gamma$ is $L$-consistent (no derivation of $\bot$)
- $\Gamma$ is maximal: for all $\varphi$, either $\varphi \in \Gamma$ or $\neg\varphi \in \Gamma$

**Lemma 5.4** (Lindenbaum)**.** *Every consistent set can be extended to a maximally consistent set.*

*Proof.* Enumerate all formulas. Add each one if consistent, otherwise add its negation. □

## 5.3 The Canonical Model

**Definition 5.5** (Canonical Model)**.** The *canonical model* for logic $L$ is $\mathcal{M}^c_L = (W^c, R^c, V^c)$ where:

- $W^c$ = all maximally $L$-consistent sets
- $\Gamma R^c \Delta$ iff for all $\varphi$: $\Box\varphi \in \Gamma \Rightarrow \varphi \in \Delta$
- $V^c(p) = \{\Gamma : p \in \Gamma\}$

**Lemma 5.6** (Truth Lemma)**.** $\mathcal{M}^c_L, \Gamma \models \varphi$ *iff* $\varphi \in \Gamma$*.*

*Proof.* Induction on $\varphi$. The key case is $\Box\varphi$:

- If $\Box\varphi \in \Gamma$ and $\Gamma R^c \Delta$, then $\varphi \in \Delta$, so by IH $\Delta \models \varphi$.
- If $\Box\varphi \notin \Gamma$, construct $\Delta$ with $\varphi \notin \Delta$ and $\Gamma R^c \Delta$.

□

## 5.4 Completeness Theorem

**Theorem 5.7** (Completeness of K)**.** *If $\varphi$ is valid in all Kripke frames, then $\varphi$ is provable in K.*

*Proof.* Contrapositive. If $\varphi$ is not provable, then $\{\neg\varphi\}$ is consistent. Extend to MCS $\Gamma$. By Truth Lemma, $\mathcal{M}^c_K, \Gamma \models \neg\varphi$, so $\varphi$ is not valid. □

## 5.5 Canonicity

**Definition 5.8** (Canonical Formula)**.** A formula $\varphi$ is *canonical* if: whenever $\varphi \in L$, the canonical frame for $L$ validates $\varphi$.

> ★ **Key Insight**
>
> Canonicity is the bridge between syntax (axioms) and semantics (frame conditions). If an axiom is canonical, adding it to a logic preserves completeness.

**Example 5.9.** The T axiom $\Box p \to p$ is canonical: if $\mathrm{T} \in L$, then $R^c$ is reflexive.

# Chapter 6

# Decidability

◎ Goals of This Chapter

- Understand the finite model property (FMP)
- Master the filtration technique
- Prove decidability of K, T, S4, S5
- Know the complexity classes involved

## 6.1 The Finite Model Property

**Definition 6.1** (Finite Model Property)**.** A logic $L$ has the *finite model property* (FMP) if: every $L$-consistent formula is satisfiable in a *finite* model.

**Theorem 6.2.** *If $L$ is finitely axiomatizable and has FMP, then $L$ is decidable.*

*Proof.* To decide if $\varphi \in L$:

- Search for a proof of $\varphi$ (complete enumeration)
- Search for a finite countermodel to $\varphi$

One must succeed. Both searches are effective. □

## 6.2 Filtration

The key technique for proving FMP.

**Definition 6.3** (Closure)**.** The *closure* $\mathrm{cl}(\varphi)$ is the smallest set containing $\varphi$ and all its subformulas, closed under single negation.

Note: $|\mathrm{cl}(\varphi)| \leq 2 \cdot |\varphi|$.

**Definition 6.4** (Filtration)**.** Let $\mathcal{M} = (W, R, V)$ and $\Sigma = \mathrm{cl}(\varphi)$. Define equivalence:

$$w \equiv_\Sigma v \iff \forall \psi \in \Sigma : (\mathcal{M}, w \models \psi \Leftrightarrow \mathcal{M}, v \models \psi)$$

A *filtration* of $\mathcal{M}$ through $\Sigma$ is $(W/{\equiv_\Sigma}, R', V')$ where:

- $V'(p) = \{[w] : w \in V(p)\}$ for $p \in \Sigma$
- $R'$ satisfies: $wRv \Rightarrow [w]R'[v]$ (min condition)
- $R'$ satisfies: $[w]R'[v] \wedge \Box\psi \in \Sigma \wedge \mathcal{M}, w \models \Box\psi \Rightarrow \mathcal{M}, v \models \psi$ (max condition)

**Theorem 6.5** (Filtration Lemma)**.** *For any filtration $\mathcal{M}'$ of $\mathcal{M}$ through $\Sigma$:*

$$\mathcal{M}, w \models \psi \iff \mathcal{M}', [w] \models \psi \quad \textit{for all } \psi \in \Sigma$$

**Corollary 6.6.** $|W/{\equiv_\Sigma}| \leq 2^{|\Sigma|}$*, so the filtration is finite.*

## 6.3 FMP for Standard Logics

**Theorem 6.7.** *K, T, S4, S5 all have the finite model property.*

**Proof Sketch**

- K: smallest filtration works
- T: take reflexive closure
- S4: take reflexive-transitive closure
- S5: take equivalence closure

## 6.4 Complexity

| **Logic** | **Complexity** |
|---|---|
| K, T, S4 | PSPACE-complete |
| S5 | NP-complete |
| PDL | EXPTIME-complete |

★ **Key Insight**

Modal logic satisfiability is generally PSPACE-complete. This is “just barely” tractable—hard, but not as bad as undecidable.

## Chapter 7

# Correspondence and Sahlqvist's Theorem

**◎ Goals of This Chapter**

- Understand frame correspondence deeply
- Learn what Sahlqvist formulas are
- Appreciate why Sahlqvist's theorem is powerful
- Know that this has been mechanized

## 7.1 The Correspondence Problem

Recall: some modal axioms correspond to first-order frame conditions.

$$\begin{array}{ll}
\Box p \to p & \forall x. Rxx \\
\Box p \to \Box\Box p & \forall xyz. Rxy \wedge Ryz \to Rxz \\
\Diamond p \to \Box\Diamond p & \forall xy. Rxy \to \forall z. Rxz \to Ryz
\end{array}$$

**Question**: Which modal formulas correspond to first-order conditions?

**Answer**: Not all! The Löb axiom $\Box(\Box p \to p) \to \Box p$ corresponds to a *second-order* condition (well-foundedness).

## 7.2 Sahlqvist Formulas

**Definition 7.1** (Sahlqvist Formula (simplified))**.** A *Sahlqvist formula* has the form:

$$\varphi \to \psi$$

where:

- $\varphi$ is *positive* in $\Box$ and built from boxed atoms ($\Box^n p$), negative formulas, and $\wedge, \vee$
- $\psi$ is *positive* (no negations in front of atoms)

**Example 7.2.** All the standard axioms are Sahlqvist:

- $\Box p \to p$ (T)
- $\Box p \to \Box\Box p$ (4)
- $p \to \Box\Diamond p$ (B)
- $\Diamond p \to \Box\Diamond p$ (5)

## 7.3 Sahlqvist's Theorem

**Theorem 7.3** (Sahlqvist 1975)**.** *Every Sahlqvist formula:*

1. *Corresponds to a first-order frame condition (computable from the formula)*
2. *Is canonical (hence its logic is complete)*

**Key Insight**

Sahlqvist's theorem is a "cookbook" for modal logic:

1. Write down your axiom (if Sahlqvist)
2. Automatically get the frame condition
3. Automatically get completeness

No need to construct canonical models by hand!

## 7.4 The Algorithm (SQEMA)

There's an algorithm called **SQEMA** (Sahlqvist-van Benthem algorithm) that:

- Takes a modal formula as input
- Outputs the corresponding first-order condition (if it exists)

The algorithm works by:

1. Translate to first-order logic with the standard translation
2. Apply Ackermann's lemma to eliminate second-order quantifiers
3. If successful, output the first-order result

## 7.5 Mechanization

**Historical Note**

Sahlqvist's theorem has been formalized in proof assistants:

- Isabelle/HOL: formalized by various authors
- Lean: partial formalizations exist

This is one of the benchmark results for modal logic mechanization.

**Exercise**

1. Verify that the McKinsey axiom $\Box\Diamond p \to \Diamond\Box p$ is *not* Sahlqvist. (Hint: it's not canonical for the expected frame class.)
2. Use the standard translation to find the frame condition for $\Box(p \lor q) \to \Box p \lor \Box q$. Is this formula valid on all frames?

# Chapter 8

# Graded Modal Logic

◎ Goals of This Chapter

- Go beyond $\forall/\exists$: counting quantifiers
- Understand graded modalities: $\Diamond_{\geq n}$, $\Diamond_{\leq n}$
- See majority and probabilistic quantifiers
- Appreciate the trade-offs (expressiveness vs decidability)

## 8.1 The Limitation of $\Box/\Diamond$

Standard modal logic has:

- $\Box\varphi$ = "all accessible worlds satisfy $\varphi$" ($\forall$)
- $\Diamond\varphi$ = "some accessible world satisfies $\varphi$" ($\exists$)

But natural reasoning uses many more quantifiers:

- "*Most* options are safe"
- "*At least 3* escape routes exist"
- "*Exactly one* successor state"
- "With *probability* $\geq 0.9$, success"

⚠ Common Pitfall

"At least 3 safe options" is very different from "some safe option." Standard modal logic can't distinguish them.

## 8.2 Graded Modalities

**Definition 8.1** (Graded Modal Logic)**.** Extend the language with counting modalities:

$$\begin{array}{ll} \Diamond_{\geq n}\varphi & \text{“at least } n \text{ accessible worlds satisfy } \varphi\text{”} \\ \Diamond_{\leq n}\varphi & \text{“at most } n \text{ accessible worlds satisfy } \varphi\text{”} \\ \Diamond_{= n}\varphi & \text{“exactly } n \text{ accessible worlds satisfy } \varphi\text{”} \end{array}$$

**Definition 8.2** (Semantics)**.**

$$\mathcal{M}, w \models \Diamond_{\geq n}\varphi \iff |\{v : wRv \text{ and } \mathcal{M}, v \models \varphi\}| \geq n$$

Note: $\Diamond\varphi \equiv \Diamond_{\geq 1}\varphi$ and $\Box\varphi \equiv \Diamond_{\leq 0}\neg\varphi$.

## 8.3 Majority Quantifier

**Definition 8.3** (Majority Modality)**.**

$$\mathsf{M}\varphi \quad \text{“most accessible worlds satisfy } \varphi\text{”}$$

Semantics:

$$\mathcal{M}, w \models \mathsf{M}\varphi \iff |\{v : wRv \land v \models \varphi\}| > |\{v : wRv \land v \not\models \varphi\}|$$

**Key Insight**

The majority quantifier $\mathsf{M}$ cannot be defined using $\forall/\exists$ (Lindström). It’s genuinely more expressive.

## 8.4 Probabilistic Modality

**Definition 8.4** (Probabilistic Modal Logic)**.** Given a probability distribution $\mu_w$ over successors of $w$:

$$\mathcal{M}, w \models P_{\geq r}\varphi \iff \mu_w(\{v : wRv \land v \models \varphi\}) \geq r$$

This requires more structure: not just accessibility $R$, but a probability measure.

## 8.5 Meta-Theoretic Trade-offs

Adding quantifiers has costs:

| **Property** | **Basic Modal** | **Graded/Majority** |
|---|---|---|
| Compactness | ✓ | Often fails |
| Finite model property | ✓ | Often fails |
| Decidability | PSPACE | Higher complexity |

**Theorem 8.5** (Lindström)**.** *First-order logic is the unique logic with compactness and Löwenheim-Skolem. Adding quantifiers breaks at least one.*

## 8.6 Connection to Description Logic

Description logics (used in OWL, knowledge graphs) have number restrictions:

$$\begin{aligned} (\geq 3\ \mathsf{hasChild.Doctor}) &\quad \text{“at least 3 children are doctors”} \\ (\leq 1\ \mathsf{hasSpouse}) &\quad \text{“at most 1 spouse”} \end{aligned}$$

These are graded modalities in disguise!

## 8.7 Why This Matters

For AI agents:

- “Ensure at least 2 fallback options” (robustness)
- “Most paths lead to success” (probabilistic planning)
- “Exactly one next state” (determinism)

Standard modal logic can’t express these. Graded modal logic can.

# Chapter 9

# Dynamic Logic

◎ Goals of This Chapter

- Understand PDL: modal logic for programs
- See how actions become modalities
- Appreciate the connection to program verification
- Connect to coalgebra (labeled transition systems)

## 9.1 Programs as Modalities

So far: $\Box\varphi$ means "in all accessible worlds, $\varphi$."

New idea: what if accessibility is *labeled by programs*?

**Definition 9.1** (PDL Syntax)**.** Programs $\alpha$ and formulas $\varphi$ are mutually defined:

$$\alpha ::= a \mid \alpha;\beta \mid \alpha \cup \beta \mid \alpha^* \mid \varphi?$$
$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid [\alpha]\varphi$$

where $a$ is an atomic action.

| **Program** | **Meaning** |
|---|---|
| $a$ | atomic action |
| $\alpha;\beta$ | do $\alpha$, then $\beta$ |
| $\alpha \cup \beta$ | choose $\alpha$ or $\beta$ (nondeterministic) |
| $\alpha^*$ | repeat $\alpha$ zero or more times |
| $\varphi?$ | test: proceed if $\varphi$, else fail |

## 9.2 Semantics

**Definition 9.2** (PDL Model)**.** A PDL model is $(W, \{R_a\}_{a\in\text{Act}}, V)$ where each atomic action $a$ has its own accessibility relation $R_a$.

Extend to compound programs:

$$\begin{aligned} R_{\alpha;\beta} &= R_\alpha \circ R_\beta \quad \text{(composition)} \\ R_{\alpha\cup\beta} &= R_\alpha \cup R_\beta \quad \text{(union)} \\ R_{\alpha^*} &= R_\alpha^* \quad \text{(reflexive-transitive closure)} \\ R_{\varphi?} &= \{(w,w) : w \models \varphi\} \quad \text{(identity on } \varphi\text{-states)} \end{aligned}$$

**Definition 9.3** (Satisfaction)**.**

$$\mathcal{M}, w \models [\alpha]\varphi \iff \forall v : (w,v) \in R_\alpha \Rightarrow \mathcal{M}, v \models \varphi$$

“After any execution of $\alpha$, $\varphi$ holds.”

Dually: $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$ means “$\alpha$ can lead to a state where $\varphi$.”

## 9.3 Examples

**Example 9.4** (Program Correctness)**.**

$$\mathsf{pre} \to [\mathsf{program}]\mathsf{post}$$

“If precondition holds, then after running the program, postcondition holds.”

This is a Hoare triple $\{\mathsf{pre}\}\mathsf{program}\{\mathsf{post}\}$ expressed in modal logic!

**Example 9.5** (Loop Invariant)**.**

$$\mathsf{inv} \to [(\mathsf{cond}?;\mathsf{body})^*](\neg\mathsf{cond} \to \mathsf{inv})$$

“If invariant holds, after any number of loop iterations, when loop exits, invariant still holds.”

**Example 9.6** (Agent Planning)**.**

$$\mathsf{atHome} \to \langle\mathsf{drive};\mathsf{park}\rangle\mathsf{atWork}$$

“From home, there exists a way (drive then park) to reach work.”

## 9.4 PDL and Automata

Programs in PDL correspond to regular expressions:

- ; = concatenation
- $\cup$ = union
- $*$ = Kleene star

**Theorem 9.7.** *PDL model checking is in P. PDL satisfiability is EXPTIME-complete.*

## 9.5 Connection to Coalgebra

A PDL model is a *labeled transition system*:

$$\gamma : W \to \prod_{a \in \mathrm{Act}} \mathcal{P}(W)$$

This is a coalgebra for the functor $F(X) = (\mathcal{P}(X))^{\mathrm{Act}}$.

