Logic Tensor Networks

Acile Chahboun

Seminar Formal Methods and Automated Reasoning

Introduction

- Logic Tensor Networks, a neurosymbolic framework combining neural networks with first-order logic
- Logical Knowledge and data are learned together in a single system
- Built around a fully differentiable logical language called Real Logic

---

Reference: Badreddine, S., d'Avila Garcez, A., Serafini, L., Spranger, M. (2021). Logic Tensor Networks. *arXiv:2012.13635*.

Key Terminology

- **Neurosymbolic AI:** An AI approach that combines neural networks for learning with symbolic logic for reasoning.
- **Neural Network:** A model that learns patterns from data by adjusting numerical parameters.
- **Symbolic AI:** An AI approach that represents knowledge explicitly using symbols, rules, and logical relations.
- **First-Order Logic:** A formal logic system built from objects, variables, predicates, functions, and quantifiers.
- **Logic Tensor Network:** A framework that represents logical knowledge in a numerical, differentiable form so it can be used with neural networks.

Key Terminology

- **Real Logic:** The logic used in LTNs. Unlike classical logic, statements can be partly true, with truth values between 0 and 1.
- **Truth Value:** A number measuring how true a statement is, e.g. $\texttt{Cat}(\texttt{image}) = 0.92$.
- **Grounding:** The process of assigning real numerical meaning to logical symbols. An image becomes a tensor, and $\texttt{Cat}(x)$ becomes a neural classifier.
- **Tensor:** A numerical object used by neural networks. Scalars, vectors, matrices, and higher-dimensional arrays are all tensors.
- **Constant:** A symbol for a specific object, e.g. `Alice`, `Paris`, `image1`.

Key Terminology

- **Variable:** A placeholder that can range over many objects, e.g. x in $\texttt{Cat}(x)$.
- **Domain:** The type or category of objects a symbol belongs to, e.g. $\texttt{Person}$, $\texttt{City}$, $\texttt{Image}$.
- **Function:** A mapping from one or more objects to another object, e.g. $\texttt{fatherOf}(x)$.
- **Predicate:** A statement about objects that returns a truth value, e.g. $\texttt{Friend}(\texttt{Alice}, \texttt{Bob})$ or $\texttt{Digit}(\texttt{image}, 8)$.
- **Formula:** A logical expression built from predicates, variables, connectives, and quantifiers, e.g. $\texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$.

Key Terminology

- **Axiom:** A formula treated as background knowledge that the model should satisfy, e.g. “All cats are animals.”
- **Connective:** A logical operator such as AND, OR, NOT, or IMPLIES.
- **Quantifier:** A logical operator that talks about many objects. $\forall x$ means “for all x” and $\exists x$ means “there exists an x.”
- **Aggregation:** The process of combining many truth values into one truth value, often used for quantifiers.
- **Satisfiability:** A score measuring how well the model satisfies all given facts and rules.

Key Terminology

- **Learning:** Adjusting the model's parameters to maximize satisfiability.
- **Querying:** Asking the trained model how true a statement is, e.g. $\texttt{Digit}(\texttt{image1}, \texttt{8}) = 0.87$.
- **Reasoning:** Using known facts and rules to determine whether another statement follows.
- **Differentiability:** A mathematical property that allows the system to be trained using gradient-based learning.
- **Gradient:** A signal that tells the model how to change its parameters during training.

Key Terminology

- **Fuzzy Logic:** A logic system where statements can be partly true rather than only true or false.
- **Knowledge Base:** A collection of facts and rules expressed as logical formulas that the model must satisfy.
- **Semantic Loss:** A training loss derived from logical constraints, penalizing the model when formulas in the knowledge base are not satisfied.
- **t-norm:** A function used to compute the fuzzy AND of two truth values, e.g. product t-norm: $a \cdot b$.
- **t-conorm:** A function used to compute the fuzzy OR of two truth values, e.g. $a + b - a \cdot b$.

Motivation

- AI systems need to do more than recognize patterns, they should be able to learn from data, reason about what they have learned, and use that knowledge to make decisions.
- Modern deep learning is strong at learning from raw data using sub-symbolic representations such as vectors and tensors.
- However, reasoning often requires a higher-level language that can express rules, relationships, and constraints clearly.

