RWTH AACHEN UNIVERSITY

Query Languages for Machine-Learning Models

Martin Grohe
joint work with Christoph Standke, Juno Steegmans, Jan Van den Bussche

Understanding ML Models

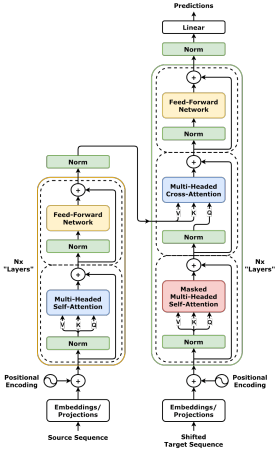


Image source: Wikipedia

Understanding ML Models

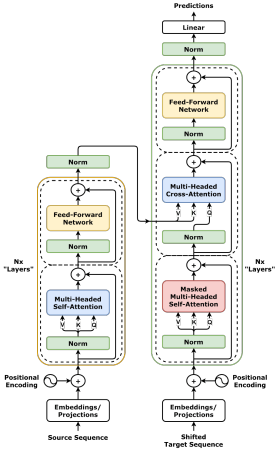


Image source: Wikipedia

Verification

- Prove that the model computes the correct function.

Understanding ML Models

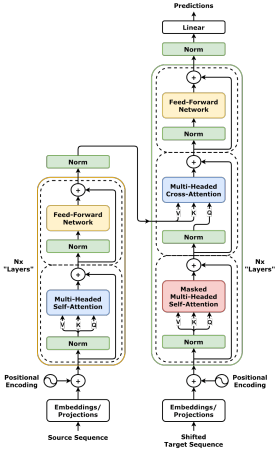


Image source: Wikipedia

Verification

- Prove that the model computes the correct function.
- Prove that the model satisfies certain requirements or desiderata.

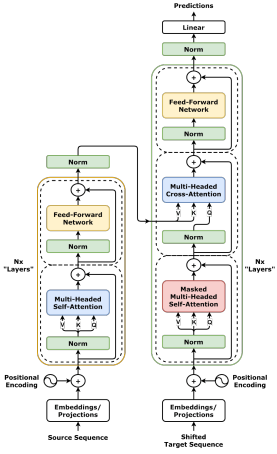


Image source: Wikipedia

Understanding ML Models

Verification

- Prove that the model computes the correct function.
- Prove that the model satisfies certain requirements or desiderata.

Explainable AI (XAI)

- Explain to humans how the model works.

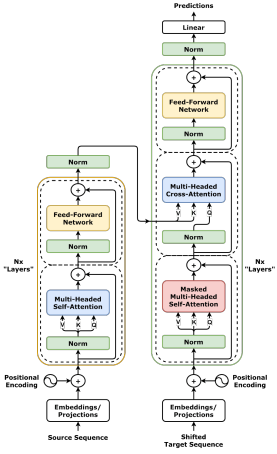


Image source: Wikipedia

Understanding ML Models

Verification

- Prove that the model computes the correct function.
- Prove that the model satisfies certain requirements or desiderata.

Explainable AI (XAI)

- Explain to humans how the model works.
- Explain why the model reached a decision.

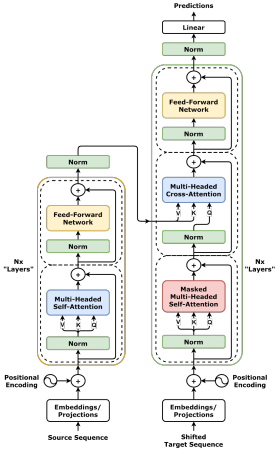


Image source: Wikipedia

Understanding ML Models

Verification

- Prove that the model computes the correct function.
- Prove that the model satisfies certain requirements or desiderata.

Explainable AI (XAI)

- Explain to humans how the model works.
- Explain why the model reached a decision.
- Explain the inner workings of a model.

Understanding ML Models

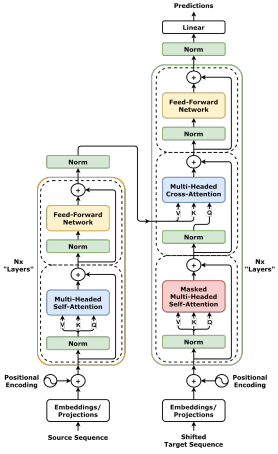


Image source: Wikipedia

Verification

- Prove that the model computes the correct function.
- Prove that the model satisfies certain requirements or desiderata.

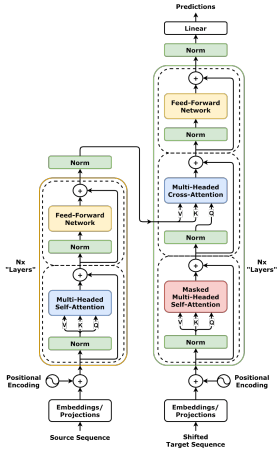
Explainable AI (XAI)

- Explain to humans how the model works.
- Explain why the model reached a decision.
- Explain the inner workings of a model.

For an excellent critique of current XAI practices, see:

T. Freiesleben and G. König. Dear XAI community, we need to talk! Fundamental misconceptions in current XAI research. Proc. World conference on explainable artificial intelligence, Springer 2023.

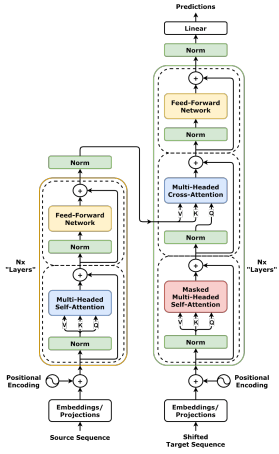
Querying ML Models



$$? \models \varphi$$

Most ML Verification and XAI tasks can be viewed as querying a model (a.k.a. model checking).

Querying ML Models



$$? \models \varphi$$

Most ML Verification and XAI tasks can be viewed as querying a model (a.k.a. model checking).

We may either ask specific fixed queries or interact with the model in an adaptive query-answer process.

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

What is the maximum value $f_{\mathcal{M}}$ can take?

$$\max_{v,d} f_{\mathcal{M}}(v, d).$$

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

What is the maximum value $f_{\mathcal{M}}$ can take?

$$\max_{v,d} f_{\mathcal{M}}(v, d).$$

Is it true that $f_{\mathcal{M}}$ is always nonnegative?

$$\forall v, d \; f_{\mathcal{M}}(v, d) \geq 0.$$

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

Is it true that if the velocity is ≥ 1 *and the distance is* ≤ 10 *then the brake intensity is maximal?*

$$\forall v, d \big(v \geq 1 \wedge d \leq 10 \to f_{\mathcal{M}}(v, d) = \max_{v', d'} f_{\mathcal{M}}(v', d') \big).$$

This is an example of a safety query.

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

Is it possible not to brake at all?

$$\exists v, d \; f_{\mathcal{M}}(v, d) = 0.$$

This is an example of a liveness query.

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

How much can small changes (of at most ϵ) of the input affect the output?

$$\max_{\delta} \forall v, v', d, d' \big(|v - v'| \leq \epsilon \wedge |d - d'| \leq \epsilon \to |f_{\mathcal{M}}(v, d) - f_{\mathcal{M}}(v', d')| \leq \delta \big).$$

This is an example of a robustness query.