★ **Key Insight**

PDL semantics is coalgebraic. Programs are built from atomic actions using regular operations. This connects modal logic of programs to automata theory.

## 9.6 Extensions

- **Game Logic**: players alternate, $[\alpha^d]\varphi$ = "player can ensure $\varphi$"
- **Concurrent PDL**: parallel composition $\alpha \| \beta$
- **$\mu$-calculus**: add fixed points, subsumes PDL and CTL

# Chapter 10

# Automata and the Chomsky Hierarchy

◎ Goals of This Chapter

- Understand automata as another way to describe “behavior”
- See the Chomsky hierarchy: what can be recognized by what machine
- Prepare for the unification: Kripke frames and automata are both coalgebras

## 10.1 Finite Automata

**Definition 10.1** (Deterministic Finite Automaton)**.** A *DFA* is a tuple $(Q, \Sigma, \delta, q_0, F)$ where:

- $Q$ is a finite set of states
- $\Sigma$ is a finite alphabet
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the initial state
- $F \subseteq Q$ is the set of accepting states

Intuition

A DFA is a machine that reads a string symbol by symbol, transitioning between states. At the end, it accepts or rejects based on whether it's in an accepting state.

**Definition 10.2** (Non-deterministic Finite Automaton)**.** An *NFA* allows multiple transitions: $\delta : Q \times \Sigma \to \mathcal{P}(Q)$.

**Theorem 10.3.** *DFA and NFA recognize the same class of languages: the* regular languages*.*

## 10.2 The Chomsky Hierarchy

| **Type** | **Language Class** | **Machine** | **Example** |
|---|---|---|---|
| Type 3 | Regular | Finite automaton | $a^*b^*$ |
| Type 2 | Context-free | Pushdown automaton | $a^nb^n$ |
| Type 1 | Context-sensitive | Linear bounded | $a^nb^nc^n$ |
| Type 0 | Recursively enumerable | Turing machine | Halting problem |

★ Key Insight

Each level adds computational power:

- Finite automaton: finite memory (just the state)
- Pushdown: unbounded stack
- Linear bounded: tape proportional to input
- Turing: unbounded tape

## 10.3 Automata as Transition Systems

Notice the similarity:

| **Structure** | **States** | **Transitions** |
|---|---|---|
| Kripke frame | Worlds $W$ | $R \subseteq W \times W$ |
| DFA | States $Q$ | $\delta : Q \times \Sigma \to Q$ |
| NFA | States $Q$ | $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ |
| Labeled TS | States $S$ | $\to \subseteq S \times \Sigma \times S$ |

They're all "states + transitions." This is not a coincidence.

Intuition

Both Kripke frames and automata describe *systems that can be in states and move between states.* Modal logic asks "what is true here?" Automata theory asks "what strings are accepted?" But the underlying structure is the same.

## 10.4 Looking Ahead

The next chapter introduces **coalgebra**, which unifies all these structures under one framework:

$$\text{state} \to F(\text{state})$$

Different choices of the functor $F$ give Kripke frames, DFAs, NFAs, Markov chains, streams, and more.

**Exercise**

1. Draw a DFA that accepts strings with an even number of $a$'s.
2. Can a finite automaton recognize $\{a^n b^n : n \geq 0\}$? Why or why not?
3. What's the relationship between a Kripke frame and an NFA without accepting states?

# Chapter 11

# Beyond Kripke: Coalgebra

◎ Goals of This Chapter

- See that Kripke frames are just one instance of a general pattern
- Understand coalgebra as the mathematics of state-based systems
- Appreciate the unification: automata, Markov chains, streams

## 11.1 The Pattern

Look at these structures:

| **Structure** | **Transition** |
|---|---|
| Kripke frame | $W \to \mathcal{P}(W)$ |
| Deterministic automaton | $Q \to 2 \times Q^A$ |
| Markov chain | $S \to \mathcal{D}(S)$ |
| Stream | $S \to A \times S$ |

They all have the form: **state** → **F(state)** for some functor $F$.

Intuition

A coalgebra answers: "What can I observe from this state, and what states can I reach next?" It's the mathematics of *behavior*.

## 11.2 Coalgebras

**Definition 11.1** (Coalgebra)**.** Let $F : \mathbf{Set} \to \mathbf{Set}$ be a functor. An $F$-*coalgebra* is a pair $(X, \gamma)$ where:

- $X$ is a set of *states*
- $\gamma : X \to F(X)$ is the *transition structure*

**Example 11.2** (Kripke Frame as Coalgebra)**.** A Kripke frame $(W, R)$ is a $\mathcal{P}$-coalgebra: $\gamma(w) = \{v : wRv\}$.

## 11.3 Final Coalgebra

**Definition 11.3** (Final Coalgebra)**.** A *final* $F$-coalgebra $(\Omega, \omega)$ is one where for any $(X, \gamma)$, there exists a unique morphism $X \to \Omega$.

★ **Key Insight**

The final coalgebra is the "universe of all possible behaviors." Two states are equivalent iff they map to the same element in $\Omega$.

# Chapter 12

# How Much Can We Express?

◎ Goals of This Chapter

- Understand expressiveness as a measure of logical power
- Compare modal logic to first-order logic
- Introduce correspondence theory

## 12.1 The Standard Translation

**Definition 12.1** (Standard Translation)**.** Define $\mathrm{ST}_x : \mathcal{L}(\Box) \to \mathcal{L}_{\mathrm{FO}}$:

$$\begin{aligned} \mathrm{ST}_x(p) &= P(x) \\ \mathrm{ST}_x(\neg\varphi) &= \neg\mathrm{ST}_x(\varphi) \\ \mathrm{ST}_x(\Box\varphi) &= \forall y(R(x,y) \to \mathrm{ST}_y(\varphi)) \end{aligned}$$

Modal logic is a *fragment* of first-order logic. But which fragment?

## 12.2 The van Benthem Characterization

**Theorem 12.2** (van Benthem)**.** *A first-order formula $\varphi(x)$ is equivalent to a modal formula iff it is invariant under bisimulation.*

★ Key Insight

Modal logic = bisimulation-invariant fragment of first-order logic.

## 12.3 Frame Correspondence

Some modal formulas correspond to first-order conditions on frames:

| **Modal Axiom** | **Frame Condition** |
|---|---|
| $\Box p \to p$ | Reflexive |
| $\Box p \to \Box\Box p$ | Transitive |
| $p \to \Box\Diamond p$ | Symmetric |

# Part II

# Learning

# Chapter 13

# What is Learning?

◎ Goals of This Chapter

- Ask: what does it mean to "learn"?
- Survey mathematical notions of "getting closer"
- Discover that most aren't operational
- Motivate our choice: gradient descent + semiring

## 13.1 The Intuition

Learning feels like **getting closer** to something.

You start far from the answer. You try something. You see how well it works. You adjust. You try again. Gradually, you approach the target.

Intuition

A child learning to throw a ball:

1. Throw. Miss.
2. Adjust. Throw. Closer.
3. Adjust. Throw. Hit!

Each attempt is "closer" to the goal. Learning is the process of closing the gap.

Can we make this precise? What mathematical structures capture "approaching"?

## 13.2 Mathematical Notions of Approximation

### 13.2.1 Topology

Topology defines “closeness” without distance.

- A sequence $x_n \to x$ means: every neighborhood of $x$ eventually contains all $x_n$
- Continuous functions preserve convergence

**Problem**: Topology *describes* convergence but doesn’t tell you *how* to get there. No algorithm.

### 13.2.2 Metric Spaces

Add a distance function $d(x, y)$.

- “Closer” means smaller $d$
- Cauchy sequences, completeness

**Problem**: Having a metric doesn’t give you a method to minimize it.

### 13.2.3 Order / Domain Theory

An information ordering: $x \sqsubseteq y$ means “$y$ is more defined than $x$.”

- Start with $\bot$ (no information)
- Iteratively refine: $\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \cdots$
- Reach a fixed point

**Problem**: Requires the structure to be a dcpo with finite chains, or continuous functions. Doesn’t apply to arbitrary parameter spaces.

### 13.2.4 Galois Connections

A pair of “best approximations” between two levels of abstraction.

$$f(x) \leq y \iff x \leq g(y)$$

Used in abstract interpretation, type systems.

**Problem**: Describes the *relationship* between approximations, not how to compute them.

### 13.2.5 Game-Theoretic

Minimax, Nash equilibrium.

- Multiple agents adjusting strategies
- Converge to equilibrium (sometimes)

**Problem**: Convergence is not guaranteed. Computationally hard.

### 13.2.6 Probabilistic

PAC learning: "Probably Approximately Correct."

- With high probability, get close to the target
- Concentration inequalities, sample complexity

**Problem**: Gives bounds, not algorithms. Still need a method to actually learn.

## 13.3 The Operational Question

⚠ Common Pitfall

Most notions of approximation are **descriptive**, not **operational**. They tell you what "getting closer" means, but not how to do it.

What we need:

1. A **space** to search in
2. A **measure** of how close we are (loss)
3. An **algorithm** that reduces the loss

## 13.4 The Problem of Semantic Convergence

There's a deeper issue with approaches that require "understanding" the structure.

In analysis, when we say a sequence $x_n \to x$, we have *tools to prove* this:

- Epsilon-delta definitions
- Cauchy criterion
- Completeness of $\mathbb{R}$

The target $x$ **exists**, and we can **prove** our approximations converge to it.

⚠ Common Pitfall

But what if we're trying to "approximate" a *meaning*?
When we build a model to capture some semantic concept—"justice," "intention," "belief"—what guarantees that our formalization converges to anything real?

This is Wittgenstein's challenge. He famously wrote:

> "Philosophy is a battle against the **bewitchment** of our intelligence by means of language."

The bewitchment: language makes us *feel* like we're referring to something. We say "meaning," "understanding," "truth," and we imagine there must be *things* these words point to—things we can approximate, converge to, capture.

But what if there's nothing there? What if the question "what is the meaning?" is itself a grammatical illusion—a question that *looks* well-formed but has no answer?

Intuition

In analysis: we prove $x_n \to x$ using the structure of $\mathbb{R}$. The limit *exists*.
In semantics: we want "model → meaning." But maybe there is no "meaning" sitting there, waiting to be approximated. The target itself may be a mirage created by language.

The domain-theoretic approach is vulnerable to this bewitchment. To set up the dcpo, to define the continuous function, you must already *believe* there's a well-defined target. You formalize your intuition about "what learning should converge to." But you can't prove your intuition refers to anything real.

★ Key Insight

Shannon's information theory succeeds by *refusing to engage* with meaning.
He doesn't ask: "What does this message mean?"
He asks: "How many bits to transmit it?"
The semantic question has no tools. The syntactic question does.

Gradient descent makes the same move:

| Approach | Question | Tools? |
|---|---|---|
| Semantic | "Does my model capture the concept?" | No |
| Operational | "Does my loss decrease?" | Yes |

We don't ask whether our neural network "understands" language. We ask whether its loss on the next-token prediction task goes down. The first question is philosophically intractable. The second is measurable.

This is why we choose gradient descent: not because it's philosophically satisfying, but because it's the only game in town with actual convergence guarantees—at the operational level, not the semantic level.

## 13.5 What Actually Works

Surprisingly few methods are both *principled* and *operational*:

| Method | Requirements | Scalable? |
|---|---|---|
| Gradient descent | Differentiable loss | Yes |
| Fixed point iteration | Monotone + chain condition | Sometimes |
| Constraint propagation | Discrete + local | Sometimes |
| LP relaxation | Linear/convex | Yes |
| MCMC / sampling | Probabilistic | Slow |
| Evolutionary | Fitness function | Slow |

★ Key Insight

**Gradient descent** dominates because it is:

- Mathematically principled (follows steepest descent)
- Computationally tractable (autodiff scales)
- General (works for any differentiable function)

## 13.6 Our Path

For learning *logical structures* (Kripke frames, automata, coalgebras):

1. The structures are **discrete** — can't directly use gradients
2. We need to **embed** them into a continuous space
3. The embedding should preserve logical meaning
4. After learning, we **extract** back to discrete

This is the **semiring relaxation**:

$$\text{Discrete structure} \xrightarrow{\text{embed}} \text{Semiring-valued} \xrightarrow{\text{gradient descent}} \text{Trained} \xrightarrow{\text{threshold}} \text{Discrete}$$

## 13.7 Why Semiring?

Why not just use $[0,1]$ (fuzzy) and be done?

Because we want:

- **Generality**: Different tasks need different interpretations
- **Compositionality**: Logical connectives should compose correctly
- **Gradient flow**: Operations should be differentiable

Semirings provide all three. They are the *minimal algebraic structure* that supports continuous relaxation of logic.

**★ Key Insight**

We arrive at semirings not by abstract preference, but by elimination:

- Need operational approximation → gradient descent
- Structures are discrete → need continuous embedding
- Want compositionality → need algebraic structure
- Semiring = the natural answer

## 13.8 What's Next

The rest of Part II develops this idea:

1. Gradient descent: the core algorithm
2. Continuization: embedding discrete into continuous
3. Fuzzy and probabilistic: concrete examples
4. Semiring: the general framework
5. Modal operators: how to handle $\Box/\Diamond$
6. The learning loop: putting it all together

# Chapter 14

# Gradient Descent

◎ Goals of This Chapter

- Review calculus: derivatives measure sensitivity
- Understand gradient descent as iterative optimization
- See Newton's method and its quadratic convergence
- Appreciate why continuous optimization is powerful

## 14.1 The Optimization Problem

We have a function $f : \mathbb{R}^n \to \mathbb{R}$ and want to find:

$$\theta^* = \arg\min_{\theta} f(\theta)$$

If $f$ is differentiable, the gradient $\nabla f(\theta)$ points "uphill."

Intuition

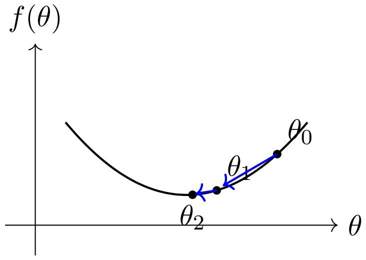
The gradient tells you: "If you want $f$ to increase, go this direction." So to minimize, go the opposite direction.

## 14.2 Gradient Descent

**Definition 14.1** (Gradient Descent)**.** Starting from $\theta_0$, iterate:

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t)$$

where $\eta > 0$ is the *learning rate.*



## 14.3 Convergence

**Theorem 14.2** (Convergence for Convex Functions)**.** *If $f$ is convex and $L$-smooth (bounded second derivatives), gradient descent with $\eta = 1/L$ satisfies:*

$$f(\theta_t) - f(\theta^*) \leq \frac{\|\theta_0 - \theta^*\|^2}{2t/L}$$

Rate: $O(1/t)$. Slow, but guaranteed.

## 14.4 Newton's Method

Use second-order information (the Hessian $H = \nabla^2 f$):

$$\theta_{t+1} = \theta_t - H(\theta_t)^{-1} \nabla f(\theta_t)$$

**Theorem 14.3** (Quadratic Convergence)**.** *Near a minimum with positive definite Hessian, Newton's method converges quadratically:*

$$\|\theta_{t+1} - \theta^*\| \leq C\|\theta_t - \theta^*\|^2$$

★ **Key Insight**

Newton's method uses curvature information to take better steps. The Hessian "rescales" the gradient to account for how steep different directions are.

## 14.5 The Problem with Discrete Spaces

All of this requires:

- A continuous space $\mathbb{R}^n$
- A differentiable function $f$

But the structures we care about (automata, Kripke frames, programs) are *discrete*.