The Core Problem

Learning / Neural	**Reasoning / Symbolic**
Learns patterns from data	Derives conclusions from facts and rules
Uses vectors and tensors	Uses symbols and logical formulas
Works well with noisy real-world inputs	Works well with explicit structured knowledge
Statistical generalization	Logical inference
Bottom-up: data $\rightarrow$ patterns	Top-down: rules $\rightarrow$ conclusions
Approximate and probabilistic	Precise and rule-based
Knowledge is implicit	Knowledge is explicit and human-readable
What is likely?	*What must follow?*

The Core Problem

Neural systems struggle with:	Symbolic systems struggle with:
Explicit logical rules	Learning directly from raw data
Long reasoning chains	Perception tasks such as images or signals
Guarantees and truthfulness	Noise, uncertainty, and exceptions
Interpretability	Scalability to large real-world datasets

Core Issue

Modern AI needs both sides: the ability to learn from data and the ability to reason with explicit knowledge. LTN addresses this by combining neural learning with differentiable first-order logic.

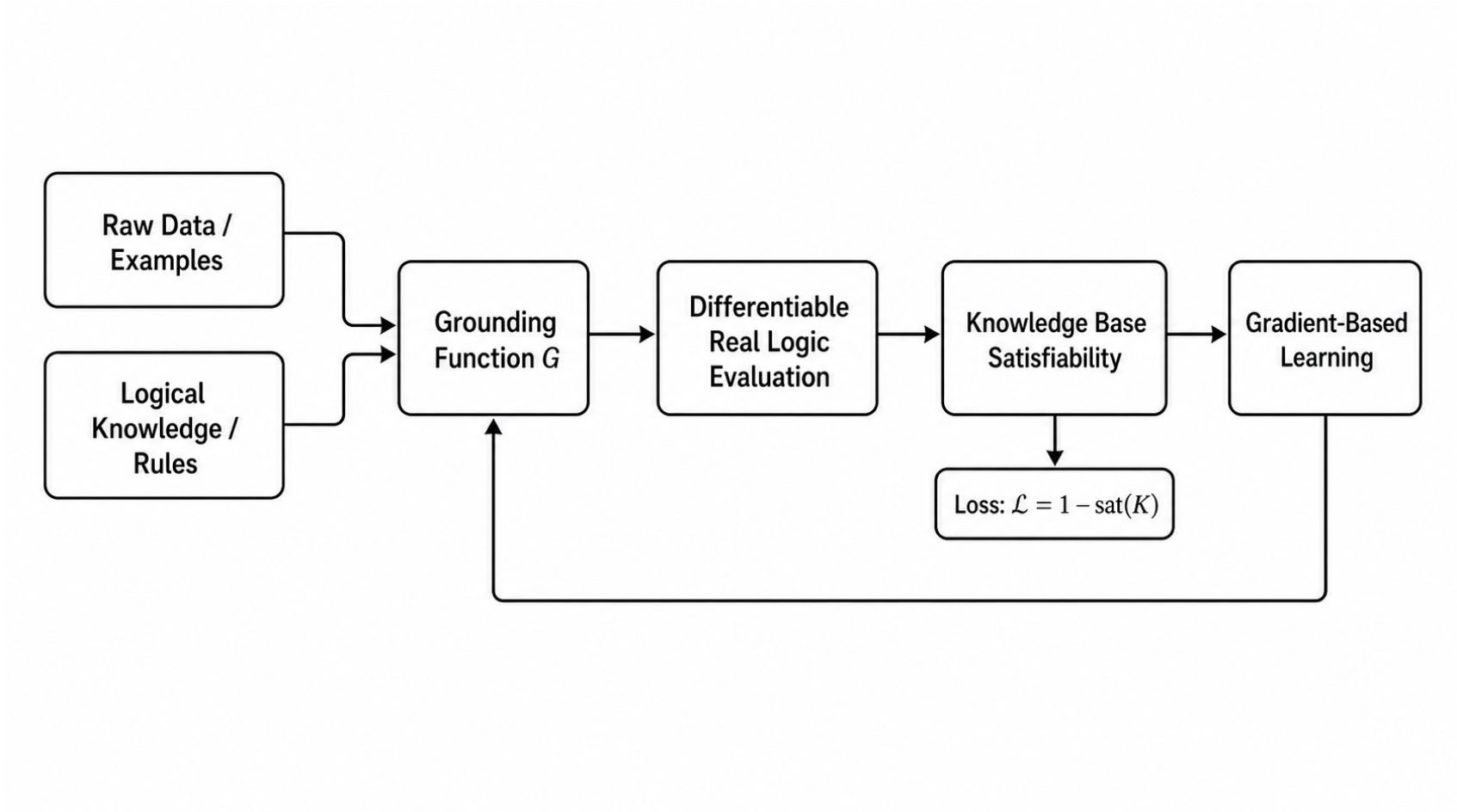
NLM Recap

- **Neural Logic Machines (NLMs)** are neural models designed to perform logic-like reasoning over objects and their relationships.
- Instead of treating data as isolated examples, they represent objects, predicates, and relations using tensors.
- This allows the model to reason about structured information such as $\texttt{friend}(x, y)$, $\texttt{parent}(x, y)$, or $\texttt{connected}(x, y)$.
- NLMs can learn relational patterns and generalize across different numbers of objects, making them useful for tasks involving graphs, rules, and object interactions.