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

Does the velocity have a noticeable effect on the brake intensity?

$$\exists v, v', d \; |f_{\mathcal{M}}(v, d) - f_{\mathcal{M}}(v', d)| > \epsilon.$$

This is a typical explainability query trying to figure out the importance of the input features.

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

How many parameters does the model have?

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

How many parameters does the model have?

Are there parameters that have no noticeable effect on the output?

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

How many parameters does the model have?

Are there parameters that have no noticeable effect on the output?

Which are the most important parameters if the velocity is greater than 10*?*

Examples

Suppose our model $\mathcal{M}$ computes a function $f_{\mathcal{M}} : \mathbb{R}^2 \to \mathbb{R}$ controlling the brake intensity of a car based on velocity and distance to an obstruction.

Queries

How many parameters does the model have?

Are there parameters that have no noticeable effect on the output?

Which are the most important parameters if the velocity is greater than 10*?*

Here we are querying the inner workings of the model and not only the function it computes.

Model-Agnostic vs Model-Specific

We always consider ML models $\mathcal{M}$ computing functions $f_{\mathcal{M}}$.

Model-Agnostic vs Model-Specific

We always consider ML models $\mathcal{M}$ computing functions $f_{\mathcal{M}}$.
The answer to a query φ against a model $\mathcal{M}$ is denoted by $[\![\varphi]\!]^{\mathcal{M}}$.

Model-Agnostic vs Model-Specific

We always consider ML models $\mathcal{M}$ computing functions $f_{\mathcal{M}}$.
The answer to a query φ against a model $\mathcal{M}$ is denoted by $[\![\varphi]\!]^{\mathcal{M}}$.

Definition

A query φ is model agnostic (over a class $\mathbf{M}$ of models) if its answer only depends on the function computed by the model, that is,

$$f_{\mathcal{M}} = f_{\mathcal{M}'} \quad \Longrightarrow \quad [\![\varphi]\!]^{\mathcal{M}} = [\![\varphi]\!]^{\mathcal{M}'} \qquad \text{for all } \mathcal{M}, \mathcal{M}' \text{ (in } \mathbf{M}).$$

Model-Agnostic vs Model-Specific

We always consider ML models $\mathcal{M}$ computing functions $f_{\mathcal{M}}$.
The answer to a query φ against a model $\mathcal{M}$ is denoted by $[\![\varphi]\!]^{\mathcal{M}}$.

Definition

A query φ is model agnostic (over a class $\mathbf{M}$ of models) if its answer only depends on the function computed by the model, that is,

$$f_{\mathcal{M}} = f_{\mathcal{M}'} \quad \Longrightarrow \quad [\![\varphi]\!]^{\mathcal{M}} = [\![\varphi]\!]^{\mathcal{M}'} \qquad \text{for all } \mathcal{M}, \mathcal{M}' \text{ (in } \mathbf{M}).$$

Examples

Model agnostic: How much can small changes (of at most ϵ) of the input affect the output?

Model-Agnostic vs Model-Specific

We always consider ML models $\mathcal{M}$ computing functions $f_{\mathcal{M}}$.
The answer to a query φ against a model $\mathcal{M}$ is denoted by $[\![\varphi]\!]^{\mathcal{M}}$.

Definition

A query φ is model agnostic (over a class $\mathbf{M}$ of models) if its answer only depends on the function computed by the model, that is,

$$f_{\mathcal{M}} = f_{\mathcal{M}'} \quad \Longrightarrow \quad [\![\varphi]\!]^{\mathcal{M}} = [\![\varphi]\!]^{\mathcal{M}'} \qquad \text{for all } \mathcal{M}, \mathcal{M}' \text{ (in } \mathbf{M}).$$

Examples

Model agnostic: How much can small changes (of at most ϵ) of the input affect the output?

Not model agnostic: Does the model have $\leq 100,000$ parameters?

Model-Agnostic vs Model-Specific

We always consider ML models $\mathcal{M}$ computing functions $f_{\mathcal{M}}$.
The answer to a query φ against a model $\mathcal{M}$ is denoted by $[\![\varphi]\!]^{\mathcal{M}}$.

Definition

A query φ is model agnostic (over a class **M** of models) if its answer only depends on the function computed by the model, that is,

$$f_{\mathcal{M}} = f_{\mathcal{M}'} \quad \Longrightarrow \quad [\![\varphi]\!]^{\mathcal{M}} = [\![\varphi]\!]^{\mathcal{M}'} \qquad \text{for all } \mathcal{M}, \mathcal{M}' \text{ (in } \mathbf{M}).$$

Examples

Model agnostic: How much can small changes (of at most ϵ) of the input affect the output?

Not model agnostic: Does the model have $\leq 100,000$ parameters?

Model agnostic: Is there a model computing the same function with at most 100,000 parameters?

Black-Box vs White-Box

Definition

A query φ is black box if it only has access to the function $f_{\mathcal{M}}$ and not to the model $\mathcal{M}$.

Black-Box vs White-Box

Definition

A query φ is black box if it only has access to the function $f_{\mathcal{M}}$ and not to the model $\mathcal{M}$.

Observation

Every black-box query is model-agnostic. The converse does not hold.

Black-Box vs White-Box

Definition

A query φ is black box if it only has access to the function $f_{\mathcal{M}}$ and not to the model $\mathcal{M}$.

Observation

Every black-box query is model-agnostic. The converse does not hold.

Remark

"Black box" is a syntactic property, whereas "model agnostic" is a semantic one.

Query languages for ML Models

- We look for declarative (database-style) query languages expressive enough for typical ML querying tasks while still allowing efficient query evaluation.

Query languages for ML Models

- We look for declarative (database-style) query languages expressive enough for typical ML querying tasks while still allowing efficient query evaluation.
- It is important to be able to query real (or rational) weights and functions over the reals (rationals) and not just Boolean models or discrete structures.

Query languages for ML Models

- We look for declarative (database-style) query languages expressive enough for typical ML querying tasks while still allowing efficient query evaluation.
- It is important to be able to query real (or rational) weights and functions over the reals (rationals) and not just Boolean models or discrete structures.
- We are interested in generic languages (like first-order logic) to study the principles of querying ML models and not so much in user-friendly syntax or individual features tailored towards specific models.

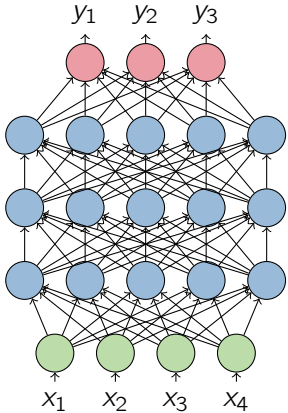
Query languages for ML Models

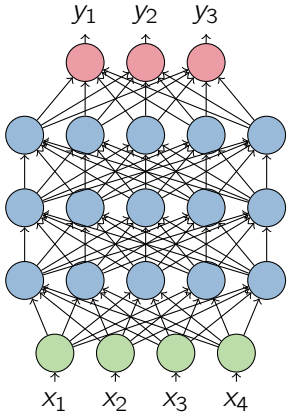
- We look for declarative (database-style) query languages expressive enough for typical ML querying tasks while still allowing efficient query evaluation.
- It is important to be able to query real (or rational) weights and functions over the reals (rationals) and not just Boolean models or discrete structures.
- We are interested in generic languages (like first-order logic) to study the principles of querying ML models and not so much in user-friendly syntax or individual features tailored towards specific models.
- A particular focus is the trade-off between expressiveness and the complexity of query evaluation.