⚠ **Common Pitfall**

You can't take the gradient of "number of states" or "which transitions exist." Discrete spaces have no derivatives.

**Solution**: Embed discrete structures into continuous spaces. This is the motivation for *semirings*.

Chapter 15

# From Discrete to Continuous

◎ Goals of This Chapter

- Understand why we need to "soften" discrete structures
- See concrete examples: soft attention, soft selection
- Introduce the idea of relaxation and rounding

## 15.1 The Continuization Trick

Discrete optimization is hard. Continuous optimization has gradients.

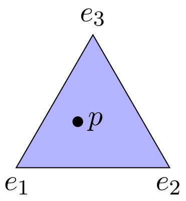
**Strategy**:

1. Embed discrete objects into a continuous space
2. Optimize in the continuous space
3. Round back to discrete

## 15.2 Example: Soft Selection

**Discrete**: Choose one of $n$ items. Represented as one-hot vector $e_i \in \{0, 1\}^n$.

**Continuous**: Use a probability distribution $p \in \Delta^{n-1}$ (the simplex).



The corners are discrete choices. The interior is "soft" mixtures.

## 15.3 Example: Soft Transitions

**Discrete**: Transition relation $R \subseteq W \times W$. Either $(w, v) \in R$ or not.

**Continuous**: Transition weights $R : W \times W \to [0, 1]$.

- $R(w, v) = 1$: definitely connected
- $R(w, v) = 0$: definitely not connected
- $R(w, v) = 0.7$: "70% connected" (soft)

## 15.4 The Softmax Function

To go from unconstrained $\mathbb{R}^n$ to the simplex:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

- Output is always a valid distribution
- Differentiable everywhere
- Temperature $\tau$: $\text{softmax}(z/\tau)$ becomes sharper as $\tau \to 0$

## 15.5 Relaxation and Rounding

**Relaxation**: Replace discrete constraints with continuous ones.

$$x \in \{0, 1\} \quad \rightsquigarrow \quad x \in [0, 1]$$

**Rounding**: After optimization, snap back to discrete.

$$x = 0.7 \quad \rightsquigarrow \quad x = 1$$

⚠ **Common Pitfall**

Rounding can fail! The continuous optimum might round to a bad discrete solution. This is why we need careful design of the relaxation.

## 15.6 What We Need

To make this systematic, we need:

1. A principled way to "soften" logical operations ($\wedge, \vee, \neg$)
2. Compatibility with gradient computation
3. Convergence guarantees (soft → crisp)

This is exactly what **semirings** provide.

# Chapter 16

# Fuzzy Logic

◎ Goals of This Chapter

- See the simplest way to make logic continuous: fuzzy truth values
- Understand fuzzy AND, OR, NOT
- Appreciate what works and what doesn't

## 16.1 Truth as a Degree

Classical logic: true $= 1$, false $= 0$.

Fuzzy logic: truth values in $[0, 1]$.

Intuition

"The water is hot" — is this true or false? Depends on temperature. At 30°C, maybe 0.3 true. At 80°C, maybe 0.95 true. Fuzzy logic captures this gradation.

## 16.2 Fuzzy Connectives

How should $\wedge$, $\vee$, $\neg$ work on $[0, 1]$?

**Negation**:

$$\neg x = 1 - x$$

If $x$ is 0.7 true, then $\neg x$ is 0.3 true.

**Conjunction** (Gödel):

$$x \wedge y = \min(x, y)$$

"$A$ and $B$" is as true as the least true conjunct.

**Disjunction** (Gödel):

$$x \vee y = \max(x, y)$$

"$A$ or $B$" is as true as the most true disjunct.

## 16.3 Example: Fuzzy Kripke Model

**Definition 16.1** (Fuzzy Kripke Model)**.** A *fuzzy Kripke model* is $(W, R, V)$ where:

- $R : W \times W \to [0, 1]$ (fuzzy accessibility)
- $V : \Phi \to (W \to [0, 1])$ (fuzzy valuation)

Satisfaction becomes a degree:

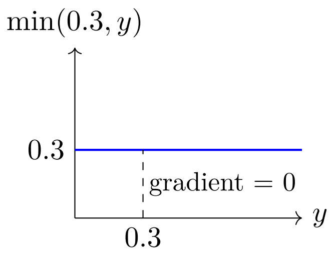
$$\begin{aligned} [\![p]\!]_w &= V(p)(w) \\ [\![\neg\varphi]\!]_w &= 1 - [\![\varphi]\!]_w \\ [\![\varphi \wedge \psi]\!]_w &= \min([\![\varphi]\!]_w, [\![\psi]\!]_w) \\ [\![\Box\varphi]\!]_w &= \min_{v \in W} (R(w, v) \Rightarrow [\![\varphi]\!]_v) \end{aligned}$$

where $a \Rightarrow b = \min(1, 1 - a + b)$ (Łukasiewicz implication) or other choices.

## 16.4 Problems with Min/Max

⚠ Common Pitfall

Min/max have zero gradients almost everywhere!
If $x = 0.3$ and $y = 0.7$, then $\min(x, y) = x$. The gradient w.r.t. $y$ is zero. Optimization gets stuck.



## 16.5 Alternative: Product Logic

Instead of min/max, use multiplication:

$$\begin{aligned} x \wedge y &= x \cdot y \\ x \vee y &= x + y - x \cdot y \end{aligned}$$

Now gradients are non-zero:

$$\frac{\partial(x \cdot y)}{\partial y} = x \neq 0$$

But this changes the algebraic properties...

## 16.6 The Pattern

We have two “fuzzy logics”:

| | **Gödel** | **Product** |
|---|---|---|
| AND | min | × |
| OR | max | + (clamped) |

They’re both trying to do the same thing: extend Boolean logic to $[0, 1]$. Is there a general framework that includes both?

# Chapter 17

# Probabilistic Reasoning

◎ Goals of This Chapter

- See probability as another way to handle uncertainty
- Understand how AND and OR work probabilistically
- Compare with fuzzy logic
- Notice the algebraic pattern

## 17.1 Probability vs Fuzziness

Fuzzy: “The water is 0.7 hot” (degree of truth)

Probabilistic: “There’s 0.7 probability the water is hot” (uncertainty about a crisp fact)

Different interpretations, but similar math.

## 17.2 Probabilistic Connectives

For independent events $A$, $B$ with probabilities $p$, $q$:

**Conjunction**:

$$P(A \wedge B) = P(A) \cdot P(B) = p \cdot q$$

**Disjunction** (inclusive):

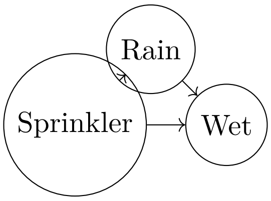
$$P(A \vee B) = P(A) + P(B) - P(A \wedge B) = p + q - pq$$

**Negation**:

$$P(\neg A) = 1 - P(A) = 1 - p$$

## 17.3 Example: Probabilistic Inference

Consider a simple Bayesian network:



Computing $P(\text{Wet})$ involves:

- Multiplying probabilities along paths (AND)
- Adding probabilities across paths (OR)

## 17.4 Semiring Structure Appears

Look at the operations:

| | **Fuzzy (Gödel)** | **Probabilistic** |
|---|---|---|
| AND ($\otimes$) | min | $\times$ |
| OR ($\oplus$) | max | $+$ |
| False ($\mathbf{0}$) | 0 | 0 |
| True ($\mathbf{1}$) | 1 | 1 |

Both satisfy:

- $\oplus$ is associative, commutative, has identity $\mathbf{0}$
- $\otimes$ is associative, commutative, has identity $\mathbf{1}$
- $\otimes$ distributes over $\oplus$
- $\mathbf{0} \otimes x = \mathbf{0}$

★ **Key Insight**

This is the structure of a **semiring**! Both fuzzy logic and probabilistic logic are instances of the same algebraic pattern.

## 17.5 Why This Matters

Once we recognize the pattern:

- We can write algorithms that work for *any* semiring
- Switching from fuzzy to probabilistic = changing the semiring
- New semirings = new kinds of reasoning

## 17.6 More Examples

| **Name** | $\oplus$ | $\otimes$ | **Use** |
|---|---|---|---|
| Boolean | $\vee$ | $\wedge$ | Classical logic |
| Fuzzy | max | min | Soft constraints |
| Probabilistic | $+$ | $\times$ | Uncertainty |
| Tropical | min | $+$ | Shortest paths |

The tropical semiring is especially interesting: "OR = take minimum", "AND = add costs". Finding a satisfying assignment becomes finding the shortest path!

Next: the general definition of semiring.

# Chapter 18

# Semiring: The Abstraction

◎ Goals of This Chapter

- See the common structure behind fuzzy, probabilistic, tropical
- Understand the formal definition of semiring
- Appreciate why this abstraction is useful

## 18.1 The Pattern We've Seen

Recall:

| **Name** | $S$ | $\oplus$ (OR) | $\otimes$ (AND) | $\mathbf{0}$ | $\mathbf{1}$ |
|---|---|---|---|---|---|
| Boolean | $\{0,1\}$ | $\vee$ | $\wedge$ | 0 | 1 |
| Fuzzy (Gödel) | $[0,1]$ | max | min | 0 | 1 |
| Probabilistic | $\mathbb{R}_{\geq 0}$ | $+$ | $\times$ | 0 | 1 |

All satisfy the same algebraic laws. Let's name this structure.

## 18.2 Definition

**Definition 18.1** (Semiring)**.** A *semiring* is a tuple $(S, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where:

1. $(S, \oplus, \mathbf{0})$ is a commutative monoid:
   - $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
   - $a \oplus b = b \oplus a$
   - $a \oplus \mathbf{0} = a$
2. $(S, \otimes, \mathbf{1})$ is a monoid:
   - $a \otimes (b \otimes c) = (a \otimes b) \otimes c$

- $a \otimes \mathbf{1} = \mathbf{1} \otimes a = a$

3. $\otimes$ distributes over $\oplus$:
    - $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
    - $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
4. $\mathbf{0}$ annihilates:
    - $a \otimes \mathbf{0} = \mathbf{0} \otimes a = \mathbf{0}$

**Intuition**

$\oplus$ = "or" / "choice" / "combine alternatives"

$\otimes$ = "and" / "sequence" / "combine steps"

$\mathbf{0}$ = "impossible" / "failure"

$\mathbf{1}$ = "trivially true" / "do nothing"

## 18.3 More Examples

**Definition 18.2** (Tropical Semiring)**.** $(\mathbb{R} \cup \{+\infty\}, \min, +, +\infty, 0)$

- $a \oplus b = \min(a, b)$: choose the better option
- $a \otimes b = a + b$: accumulate costs
- $\mathbf{0} = +\infty$: infinite cost = impossible
- $\mathbf{1} = 0$: zero cost = free

**Key Insight**

In the tropical semiring, logical inference becomes optimization! "Find a satisfying assignment" becomes "find the minimum-cost path."

**Definition 18.3** (Viterbi Semiring)**.** $([0, 1], \max, \times, 0, 1)$
Used in HMMs: find the most probable path.

**Definition 18.4** (Log Semiring)**.** $(\mathbb{R} \cup \{-\infty\}, \mathrm{logsumexp}, +, -\infty, 0)$
Numerically stable version of probabilistic semiring.

## 18.4 Why Semirings?

1. **Genericity**: Write algorithm once, instantiate for different semirings
2. **Correctness**: Algebraic laws guarantee properties
3. **Modularity**: Change interpretation without changing structure

**Example 18.5.** The same dynamic programming algorithm:

- Boolean semiring → reachability (is there a path?)
- Tropical semiring → shortest path (what's the minimum cost?)
- Probabilistic semiring → most probable path
- Counting semiring $(\mathbb{N}, +, \times, 0, 1)$ → count paths

# Chapter 19

# The Gradient Semiring

◎ Goals of This Chapter

- Understand automatic differentiation algebraically
- See how gradients flow through semiring operations
- Appreciate: inference and learning in one pass

## 19.1 The Problem

We want to optimize over semiring computations.

Given a semiring expression (e.g., probabilistic inference), we need gradients w.r.t. parameters.

Naïve approach: compute forward, then backpropagate.

Better approach: **embed gradients into the semiring itself**.

## 19.2 Dual Numbers

**Definition 19.1** (Dual Numbers)**.** A *dual number* is $a + b\epsilon$ where $\epsilon^2 = 0$.

Arithmetic:

$$\begin{aligned}(a + b\epsilon) + (c + d\epsilon) &= (a + c) + (b + d)\epsilon \\ (a + b\epsilon) \times (c + d\epsilon) &= ac + (ad + bc)\epsilon\end{aligned}$$

★ Key Insight

If $f(x) = a + b\epsilon$ when we plug in $x = x_0 + 1 \cdot \epsilon$, then:

- $a = f(x_0)$ (the value)
- $b = f'(x_0)$ (the derivative!)

## 19.3 The Gradient Semiring

**Definition 19.2** (Gradient Semiring)**.** Elements: pairs $(v, g)$ where $v =$ value, $g =$ gradient.

Operations:

$$
\begin{aligned}
(v_1, g_1) \oplus (v_2, g_2) &= (v_1 + v_2,\ g_1 + g_2) \\
(v_1, g_1) \otimes (v_2, g_2) &= (v_1 \cdot v_2,\ v_1 \cdot g_2 + v_2 \cdot g_1)
\end{aligned}
$$

Identities: $\mathbf{0} = (0, 0)$, $\mathbf{1} = (1, 0)$.

**Proposition 19.3.** *The gradient semiring is a semiring.*

*Proof.* Check the axioms. The key is that $\otimes$ follows the product rule. □

## 19.4 Forward-Mode Automatic Differentiation

To compute $f(x)$ and $\frac{\partial f}{\partial x}$ simultaneously:

1. Replace $x$ with $(x, 1)$ (value $x$, derivative 1)

2. Compute using gradient semiring operations

3. Extract: first component = value, second = derivative

**Example 19.4.** Compute $f(x) = x^2 + x$ at $x = 3$:

$$
\begin{aligned}
x &= (3, 1) \\
x^2 &= (3, 1) \otimes (3, 1) = (9, 6) \\
x^2 + x &= (9, 6) \oplus (3, 1) = (12, 7)
\end{aligned}
$$

So $f(3) = 12$ and $f'(3) = 7$. Check: $f'(x) = 2x + 1$, so $f'(3) = 7$.

## 19.5 Application: Differentiable Logic

Any semiring computation can be made differentiable:

1. Write your logic in semiring form

2. Instantiate with gradient semiring

3. Get gradients for free

★ **Key Insight**

This is how DeepProbLog works: probabilistic logic programming with gradients. Inference and learning happen in a single forward pass.

## 19.6 Limitations

Forward-mode computes $\frac{\partial f}{\partial x_i}$ for one $x_i$ at a time.

For many parameters, need reverse-mode (backpropagation).

Reverse-mode is also algebraic, but more complex (requires “transposing” the computation).

# Chapter 20

# Semiring-Valued Coalgebra

◎ Goals of This Chapter

- Combine Part I (coalgebra) with Part II (semiring)
- See how to "soften" any coalgebra
- Understand the learning setup

## 20.1 Recall: Coalgebra

From Part I: an $F$-coalgebra is $(X, \gamma : X \to F(X))$.

Examples:

- Kripke frame: $X \to \mathcal{P}(X)$
- DFA: $X \to 2 \times X^A$
- Stream: $X \to A \times X$

These are all *crisp*: transitions either exist or don't.

## 20.2 Semiring-Valued Transitions

**Definition 20.1** (Semiring-Valued Coalgebra)**.** Fix a semiring $S$. An *$S$-valued $F$-coalgebra* replaces sets with $S$-valued functions.