NLM Recap

- The logic in NLMs is learned **implicitly** from data. The model may discover rule-like patterns, but those rules are not always written explicitly in a human-readable form.
- **This is where LTNs differ:** LTNs allow logical knowledge to be stated directly as formulas, grounded into tensors, and optimized through differentiable Real Logic.

Logic Tensor Networks Overview



Real Logic: Syntax

- **Real Logic uses a first-order logical language:** Built from symbols that represent objects, variables, functions, and relations.
- **Domains** define types of objects, e.g. $\texttt{Person}$, $\texttt{City}$, $\texttt{Image}$, $\texttt{Digit}$.
- **Constants** name specific objects, e.g. $\texttt{Alice}$, $\texttt{Paris}$, $\texttt{img}_1$.
- **Variables** range over objects in a domain, e.g. x may range over all images.
- **Functions** map objects to other objects, e.g. $\texttt{fatherOf}(x)$ returns a person.
- **Predicates** express relations or properties, e.g. $\texttt{Cat}(x)$, $\texttt{Friend}(x,y)$, $\texttt{Digit}(\texttt{img},8)$.

- **Formulas** combine predicates using logical operators, e.g. $\texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$ and $\texttt{Friend}(x,y) \wedge \texttt{Friend}(y,z)$.
- **Quantifiers** express rules over many objects, e.g. $\forall x\ \texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$ and $\exists x\ \texttt{Digit}(x,8)$.

To summarize

Real Logic syntax defines what counts as a valid logical expression before those expressions are grounded into tensors and evaluated numerically.

Real Logic: Semantics

- **Semantics gives meaning to the syntax:** It explains how logical symbols and formulas are interpreted numerically.
- **Symbols are grounded into real-valued objects:** Constants, variables, functions, and predicates are mapped to tensors or tensor operations.
- **Predicates return truth values in** $[0, 1]$**:** e.g. $\mathtt{Cat}(\mathtt{img}_1) = 0.92$ means the model strongly believes $\mathtt{img}_1$ is a cat.
- **Functions become real-valued mappings:** e.g. $\mathtt{fatherOf}(x)$ is interpreted as a function over grounded objects.

- **Logical connectives become fuzzy operators:** AND, OR, NOT, and IMPLIES are computed using differentiable fuzzy-logic operations.
- **Quantifiers become aggregators:** $\forall x$ and $\exists x$ combine many truth values into one score. e.g. $\forall x \, \texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$ is evaluated by checking the rule over many grounded examples and aggregating the results.

To summarize

Real Logic semantics turns logical formulas into differentiable numerical computations, so they can be optimized during neural-network training.

Grounding Intuition

- **Grounding connects logic to data:** Logical symbols are given numerical meaning so a neural model can process them.
- **Objects become tensors:** e.g. an image becomes a pixel tensor; a person or word may become an embedding vector.
- **Predicates become functions or neural networks:** e.g. $\texttt{Cat}(x)$ can be a classifier that returns how true it is that x is a cat.
- **Relations become truth-valued mappings:** e.g. $\texttt{Friend}(x, y)$ returns a score between 0 and 1 for each pair of objects.

- **Rules become differentiable computations:** A formula such as $\texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$ is evaluated numerically, not just symbolically.
- **Learning adjusts the grounding:** The model changes embeddings, predicates, and functions so the logical rules become more satisfied.

To summarize

Grounding is the bridge between symbolic logic and neural learning: it turns symbols and formulas into tensors, functions, and truth scores.

Semantics of Real Logic: Tensor Intuition

Classical first-order logic

A variable assignment gives each variable one value at a time:

$$x = a_i, \quad y = b_j \qquad \Rightarrow \qquad I(f(x,y)) = I(f)(a_i, b_j)$$

Real Logic / LTNs

Variables are grounded over whole batches:

$$\mathscr{G}(x) = A = [a_1, a_2, \ldots] \qquad \mathscr{G}(y) = B = [b_1, b_2, \ldots]$$

$$\mathscr{G}(f(x,y))_{i,j} = \mathscr{G}(f)(A[i], B[j])$$

Semantics of Real Logic: Tensor Intuition

Tensor view

	b_1	b_2	b_3
a_1	$f(a_1, b_1)$	$f(a_1, b_2)$	$f(a_1, b_3)$
a_2	$f(a_2, b_1)$	$f(a_2, b_2)$	$f(a_2, b_3)$

Each entry: $x = A[i],\ y = B[j]$.