Query languages for ML Models

- We look for declarative (database-style) query languages expressive enough for typical ML querying tasks while still allowing efficient query evaluation.
- It is important to be able to query real (or rational) weights and functions over the reals (rationals) and not just Boolean models or discrete structures.
- We are interested in generic languages (like first-order logic) to study the principles of querying ML models and not so much in user-friendly syntax or individual features tailored towards specific models.
- A particular focus is the trade-off between expressiveness and the complexity of query evaluation.
- The foundation of our approach is Grädel and Gurevich's meta-finite model theory (1998).

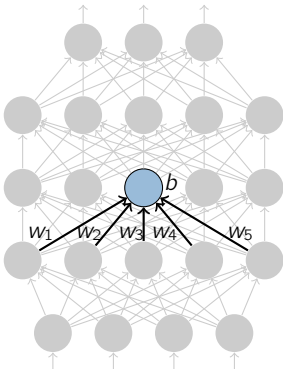
Feedforward Neural Networks (FNNs)





Feedforward Neural Networks (FNNs)

- Nodes and edges are weighted.



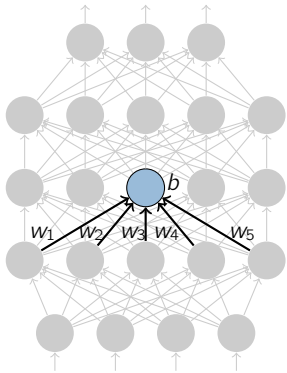
Feedforward Neural Networks (FNNs)

- Nodes and edges are weighted.
- Node with bias $b \in \mathbb{R}$ and incoming edges with weights $w_1, \ldots, w_k \in \mathbb{R}$ computes function

$$x_1, \ldots, x_k \mapsto \sigma\left(b + \sum_{i=1}^{k} w_i x_i\right),$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function.

Feedforward Neural Networks (FNNs)

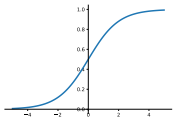


- Nodes and edges are weighted.
- Node with bias $b \in \mathbb{R}$ and incoming edges with weights $w_1, \ldots, w_k \in \mathbb{R}$ computes function

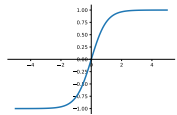
$$x_1, \ldots, x_k \mapsto \sigma\left(b + \sum_{i=1}^{k} w_i x_i\right),$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function.

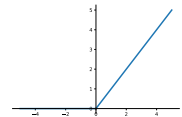
Typical activation functions are:



sigmoid

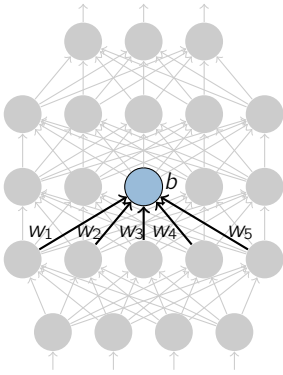


hyperbolic tangent



rectified linear unit (ReLU)

Feedforward Neural Networks (FNNs)

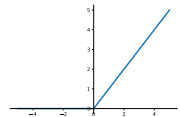


- Nodes and edges are weighted.
- Node with bias $b \in \mathbb{R}$ and incoming edges with weights $w_1, \dots, w_k \in \mathbb{R}$ computes function

$$x_1, \dots, x_k \mapsto \sigma\left(b + \sum_{i=1}^{k} w_i x_i\right),$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function.

In this talk, we always use $\mathrm{ReLU} : x \mapsto \max\{0, x\}$ as activation function for the internal nodes ("hidden layers") and $\mathrm{id} : x \mapsto x$ for the output nodes.



rectified linear unit (ReLU)

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Weighted Structures

Υ-structure $\mathcal{A}$ consist of:

- finite nonempty universe A,

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Weighted Structures

Υ-structure $\mathcal{A}$ consist of:

- finite nonempty universe A,
- relation $R^{\mathcal{A}} \subseteq A^k$ for each k-ary relation symbol $R \in \Upsilon$,

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Weighted Structures

Υ-structure $\mathcal{A}$ consist of:

- finite nonempty universe A,
- relation $R^{\mathcal{A}} \subseteq A^k$ for each k-ary relation symbol $R \in \Upsilon$,
- function $F^{\mathcal{A}} \colon A^k \to \underbrace{\mathbb{R}_\bot}_{:=\mathbb{R}\cup\{\bot\}}$ for each k-ary weight function symbol $F \in \Upsilon$.

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Weighted Structures

Υ-structure $\mathcal{A}$ consist of:

- finite nonempty universe A,
- relation $R^{\mathcal{A}} \subseteq A^k$ for each k-ary relation symbol $R \in \Upsilon$,
- function $F^{\mathcal{A}} \colon A^k \to \underbrace{\mathbb{R}_\perp}_{:=\mathbb{R}\cup\{\perp\}}$ for each k-ary weight function symbol $F \in \Upsilon$.

Example

FNNs are weighted structures of vocabulary $\Upsilon_{\mathrm{FNN}} := \{\mathrm{wt}, \mathrm{bias}, \leq_{\mathrm{in}} \leq_{\mathrm{out}}\}$,

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Weighted Structures

Υ-structure $\mathcal{A}$ consist of:

- finite nonempty universe A,
- relation $R^{\mathcal{A}} \subseteq A^k$ for each k-ary relation symbol $R \in \Upsilon$,
- function $F^{\mathcal{A}} \colon A^k \to \underbrace{\mathbb{R}_\perp}_{:=\mathbb{R}\cup\{\perp\}}$ for each k-ary weight function symbol $F \in \Upsilon$.

Example

FNNs are weighted structures of vocabulary $\Upsilon_{\mathsf{FNN}} := \{\mathsf{wt}, \mathsf{bias}, \leq_{\mathsf{in}} \leq_{\mathsf{out}}\}$, where

- $\mathsf{wt}, \mathsf{bias}$ are binary resp. unary symbols interpreted by the weights of the edges and biases of the nodes,

Vocabularies

Set Υ of relation symbols R and weight function symbols F, each with a non-negative arity.

Weighted Structures

Υ-structure $\mathcal{A}$ consist of:

- finite nonempty universe A,
- relation $R^{\mathcal{A}} \subseteq A^k$ for each k-ary relation symbol $R \in \Upsilon$,
- function $F^{\mathcal{A}} \colon A^k \to \underbrace{\mathbb{R}_\perp}_{:=\mathbb{R}\cup\{\perp\}}$ for each k-ary weight function symbol $F \in \Upsilon$.

Example

FNNs are weighted structures of vocabulary $\Upsilon_{\mathrm{FNN}} := \{\mathrm{wt}, \mathrm{bias}, \leq_{\mathrm{in}} \leq_{\mathrm{out}}\}$, where

- $\mathrm{wt}, \mathrm{bias}$ are binary resp. unary symbols interpreted by the weights of the edges and biases of the nodes,
- $\leq_{\mathrm{in}}, \leq_{\mathrm{out}}$ are binary relation symbols interpreted by linear orders on the input nodes and on the output nodes, respectively.