For $F = \mathcal{P}$ (Kripke-like):

$$\gamma : X \to (X \to S)$$

i.e., $\gamma : X \times X \to S$. Each transition has a weight in $S$.

**Example 20.2** (Fuzzy Kripke Frame)**.** $S = [0, 1]$. Then $\gamma(w, v) = 0.7$ means "70% connected."

**Example 20.3** (Probabilistic Automaton)**.** $S = \mathbb{R}_{\geq 0}$. Then $\gamma(q, a, q')$ is the probability of transitioning from $q$ to $q'$ on input $a$.

## 20.3 Parameterized Coalgebras

For learning, we parameterize the coalgebra:

**Definition 20.4.** A *parameterized $S$-coalgebra* is a family $\gamma_\theta$ indexed by $\theta \in \Theta$.

Typically: $\theta$ are real numbers, $\gamma_\theta$ uses softmax or sigmoid to produce $S$-values.

**Example 20.5.** For a fuzzy DFA with $n$ states and alphabet $A$:

$$\theta \in \mathbb{R}^{n \times |A| \times n}$$

$$\gamma_\theta(q, a, q') = \mathrm{softmax}_j(\theta_{q,a,j})_{q'}$$

## 20.4 Semantics in Semirings

Modal formulas can be evaluated in semiring-valued models:

$$\begin{aligned}
[\![p]\!]_w &\in S \\
[\![\varphi \wedge \psi]\!]_w &= [\![\varphi]\!]_w \otimes [\![\psi]\!]_w \\
[\![\varphi \vee \psi]\!]_w &= [\![\varphi]\!]_w \oplus [\![\psi]\!]_w \\
[\![\Box\varphi]\!]_w &= \bigotimes_v \gamma(w, v) \Rightarrow_S [\![\varphi]\!]_v
\end{aligned}$$

where $\Rightarrow_S$ is an $S$-valued implication (semiring-dependent).

## 20.5 The Bridge

$$\begin{tikzcd}
\text{Crisp Coalgebra} \arrow[r, leftrightarrow, "\text{soften}", "\text{threshold}"'] \arrow[d, dashed, no head] & S\text{-valued Coalgebra} \arrow[d, dashed, no head] \\
\text{Discrete, not learnable} & \text{Continuous, differentiable}
\end{tikzcd}$$

★ **Key Insight**

Semiring-valued coalgebra is the bridge between the discrete world of Part I and the continuous world of optimization.

# Chapter 21

# Modal Operators in Semirings

◎ Goals of This Chapter

- Face the core question: how to interpret $\Box/\Diamond$ over semirings
- Survey existing approaches: fuzzy, many-valued, residuated lattices
- Understand what's solved and what's open
- Know the implications for learning

## 21.1 The Problem

In classical modal logic:

$$\mathcal{M}, w \models \Box\varphi \iff \forall v : wRv \Rightarrow \mathcal{M}, v \models \varphi$$

This involves:

- $\forall$ — universal quantification (AND over all successors)
- $\Rightarrow$ — implication (if accessible, then...)

In semiring-valued setting:

- $R(w, v) \in S$ — soft accessibility
- $[\![\varphi]\!]_v \in S$ — soft truth value

**Question**: What is $[\![\Box\varphi]\!]_w$?

## 21.2 Existing Work: Three Traditions

This is *not* a completely open problem. There's substantial literature.

### 21.2.1 Modal Semirings (Algebraic)

Möller, Desharnais, and others developed *modal semirings* as an algebraic axiomatization.

- Define *domain* operation: $\mathrm{dom}(x) = |x\rangle 1$
- Define box via de Morgan: $|x]p = \neg|x\rangle\neg p$
- Focus on algebraic laws, not many-valued truth

Key reference: "Some Uses of Modal Semirings" (Möller & Desharnais, WADT 2024).

This approach is about *algebraic structure*, not about truth values in $[0, 1]$.

### 21.2.2 Many-Valued Modal Logic (Lattice-Valued)

Bou, Esteva, Godo, and others study modal logic over *residuated lattices.*

**Definition 21.1** (Lattice-Valued Kripke Model)**.** Over a residuated lattice $A = \langle A, 0, 1, \wedge, \vee, \otimes, \to\rangle$:

- $R : W \times W \to A$ (many-valued accessibility)
- $e : \mathrm{Var} \times W \to A$ (many-valued valuation)

The semantics of $\Box$:

$$[\![\Box\varphi]\!]_w = \bigwedge_v (R(w, v) \to [\![\varphi]\!]_v)$$

where $\to$ is the residual of $\otimes$.

Key reference: "On the Minimum Many-Valued Modal Logic over a Finite Residuated Lattice" (Bou et al., 2009).

### 21.2.3 Fuzzy Modal Logic (T-Norm Based)

Fuzzy modal logic uses t-norms and their residuals.

**Definition 21.2** (Fuzzy Modal Semantics)**.** For a t-norm $\otimes$ with residual $\Rightarrow$:

$$[\![\Box\varphi]\!]_w = \inf_v (R(w, v) \Rightarrow [\![\varphi]\!]_v)$$
$$[\![\Diamond\varphi]\!]_w = \sup_v (R(w, v) \otimes [\![\varphi]\!]_v)$$

Common choices:

| **Name** | $a \otimes b$ | $a \Rightarrow b$ |
|---|---|---|
| Gödel | $\min(a, b)$ | $\begin{cases} 1 & a \leq b \\ b & a > b \end{cases}$ |
| Łukasiewicz | $\max(0, a + b - 1)$ | $\min(1, 1 - a + b)$ |
| Product | $a \cdot b$ | $\begin{cases} 1 & a \leq b \\ b/a & a > b \end{cases}$ |

## 21.3 What's Known

**Theorem 21.3** (Failure of Axiom K)**.** *In many-valued modal logic with non-Boolean accessibility, the axiom*

$$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$$

*generally **fails**.*

**Theorem 21.4** (Non-Interdefinability)**.** *In general, $\Diamond\varphi \not\equiv \neg\Box\neg\varphi$ when $R$ is many-valued.*

| **Property** | **Status** |
|---|---|
| Semantics for $\Box/\Diamond$ | Defined (inf/sup with residual) |
| Axiom K | Fails in general |
| $\Box/\Diamond$ duality | Fails in general |
| Completeness | Case-by-case, often unknown |
| Decidability | Case-by-case |

## 21.4 What's Still Open

1. **General semirings**: Most work uses residuated lattices (which have a specific implication $\to$). Arbitrary semirings don't have a canonical implication.

2. **Tropical semiring**: $(\mathbb{R} \cup \{\infty\}, \min, +)$ — what's the right modal semantics? Not well-studied.

3. **Gradient semiring**: What happens when we track derivatives through modal operators? Unexplored.

4. **Best semantics for learning**: Which choice of t-norm/residual leads to best optimization landscape? No one has studied this.

5. **Convergence**: When does soft modal semantics converge to crisp? Conditions unclear.

## 21.5 For Learning: Practical Choices

Given the theory, what should we use for gradient-based learning?

⚠ **Common Pitfall**

inf and sup have zero gradients almost everywhere. Not suitable for learning.

### 21.5.1 Option 1: Soft Inf/Sup

Replace inf with softmin:

$$\mathrm{softmin}_\tau(x_1, \ldots, x_n) = -\tau \log \sum_i e^{-x_i/\tau}$$

As $\tau \to 0$, this approaches true min.

### 21.5.2 Option 2: Product Semantics

Use product t-norm throughout:

$$\begin{aligned} [\![\Box\varphi]\!]_w &= \prod_v (1 - R(w,v) \cdot (1 - [\![\varphi]\!]_v)) \\ [\![\Diamond\varphi]\!]_w &= 1 - \prod_v (1 - R(w,v) \cdot [\![\varphi]\!]_v) \end{aligned}$$

Fully differentiable, gradients everywhere.

### 21.5.3 Option 3: Weighted Average

For $\Box$ as "expected truth over successors":

$$[\![\Box\varphi]\!]_w = \frac{\sum_v R(w,v) \cdot [\![\varphi]\!]_v}{\sum_v R(w,v)}$$

Simple, differentiable, but doesn't match classical semantics.

## 21.6 Recommendation

★ Key Insight

For learning:

1. Use **product t-norm** or **softmin/softmax**
2. Accept that axiom K may fail during training
3. Add regularization to push toward crisp values
4. After convergence, verify classical properties

## 21.7 Summary

| **Aspect** | **Status** | **Reference** |
|---|---|---|
| Algebraic axioms | Solved | Möller et al. |
| Residuated lattice semantics | Solved | Bou et al. |
| Fuzzy/t-norm semantics | Solved | Hájek, Fitting |
| Arbitrary semirings | Partially open | — |
| Differentiable semantics | Open | — |
| Best choice for learning | Open | — |

The mathematics of many-valued modal logic is well-developed. The question of which variant works best for gradient-based learning is new territory.

# Chapter 22

# The Learning Loop

◎ Goals of This Chapter

- Put everything together: the full learning pipeline
- Understand loss, forward pass, backward pass
- See convergence from soft to crisp

## 22.1 The Setup

We have:

- A functor $F$ (what we're learning: DFA, Kripke, etc.)
- A semiring $S$ (how we soften: fuzzy, probabilistic, etc.)
- Parameters $\theta \in \Theta \subseteq \mathbb{R}^d$
- A parameterized $S$-coalgebra $\gamma_\theta$
- Training data $\mathcal{D}$

## 22.2 Loss Function

The loss measures how well $\gamma_\theta$ explains the data.

**Example 22.1** (Automaton Learning)**.** Data: strings labeled accept/reject.

$$L(\theta) = -\sum_{(x,y)\in\mathcal{D}} \log P_\theta(y \mid x)$$

where $P_\theta$ is computed by running the soft automaton on $x$.

**Example 22.2** (Modal Satisfaction)**.** Data: (model, world, formula, truth value) tuples.

$$L(\theta) = \sum_{(w,\varphi,v)} \left( [\![\varphi]\!]_w^\theta - v \right)^2$$

## 22.3 Forward Pass

Compute $[\![\cdot]\!]^\theta$ using semiring operations.

If using gradient semiring: get gradients simultaneously.

## 22.4 Backward Pass

Compute $\nabla_\theta L$ using:

- Gradient semiring (forward-mode), or
- Standard backpropagation (reverse-mode)

## 22.5 Update

$$\theta \leftarrow \theta - \eta \nabla_\theta L$$

Repeat until convergence.

## 22.6 Convergence to Crisp

As training progresses, soft values should approach $\{0, 1\}$.

**Definition 22.3** (Temperature Annealing)**.** Replace softmax with:

$$\text{softmax}_\tau(z)_i = \frac{e^{z_i/\tau}}{\sum_j e^{z_j/\tau}}$$

Gradually decrease $\tau \to 0$. At $\tau = 0$, this is hard argmax.

**Definition 22.4** (Sparsity Regularization)**.** Add to loss:

$$L_{\text{sparse}}(\theta) = \lambda \sum_{i,j} H(\gamma_\theta(i, j))$$

where $H(p) = -p \log p - (1 - p) \log(1 - p)$ is entropy. Pushes toward 0 or 1.

## 22.7 Extraction

After training, extract crisp coalgebra:

$$\gamma_{\text{crisp}}(x, y) = \begin{cases} 1 & \text{if } \gamma_\theta(x, y) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

★ Key Insight

The full pipeline:

$$\text{Data} \xrightarrow{\text{loss}} \text{Soft Coalgebra} \xrightarrow{\text{optimize}} \text{Trained Soft} \xrightarrow{\text{threshold}} \text{Crisp Coalgebra}$$

We learn in continuous space, then extract a discrete program.

## 22.8 Example: Learning a DFA

**Task**: Learn a DFA from positive/negative examples.
**Setup**:

- $n$ states, alphabet $\{0, 1\}$
- $\theta \in \mathbb{R}^{n \times 2 \times n + n}$ (transitions + accepts)
- Soft transitions via softmax
- Soft accepts via sigmoid

**Training**: Standard gradient descent on cross-entropy loss.
**Result**: After convergence, threshold to get a crisp DFA.
This is what we'll implement in the code for Part II.

# Chapter 23

# Differentiable Graded Modalities

◎ **Goals of This Chapter**

- Combine graded quantifiers with differentiable learning
- See how to soften counting modalities
- Understand this as a potential novel contribution

## 23.1 The Gap

From Part I: graded modalities $\Diamond_{\geq n}$, $\mathsf{M}$, $P_{\geq r}$ are expressive.

From Part II: semiring-valued coalgebra enables learning.

But existing work doesn't combine them:

| **Work** | **Learns $R$?** | **Graded?** | **Differentiable?** |
|---|---|---|---|
| MLNNs (2024) | ✓ | × (only $\Box/\Diamond$) | ✓ |
| Graded Modal | × | ✓ | × |
| This chapter | ✓ | ✓ | ✓ |

## 23.2 Softening Graded Modalities

Recall the crisp semantics:

$$\mathcal{M}, w \models \Diamond_{\geq n}\varphi \iff |\{v : wRv \wedge v \models \varphi\}| \geq n$$

This is a hard threshold. Not differentiable.

**Definition 23.1** (Soft Graded Modality)**.** For fuzzy $R : W \times W \to [0,1]$ and $[\![\varphi]\!]_v \in [0,1]$:

$$[\![\Diamond_{\geq n}\varphi]\!]_w = \sigma\left(\sum_v R(w,v)\cdot[\![\varphi]\!]_v - n + \frac{1}{2}\right)$$

where $\sigma$ is the sigmoid function.

**Intuition**

The sum $\sum_v R(w,v)\cdot[\![\varphi]\!]_v$ is a soft count of how many successors satisfy $\varphi$. The sigmoid turns "soft count $\geq n$" into a value in $[0,1]$.

## 23.3 Other Soft Quantifiers

| **Quantifier** | **Soft Version** |
|---|---|
| $\Diamond_{\geq n}\varphi$ | $\sigma(\sum_v R\cdot[\![\varphi]\!]_v - n + 0.5)$ |
| $\Diamond_{\leq n}\varphi$ | $\sigma(n + 0.5 - \sum_v R\cdot[\![\varphi]\!]_v)$ |
| $\mathsf{M}\varphi$ | $\sigma(\sum_v R\cdot[\![\varphi]\!]_v - \sum_v R\cdot[\![\neg\varphi]\!]_v)$ |
| $P_{\geq r}\varphi$ | $\sigma(\frac{\sum_v R\cdot[\![\varphi]\!]_v}{\sum_v R(w,v)} - r)$ |

## 23.4 Learning with Graded Constraints

Now we can train:

**Example 23.2** (Robust Planning)**.** "Learn a transition system where every state has at least 2 safe successors."
Constraint: $\forall w.\Diamond_{\geq 2}\mathsf{safe}$
Loss: $L = \sum_w (1 - [\![\Diamond_{\geq 2}\mathsf{safe}]\!]_w)^2$
Optimize: gradient descent on $R$ (soft accessibility).

**Example 23.3** (Probabilistic Requirement)**.** "With probability $\geq 0.8$, action leads to goal."
Constraint: $P_{\geq 0.8}\mathsf{goal}$
Loss derived from soft semantics, optimize $R$.

## 23.5 Convergence

**Conjecture 23.4.** *With appropriate regularization (entropy penalty, temperature annealing), soft graded modalities converge to crisp:*

- $R(w,v) \to \{0,1\}$
- $[\![\Diamond_{\geq n}\varphi]\!]_w \to \{0,1\}$

*The learned structure is a classical Kripke model satisfying the graded constraints.*

## 23.6 Open Questions

1. What are the best soft approximations? (Sigmoid? Softplus?)
2. Convergence guarantees?
3. How do meta-theoretic properties (decidability, completeness) transfer?
4. Approximation bounds: how close is soft to crisp?