Main idea

variables over domains $\rightarrow$ all assignments $\rightarrow$ tensor of truth values in $[0, 1]$

Each entry is a fuzzy truth value in $[0, 1]$.

Grounding Terms & Predicates

- **Grounding** maps logical symbols into numerical objects via a grounding function $\mathscr{G}$:

$$\mathscr{G} : \text{logical symbols} \rightarrow \text{numerical objects}$$

- **Constants become tensors:** e.g. $\mathscr{G}(\mathtt{img}_1) = v_{\mathtt{img}_1} \in \mathbb{R}^{28\times 28\times 1}$
- **Variables become batches of tensors:** if $x = [\mathtt{img}_1, \mathtt{img}_2, \mathtt{img}_3]$ then $\mathscr{G}(x) \in \mathbb{R}^{n\times H\times W\times C}$
- **Functions become numerical mappings:**

$$\mathscr{G}(f(x)) = \mathscr{G}(f)(\mathscr{G}(x)), \quad \mathscr{G}(f) : \mathbb{R}^d \rightarrow \mathbb{R}^m$$

Grounding Terms & Predicates

- **Predicates become truth-valued functions:**

$$\mathscr{G}(\texttt{Cat}(x)) = \mathscr{G}(\texttt{Cat})(\mathscr{G}(x)) = \sigma(\mathrm{MLP}(v_{\texttt{img}_1})) = 0.92$$

- **Predicates over batches return truth-value tensors:**

$$\mathscr{G}(\texttt{Cat}(x)) = [0.92,\ 0.13,\ 0.77], \quad \mathscr{G}(\texttt{Friend}(x,y)) \in [0,1]^{n \times m}$$

- **Rules become differentiable computations:**

$$\mathscr{G}(\texttt{Cat}(x) \rightarrow \texttt{Animal}(x)) = [0.99,\ 1.00,\ 0.93]$$

$$\mathscr{G}(\forall x\, \texttt{Cat}(x) \rightarrow \texttt{Animal}(x)) = \mathrm{Agg}([0.99, 1.00, 0.93]) = 0.97$$

Connectives

Logical connectives combine statements into larger formulas. In Real Logic, truth values lie in $[0,1]$, so connectives combine numbers, not just Boolean values.

Negation (NOT)

$$\neg P(x) = 1 - P(x) \qquad \text{e.g. } 1 - 0.8 = 0.2$$

Conjunction (AND)

$$P(x) \wedge Q(x) = P(x) \cdot Q(x) \qquad \text{e.g. } 0.8 \cdot 0.6 = 0.48$$

Disjunction (OR)

$$P(x) \vee Q(x) = P(x) + Q(x) - P(x) \cdot Q(x) \qquad \text{e.g. } 0.8 + 0.6 - 0.48 = 0.92$$

Connectives

Implication (IF...THEN)

$$P(x) \rightarrow Q(x)$$

The rule is violated mainly when $P(x)$ is high but $Q(x)$ is low. e.g. $\texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$ is satisfied when anything classified as a cat is also classified as an animal.

Why connectives matter

Connectives turn logical rules into differentiable numerical computations:

$$\texttt{Cat}(x) \wedge \texttt{Small}(x) \longrightarrow \mathscr{G}(\texttt{Cat}(x)) \cdot \mathscr{G}(\texttt{Small}(x))$$

Logical formulas can then be evaluated, optimized, and learned through gradient-based training.

Quantifiers & Aggregation

Quantifiers evaluate a rule over many examples and return one final truth score.

Universal quantifier $\forall$

$\forall x\ \texttt{Cat}(x) \rightarrow \texttt{Animal}(x)$ — for every x, if x is a cat, then x is an animal.

$$\text{Agg}_{\forall}([0.99,\ 0.88,\ 0.94,\ 0.30]) \rightarrow \text{one score}$$

Behaves like a **soft minimum** — a low score pulls the final result down.

Existential quantifier $\exists$

$\exists x\ \texttt{Red}(x)$ — there exists at least one red object.

$$\text{Agg}_{\exists}([0.10,\ 0.20,\ 0.95,\ 0.15]) \rightarrow \text{one score}$$

Behaves like a **soft maximum** — a high score raises the final result.

Aggregation: generalized mean

$$M_p(t_1, \ldots, t_n) = \left(\frac{t_1^p + \cdots + t_n^p}{n} \right)^{1/p}$$

- $p = 1 \rightarrow$ normal average
- large $p \rightarrow$ closer to maximum ($\exists$)
- soft-min form $\rightarrow$ closer to minimum ($\forall$)

$$\mathrm{Agg}_{\forall}([t_1, \ldots, t_n]) = 1 - M_p(1 - t_1, \ldots, 1 - t_n)$$

Main idea

$\forall$ = soft "all" $\exists$ = soft "at least one", aggregation makes quantified rules differentiable and trainable.