Weight Aggregation Logics

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

Syntax

Formulas φ and terms θ formed by the following grammar:

$$\varphi ::= x = y \mid R(x_1, \ldots, x_k) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi$$

$$\theta ::= 0 \mid 1 \mid F(x_1, \ldots, x_k) \mid (\theta \circ \theta) \mid \text{if } \varphi \text{ then } \theta \text{ else } \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta.$$

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

Syntax

k-ary relation symbol

$\in \{\vee, \wedge, \rightarrow\}$

x, y, x_i variables

terms θ formed by the following grammar:

$\in \{\exists, \forall\}$

$$\varphi ::= x = y \mid R(x_1, \ldots, x_k) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi$$

$$\theta ::= 0 \mid 1 \mid F(x_1, \ldots, x_k) \mid (\theta \circ \theta) \mid \text{if } \varphi \text{ then } \theta \text{ else } \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta.$$

k-ary weight function symbol

$\in \{+, -, \cdot, /\}$

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

Syntax

Formulas φ and terms θ formed by the following grammar:

$$\varphi ::= x = y \mid R(x_1, \ldots, x_k) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi$$

$$\theta ::= 0 \mid 1 \mid F(x_1, \ldots, x_k) \mid (\theta \circ \theta) \mid \text{if } \varphi \text{ then } \theta \text{ else } \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta.$$

Semantics

Let $\mathcal{A}$ be a structure and $a_1, \ldots, a_k \in A$.

- Variables range over A (*not* $\mathbb{R}$).

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

Syntax

Formulas φ and terms θ formed by the following grammar:

$$\varphi ::= x = y \mid R(x_1, \ldots, x_k) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi$$

$$\theta ::= 0 \mid 1 \mid F(x_1, \ldots, x_k) \mid (\theta \circ \theta) \mid \text{if } \varphi \text{ then } \theta \text{ else } \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta.$$

Semantics

Let $\mathcal{A}$ be a structure and $a_1, \ldots, a_k \in A$.

- Variables range over A (*not* $\mathbb{R}$).
- For every formula $\varphi(x_1, \ldots, x_k)$ we define a Boolean value $[\![\varphi]\!]^{\mathcal{A}}(a_1, \ldots, a_k) \in \{0, 1\}$.

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

Syntax

Formulas φ and terms θ formed by the following grammar:

$$\varphi ::= x = y \mid R(x_1, \ldots, x_k) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi$$

$$\theta ::= 0 \mid 1 \mid F(x_1, \ldots, x_k) \mid (\theta \circ \theta) \mid \text{if } \varphi \text{ then } \theta \text{ else } \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta.$$

Semantics

Let $\mathcal{A}$ be a structure and $a_1, \ldots, a_k \in A$.

- Variables range over A (*not* $\mathbb{R}$).
- For every formula $\varphi(x_1, \ldots, x_k)$ we define a Boolean value $[\![\varphi]\!]^{\mathcal{A}}(a_1, \ldots, a_k) \in \{0, 1\}$.
- For every term $\theta(x_1, \ldots, x_k)$ we define a numerical value $[\![\theta]\!]^{\mathcal{A}}(a_1, \ldots, a_k) \in \mathbb{R}_\perp$.

The Basic Logic: FO(SUM)

We define first-order logic with weight aggregation FO(SUM).

Syntax

Formulas φ and terms θ formed by the following grammar:

$$\varphi ::= x = y \mid R(x_1, \ldots, x_k) \mid \theta \leq \theta \mid \neg\varphi \mid (\varphi * \varphi) \mid Qx\varphi$$

$$\theta ::= 0 \mid 1 \mid F(x_1, \ldots, x_k) \mid (\theta \circ \theta) \mid \text{if } \varphi \text{ then } \theta \text{ else } \theta \mid \sum_{(x_1,\ldots,x_k):\varphi} \theta.$$

Semantics

Let $\mathcal{A}$ be a structure and $a_1, \ldots, a_k \in A$.

- Variables range over A (*not* $\mathbb{R}$).
- For every formula $\varphi(x_1, \ldots, x_k)$ we define a Boolean value $[\![\varphi]\!]^{\mathcal{A}}(a_1, \ldots, a_k) \in \{0, 1\}$.
- For every term $\theta(x_1, \ldots, x_k)$ we define a numerical value $[\![\theta]\!]^{\mathcal{A}}(a_1, \ldots, a_k) \in \mathbb{R}_\perp$.

Both are straightforward inductive definitions.

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* eval-node$_d(x)$ *evaluating an FNN at node x of depth d.*

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* eval-node$_d(x)$ *evaluating an FNN at node x of depth d.*

That is, for every $m \in \mathbb{N}$, every FNN $\mathcal{N}$ of input dimension m, every node $v \in V^{\mathcal{N}}$ of depth at most d, and every $(r_1, \ldots, r_m) \in \mathbb{R}^m$ it holds that

$$[\![\text{eval-node}_d]\!]^{\mathcal{N}(r_1,\ldots,r_m)}(v) = \begin{cases} f_v^{\mathcal{N}}(r_1, \ldots, r_m) & \text{if the depth of } v \text{ is at most } d, \\ \bot & \text{otherwise.} \end{cases}$$

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* eval-node$_d(x)$ *evaluating an FNN at node x of depth d.*

That is, for every $m \in \mathbb{N}$, every FNN $\mathcal{N}$ of input dimension m, every node $v \in V^{\mathcal{N}}$ of depth at most d, and every $(r_1, \ldots, r_m) \in \mathbb{R}^m$ it holds that

$$[\![\text{eval-node}_d]\!]^{\mathcal{N}(r_1,\ldots,r_m)}(v) = \begin{cases} f_v^{\mathcal{N}}(r_1, \ldots, r_m) & \text{if the depth of } v \text{ is at most } d, \\ \bot & \text{otherwise.} \end{cases}$$

expansion of $\mathcal{N}$ by unary weight function inp mapping the ith input to r_i

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* $\text{eval-node}_d(x)$ *evaluating an FNN at node x of depth d.*

We let

$$\text{eval-node}_0(x) \coloneqq \text{inp}(x)$$

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* $\text{eval-node}_d(x)$ *evaluating an FNN at node x of depth d.*

We let

$$\begin{aligned} \text{eval-node}_0(x) &:= \text{inp}(x) \\ \text{eval-node}_{d+1}(x) &:= \text{if } \text{inp}(x) \neq \bot \text{ then } \text{inp}(x) \\ &\qquad \text{else relu}\left(\text{bias}(x) + \sum_{y:\text{edge}(y,x)} \text{wt}(y,x) \cdot \text{eval-node}_d(y)\right). \end{aligned}$$

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* $\text{eval-node}_d(x)$ *evaluating an FNN at node x of depth d.*

We let

$$\begin{aligned}\text{eval-node}_0(x) &:= \text{inp}(x)\\ \text{eval-node}_{d+1}(x) &:= \text{if } \text{inp}(x) \neq \bot \text{ then } \text{inp}(x)\\ &\qquad \text{else relu}\left(\text{bias}(x) + \sum_{y:\text{edge}(y,x)} \text{wt}(y,x)\cdot \text{eval-node}_d(y)\right).\end{aligned}$$

Here $\text{edge}(y,x) := \text{wt}(y,x) \neq \bot$ is a formula defining the edge relation

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* $\text{eval-node}_d(x)$ *evaluating an FNN at node x of depth d.*

We let

$$\begin{aligned}\text{eval-node}_0(x) &:= \text{inp}(x)\\ \text{eval-node}_{d+1}(x) &:= \text{if } \text{inp}(x) \neq \bot \text{ then } \text{inp}(x)\\ &\qquad \text{else relu}\left(\text{bias}(x) + \sum_{y:\text{edge}(y,x)} \text{wt}(y,x)\cdot \text{eval-node}_d(y)\right).\end{aligned}$$

Here $\text{edge}(y,x) := \text{wt}(y,x) \neq \bot$ is a formula defining the edge relation and

$$\text{relu}(\theta) := \text{if } \theta \geq 0 \text{ then } \theta \text{ else } 0\cdot\theta$$

defines the ReLU function.