★ **Key Insight**

Differentiable graded modalities let us *learn* structures satisfying counting constraints. This combines the expressiveness of graded modal logic with the learnability of neural networks.

# Part III

# Learning Well

# Chapter 24

# Learning as a Formal Object

**Goals of This Chapter**

- Distinguish *learning task* from *learning*
- Define behavior via final coalgebra
- See how task and learner connect through observation
- Understand performance as a multi-dimensional tuple

## 24.1 Behavior: What We're Trying to Learn

Before we can talk about learning, we need to say what it means for two systems to "behave the same."

**Definition 24.1** (Final Coalgebra and Behavior)**.** Let $H : C \to C$ be a functor. A *final $H$-coalgebra* is a coalgebra $(Z, \omega : Z \to HZ)$ such that for any $H$-coalgebra $(X, \alpha : X \to HX)$, there exists a unique morphism $: X \to Z$ making the diagram commute:

$$X[r, ”\alpha”][d, ””'] HX[d, ”H()”] Z[r, ”\omega”'] HZ$$

The element $(x) \in Z$ is called the *behavior* of state $x$.

The final coalgebra $Z$ is the "universe of all possible behaviors." The unique map  extracts the complete observable behavior of any state.

**Example 24.2** (Behaviors for Common Functors)**.**

| **System** | **Functor $H$** | **Final coalgebra $Z$** |
|---|---|---|
| DFA | $2 \times (-)^{\Sigma}$ | $\mathcal{P}(\Sigma^*)$ (languages) |
| Mealy | $O \times (-)^I$ | $I^* \to O^*$ (causal functions) |
| Labeled transition | $\mathcal{P}(A \times -)$ | infinite trees up to bisim |

Two states have the same behavior iff they are bisimilar.

## 24.2 Learning Task vs Learning

Here is the key distinction:

**Definition 24.3** (Learning Task)**.** A *learning task* is a tuple:

$$T = (H, C_{\text{target}})$$

where:

- $H$ is a functor (the type of system)
- $C_{\text{target}}$ is an $H$-coalgebra that **exists but is unknown**

The task is: find a coalgebra that behaves like $C_{\text{target}}$.

But we don't have direct access to $C_{\text{target}}$. We only see partial observations.

**Definition 24.4** (Observation)**.** An *observation* $D$ is partial information about the behavior of $C_{\text{target}}$.

Formally, let $(Z, \omega)$ be the final $H$-coalgebra. Then:

$$D \subseteq Z_n$$

where $Z_n$ is the "truncation" of $Z$ to depth $n$—behaviors observed up to $n$ steps.

**Example 24.5** (Observations in Practice)**.**

- For DFA: $D$ = finite set of (word, accept/reject) pairs
- For Mealy: $D$ = finite set of input-output traces
- For a program: $D$ = input-output examples, or a queryable oracle

Now we can define learning itself:

**Definition 24.6** (Learning)**.** A *learning configuration* (or *learner*) is a tuple:

$$L = (H, S, \Theta, \gamma)$$

where:

- $H$ : functor (structural assumption—what kind of system)
- $S$ : semiring (how we measure—fuzzy, probabilistic, tropical...)
- $\Theta$ : parameter space, indexing a family of coalgebras $\{C_\theta\}_{\theta \in \Theta}$
- $\gamma$ : optimizer (how we search $\Theta$)

★ **Key Insight**

The learning task $T$ contains the unknown $C_{\text{target}}$.
The learner $L$ contains the machinery to find an approximation.
They are separate objects.

## 24.3 How Task and Learner Connect

The relationship:

$$[columnsep = large, rowsep = large]T = (H, C_{\text{target}})[d, \text{"observe"}']D[r, \text{"}L\text{"}]\theta^* \in \Theta$$

1. The task $T$ contains the unknown target $C_{\text{target}}$
2. We observe $D$, a partial view of $C_{\text{target}}$'s behavior
3. The learner $L$ takes $D$ and produces $\theta^*$
4. We hope $C_{\theta^*} \approx C_{\text{target}}$

More categorically:

$$\begin{aligned} \text{Task}(H) &= \text{category of learning tasks for functor } H \\ \text{Obs} &= \text{category of observations} \\ \text{Learner}(H,S) &= \text{category of } (H,S)\text{-learners} \end{aligned}$$

We have:

- observe : $\text{Task}(H) \to \text{Obs}$ (forgets $C_{\text{target}}$, keeps observable part)
- $L : \text{Obs} \to \Theta$ (learner maps observations to parameters)

## 24.4 Performance is Not a Number

Given a learner $L$ and a task $T$, how do we evaluate?

**Not** with a single number. Performance is multi-dimensional:

**Definition 24.7** (Performance Tuple)**.**

$$\text{Perf}(L, T) = (\varepsilon_{\text{opt}}, \varepsilon_{\text{gen}}, \varepsilon_{\text{exp}})$$

where:

- $\varepsilon_{\text{opt}}$ = **optimization**: how well does $\gamma$ converge on $D$?
- $\varepsilon_{\text{gen}}$ = **generalization**: how well does $C_{\theta^*}$ perform on unseen behavior?
- $\varepsilon_{\text{exp}}$ = **expressiveness**: $\inf_{\theta \in \Theta} d_H^S(C_\theta, C_{\text{target}})$—can we even represent the target?

These dimensions interact:

- More expressive $\Theta$ (lower $\varepsilon_{\text{exp}}$) often means harder optimization (higher $\varepsilon_{\text{opt}}$)
- Better optimization on $D$ may hurt generalization (overfitting)

★ Key Insight

"Learning well" is not a scalar. It's a point in a multi-dimensional performance space.
Different applications care about different trade-offs.

## 24.5 The Geometry Induced by $(H, S)$

Here's the deep point: the pair $(H, S)$ induces geometric structure on $\Theta$.

**Definition 24.8** ($S$-Bisimilarity)**.** For $H$-coalgebras over semiring $S$, the *$S$-bisimilarity* is:

$$d_H^S : \mathrm{Coalg}(H) \times \mathrm{Coalg}(H) \to S$$

defined via Kantorovich lifting of $H$.

Now fix a target. The loss function emerges:

$$\ell : \Theta \to S, \quad \theta \mapsto d_H^S(C_\theta, C_{\text{target}})$$

The pair $(H, S)$ determines:

- The "shape" of the loss landscape on $\Theta$
- Which directions in $\Theta$ correspond to behavioral changes
- The condition number of optimization

★ Key Insight

The optimizer $\gamma$ operates on the geometry that $(H, S)$ induces on $\Theta$.
"Learning well" = $\gamma$ converges efficiently on this geometry.

## 24.6 The Question of Part III

We now have precise language:

> Given $(H, S, \Theta, \gamma)$, what determines whether $\gamma$ converges well on the $(H, S)$-induced geometry of $\Theta$?

Equivalently:

- When does parameter distance match behavioral distance?
- What is the “condition number” of the map $\theta \mapsto C_\theta$?
- How does the choice of $S$ affect optimization?

The rest of Part III explores these questions experimentally and theoretically.

Chapter 25

# Experiments: What Works and What Doesn't

◎ Goals of This Chapter

- See concrete examples where architecture matters
- Observe patterns: symmetry seems to help
- Gather empirical evidence before building theory

## 25.1 Experiment 1: Counting

**Task**: Given a sequence, count occurrences of symbol $a$.

**Setup A**: Fully connected network

- Input: one-hot sequence, flattened
- Output: count (regression)
- Parameters: $O(n^2)$ where $n$ = sequence length

**Setup B**: Recurrent network (counter)

- State: running count
- Update: if $a$, increment; else, keep
- Parameters: $O(1)$

**Result**: Setup B learns instantly. Setup A struggles, needs much more data.

## 25.2 Experiment 2: Image Classification

**Task**: Classify images (e.g., MNIST digits).

**Setup A**: Fully connected

- Flatten image to vector
- Dense layers
- Parameters: $O(n^2)$ for $n$ pixels

**Setup B**: Convolutional

- Local kernels, shared across positions
- Parameters: $O(k^2)$ for kernel size $k$

**Result**: CNN learns much faster, generalizes better, needs less data.

## 25.3 Experiment 3: Set Functions

**Task**: Given a set of numbers, compute some function (e.g., sum, max).

**Setup A**: Feed as ordered sequence to MLP

**Setup B**: Permutation-invariant architecture (DeepSets)

$$f(\{x_1, \dots, x_n\}) = \rho\left(\sum_i \phi(x_i)\right)$$

**Result**: Setup A fails badly (learns spurious order dependence). Setup B works.

## 25.4 The Pattern

| **Task** | **Task Symmetry** | **Best Architecture** |
|---|---|---|
| Counting | Time-shift invariant | RNN / Counter |
| Image | Translation invariant | CNN |
| Set function | Permutation invariant | DeepSets |
| Graph function | Node permutation | GNN |

★ **Key Insight**

When the architecture respects the task's symmetry, learning is easy. When it doesn't, the network wastes capacity learning what it should get for free.

## 25.5 Questions

1. **Why** does matching symmetry help?

2. Can we **quantify** how much it helps?

3. Given a task, can we **derive** the right architecture?

The next chapters develop the theory to answer these.

## 25.6 Numerical Verification: Distance Matching

We ran experiments to test the **distance matching hypothesis**: good parameterization means parameter distance ≈ function distance.

### 25.6.1 Methodology

For each architecture:

1. Sample many random parameter pairs $(\theta_1, \theta_2)$

2. Compute parameter distance $d_\Theta(\theta_1, \theta_2) = \|\theta_1 - \theta_2\|$

3. Compute function distance $d_\mathcal{F}(f_{\theta_1}, f_{\theta_2}) = \|f_{\theta_1}(x) - f_{\theta_2}(x)\|$

4. Measure correlation and coefficient of variation of the ratio

High correlation + low CV = good distance matching.

### 25.6.2 Result 1: Redundancy Destroys Matching

| **Model** | **Correlation** | **Rating** |
|---|---|---|
| Linear (direct) | 0.96 | Excellent |
| Linear $(a - b)$ parameterization | 0.48 | Poor |
| Linear $(a - b + c - d)$ | 0.26 | Terrible |
| Matrix $A \cdot B$ factorization | 0.26 | Terrible |

★ **Key Insight**

Redundant parameterization destroys distance matching. The more redundancy, the worse.

### 25.6.3 Result 2: Depth Destroys Matching (Even Without Nonlinearity!)

| Model | Correlation | Rating |
|---|---|---|
| 1-layer linear | 0.97 | Excellent |
| 2-layer linear ($W_2 W_1 x$) | 0.26 | Terrible |
| 3-layer linear | 0.31 | Terrible |
| 4-layer linear | 0.28 | Terrible |

★ Key Insight

Deep linear networks have the *same function space* as shallow ones (all linear functions), yet distance matching collapses!
Why? Because $W_2 \cdot W_1 = W_2' \cdot W_1'$ has infinitely many solutions. Depth introduces *implicit redundancy.*

### 25.6.4 Result 3: Sparsity Helps

| Model | Correlation | Rating |
|---|---|---|
| Dense linear (20 params) | 0.91 | Excellent |
| Sparse linear (5 params, fixed positions) | 0.98 | Excellent |
| Soft-sparse (sigmoid mask) | 0.48 | Poor |

Fixed sparsity reduces parameters and improves matching. "Soft" sparsity with learnable masks introduces redundancy.

### 25.6.5 Result 4: Matrix Factorization Always Hurts

| Model | Correlation | Rating |
|---|---|---|
| Direct linear | 0.96 | Excellent |
| Low-rank $k = 2$ | 0.31 | Terrible |
| Low-rank $k = 5$ | 0.29 | Terrible |
| SVD parameterization | 0.32 | Terrible |

Any factorization introduces redundancy.

### 25.6.6 Result 5: Nonlinearity Is Not The Main Culprit

| 2-layer MLP with: | Correlation | Rating |
|---|---|---|
| Identity (linear) | 0.13 | Terrible |
| ReLU | 0.20 | Terrible |
| Tanh | 0.19 | Terrible |
| Sigmoid | 0.17 | Terrible |

Even the "linear" 2-layer network (identity activation) has terrible matching!

★ Key Insight

The problem is **depth**, not nonlinearity. The 2-layer structure $W_2 \cdot W_1$ already has redundancy, regardless of activation function.

### 25.6.7 Summary: Hierarchy of Destruction

Factors that destroy distance matching (in order of importance):

1. **Explicit redundancy**: $a - b$ parameterization, matrix factorization
2. **Depth**: even linear depth creates implicit redundancy
3. **Complex structure**: attention (QKV), soft masks

### 25.6.8 The Puzzle

If depth destroys distance matching, why do deep networks work at all?

Possible answers:

- They barely work (training is indeed hard)
- Skip connections help (ResNet, see next section)
- Expressiveness gains outweigh optimization costs
- Special initialization/normalization compensate

## 25.7 Open Question: Why Does ResNet Work?

Our experiments show:

- Plain deep networks: poor matching, degrades with depth
- ResNet: CV (variability) stays more stable with depth

Skip connections make the network “closer to identity,” which may preserve some distance matching properties. This needs more investigation.

## Chapter 26

# The Distance Matching Principle

◎ Goals of This Chapter

- Propose a core principle: good parameterization matches distances
- See how this explains known successes
- Understand what remains to be formalized

## 26.1 The Observation

Why is linear regression optimal for linear functions?

1. Expressiveness: linear model can represent all linear functions
2. No redundancy: different parameters give different functions
3. Convex optimization: unique global minimum
4. Gauss-Markov: statistically optimal (BLUE)

But there's a deeper reason unifying all of these:

★ Key Insight

In linear regression, parameter distance is proportional to function distance.
If $\theta_1$ and $\theta_2$ are close in $\mathbb{R}^{d+1}$, then $f_{\theta_1}$ and $f_{\theta_2}$ are close as functions.
The parameterization "matches" the function space geometry.

## 26.2 Distance Matching

**Definition 26.1** (Informal)**.** A parameterization $\gamma : \Theta \to \mathcal{F}$ **matches distances** if:

$$d_\Theta(\theta_1, \theta_2) \approx c \cdot d_{\mathcal{F}}(\gamma(\theta_1), \gamma(\theta_2))$$

for some constant $c > 0$.

Strict version: $\gamma$ is an **isometry** (distance-preserving map).
Weak version: distances are **proportional** (up to bounded distortion).

## 26.3 Why Distance Matching Helps Optimization

Gradient descent moves in the steepest direction in parameter space.

**If distances match**: steepest in $\Theta$ = steepest in $\mathcal{F}$. Gradient descent does the "right thing."

**If distances don't match**:

- Some directions in $\Theta$: move far, function barely changes (redundant)
- Some directions in $\Theta$: move little, function changes a lot (sensitive)
- Gradient descent zigzags, wastes effort

## 26.4 Case Studies

### 26.4.1 Linear Regression: Perfect Match

| | |
|---|---|
| Parameter space | $\mathbb{R}^{d+1}$ with Euclidean distance |
| Function space | Linear functions with $L^2$ distance |
| Relationship | $\|f_{\theta_1} - f_{\theta_2}\|_{L^2} \propto \|\theta_1 - theta_2\|$ |

Linear relationship. Perfect match. Optimization is easy.

### 26.4.2 MLP Learning a Linear Function: Mismatch

| | |
|---|---|
| Parameter space | High-dimensional (all weight matrices) |
| Function space | Linear functions (small subset) |
| Relationship | Many $\theta$ map to same function |

Severe redundancy. Most directions in $\Theta$ don't change the function. Distances don't match.