Quantifier Subtlety

In classical logic, some formulas are logically equivalent. In LTNs, they may **not** be numerically equivalent because quantifiers use **soft aggregation**.

Classical logic equivalence

$$\forall x\,(\phi(x) \wedge \psi(x)) \;\equiv\; (\forall x\,\phi(x)) \wedge (\forall x\,\psi(x))$$

Both sides evaluate to **False** when any instance fails, the equivalence holds.

Quantifier Subtlety

In LTNs: soft conjunction $a \wedge b := a \cdot b$, universal aggregation $A_\forall =$ average.

Example values

x	$\phi(x)$	$\psi(x)$	$\phi(x) \cdot \psi(x)$
x_1	0.9	0.8	$0.9 \cdot 0.8 = 0.72$
x_2	0.4	0.7	$0.4 \cdot 0.7 = 0.28$

Left-hand side

$$\forall x\,(\phi(x) \wedge \psi(x)) = A_\forall([0.72,\ 0.28]) = \frac{0.72 + 0.28}{2} = 0.50$$

Right-hand side

$$(\forall x\,\phi(x)) \wedge (\forall x\,\psi(x)) = \frac{0.9 + 0.4}{2} \cdot \frac{0.8 + 0.7}{2} = 0.65 \cdot 0.75 = 0.4875$$

Quantifier Subtlety

Final comparison

Setting	Formula 1	Formula 2	Same?
Classical logic	False	False	Yes
LTN / Real Logic	0.50	0.4875	No

Main idea

In classical logic:

$$\forall x(\phi(x) \wedge \psi(x)) \equiv (\forall x\, \phi(x)) \wedge (\forall x\, \psi(x))$$

But in LTNs:

$$A_\forall(\phi(x) \wedge \psi(x)) \neq A_\forall(\phi(x)) \wedge A_\forall(\psi(x))$$

because aggregation does **not** generally commute with fuzzy connectives.

Quantifier Subtlety: Variable Groundings

Classical logic

In classical logic, renaming a bound variable does not change the meaning:

$$\forall x\, p(x) \equiv \forall y\, p(y)$$

Both are interpreted over the same domain.

Real Logic / LTNs

Each variable name is associated with its own grounding $\mathscr{G}(x)$, $\mathscr{G}(y)$. So the formulas behave like:

$$\forall x \in \mathscr{G}(x)\, p(x) \qquad \forall y \in \mathscr{G}(y)\, p(y)$$

If $\mathscr{G}(x) \neq \mathscr{G}(y)$, the two formulas are evaluated over different tensors and can have different meanings.

Quantifier Subtlety: Binary Classification

Two variables, two groundings

$$\mathscr{G}(x^+) = \text{batch of positive points} \qquad \mathscr{G}(x^-) = \text{batch of negative points}$$

The two axioms:

$$\forall x^+ \, A(x^+) \qquad \forall x^- \, \neg A(x^-)$$

are **not contradictory** — the quantifiers range over different grounded domains.

Binary classification view

Formula	Grounding	Meaning
$\forall x^+ \, A(x^+)$	positive examples	positives should satisfy A
$\forall x^- \, \neg A(x^-)$	negative examples	negatives should not satisfy A

Main idea

Variable names are tied to their groundings:

$$\text{variable name} \rightarrow \text{grounded tensor} \rightarrow \text{aggregation range}$$

Quantifiers determine **which tensor or domain is aggregated over**.

Diagonal Quantification

Diagonal quantification is used when two variables are already **paired** in the data.

Example pairing

Image	Correct label
$\texttt{img}_1$	cat
$\texttt{img}_2$	dog
$\texttt{img}_3$	bird

Normal quantification — all combinations

Evaluates $x \times y$: every image with every label $\rightarrow 3 \times 3 = 9$ comparisons. May include irrelevant pairs such as $\texttt{CorrectLabel}(\texttt{img}_1, \texttt{dog})$.