Example: Evaluating FNNs

For every d there is an FO(SUM)*-term* eval-node$_d(x)$ *evaluating an FNN at node x of depth d.*

We can also define a closed FO(SUM)-term eval_d evaluating an FNN of output dimension 1 and depth d.

Example: Aggregation

Counting: *Count the number of* $\mathbf{x}$ *satisfying a formula* φ.

$$\mathsf{count}_{\mathbf{x}:\varphi} := \sum_{\mathbf{x}:\varphi} 1.$$

Example: Aggregation

Counting: *Count the number of* $\mathbf{x}$ *satisfying a formula* φ.

$$\mathsf{count}_{\mathbf{x}:\varphi} := \sum_{\mathbf{x}:\varphi} 1.$$

As an application, we can count the parameters of an FNN:

$$\mathsf{count}_{x:\mathsf{bias}(x)\neq\bot} + \mathsf{count}_{(x,y):\mathsf{wt}(x,y)\neq\bot}.$$

Example: Aggregation

Counting: *Count the number of* $\mathbf{x}$ *satisfying a formula* φ.

$$\mathsf{count}_{\mathbf{x}:\varphi} := \sum_{\mathbf{x}:\varphi} 1.$$

As an application, we can count the parameters of an FNN:

$$\mathsf{count}_{x:\mathsf{bias}(x)\neq\bot} + \mathsf{count}_{(x,y):\mathsf{wt}(x,y)\neq\bot}.$$

Averaging: *Take the average of a term* θ *over the set defined by formula* φ.

$$\mathsf{avg}_{\mathbf{x}:\varphi}\theta := \Big(\sum_{\mathbf{x}:\varphi} \theta\Big)/\mathsf{count}_{\mathbf{x}:\varphi}.$$

Example: Aggregation

Counting: *Count the number of* $\mathbf{x}$ *satisfying a formula* φ.

$$\mathsf{count}_{\mathbf{x}:\varphi} \coloneqq \sum_{\mathbf{x}:\varphi} 1.$$

As an application, we can count the parameters of an FNN:

$$\mathsf{count}_{x:\mathsf{bias}(x)\neq\bot} + \mathsf{count}_{(x,y):\mathsf{wt}(x,y)\neq\bot}.$$

Averaging: *Take the average of a term* θ *over the set defined by formula* φ.

$$\mathsf{avg}_{\mathbf{x}:\varphi}\theta \coloneqq \Big(\sum_{\mathbf{x}:\varphi} \theta\Big)/\mathsf{count}_{\mathbf{x}:\varphi}.$$

Maximum: *Take the maximum of a term* θ *over the set defined by formula* φ.

We let $\mathsf{max}_{\mathbf{x}:\varphi}\theta \coloneqq \mathsf{avg}_{\mathbf{x}:\varphi_{\mathsf{max}}}\theta$, where

$$\varphi_{\mathsf{max}} \coloneqq \varphi(\mathbf{x}) \wedge \forall \mathbf{x}'\big(\varphi(\mathbf{x}') \to \theta(\mathbf{x}') \leq \theta(\mathbf{x})\big).$$

Example: Integration

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

For fixed m, d, we can define integrals of functions computed by FNNs of input dimension m, output dimension 1, and depth at most d.

Example: Integration

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

For fixed m, d, we can define integrals of functions computed by FNNs of input dimension m, output dimension 1, and depth at most d.

That is, there is an FO(SUM) *term* $\mathrm{integrate}_{d,m}$ *such that for all FNNs $\mathcal{N}$ of input dimension m, output dimension 1, and depth at most d we have*

$$[\![\mathrm{integrate}_{d,m}]\!]^{\mathcal{N}} = \int_0^1 \cdots \int_0^1 f^{\mathcal{N}}(x_1, \ldots, x_m) dx_1 \ldots dx_m.$$

Example: Integration

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

For fixed m, d, we can define integrals of functions computed by FNNs of input dimension m, output dimension 1, and depth at most d.

That is, there is an FO(SUM) *term* $\mathrm{integrate}_{d,m}$ *such that for all FNNs $\mathcal{N}$ of input dimension m, output dimension 1, and depth at most d we have*

$$[\![\mathrm{integrate}_{d,m}]\!]^{\mathcal{N}} = \int_0^1 \cdots \int_0^1 f^{\mathcal{N}}(x_1, \ldots, x_m) dx_1 \ldots dx_m.$$

Remark

Here we choose integration boundaries $0, 1$ for all variables, but the result can be extended to arbitrary boundaries (passed via unary weight functions to the FNN).

Integration (cont'd)

The unary case $m = 1$ is easy:

Integration (cont'd)

The unary case $m = 1$ is easy:

- We use the simple fact that functions computed by FNNs with ReLU and id activations are piecewise linear.

Integration (cont'd)

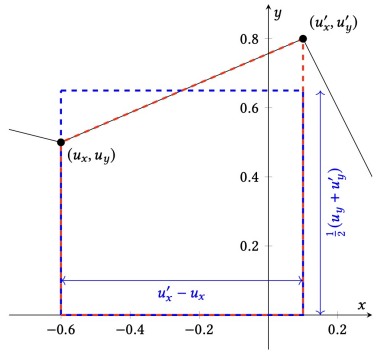
The unary case $m = 1$ is easy:

- We use the simple fact that functions computed by FNNs with ReLU and id activations are piecewise linear.
- We can associate the breakpoints of the piecewise linear unary function with nodes of the FNN and use FO(SUM)-terms to compute the coordinates of the breakpoints.

Integration (cont'd)

The unary case $m = 1$ is easy:

- We use the simple fact that functions computed by FNNs with ReLU and id activations are piecewise linear.
- We can associate the breakpoints of the piecewise linear unary function with nodes of the FNN and use FO(SUM)-terms to compute the coordinates of the breakpoints.
- Then we define the integrals over the linear pieces and use summation to combine them.



Model-Agnostic Queries in FO(SUM)

Recall

An FO(SUM) expression ξ (term or formula) is model agnostic over a class $\mathbf{N}$ of FNNs if for all $\mathcal{N}, \mathcal{N}' \in \mathbf{N}$ it holds that

$$f^{\mathcal{N}} = f^{\mathcal{N}'} \quad \Longrightarrow \quad [\![\xi]\!]^{\mathcal{N}} = [\![\xi]\!]^{\mathcal{N}'}.$$

Model-Agnostic Queries in FO(SUM)

Recall

An FO(SUM) expression ξ (term or formula) is model agnostic over a class $\mathbf{N}$ of FNNs if for all $\mathcal{N}, \mathcal{N}' \in \mathbf{N}$ it holds that

$$f^{\mathcal{N}} = f^{\mathcal{N}'} \quad \Longrightarrow \quad [\![\xi]\!]^{\mathcal{N}} = [\![\xi]\!]^{\mathcal{N}'}.$$

$\mathbf{N}(m, n)$ denotes the class of all FNNs of input dimension m and output dimension n.