### 26.4.3 CNN for Translation-Invariant Functions: Approximate Match

| | |
|---|---|
| Parameter space | Convolution kernels (small) |
| Function space | Translation-invariant functions |
| Relationship | Different kernels → different functions (near bijection) |

Much better match than MLP. This may explain why CNNs work well on images.

### 26.4.4 Fully-Connected for Translation-Invariant: Mismatch

| | |
|---|---|
| Parameter space | $O(n^2)$ weights |
| Function space | Translation-invariant functions |
| Relationship | Huge redundancy |

Most weight configurations give the same translation-invariant function. Terrible mismatch.

### 26.4.5 ResNet: Improved Match via Skip Connections

Skip connections have an interesting effect:

- Identity function corresponds to $F = 0$ (zero residual)
- Small perturbation of parameters → small perturbation of function
- The mapping $\theta \mapsto f_\theta$ is more "linear" near identity

This may make the distance relationship more proportional.

## 26.5 Measuring Distance Match

How to quantify whether distances match?

### 26.5.1 Jacobian Singular Values

The Jacobian $J = \partial f / \partial \theta$ tells us how parameter changes map to function changes.

- Singular values all similar → uniform in all directions → good match
- Singular values vary widely → some directions stretched, some compressed → bad match

**Metric**: ratio of largest to smallest singular value (condition number).

### 26.5.2 Fisher Information

The Fisher information matrix $F(\theta)$ measures sensitivity of output distribution to parameter changes.

- $F$ well-conditioned → all directions equally informative → good match
- $F$ ill-conditioned → some directions redundant → bad match

### 26.5.3 Connection

Fisher information and Jacobian singular values are related:

For regression with Gaussian noise, $F \propto J^\top J$.

Condition number of $F$ = square of condition number of $J$.

## 26.6 Natural Gradient: A Partial Solution

Amari's natural gradient addresses mismatch by *correcting* gradients:

$$\tilde{\nabla} L = F^{-1} \nabla L$$

This makes gradient descent behave as if distances matched.

**Problem**: Requires computing $F^{-1}$, which is expensive.

**Our question**: Can we choose $\Theta$ so that ordinary gradient already works well?

## 26.7 The Design Principle

★ **Key Insight**

**Conjecture**: The optimal parameterization for a function class $\mathcal{F}$ is one where parameter distance matches function distance.
Formally: find $\Theta$ and $\gamma : \Theta \to \mathcal{F}$ such that $\gamma$ is (approximately) an isometry.

This would unify:

- Linear regression: isometry by construction
- CNNs: approximate isometry for translation-invariant functions
- Equivariant networks: remove redundant directions, improve match

## 26.8 Categorical Formalization: Lawvere Metric Spaces

The distance matching principle has a natural formalization using **enriched category theory**.

### 26.8.1 Lawvere's Insight

Lawvere (1973) observed that metric spaces are categories enriched over $([0, \infty], \geq, +)$:

- Objects: points in the space
- $\text{Hom}(A, B) = d(A, B) \in [0, \infty]$
- Composition: triangle inequality $d(A, C) \leq d(A, B) + d(B, C)$
- Identity: $d(A, A) = 0$

In this view, a **functor** between Lawvere metric spaces is a **distance non-increasing map**:

$$d_Y(F(a), F(b)) \leq d_X(a, b)$$

That is, a map with Lipschitz constant $\leq 1$.

### 26.8.2 Behavioral Metrics for Coalgebras

For coalgebras, there is a well-developed theory of **behavioral metrics**:

Given a coalgebra $\alpha : X \to HX$, we can define a pseudometric $d$ on states where:

- $d(s, t) = 0$ iff $s$ and $t$ are bisimilar
- $d(s, t)$ small means $s$ and $t$ behave similarly
- $d(s, t)$ large means very different behaviors

This is constructed by lifting the functor $H$ to the category of (pseudo)metric spaces, using Kantorovich or Wasserstein constructions.

### 26.8.3 Distance Matching as Metric Functor

Now we can formalize distance matching:

**Definition 26.2** (Distance Matching — Formal)**.** A parameterization $\gamma : \Theta \to \text{Coalg}(H)$ is **distance matching** if it is a bi-Lipschitz map:

$$c_1 \cdot d_\Theta(\theta_1, \theta_2) \leq d_{\text{behavior}}(\gamma(\theta_1), \gamma(\theta_2)) \leq c_2 \cdot d_\Theta(\theta_1, \theta_2)$$

for constants $0 < c_1 \leq c_2 < \infty$.

- Upper bound ($\leq c_2$): $\gamma$ is Lipschitz. Small parameter change → small behavior change. **Continuity.**
- Lower bound ($\geq c_1$): $\gamma^{-1}$ is Lipschitz. Different behaviors → different parameters. **No redundancy.**

If $c_1 = c_2$, we have an **isometry** (up to scaling).

### 26.8.4 The Picture

$$\begin{tikzcd}
\text{Parameter space } \Theta \arrow[r, "\gamma"] \arrow[d, "\text{Euclidean}"'] & \text{Coalgebra space} \arrow[d, "\text{Behavioral}"] \\
\text{Lawvere metric space} \arrow[r, "\text{bi-Lipschitz?}"'] & \text{Lawvere metric space}
\end{tikzcd}$$

Distance matching asks: is the bottom arrow bi-Lipschitz?

## 26.9 Connection to Semiring Relaxation

This framework connects Part II and Part III:

### 26.9.1 Why Semiring Relaxation Helps

In Part II, we softened discrete coalgebras to semiring-valued coalgebras to enable gradient descent.

Now we see another benefit: **softening makes behavioral distance continuous**.

- Hard DFA: small parameter change can cause discrete behavior jump (0 → 1)
- Soft DFA: parameter change causes proportional behavior change

Softening is necessary not just for gradients, but for distance matching!

### 26.9.2 The Full Picture

1. **Part I**: Define the structure (coalgebra, bisimulation)
2. **Part II**: Soften to enable gradients (semiring relaxation)
3. **Part III**: Parameterize to match distances (behavioral metric)

Together: a principled pipeline from structure to learnable system.

## 26.10 Implications

If this framework is correct:

★ Key Insight

Architecture design becomes **geometry matching**:

1. Compute the behavioral metric of your target coalgebra class
2. Construct a parameter space with matching metric geometry
3. Gradient descent then "does the right thing"

This is no longer alchemy. It is engineering from first principles.

## 26.11 Open Problems

1. **Compute behavioral metrics**: For specific functors (DFA, Markov, Kripke), what is the behavioral metric explicitly?
2. **Construct matching parameterizations**: Given a behavioral metric, how to design $\Theta$ to match it?
3. **Verify known architectures**: Do successful architectures (CNN, ResNet, Transformer) achieve approximate distance matching? Can we measure this?
4. **Generalization**: Distance matching addresses optimization. Does it also help generalization?
5. **Approximation**: When perfect matching is impossible, what's the best approximation? Is there a canonical "closest" parameterization?

These are the questions for future research.

# Chapter 27

# Distance Matching: Theory and Experiments

◎ Goals of This Chapter

- Formalize the distance matching principle
- Connect it to optimization theory via condition number
- Verify empirically with diverse architectures and tasks
- Show that loss function induces the behavioral metric

## 27.1 The Core Principle

★ Key Insight

**Distance Matching Principle**: A parameterization $\gamma : \Theta \to \mathcal{F}$ is "good" if parameter distance is proportional to function distance:

$$c_1 \cdot d_\Theta(\theta_1, \theta_2) \leq d_\mathcal{F}(\gamma(\theta_1), \gamma(\theta_2)) \leq c_2 \cdot d_\Theta(\theta_1, \theta_2)$$

When $c_1 \approx c_2$, the mapping is approximately an isometry.

## 27.2 Loss Function Induces Behavioral Metric

A key insight: the loss function *defines* the behavioral metric, not the other way around.

For MSE loss $L(\theta) = \|f_\theta(X) - Y\|_2^2$, the natural behavioral distance is:

$$d_{\text{behavior}}(f, g) = \|f(X) - g(X)\|_2$$

This is not an arbitrary choice—it is *induced* by the loss function:

| **Loss Function** | **Output Metric** | **Behavioral Distance** |
|---|---|---|
| MSE | $L_2$ | $\|f(X) - g(X)\|_2$ |
| MAE | $L_1$ | $\|f(X) - g(X)\|_1$ |
| Cross-entropy | KL divergence | $D_{KL}(f\|g)$ |
| Wasserstein | Wasserstein | $W(f, g)$ |

★ Key Insight

The loss function simultaneously defines:

1. What we optimize (minimize loss)
2. The behavioral metric (distance on function space)

These are two sides of the same coin.

## 27.3 Connection to Optimization: The Jacobian

Let $J = \frac{\partial f_\theta(X)}{\partial \theta}$ be the Jacobian of the parameter-to-output mapping.

### 27.3.1 Condition Number

The **condition number** $\kappa(J) = \sigma_{\max}(J)/\sigma_{\min}(J)$ measures how well distances are preserved:

- $\kappa(J) = 1$: perfect isometry
- $\kappa(J)$ large: some directions stretched, others compressed
- $\kappa(J) = \infty$: null space exists (redundant parameters)

### 27.3.2 Why $\kappa(J)$ Determines Learnability

For MSE loss, the Hessian (Gauss-Newton approximation) is $H \approx J^\top J$.

**Theorem 27.1.** *Gradient descent converges at rate $O\left(\left(1 - \frac{1}{\kappa(H)}\right)^t\right)$, where $\kappa(H) = \kappa(J)^2$.*

Therefore:

$$\kappa(J) \text{ small} \implies \kappa(H) \text{ small} \implies \text{fast convergence}$$

★ Key Insight

$\kappa(J)$ is the **precise measure of learnability**:

- $\kappa(J)$ small → gradient descent converges quickly
- $\kappa(J)$ large → convergence is slow or unstable

## 27.4 Experimental Verification

We conducted extensive experiments to verify the theory.

### 27.4.1 Experiment 1: Same Function Space, Different Parameterizations

All models below express the *same* function space (linear functions), but with different parameterizations:

| **Parameterization** | $\kappa(J)$ | **Final Loss** | **CV** |
|---|---|---|---|
| Direct: $f(x) = x \cdot \theta$ | 1.5 | 0.0000 | 0.00 |
| Redundant: $f(x) = x \cdot (a - b)$ | $\infty$ | 0.0000 | 0.00 |
| 2-layer: $f(x) = x \cdot W_1 \cdot W_2$ | $\infty$ | 0.1390 | 1.29 |
| MLP (overkill) | $\infty$ | 2.5360 | 0.38 |

Key observation: same function space, but parameterization determines $\kappa(J)$ and learning stability.

### 27.4.2 Experiment 2: Depth Destroys Distance Matching

Even for *linear* networks (no nonlinearity), depth introduces implicit redundancy:

| **Model** | $\kappa(J)$ | **Rating** |
|---|---|---|
| 1-layer linear | 1.9 | Excellent |
| 2-layer linear ($W_2 W_1 x$) | $\infty$ | Poor |
| 3-layer linear | $\infty$ | Poor |

Why? Because $W_2 \cdot W_1 = W_2' \cdot W_1'$ has infinitely many solutions.

### 27.4.3 Experiment 3: Classic Task-Architecture Matches

When architecture matches task structure, $\kappa(J)$ is small:

| **Task** | **Architecture** | $\kappa(J)$ | **Loss** | **CV** |
|---|---|---|---|---|
| Permutation invariant $\sum x_i$ | DeepSets | **1.0** | 0.0000 | 0.87 |
| Multiplication $x_1 \cdot x_2$ | Mult gate | **1.0** | 0.0000 | **0.00** |
| Counting $\#(x > 0)$ | Counter | **1.0** | 0.1010 | 0.05 |
| Sparse $x_0 + x_1$ | Sparse (2 params) | **1.2** | 0.0000 | 0.78 |
| Linear | MLP | $\infty$ | 2.54 | – |
| Permutation invariant | MLP | 336 | 2.02 | – |
| Sparse | MLP | 60 | 1.08 | – |

The multiplication gate achieves **perfect** distance matching: $\kappa = 1$, loss $= 0$, CV $= 0$.

### 27.4.4 Experiment 4: $\kappa(J)$ Determines Distribution of Outcomes

We ran 30 random seeds for each architecture:

| **Architecture** | $\kappa(J)$ | **Mean Loss** | **Std Loss** | **CV** |
|---|---|---|---|---|
| Linear direct | 1.5 | 0.008 | 0.000 | **0.00** |
| Linear (a-b) | $\infty$ | 0.008 | 0.000 | **0.00** |
| MLP 2-layer | $\infty$ | 0.59 | 0.57 | **0.96** |
| ResNet 2-layer | 15667 | 0.11 | 0.03 | 0.26 |

★ Key Insight

$\kappa(J)$ determines not a single convergence speed, but the **distribution** of outcomes:

- $\kappa(J)$ small → narrow distribution (all seeds converge well)
- $\kappa(J)$ large → wide distribution (depends on luck/initialization)

This explains why "good architectures" are robust to initialization.

## 27.5 Two Failure Modes

An architecture can fail in two ways:

1. **Lack of expressiveness**: Cannot represent the target function.
   - Example: Linear model for multiplication
   - Symptom: $\kappa(J)$ small but loss high
2. **Redundancy**: Can represent but has unnecessary parameters.
   - Example: MLP for linear function

- Symptom: $\kappa(J)$ large, training unstable

The ideal architecture has:

$$\text{Sufficient expressiveness} + \text{No redundancy} = \kappa(J) \text{ minimized}$$

## 27.6 Connection to Category Theory

The framework connects to enriched category theory:

1. **Loss function** defines the enrichment (metric on output space)
2. This induces a **Lawvere metric** on the function space
3. Distance matching asks: is $\gamma : \Theta \to \mathcal{F}$ a bi-Lipschitz map?
4. $\kappa(J)$ is the Lipschitz constant ratio

$$\begin{tikzcd}
\text{Loss Function (MSE)} \arrow[d, "\text{induces}"] & \\
\mathbb{R}^n \text{ with } L_2 \text{ metric} \arrow[d, "\text{induces}"] & \\
\text{Function space with behavioral metric} \arrow[r, leftrightarrow, "\kappa(J) \text{ measures this}", "\text{bi-Lipschitz?}"'] & \text{Parameter space } \Theta
\end{tikzcd}$$

## 27.7 Summary

1. **Behavioral metric** is induced by the loss function (not arbitrary)
2. **Distance matching** = parameter distance ≈ behavioral distance
3. $\kappa(J)$ is the precise measure of distance matching quality
4. $\kappa(J)$ **small** ⇔ fast, stable learning
5. $\kappa(J)$ **large** ⇔ slow, unstable, seed-dependent
6. **Good architecture** = matches task structure = minimizes $\kappa(J)$

This provides a principled answer to "why does architecture matter": the right architecture achieves distance matching, making optimization efficient.

## 27.8 Related Work

The ideas in this chapter have deep connections to existing literature.

### 27.8.1 Dynamical Isometry (Saxe et al., 2013)

The concept of $\kappa(J) \approx 1$ was introduced as **dynamical isometry**:

> "When the singular values of the input-output Jacobian are all clustered around one, the network achieves dynamical isometry and trains efficiently despite being very deep."

Key results:

- Orthogonal initialization achieves dynamical isometry
- Deep linear networks can have depth-independent learning times with proper initialization
- This explains why certain initializations work better than others

### 27.8.2 Neural Tangent Kernel (NTK) Theory

The NTK literature explicitly connects eigenvalues to trainability:

- $\lambda_{\min}(\text{NTK})$ determines the slowest convergence direction
- The condition number $\kappa = \lambda_{\max}/\lambda_{\min}$ is used as a **trainability measure**
- Convergence rate is $O(e^{-\lambda_{\min} t})$ along the slowest eigenvector

### 27.8.3 Natural Gradient (Amari, 1998)

Amari's natural gradient addresses distance mismatch by correcting gradients:

$$\tilde{\nabla} L = F^{-1} \nabla L$$

where $F$ is the Fisher information matrix. This makes gradient descent behave *as if* distances matched.