Diagonal quantification — matched pairs only

Evaluates only $(\texttt{img}_1, \texttt{cat})$, $(\texttt{img}_2, \texttt{dog})$, $(\texttt{img}_3, \texttt{bird}) \rightarrow 3$ comparisons.

$$\forall \mathrm{Diag}(x, y)\ \texttt{CorrectLabel}(x, y)$$

Diagonal Quantification

Comparison

Normal quantification	Diagonal quantification
Every x with every y	Only x_i with y_i
$n \times n$ pairs	n pairs
Useful for all-pairs reasoning	Useful for paired datasets
May include irrelevant pairs	Preserves intended pairs
More computationally expensive	More efficient

Main idea

Normal quantification compares everything with everything. Diagonal quantification compares each item only with its paired item. Especially useful for:

$$\text{image} \to \text{label} \qquad \text{question} \to \text{answer} \qquad \text{input} \to \text{target}$$

Guarded Quantifiers

Guarded quantifiers apply a rule only to a **specific subset** of examples.

$$\forall x\ P(x) \quad \longrightarrow \quad \forall x : \mathit{Guard}(x)\ P(x)$$

Simple intuition

$$\text{guarded quantifier} = \text{quantifier} + \text{filter}$$

$$\forall x : \texttt{AnimalImage}(x)\ \texttt{HasAnimalLabel}(x)$$

For every image that is an animal image, it should have an animal label.

Effect of the guard

Example	**AnimalImage(x)**	**Included?**
cat image	True	✓
dog image	True	✓
car image	False	
chair image	False	

Guarded Quantifiers

Guard as a mask

$$P(x) = [0.91,\ 0.84,\ 0.30,\ 0.20] \qquad \text{Mask}(x) = [1,\ 1,\ 0,\ 0]$$

The guarded quantifier aggregates only $[0.91,\ 0.84]$, not the full vector.

$$\text{Mask}(x) = \begin{cases} 1 & \text{if } \mathit{Guard}(x) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

Guarded quantifier vs implication

Guarded quantifier	Implication form
Filters examples first	Evaluates every example
Ignores out-of-guard examples	Still scores them
Aggregates selected cases only	Aggregates all cases
Better for subset rules	Affected by irrelevant cases

Main idea

Do not apply this rule to everything, apply it only to examples selected by the guard.
Makes rules more precise when only part of the dataset should contribute.

Objective / Thesis

LTNs are not only a supervised learning method — they are a general **neurosymbolic framework** for expressing different AI tasks as logical constraints.

Core technical idea

Instead of only minimizing prediction error, LTNs maximize knowledge base satisfiability:

$$\max_{\theta} \mathit{Sat}_{\mathscr{G}_\theta}(\mathscr{K})$$

where $\mathscr{K}$ = knowledge base, $\mathscr{G}_\theta$ = grounding with learnable parameters, Sat = satisfiability score.

Training objective

$$\mathscr{L}(\theta) = 1 - \mathit{Sat}_{\mathscr{G}_\theta}(\mathscr{K}) \qquad \Rightarrow \qquad \min_{\theta} \mathscr{L}(\theta)$$

The model adjusts θ so the facts and rules in the knowledge base become as true as possible.

Objective / Thesis

Different tasks, one framework

Task	LTN view
Supervised learning	Labeled facts should be satisfied
Semi-supervised learning	Labels plus rules over unlabeled data
Classification	Class predicates output truth scores
Regression	Functions grounded as learnable mappings
Clustering	Similar objects share cluster structure
Relational learning	Predicates model relations between objects
Query answering	Ask for the truth value of a formula
Reasoning	Test whether a conclusion follows from $\mathscr{K}$

Main thesis

$$\text{data} + \text{rules} \rightarrow \text{grounded formulas} \rightarrow \text{satisfiability} \rightarrow \text{gradient-based learning}$$

LTNs turn machine-learning problems into optimization over logical satisfiability, using one differentiable logical language for learning, reasoning, and querying.

Learning & Reasoning

Learning

Adjust learnable parameters θ so the knowledge base becomes as satisfied as possible:

$$\max_{\theta} \mathit{Sat}_{\mathcal{G}_\theta}(\mathscr{K}) \quad \Leftrightarrow \quad \min_{\theta} \mathscr{L}(\theta) = 1 - \mathit{Sat}_{\mathcal{G}_\theta}(\mathscr{K})$$

What gets learned?

Component	Example
Predicate grounding	Learn $\texttt{Cat}(x)$ as a classifier
Function grounding	Learn $\texttt{price}(x)$ as a regression function
Object embeddings	Learn vector representations for entities
Relation predicates	Learn $\texttt{Friend}(x,y)$, $\texttt{Similar}(x,y)$

Learning & Reasoning

Reasoning

Use the learned knowledge base to evaluate or infer new statements.

$$\texttt{Cat}(x) \rightarrow \texttt{Animal}(x), \quad \texttt{Cat}(\texttt{img}_1) = 0.95$$

$$\text{Query: } \texttt{Animal}(\texttt{img}_1)? \quad \Rightarrow \quad \texttt{Animal}(\texttt{img}_1) = 0.93$$

Learning vs reasoning

Learning	Reasoning
Adjusts model parameters	Uses the trained model
Maximizes satisfiability	Evaluates new formulas
Changes θ	Keeps θ fixed
Happens during training	Happens during evaluation

Main idea

Learning: make the knowledge base true. Reasoning: ask what follows from it. Both are done through grounded formulas with truth values in $[0, 1]$.