Model-Agnostic Queries in FO(SUM)

Recall

An FO(SUM) expression ξ (term or formula) is model agnostic over a class $\mathbf{N}$ of FNNs if for all $\mathcal{N}, \mathcal{N}' \in \mathbf{N}$ it holds that

$$f^{\mathcal{N}} = f^{\mathcal{N}'} \quad \Longrightarrow \quad [\![\xi]\!]^{\mathcal{N}} = [\![\xi]\!]^{\mathcal{N}'}.$$

$\mathbf{N}(m, n)$ denotes the class of all FNNs of input dimension m and output dimension n.

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

Only trivial FO(SUM) *expressions are model agnostic over* $\mathbf{N}(1, 1)$.

Model-Agnostic Queries in FO(SUM)

Recall

An FO(SUM) expression ξ (term or formula) is model agnostic over a class $\mathbf{N}$ of FNNs if for all $\mathcal{N}, \mathcal{N}' \in \mathbf{N}$ it holds that

$$f^{\mathcal{N}} = f^{\mathcal{N}'} \quad \Longrightarrow \quad [\![\xi]\!]^{\mathcal{N}} = [\![\xi]\!]^{\mathcal{N}'}.$$

$\mathbf{N}(m, n)$ denotes the class of all FNNs of input dimension m and output dimension n.

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

Only trivial FO(SUM) *expressions are model agnostic over* $\mathbf{N}(1, 1)$.

That is, if an FO(SUM) *expression* ξ *is model agnostic over* $\mathbf{N}(1, 1)$ *then* $[\![\xi]\!]^{\mathcal{N}} = [\![\xi]\!]^{\mathcal{N}'}$ *for all* $\mathcal{N}, \mathcal{N}' \in \mathbf{N}(1, 1)$.

$\mathbf{N}_d(m, n)$ denotes the class of all FNNs of input dimension m, output dimension 1, and depth at most d.

$\mathbf{N}_d(m, n)$ denotes the class of all FNNs of input dimension m, output dimension 1, and depth at most d.

Example (Integration)

For all d, m, the term $\text{integrate}_{d,m}$ is model-agnostic over $\mathbf{N}_d(m, 1)$.

$\mathbf{N}_d(m, n)$ denotes the class of all FNNs of input dimension m, output dimension 1, and depth at most d.

Example (Integration)

For all d, m, the term $\text{integrate}_{d,m}$ is model-agnostic over $\mathbf{N}_d(m, 1)$.

Example (Evaluation)

For all d, m, the term eval_d is model agnostic over the class of all $\mathcal{N}(\mathbf{r})$, where $\mathcal{N} \in \mathbf{N}_d(m, 1)$ and $\mathbf{r} \in \mathbb{R}^m$.

A Black-Box Language

FO(SUM)-expressions have full access to the models they query; we may call FO(SUM) a white-box language.

A Black-Box Language

FO(SUM)-expressions have full access to the models they query; we may call FO(SUM) a white-box language.

Black-box logic

For an m-ary function symbol $f^{(m)}$, we define the black-box logic FO($\mathcal{R}_{\text{lin}}, f^{(m)}$).

A Black-Box Language

FO(SUM)-expressions have full access to the models they query; we may call FO(SUM) a white-box language.

Black-box logic

For an m-ary function symbol $f^{(m)}$, we define the black-box logic $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$.

Syntax: first-order logic in the language $\{+, \leq, (q)_{q\in\mathbb{Q}}, f^{(m)}\}$.

A Black-Box Language

FO(SUM)-expressions have full access to the models they query; we may call FO(SUM) a white-box language.

Black-box logic

For an m-ary function symbol $f^{(m)}$, we define the black-box logic $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$.

Syntax: first-order logic in the language $\{+, \leq, (q)_{q\in\mathbb{Q}}, f^{(m)}\}$.

Semantics: interpret $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ formulas over expansions $(\mathcal{R}_{\mathrm{lin}}, f)$ of linear real arithmetic $\mathcal{R}_{\mathrm{lin}} = (\mathbb{R}, +, (q)_{q\in\mathbb{Q}}, \leq)$ by an m-ary function $f : \mathbb{R}^m \to \mathbb{R}$.

A Black-Box Language

FO(SUM)-expressions have full access to the models they query; we may call FO(SUM) a white-box language.

Black-box logic

For an m-ary function symbol $f^{(m)}$, we define the black-box logic $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$.

Syntax: first-order logic in the language $\{+, \leq, (q)_{q\in\mathbb{Q}}, f^{(m)}\}$.

Semantics: interpret $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ formulas over expansions $(\mathcal{R}_{\mathrm{lin}}, f)$ of linear real arithmetic $\mathcal{R}_{\mathrm{lin}} = (\mathbb{R}, +, (q)_{q\in\mathbb{Q}}, \leq)$ by an m-ary function $f\colon \mathbb{R}^m \to \mathbb{R}$.

FNN queries: For every $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ sentence φ and every FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ we define a Boolean value $[\![\varphi]\!]^{\mathcal{N}} \in \{0, 1\}$ by

$$[\![\varphi]\!]^{\mathcal{N}} = 1 \iff (\mathcal{R}_{\mathrm{lin}}, f^{\mathcal{N}}) \models \varphi.$$

A Black-Box Language

FO(SUM)-expressions have full access to the models they query; we may call FO(SUM) a white-box language.

Black-box logic

For an m-ary function symbol $f^{(m)}$, we define the black-box logic $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$.

Syntax: first-order logic in the language $\{+, \leq, (q)_{q\in\mathbb{Q}}, f^{(m)}\}$.

Semantics: interpret $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ formulas over expansions $(\mathcal{R}_{\mathrm{lin}}, f)$ of linear real arithmetic $\mathcal{R}_{\mathrm{lin}} = (\mathbb{R}, +, (q)_{q\in\mathbb{Q}}, \leq)$ by an m-ary function $f\colon \mathbb{R}^m \to \mathbb{R}$.

FNN queries: For every $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ sentence φ and every FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ we define a Boolean value $[\![\varphi]\!]^{\mathcal{N}} \in \{0, 1\}$ by

$$[\![\varphi]\!]^{\mathcal{N}} = 1 \iff (\mathcal{R}_{\mathrm{lin}}, f^{\mathcal{N}}) \models \varphi.$$

Observation

All FNN queries defined in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ *are model agnostic.*

The zero query

Let Zero_m be the query saying that an FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ computes the constant-0 function. The $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ sentence $\forall x_1 \ldots \forall x_m \; f^{(m)}(x_1, \ldots, x_m) = 0.$ defines Zero_m.

The zero query

Let Zero_m be the query saying that an FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ computes the constant-0 function. The $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ sentence $\forall x_1 \ldots \forall x_m \; f^{(m)}(x_1, \ldots, x_m) = 0.$ defines Zero_m.

Irrelevant features

For $1 \leq i \leq m$ and rational $\epsilon > 0$, let $\mathrm{Irrelevant}_{m,i,\epsilon}$ be the query saying that the ith input feature of an FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ is ϵ-irrelevant, that is, for all $r_1, \ldots, r_m, r_i' \in \mathbb{R}$ it holds that

$$|f^{\mathcal{N}}(r_1, \ldots, r_m) - f^{\mathcal{N}}(r_1, \ldots, r_{i-1}, r_i', r_{i+1}, \ldots, r_m)| \leq \epsilon.$$

The query can easily be expressed in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$.