Our perspective: instead of correcting gradients, choose a parameterization where ordinary gradients already work well.

### 27.8.4 Coalgebraic Behavioral Metrics

The categorical framework for behavioral metrics is well-developed:

- Given functor $H$, the Kantorovich/Wasserstein lifting defines a behavioral metric
- $d(s, t) = 0$ iff $s$ and $t$ are bisimilar
- This provides a principled way to define "behavioral distance"

### 27.8.5 Our Contribution

What we add to the existing picture:

1. **Loss function as primitive**: The loss function *induces* the behavioral metric (not a separate choice)

2. **Distribution perspective**: $\kappa(J)$ determines the *distribution* of outcomes over random seeds, not just expected convergence

3. **Task-architecture matching**: Systematic experiments showing that matching architecture to task structure minimizes $\kappa(J)$

4. **Categorical connection**: Framing in terms of Lawvere metric spaces and bi-Lipschitz maps

## 27.9 Open Questions

1. Can we **compute** $\kappa(J)$ efficiently for large networks?

2. Given a task, can we **derive** the optimal architecture that minimizes $\kappa(J)$?

3. How does $\kappa(J)$ relate to **generalization**? (We only addressed optimization)

4. For non-MSE losses, what is the correct behavioral metric?

5. Can the coalgebraic behavioral metric framework give us new architectures?

# Chapter 28

# From Functor to Parameterization: A Natural Emergence

◎ Goals of This Chapter

- Show that the functor structure naturally induces a behavioral metric
- Demonstrate that this metric matches parameter distance (distance matching)
- Connect Parts I, II, and III into a unified framework

## 28.1 The Question

We have established:

- **Part I**: Coalgebras define structure; bisimulation defines behavioral equivalence
- **Part II**: Semiring relaxation enables gradients
- **Part III**: Distance matching ($\kappa(J)$ small) implies good optimization

The missing link: **Given a functor $H$, what parameterization achieves distance matching?**

## 28.2 The Insight: Functor Induces Metric

For a functor $H$, there is a standard construction called **Kantorovich lifting** that produces a behavioral metric.

### 28.2.1 Example: DFA Functor

For deterministic automata, $H = 2 \times (-)^{\Sigma}$:

- A coalgebra is $(S, \alpha : S \to 2 \times S^{\Sigma}) = (S, \text{accept}, \delta)$
- Bisimulation: $s \sim t$ iff they accept the same language

### 28.2.2 Soft DFA (Relaxation)

Applying semiring relaxation (Part II):

- Soft transitions: $\delta : S \times \Sigma \to \text{Dist}(S)$ (stochastic matrices)
- Soft accept: $a : S \to [0, 1]$

### 28.2.3 Two Behavioral Metrics

The functor structure suggests two natural metrics:

**1. One-Step Metric (Local)**

$$d_{\text{one-step}}(A, B) = \|a_1 - a_2\|_1 + \sum_{\sigma \in \Sigma} \sum_{s \in S} \text{TV}(T_1^{\sigma}[s], T_2^{\sigma}[s])$$

where TV is total variation distance.

This is **one iteration** of the Kantorovich lifting.

**2. Language Metric (Global)**

$$d_{\text{language}}(A, B) = \sum_{w \in \Sigma^*} 2^{-|w|} |A(w) - B(w)|$$

where $A(w)$ is the acceptance probability of word $w$.

This is the **full behavioral distance** (bisimulation metric).

## 28.3 Experimental Discovery

We tested which behavioral metric matches parameter distance:

| **Behavioral Metric** | **Param Metric** | **Correlation** | **CV** |
|---|---|---|---|
| Structure (trivial) | Euclidean | 1.000 | 0.000 |
| **One-step** | **Fisher-Rao** | **0.802** | **0.101** |
| One-step | Euclidean | 0.761 | 0.106 |
| Fixed-point (iterated) | Euclidean | 0.435 | 0.352 |
| Language (global) | Euclidean | 0.279 | 0.490 |

★ Key Insight

The **one-step** behavioral metric (one Kantorovich iteration) matches parameter distance very well (correlation $> 0.8$).
The **global** language metric matches poorly (correlation $\approx 0.3$).

## 28.4 Interpretation

### 28.4.1 Why One-Step Works

The one-step metric compares *local* structure:

- Accept probabilities (directly)
- Transition distributions (one step)

This is exactly what the parameters encode! Small parameter changes → small one-step behavioral changes.

### 28.4.2 Why Language Metric Fails

The language metric compares *global* behavior:

- Involves arbitrarily long words
- Small parameter changes can compound over many steps
- Similar to deep networks: small weight changes → large output changes

### 28.4.3 The Depth Analogy

| **Automata** | **Neural Networks** |
|---|---|
| Word length $\lvert w\rvert$ | Network depth |
| One-step metric | One-layer comparison |
| Language metric | End-to-end comparison |
| Long words break matching | Deep networks break matching |

## 28.5 The Natural Emergence

We can now state the principle:

★ **Key Insight**

**Natural Parameterization Principle**:
Given functor $H$ defining a class of coalgebras:

1. **Structure**: $H$ defines the parameter space (e.g., stochastic matrices for soft DFA)
2. **Metric**: One-step Kantorovich lifting defines the behavioral metric
3. **Matching**: This metric naturally matches parameter distance
4. **Consequence**: Gradient descent works well for learning “one-step” objectives

## 28.6 Implications for Learning

### 28.6.1 What Can Be Learned Easily

Tasks where the objective is “local” (one-step behavioral):

- Predicting next-state distributions
- Matching transition structure
- Local acceptance probabilities

### 28.6.2 What Is Hard to Learn

Tasks requiring “global” behavioral matching:

- Language equivalence (accepting same strings)
- Long-horizon properties
- End-to-end sequence behavior

### 28.6.3 The Gap

$$\textbf{Local (one-step)} \xrightarrow{\text{iterations / depth}} \textbf{Global (language)}$$

| | |
|---|---|
| Good distance matching | Poor distance matching |
| Easy to optimize | Hard to optimize |

This explains why:

- RNNs struggle with long sequences (global objective, poor matching)
- Transformers with attention help (attention provides “shortcuts” that reduce effective depth)
- Curriculum learning helps (start with short sequences, gradually increase)

## 28.7 The Unified Framework

We can now connect all three parts:

$$\begin{tikzcd}
 & \textbf{Part I}\text{: Functor } H \arrow[d] & \\
 & \text{Coalgebra structure} \arrow[d, "\text{semiring relax}"] & \\
 & \textbf{Part II}\text{: Soft coalgebra} \arrow[dl] \arrow[dr, "\text{Kantorovich}"] & \\
\text{Parameter space } \Theta \arrow[dr] & & \text{Behavioral metric } d_H \arrow[dl] \\
 & \textbf{Part III}\text{: Distance matching} \arrow[d, "\kappa(J) \text{ small}"] & \\
 & \text{Learnable!} &
\end{tikzcd}$$

## 28.8 Open Questions

1. **Other functors**: Does this work for Markov chains ($H = \text{Dist}$), Kripke frames, etc.?
2. **Bridging local and global**: Can we design parameterizations that achieve good matching for global metrics?
3. **Attention and shortcuts**: Can attention be understood as reducing “effective depth” in this framework?
4. **Automatic architecture**: Given $H$, can we automatically derive the optimal neural architecture?

## 28.9 Related Work

Our framework connects to several established research directions:

### 28.9.1 Bisimulation Metrics for MDPs

Ferns, Panangaden, and Precup (2004, 2011) developed **bisimulation metrics** for Markov Decision Processes—quantitative relaxations of bisimulation equivalence. Their key insight: the Kantorovich functional (from optimal transport) naturally lifts metrics on state spaces to metrics on distributions, enabling recursive computation of behavioral distance.

For MDPs with discount factor $\gamma$, they proved that the bisimulation metric gives **tight bounds on optimal value functions**:

$$|V^*(s) - V^*(t)| \leq \frac{1}{1-\gamma} d_{\text{bisim}}(s,t)$$

This is precisely the kind of "behavioral metric bounds function distance" relationship we seek.

### 28.9.2 Deep Bisimulation for Control

Zhang et al. (2020) applied bisimulation metrics to deep reinforcement learning. Their method, **Deep Bisimulation for Control (DBC)**, trains encoders such that distances in latent space equal bisimulation distances in state space.

Key theoretical result: the optimal value function is **Lipschitz with respect to the bisimulation metric**. This provides formal bounds on suboptimality when learning from the induced representation.

★ Key Insight

DBC essentially implements our framework in reverse:

- We ask: given parameters, does parameter distance match behavioral distance?
- DBC asks: can we learn parameters such that latent distance equals behavioral distance?

Both are manifestations of the same principle: good representations preserve behavioral structure.

### 28.9.3 Coalgebraic Behavioral Metrics

The coalgebra community has developed a general theory of behavioral metrics. Given a functor $H$ on **Set**, there are systematic ways to lift it to functors on (pseudo)metric spaces:

- **Wasserstein lifting**: Based on optimal transport couplings

- **Kantorovich lifting**: Based on Lipschitz functions (dual formulation)

Baldan et al. (2018) and subsequent work established that these liftings give rise to behavioral pseudometrics that generalize bisimulation to the quantitative setting.

Our contribution is the observation that the **one-step** Kantorovich lifting (not the full fixed-point) matches parameter distance, explaining why local objectives are easier to optimize than global ones.

## 28.10 Summary

★ Key Insight

The functor $H$ doesn't just define *what* we're learning (coalgebra structure).
It also defines *how well* we can learn it:

- One-step Kantorovich metric: learnable (good distance matching)
- Full behavioral metric: hard (poor distance matching)

This is not a failure of our methods—it's a fundamental property of the structure being learned.

# Chapter 29

# Entanglement and Expressiveness

◎ Goals of This Chapter

- Understand the tensor network perspective on neural networks
- See how entanglement entropy characterizes expressiveness
- Know what this explains and what it doesn't

## 29.1 Functions as Tensors

A function $f(x_1, \ldots, x_n)$ where each $x_i \in \{0, 1\}$ can be viewed as a tensor with $2^n$ entries.

Naive representation: store all $2^n$ values. Exponential in $n$.

But many functions have **structure** that allows compression.

## 29.2 Tensor Decomposition

Different decompositions correspond to different dependency structures:

### 29.2.1 Fully Factorized (No Dependencies)

$$f(x_1, \ldots, x_n) = g_1(x_1) \cdot g_2(x_2) \cdots g_n(x_n)$$

Variables are independent. Only $O(n)$ parameters needed.

### 29.2.2 Matrix Product State (Chain Dependencies)

$$f(x_1, \ldots, x_n) = A_1(x_1) \cdot A_2(x_2) \cdots A_n(x_n)$$

where each $A_i$ is a matrix. Nearby variables correlated.
Parameters: $O(n \cdot r^2)$ where $r$ is the "bond dimension."
This is like an RNN.

### 29.2.3 Tree Tensor Network (Hierarchical Dependencies)

Variables are grouped hierarchically:

- First layer: pairs $(x_1, x_2), (x_3, x_4), \ldots$
- Second layer: groups of pairs
- Continue until single output

This is like a CNN or a deep network with pooling.

### 29.2.4 Fully Entangled (All Correlated)

No structure to exploit. Need $O(2^n)$ parameters.

## 29.3 Entanglement Entropy

Given a partition of variables into sets $A$ and $B$:

**Definition 29.1** (Entanglement Entropy)**.** The entanglement entropy $S(A : B)$ measures how much $A$ and $B$ are "inseparably correlated."

$$S(A : B) = -\mathrm{Tr}(\rho_A \log \rho_A)$$

where $\rho_A$ is the reduced density matrix.

**Intuition**

- $S = 0$: $A$ and $B$ are independent, $f = g(A) \cdot h(B)$
- $S$ large: $A$ and $B$ deeply entangled, must be processed together

## 29.4 The Deep vs Shallow Separation

**Theorem 29.2** (Cohen et al., 2016)**.** *Consider functions with hierarchical entanglement structure (local variables highly entangled, distant variables less so).*

- *Deep networks (hierarchical tensor decomposition): $O(poly(n))$ parameters*
- *Shallow networks (flat decomposition): $O(2^n)$ parameters*

*This is an exponential separation.*

★ Key Insight

Depth is necessary when the target function has hierarchical entanglement.
Deep networks match this structure. Shallow networks don't.

## 29.5 Matching Entanglement Structure

The principle:

> **If the architecture's tensor structure matches the target's entanglement structure, learning is efficient.**

### 29.5.1 Images

- Local pixels: high entanglement (edges, textures)
- Distant pixels: low entanglement
- Structure: hierarchical, local-to-global
- Matching architecture: CNN (hierarchical, local kernels)

### 29.5.2 Sequences

- May have long-range dependencies
- Entanglement not purely local
- Need architecture that can "skip" hierarchy levels
- Matching architecture: Transformer? (direct connections)

## 29.6 Relation to Distance Matching

Entanglement structure tells us about **expressiveness**: can the architecture represent the function?

Distance matching tells us about **optimization**: can we find the right parameters?

They are complementary:

- Entanglement: which architecture class is sufficient?
- Distance: within that class, which parameterization is optimal?

## 29.7 Limitations

Tensor network theory explains:

- Why depth matters
- Why local structure (convolution) helps for local entanglement

But doesn't explain:

- Skip connections (ResNet)
- Attention mechanisms (Transformer)
- Specific architectural choices beyond depth

It's one dimension of the story, not the whole story.

## 29.8 Open Questions

1. How to measure entanglement structure of a learning task from data?
2. Can we automatically select architecture based on measured entanglement?
3. How does entanglement relate to the distance matching principle?
4. What's the entanglement structure of language? (Needed to understand Transformers)

Chapter 30

# Fisher Information and Distance

◎ Goals of This Chapter

- Connect Fisher information to the distance matching principle
- Understand condition number as a measure of mismatch
- See natural gradient as a correction for mismatch

## 30.1 The Jacobian Perspective

A parameterization $\gamma : \Theta \to \mathcal{F}$ maps parameters to functions.

The Jacobian $J(\theta) = \partial\gamma/\partial\theta$ tells us:

> How do small parameter changes translate to function changes?

## 30.2 Singular Values and Distance Distortion

The singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k$ of $J$ measure:

- $\sigma_i$ large: moving in direction $i$ changes the function a lot
- $\sigma_i$ small: moving in direction $i$ barely changes the function
- $\sigma_i = 0$: direction $i$ is redundant (multiple $\theta$ give same function)

## 30.3 Condition Number

**Definition 30.1.** The condition number is $\kappa = \sigma_{\max}/\sigma_{\min}$.

| | |
|---|---|
| $\kappa \approx 1$ | All directions similar. Distance matching. Easy optimization. |
| $\kappa \gg 1$ | Some directions stretched, others compressed. Mismatch. Hard. |
| $\kappa = \infty$ | Some directions have $\sigma = 0$. Redundancy. Very hard. |

**Key Insight**

Condition number measures how far the parameterization is from distance matching.
$\kappa = 1$ means perfect isometry. Large $\kappa$ means severe distortion.

## 30.4 Fisher Information Matrix

For probabilistic models, the Fisher information matrix is:

$$F_{ij}(\theta) = \mathbb{E}\left[\frac{\partial \log p_\theta}{\partial \theta_i} \cdot \frac{\partial \log p_\theta}{\partial \theta_j}\right]$$

For regression with Gaussian noise: $F \propto J^\top J$.
The eigenvalues of $F$ are the squared singular values of $J$.
Condition number of $F = \kappa^2$.