Reasoning by Refutation

Reasoning by refutation tests a conclusion by trying to **break it**: can the knowledge base stay satisfied while the conclusion becomes false?

Setup

We want to know whether $\mathscr{K} \models \phi$, i.e. does ϕ follow from the knowledge base?

Refutation idea

Try to find a counterexample where $\mathscr{K}$ is true but ϕ is false:

$$Sat(\mathscr{K}) \text{ should be high} \qquad \text{while} \qquad \mathscr{G}(\phi) \text{ should be low}$$

Optimization view

$$\max Sat(\mathscr{K}) \quad \text{while minimizing} \quad \mathscr{G}(\phi)$$

Make $\mathscr{K}$ true and ϕ false.

Reasoning by Refutation

Possible outcomes

Outcome	Meaning
Counterexample found	ϕ does not follow from $\mathscr{K}$
No counterexample found	ϕ is supported by $\mathscr{K}$
$\mathscr{K}$ unsatisfiable	Knowledge base may be inconsistent

Simple example

$$\texttt{Cat}(x) \rightarrow \texttt{Animal}(x), \quad \texttt{Cat}(\texttt{img}_1) = 0.95$$

Refutation attempt: set $\texttt{Animal}(\texttt{img}_1) \approx 0$. But then $\texttt{Cat}(\texttt{img}_1) \rightarrow \texttt{Animal}(\texttt{img}_1)$ becomes poorly satisfied. So the conclusion is **supported**.

Main idea

Can the rules be true while the conclusion is false?

Yes $\Rightarrow$ conclusion fails No $\Rightarrow$ conclusion is supported

Reasoning by refutation turns logical reasoning into an optimization problem.

Example: Reasoning by Refutation

Setup

Knowledge base: $\mathscr{K} = \{A \vee B\}$ Query: $(A \vee B) \models_q A$?
Does A necessarily follow if $A \vee B$ is true? **Answer: No.**

Grounding

$$\mathscr{G}(A) = a, \quad \mathscr{G}(B) = b, \quad a, b \in [0,1]$$

$$\mathscr{G}(A \vee B) = a + b - ab \qquad \text{(probabilistic sum)}$$

Possible groundings

Grounding	$\mathscr{G}(A)$	$\mathscr{G}(B)$	$\mathscr{G}(A \vee B)$	**Interpretation**
$\mathscr{G}_1$	1	1	1	both true
$\mathscr{G}_2$	1	0	1	only A true
$\mathscr{G}_3$	0	1	1	only B true

$\mathscr{G}_3$ is the counterexample: $\mathscr{G}_3(A \vee B) = 1$ but $\mathscr{G}_3(A) = 0$.

Example: Reasoning by Refutation

Step 4 & 5: Try a candidate grounding

Choose $\mathscr{G}(A) = 0$, $\mathscr{G}(B) = 1$. Then:

$$\mathscr{G}(A \vee B) = 0 + 1 - (0)(1) = 1 \geq 0.95 \quad \checkmark$$

The knowledge base is satisfied.

Step 6 & 7: Check the conclusion

$$\mathscr{G}(A) = 0 < 0.95$$

The conclusion is not satisfied. Counterexample found:

$\mathscr{G}(A)$	$\mathscr{G}(B)$	$\mathscr{G}(A \vee B)$	$\mathscr{G}(A) < q$?
0	1	1	yes

$$\therefore \quad (A \vee B) \not\models_q A$$

Example: Reasoning by Refutation

Main idea

$A \vee B$ does not imply A because B alone can make $A \vee B$ true.
Reasoning by refutation searches for:

knowledge base true enough but conclusion false enough

If such a case exists, the conclusion does **not** follow.

Experimental Setup: Binary Classification

Goal

Classify examples as positive or negative using a predicate $A(x)$, where $A(x)$ is the truth value that x belongs to the positive class.

Data & grounding

$$x^+ = \text{positive examples} \qquad x^- = \text{negative examples}$$

$$\mathscr{G}(A)(x) = \sigma(\mathrm{MLP}(x)) \in [0,1]$$

e.g. $A(x) = 0.91$ means the model strongly believes x is positive.

Training objective

$$\max_\theta \ Sat_{\mathscr{G}_\theta}(\mathscr{K}) \quad \Leftrightarrow \quad \min_\theta \ \mathscr{L}(\theta) = 1 - Sat_{\mathscr{G}_\theta}(\mathscr{K})$$

Training pushes $A(x^+) \to 1$ and $A(x^-) \to 0$.