Examples (cont'd)

Robustness

For $m \in \mathbb{N}$ and rational $\epsilon, \delta >$, let $\text{Robust}_{m,\epsilon,\delta}$ be the query saying that an FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ is ϵ, δ robust, that is, changing the input features by at most ϵ changes the output by at most δ.

Robustness

For $m \in \mathbb{N}$ and rational $\epsilon, \delta >$, let $\text{Robust}_{m,\epsilon,\delta}$ be the query saying that an FNN $\mathcal{N} \in \mathbf{N}(m, 1)$ is ϵ, δ robust, that is, changing the input features by at most ϵ changes the output by at most δ.

$\text{Robust}_{m,\epsilon,\delta}$ can be expressed in $\text{FO}(\mathcal{R}_{\text{lin}}, f^{(m)})$:

$$\forall x_1 \ldots \forall x_m \forall y_1 \ldots \forall y_m \left(\bigwedge_{i=1}^{m} |x_1 - y_i| \leq \epsilon \rightarrow \left| f^{(m)}(x_1, \ldots, x_m) - f^{(m)}(y_1, \ldots, y_m) \right| \leq \delta \right).$$

Black Box vs White Box (I)

Corollary

The queries Zero_m, $\mathsf{Irrelevant}_{m,i,\epsilon}$, *and* $\mathsf{Robust}_{m,\epsilon,\delta}$ *are not expressible in* FO(SUM), *because they are nontrivial model-agnostic queries.*

Black Box vs White Box (I)

Corollary

The queries Zero_m, $\mathrm{Irrelevant}_{m,i,\epsilon}$, *and* $\mathrm{Robust}_{m,\epsilon,\delta}$ *are not expressible in* FO(SUM), *because they are nontrivial model-agnostic queries.*

It is not obvious if these queries can be expressed on FNNs of bounded depth d.

Black Box vs White Box (I)

Corollary

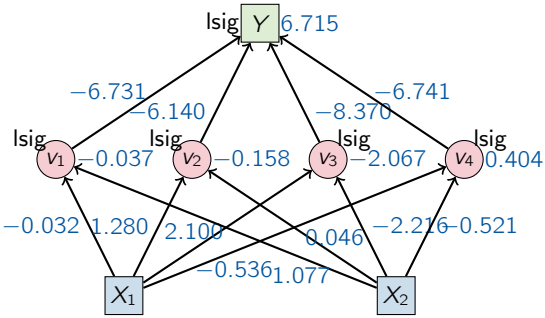
The queries Zero_m, $\mathrm{Irrelevant}_{m,i,\epsilon}$, *and* $\mathrm{Robust}_{m,\epsilon,\delta}$ *are not expressible in* FO(SUM), *because they are nontrivial model-agnostic queries.*

It is not obvious if these queries can be expressed on FNNs of bounded depth d.

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

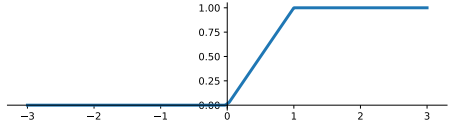
Let $d, m \in \mathbb{N}$. *Then every query on* $\mathbf{N}_d(m, 1)$ *that is expressible in* $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ *is expressible in* FO(SUM).

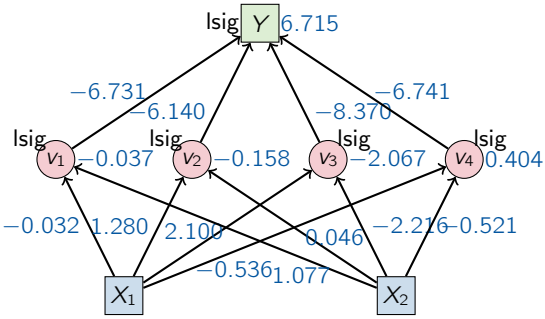
Example


lsig
Y
6.715
−6.731
−6.140
−8.370
−6.741
lsig
lsig
lsig
lsig
v1
−0.037
v2
−0.158
v3
−2.067
v4
0.404
−0.032
1.280
2.100
0.046
−2.216
−0.521
−0.536
1.077
X1
X2

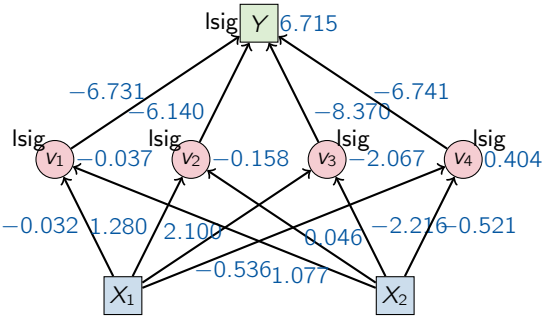
Example

Uses linearised sigmoid (lsig) activations:

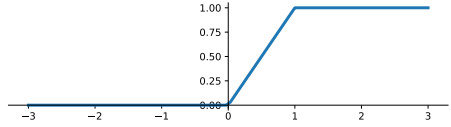




Example

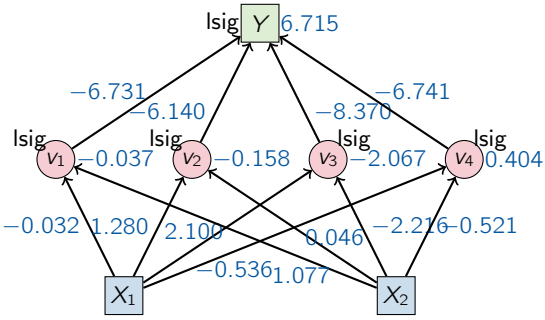


Uses linearised sigmoid (lsig) activations:

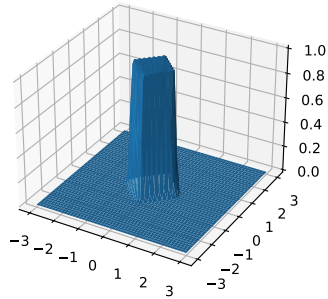


Note: $\mathsf{lsig}(x) = \mathsf{ReLU}(x) - \mathsf{ReLU}(x-1)$.

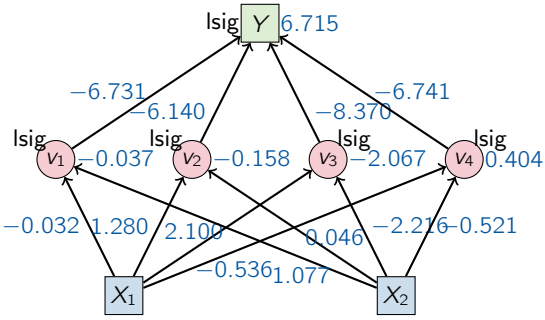
Example



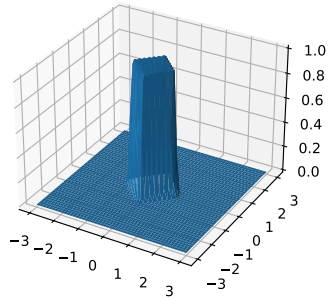
Computes the (piecewise linear) function



Example



Computes the (piecewise linear) function



An FO($\mathcal{R}_{\text{lin}}, f^{(2)}$) query might be

$$\forall x \forall y \big(|x| + |y| \leq 1 \rightarrow f^{(2)}(x, y) = 1 \big).$$

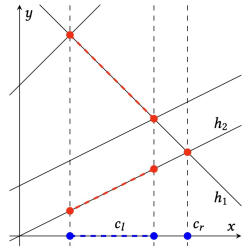
Proof Idea

- The difficulty in proving the theorem is that in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$ we can quantify over arbitrary real numbers, whereas in FO(SUM) we can only construct numbers from the weights of the input FNN using rational terms.