## 30.5 Natural Gradient

Ordinary gradient descent:

$$\theta_{t+1} = \theta_t - \eta \nabla L$$

This moves in the steepest direction in parameter space.
Natural gradient:

$$\theta_{t+1} = \theta_t - \eta F^{-1} \nabla L$$

This moves in the steepest direction in function space.

**Key Insight**

Natural gradient corrects for distance mismatch by rescaling directions according to their sensitivity.
It makes gradient descent behave as if the parameterization were isometric.

## 30.6 The Design Question

Natural gradient is a **post-hoc fix**: given a parameterization, correct the gradients.

But we want **a priori design**: choose a parameterization where ordinary gradient already works.

> Can we design $\Theta$ such that $F \approx I$ (identity)?

If yes, natural gradient = ordinary gradient. No correction needed.

## 30.7 When Is $F \approx I$ Possible?

- Linear regression with orthonormal features: $F = I$ exactly
- Parameterization that "respects" the function space geometry
- Quotient by symmetry group (remove redundant directions)

This connects to the symmetry/equivariance view: equivariant parameterizations remove redundancy, improving conditioning.

## 30.8 Summary

| **Concept** | **Role** |
|---|---|
| Jacobian $J$ | How parameters map to functions |
| Singular values | Stretch/compression in each direction |
| Condition number | Measure of distance mismatch |
| Fisher matrix $F$ | $J^\top J$, captures local geometry |
| Natural gradient | Corrects for mismatch |
| Optimal parameterization | Achieves $\kappa \approx 1$ by design |

# Chapter 31

# Symmetry as a Special Case

◎ Goals of This Chapter

- See equivariant architectures through the distance matching lens
- Understand why symmetry helps: it removes redundancy
- Know the limitations: not all structure is group-theoretic

## 31.1 The Redundancy Problem

Consider learning a translation-invariant function with a fully-connected network.

- The FC network can express many non-invariant functions
- Many different weight configurations give the same invariant function
- The parameter space is much larger than the function space

Result: severe distance mismatch. High condition number. Hard to optimize.

## 31.2 Equivariance Removes Redundancy

A $G$-equivariant architecture can only express $G$-equivariant functions.

- Parameter space is smaller
- Different parameters give different functions (less redundancy)
- Better match between parameter distance and function distance

★ Key Insight

Equivariance improves distance matching by constraining the parameter space to match the function space.

## 31.3 Example: CNN vs FC for Images

**Target**: translation-invariant image functions.

**FC network**:

- Parameters: $O(n^2)$ weights
- Can express all functions, not just invariant ones
- Huge redundancy for invariant targets
- Poor distance matching

**CNN**:

- Parameters: $O(k^2)$ kernel weights
- Can only express translation-equivariant functions
- Little redundancy
- Good distance matching

## 31.4 Quotient by Symmetry Group

Mathematically, if function space has symmetry group $G$:

$$\text{Effective function space} = \mathcal{F}/G$$

An equivariant parameterization directly parameterizes $\mathcal{F}/G$, avoiding redundancy.

A non-equivariant parameterization has $|G|$-fold redundancy.

## 31.5 Limitations

Symmetry-based design works when:

- The target has a known group symmetry $G$
- We can construct $G$-equivariant layers

But many learning problems don't have obvious group structure:

- Automata, Kripke frames: what's the group?
- Language: partial symmetries at best
- General coalgebras: symmetry unclear

## 31.6 Symmetry Within the Larger Picture

| | |
|---|---|
| **General principle** | Distance matching |
| **Special case** | Symmetry/equivariance |
| **Mechanism** | Remove redundancy by quotienting |
| **Limitation** | Requires known group structure |

The distance matching principle is more general. Symmetry is one way to achieve it, but not the only way.

## 31.7 Open Question

For structures without group symmetry (coalgebras, automata), what plays the role of "quotient by $G$"?

Possibility: quotient by bisimulation equivalence?

This connects back to our setting in Part I: bisimulation is the right equivalence for coalgebras, just as $G$-orbits are the right equivalence for $G$-symmetric functions.

# Chapter 32

# Open Questions and Other Directions

◎ Goals of This Chapter

- Summarize what we've established
- State the remaining open questions precisely
- Briefly note other directions we chose not to pursue

## 32.1 What We Have

Two candidate principles for "learning well":

### 32.1.1 Distance Matching (Optimization)

> Good parameterization = parameter distance matches function distance.

- Explains: linear regression optimality, CNN success, ResNet stability
- Measurable via: condition number, Fisher information
- Achievable via: symmetry/equivariance (special case)

### 32.1.2 Entanglement Matching (Expressiveness)

> Good architecture = tensor structure matches target's entanglement structure.

- Explains: why depth matters, why locality helps for images
- Measurable via: entanglement entropy

- Limitation: doesn't explain specific architectural choices (skip, attention)

## 32.2 How They Relate

These are complementary:

| **Principle** | **Asks** | **Answers** |
|---|---|---|
| Entanglement | Which architecture class? | Depth, locality structure |
| Distance | Which parameterization within class? | How to set up parameters |

Together they might give: given a target, first match entanglement structure to choose architecture class, then match distances to choose parameterization.

## 32.3 Open Questions

### 32.3.1 Formalizing Distance Matching

1. How to define "function distance" for general function classes?
2. Given function class $\mathcal{F}$, how to construct a matching $\Theta$?
3. When is perfect matching impossible? What's the best approximation?

### 32.3.2 Measuring Entanglement

1. How to estimate entanglement structure from finite data?
2. What's the entanglement structure of language/sequences?
3. Can we automatically select architecture based on measured entanglement?

### 32.3.3 Connecting to Coalgebras

1. For $F$-coalgebras, what is the "natural" parameterization?
2. Does bisimulation play the role that group symmetry plays for equivariant networks?
3. Can we define "distance" on coalgebra space via behavioral metrics?

### 32.3.4 Unification

1. Is there a single principle that subsumes both distance and entanglement?

2. How does generalization fit in? (Neither principle directly addresses it)

3. What about optimization dynamics? (We focused on landscape, not trajectory)

## 32.4 Directions Not Pursued

We considered but deprioritized:

### 32.4.1 Information Bottleneck

Claim: learning = fitting then compressing.

Problem: controversial. ReLU networks don't compress but still generalize. The phenomenon may be activation-function-specific.

### 32.4.2 Neural Tangent Kernel

Claim: infinite-width networks are kernel machines.

Problem: only applies in "lazy training" regime. Real networks do feature learning, which NTK can't explain.

### 32.4.3 Double Descent

Claim: more parameters can be better after the interpolation threshold.

Problem: describes a phenomenon, doesn't give design guidance.

### 32.4.4 Categorical Deep Learning

Claim: category theory unifies all architectures.

Problem: currently descriptive, not prescriptive. Doesn't tell you which architecture to use.

These may still be valuable, but they don't directly answer our question: how to design parameterizations for learning.

## 32.5 The Research Program

1. **Formalize**: Make distance matching mathematically precise

2. **Compute**: For specific function classes, construct matching parameterizations

3. **Verify**: Check if known good architectures achieve good distance matching

4. **Generalize**: Extend from group symmetry to coalgebraic structure

5. **Apply**: Use the principles to design new architectures

## 32.6 Connection to the Book's Goal

Recall: we want to learn logical structures (coalgebras, Kripke frames, automata).

### 32.6.1 The Emerging Framework

The pieces fit together:

1. **Part I**: Coalgebras define structure. Bisimulation defines equivalence.

2. **Part II**: Semiring relaxation enables gradients and makes behavior *continuous*.

3. **Part III**: Behavioral metrics (from coalgebra theory) define the "right" distance on function space. Distance matching says: make parameter distance match behavioral distance.

### 32.6.2 The Vision

If this framework is correct, architecture design becomes principled:

1. **Specify**: What coalgebra type (functor $H$)?

2. **Compute**: What is the behavioral metric for $H$-coalgebras?

3. **Construct**: Design parameter space $\Theta$ with matching metric.

4. **Train**: Gradient descent converges efficiently because distances match.

5. **Extract**: Threshold back to discrete coalgebra.

This is no longer alchemy. It is engineering from first principles.

### 32.6.3 What Remains

The framework is conceptually complete, but:

- We don't yet know how to *compute* behavioral metrics for general functors

- We don't yet know how to *construct* matching parameterizations
- We don't yet know if this explains *all* successful architectures

These are the open problems. But we now have a clear direction.

### 32.6.4 The Guiding Hypotheses

**Distance matching**: Parameterize so that parameter distance $\approx$ behavioral distance.

**Entanglement matching**: Match architecture depth/structure to the target's entanglement structure.

Together, these may give a complete theory of "learning well."

# Part IV

# Limits

# Chapter 33

# How Far Can We Go?

◎ Goals of This Chapter

- Explore meta-learning: learning to learn
- Ask whether fixed points exist
- Connect to Kolmogorov complexity

## 33.1 Meta-Learning

If learning is coalgebraic (fold/unfold), then:

- Learning algorithms are themselves structures
- We can learn *them* too

$$\begin{aligned} \text{Level 0:} \quad & \text{Data} \to \text{Learn}_0 \to \text{Program} \\ \text{Level 1:} \quad & \text{Learn}_0 \to \text{Learn}_1 \to \text{Better Learner} \\ & \vdots \end{aligned}$$

## 33.2 Fixed Points

**Definition 33.1** (Universal Learner)**.** A *universal learner* is a fixed point: $L = \text{MetaLearn}(L)$.

**Theorem 33.2** (Bounded Complexity)**.** *If* $L = \text{MetaLearn}(L)$ *exists, then* $\text{K}(L) \leq \text{K}(\text{MetaLearn}) + O(1)$.

Proof Sketch

$L$ can be described as "the fixed point of MetaLearn."

## 33.3 Existence Questions

- Does the fixed point exist?
- Is it unique?
- Can we construct it?

# Chapter 34

# The Boundary of Intelligence

◎ Goals of This Chapter

- Understand Kolmogorov complexity and its uncomputability
- Prove that incomprehensible objects exist
- Appreciate the fundamental limits of intelligence

## 34.1 Intelligence as Compression

Following Hutter: **intelligence is the ability to compress**.

**Definition 34.1** (Kolmogorov Complexity). $\mathrm{K}(x) = \min\{|p| : U(p) = x\}$

## 34.2 The Compression Chain

For any object $X$:

$$|X| \geq |\text{compress}(X)| \geq |\text{compress}^2(X)| \geq \cdots$$

**Lemma 34.2.** *Every descending chain in* $\mathbb{N}$ *stabilizes.*

## 34.3 Incomprehensible Objects

**Theorem 34.3** (Existence). *There exist objects* $L$ *with* $\mathrm{K}(L) = |L|$*—no compression is possible.*

**Theorem 34.4** (Prevalence). *For strings of length* $n$*: at least* $(1 - 2^{-c})$ *fraction have* $\mathrm{K}(x) > n - c$*.*

## 34.4 The Uncomputability Barrier

**Theorem 34.5.** K *is uncomputable.*

★ Key Insight

The compression chain exists (by well-foundedness), but we cannot walk it (by uncomputability). Intelligence has a boundary—not because there's nothing beyond, but because **the path is uncomputable**.

**Corollary 34.6** (The Boundary)**.**

- *Incompressible objects exist*
- *They cannot be identified algorithmically*
- *Most objects are incompressible*
- *Intelligence is bounded*

# Appendix A

# Application: Modal Logic for AI Agents

◎ Goals of This Chapter

- See how modal logic applies to AI agent design
- Understand the two-layer architecture
- Connect PDL to agent planning
- Appreciate the synthesis: logic + learning

## A.1 The Problem with Current Agents

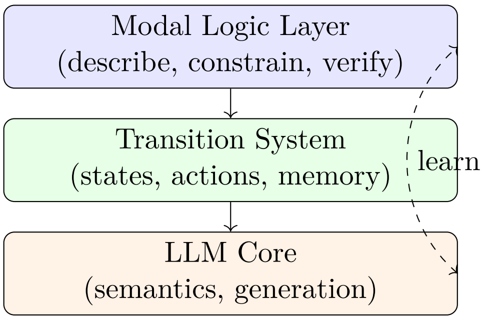
LLM-based agents (ReAct, CoT, Tool Use) are:

- Engineering patchwork, no theoretical foundation
- Hard to verify or constrain
- Essentially automata, but not formalized as such

⚠ Common Pitfall

Current agent frameworks rediscover automata theory and modal logic concepts, but without the rigor. We can do better.

## A.2 The Two-Layer Architecture



**Modal Logic Layer**: Specify constraints, verify properties
**Transition System**: Concrete state machine (coalgebra!)
**LLM Core**: Handle natural language, generate content

## A.3 PDL for Agent Actions

Agent capabilities as atomic actions:

- $\mathsf{search}$ — search the web
- $\mathsf{compute}$ — run calculation
- $\mathsf{ask}$ — query user
- $\mathsf{store}$, $\mathsf{retrieve}$ — memory operations

Complex behaviors as PDL programs:

$$[\mathsf{search}; \mathsf{summarize}]\mathsf{hasAnswer}$$

“search then summarize guarantees an answer”

$$\langle(\mathsf{try}_1 \cup \mathsf{try}_2 \cup \mathsf{try}_3)\rangle\mathsf{success}$$

“one of three attempts can succeed”

## A.4 Relevant Modal Logics

| **Logic** | **Agent Application** |
|---|---|
| Epistemic ($K_a\varphi$) | “Agent knows $\varphi$” |
| Deontic ($O\varphi$, $P\varphi$) | “Agent must/may do $\varphi$” |
| PDL ($[\alpha]\varphi$) | “After action $\alpha$, $\varphi$ holds” |
| Temporal (LTL/CTL) | “Eventually goal”, “Always safe” |
| Graded ($\Diamond_{\geq n}\varphi$) | “At least $n$ options available” |

## A.5 The Synthesis

**From logic**: precision, verifiability, constraints

**From learning**: adaptability, learning from data

**Combined**:

1. Specify agent behavior in modal logic
2. Use semiring-valued coalgebra for soft version
3. Train via gradient descent
4. Extract crisp transition system
5. Verify against modal specifications

**Example A.1** (Safe Agent)**.** Specification: "Always have at least 2 safe actions available."

$$\mathsf{AG}\,(\Diamond_{\geq 2}\mathsf{safe})$$

Train soft model → converge to crisp → verify specification holds.

## A.6 Learning Modal Abstractions

Key insight: don't expose raw memory/tape to reasoning.

Instead, abstract into modal operators:

$$\begin{array}{rl} \Box_{\mathsf{memory}}\varphi & \text{“everything in memory satisfies } \varphi\text{”} \\ \Diamond_{\mathsf{action}}\mathsf{goal} & \text{“some action can reach goal”} \end{array}$$

The agent reasons at the modal level. The transition system implements it.

★ Key Insight

Modal logic provides the *specification language* for agents. Coalgebra provides the *implementation structure*. Semiring learning provides the *training method*. Together: learnable, verifiable agents.

## A.7 Open Problems

1. How to automatically translate natural language goals to modal formulas?
2. How to handle uncertainty? (Probabilistic modal logic)
3. How to learn the right modal abstractions?
4. Scalability to real-world agent complexity?

# Afterword

We began with logic, passed through coalgebra and semirings, and arrived at the limits of intelligence.

The journey reveals a unity: **expression, learning, and limits are all algebraic**. The same structures—functors, fixed points, descent in well-founded orders—appear at every level.

Much remains unknown. The main conjecture (syntax-semantics optimality) is unproven. The existence of universal learners is unclear. The boundary between computable and incomputable intelligence is sharp, but its exact location is undetermined.

These are questions for the future.