Experimental Setup: Binary Classification

Knowledge base

$$\mathscr{K} = \{\, \forall x^+ \; A(x^+), \quad \forall x^- \; \neg A(x^-) \,\}$$

Formula	Meaning
$\forall x^+ \; A(x^+)$	all positive examples should be classified positive
$\forall x^- \; \neg A(x^-)$	all negative examples should not be classified positive

Results: Binary Classification

Expected behavior after training

Example type	Predicate value	Meaning
Positive x^+	$A(x^+) \approx 1$	classified as positive
Negative x^-	$A(x^-) \approx 0$	classified as negative
Negative x^-	$\neg A(x^-) \approx 1$	satisfies negative rule

Result in LTN terms

$$\forall x^+ \; A(x^+) \approx 1 \qquad \forall x^- \; \neg A(x^-) \approx 1$$

$$Sat_{\mathcal{G}_\theta}(\mathcal{K}) \to 1 \qquad \mathcal{L}(\theta) \to 0$$

Accuracy vs satisfiability

Metric	What it asks
Accuracy	Did the prediction cross the threshold?
Satisfiability	How strongly did it satisfy the logical formula?

e.g. $A(x^+) = 0.60$ may count as correct with threshold 0.5, but the rule is only weakly satisfied in LTN terms.

Strengths

Main strengths

Strength	Why it matters
Unified framework	Many tasks written in the same Real Logic language
Uses background knowledge	Rules, constraints, and labels in one knowledge base
Differentiable logic	Formulas trained with gradient-based methods
Handles uncertainty	Truth values in $[0, 1]$, not just true/false
Supports multiple tasks	Classification, regression, clustering, querying
More interpretable	Some knowledge explicitly written as formulas
Works with neural predicates	Predicates grounded as MLPs or classifiers

Strengths

Main strength

A normal neural network learns from data: $x \rightarrow y$. An LTN learns from data **and** rules:

$$\text{examples} + \text{logical constraints} \rightarrow \text{satisfiability optimization}$$

LTNs make logic trainable, turning symbolic knowledge into differentiable objectives that guide neural learning.

Challenges

Main challenges

Challenge	Why it matters
Choosing the right formulas	Poorly written rules can mislead the model
Aggregation sensitivity	Formula structure affects satisfaction and gradients
Fuzzy semantics choices	Different t-norms change model behavior
Scalability	Quantifiers over many variables create large products
Optimization difficulty	Nonlinear objectives can be hard to train
Partial interpretability	Neural groundings can still be black boxes
No perfect reasoning guarantee	Optimization may find local optima

Main challenge

Writing a formula is also designing a training objective. $\forall x(\phi(x) \wedge \psi(x))$ and $(\forall x\, \phi(x)) \wedge (\forall x\, \psi(x))$ behave differently during training. The question is not only *what* to encode, but *how* to encode it so the optimization behaves correctly.

logical syntax affects numerical learning

Open Questions

Key open questions

Question	Why it matters
Best fuzzy logic operators?	Different t-norms change training behavior
How to write good formulas?	Syntax affects the final loss and gradients
Scalability for large KBs?	Many variables create large tensor computations
Reliability of refutation?	No counterexample found $\neq$ none exists
Handling conflicting rules?	Real KBs may contain noisy or inconsistent rules
Interpretability of groundings?	Learned predicates may still be black boxes
Compete with task-specific models?	Specialized models may outperform on benchmarks

Open Questions

Main idea

LTNs give a powerful framework, but practical success depends on:

$$\text{good rules} + \text{good grounding} + \text{stable optimization} + \text{scalable computation}$$

The bigger question: can we build AI systems that learn from data, reason with rules, and remain scalable?

Conclusion

What LTNs achieve

Component	Role in LTN
Data	Provides examples and observations
Logic	Provides rules, constraints, and structure
Grounding	Maps symbols to tensors, functions, and predicates
Fuzzy truth values	Allow statements to be partly true
Satisfiability	Measures how well the knowledge base is respected
Gradient learning	Updates neural components to increase satisfiability

Final takeaway

$$\text{LTNs} = \text{neural networks} + \text{first-order logic} + \text{differentiable satisfiability}$$

LTNs show that logic does not have to remain separate from deep learning:

$$\text{logical formulas} \rightarrow \text{differentiable computations} \rightarrow \text{trainable AI systems}$$

Logic Tensor Networks are a flexible neurosymbolic framework for **learning**, **reasoning**, and **querying** under logical constraints.

Thank you for listening.

Questions?

Acile Chahboun
Seminar Formal Methods and Automated Reasoning