Proof Idea

- The difficulty in proving the theorem is that in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$ we can quantify over arbitrary real numbers, whereas in FO(SUM) we can only construct numbers from the weights of the input FNN using rational terms.

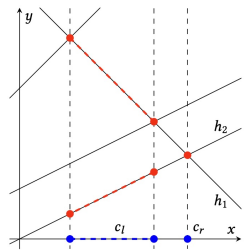
- We use a technique known as cylindrical cell decompositions, known from algebraic geometry, the model theory of o-minimal structures, and the theory of constraint databases.



Proof Idea

- The difficulty in proving the theorem is that in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$ we can quantify over arbitrary real numbers, whereas in FO(SUM) we can only construct numbers from the weights of the input FNN using rational terms.

- We use a technique known as cylindrical cell decompositions, known from algebraic geometry, the model theory of o-minimal structures, and the theory of constraint databases.

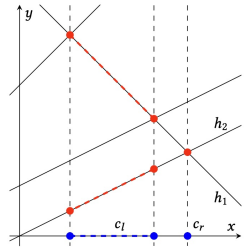


- The main part of the proof is to show that such a cell decomposition is definable in FO(SUM).

Proof Idea

- The difficulty in proving the theorem is that in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$ we can quantify over arbitrary real numbers, whereas in FO(SUM) we can only construct numbers from the weights of the input FNN using rational terms.

- We use a technique known as cylindrical cell decompositions, known from algebraic geometry, the model theory of o-minimal structures, and the theory of constraint databases.



- The main part of the proof is to show that such a cell decomposition is definable in FO(SUM).
- The individual cells can be indexed by tuples of nodes of the input FNN. As nodes in each cell behave essentially in the same way, this allows us to simulate quantification over the reals by quantification over tuples of nodes.

Black Box vs White Box (II)

Theorem (G., Standke, Steegmans, Van den Bussche 2025)
The query $\int_0^1 f^{\mathcal{N}}(x)dx = 0$ *over* $\mathbf{N}_2(1,1)$ *is not expressible in* $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$.

Black Box vs White Box (II)

Theorem (G., Standke, Steegmans, Van den Bussche 2025)

The query $\int_0^1 f^{\mathcal{N}}(x)dx = 0$ *over* $\mathbf{N}_2(1,1)$ *is not expressible in* $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$.

Corollary

There are model-agnostic queries of FNNs of depth 2 *that are expressible in* FO(SUM)*, but not in* $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f)$.

Beyond Bounded Depth

- FO(SUM) is quite expressive on FNNs of bounded depth, but can only express trivial (model-agnostic) queries on arbitrary FNNs.

Beyond Bounded Depth

- FO(SUM) is quite expressive on FNNs of bounded depth, but can only express trivial (model-agnostic) queries on arbitrary FNNs.
- To be able to go beyond bounded depth, we extend the logic by a fixed-point operator formalising inductive definitions.

Fixed-Point Logic with Weight Aggregation (Syntax)

We define the extension IFP(SUM) of FO(SUM).

Fixed-Point Logic with Weight Aggregation (Syntax)

We define the extension IFP(SUM) of FO(SUM).

We add one term-formation rule to the rules for FO(SUM):

$$\theta ::= \mathsf{ifp}\big(F(x_1, \ldots, x_k) \leftarrow \theta\big)(x'_1, \ldots, x'_k). \qquad (\star)$$

Fixed-Point Logic with Weight Aggregation (Syntax)

We define the extension IFP(SUM) of FO(SUM).

We add one term-formation rule to the rules for FO(SUM):

$$\theta ::= \mathrm{ifp}\big(F(x_1, \ldots, x_k) \leftarrow \theta\big)(x'_1, \ldots, x'_k). \qquad (\star)$$

The ifp-operator binds the weight-function symbol F, so if the vocabulary of θ is Υ then the vocabulary of $\mathrm{ifp}\big(F(x_1, \ldots, x_k) \leftarrow \theta\big)(x'_1, \ldots, x'_k)$ is $\Upsilon \setminus \{F\}$

Consider an ifp-term

$$\eta(\mathbf{x}') = \mathrm{ifp}\big(F(\mathbf{x}) \leftarrow \theta\big)(\mathbf{x}')$$

of vocabulary $\Upsilon \setminus \{F\}$.

Consider an ifp-term

$$\eta(\mathbf{x}') = \mathrm{ifp}\big(F(\mathbf{x}) \leftarrow \theta\big)(\mathbf{x}')$$

of vocabulary $\Upsilon \setminus \{F\}$.

Let $\mathcal{A}$ be an $\Upsilon \setminus \{F\}$-structure.

Consider an ifp-term

$$\eta(\mathbf{x}') = \mathrm{ifp}\big(F(\mathbf{x}) \leftarrow \theta\big)(\mathbf{x}')$$

of vocabulary $\Upsilon \setminus \{F\}$.

Let $\mathcal{A}$ be an $\Upsilon \setminus \{F\}$-structure.

- We define a sequence $(F_i)_{i \geq 0}$ of function $F_i : A^k \to \mathbb{R}_\perp$

Consider an ifp-term

$$\eta(\mathbf{x}') = \mathrm{ifp}\big(F(\mathbf{x}) \leftarrow \theta\big)(\mathbf{x}')$$

of vocabulary $\Upsilon \setminus \{F\}$.

Let $\mathcal{A}$ be an $\Upsilon \setminus \{F\}$-structure.

- We define a sequence $(F_i)_{i \geq 0}$ of function $F_i : A^k \to \mathbb{R}_\perp$ by

$$F_0(\mathbf{a}) \coloneqq \perp$$

for all $\mathbf{a} \in A^k$.

Consider an ifp-term

$$\eta(\mathbf{x}') = \mathrm{ifp}\big(F(\mathbf{x}) \leftarrow \theta\big)(\mathbf{x}')$$

of vocabulary $\Upsilon \setminus \{F\}$.

Let $\mathcal{A}$ be an $\Upsilon \setminus \{F\}$-structure.

- We define a sequence $(F_i)_{i \geq 0}$ of function $F_i : A^k \to \mathbb{R}_\perp$ by

$$F_0(\mathbf{a}) \coloneqq \perp$$

$$F_{i+1}(\mathbf{a}) \coloneqq \begin{cases} [\![\theta]\!]^{(\mathcal{A}, F_i)}(\mathbf{a}) & \text{if } F_i(\mathbf{a}) = \perp, \\ F_i(\mathbf{a}) & \text{if } F_i(\mathbf{a}) \neq \perp, \end{cases}$$

  for all $\mathbf{a} \in A^k$.

Consider an ifp-term

$$\eta(\mathbf{x}') = \mathsf{ifp}\big(F(\mathbf{x}) \leftarrow \theta\big)(\mathbf{x}')$$

of vocabulary $\Upsilon \setminus \{F\}$.

Let $\mathcal{A}$ be an $\Upsilon \setminus \{F\}$-structure.

- We define a sequence $(F_i)_{i \geq 0}$ of function $F_i : A^k \to \mathbb{R}_\perp$ by

$$F_0(\mathbf{a}) \coloneqq \perp$$

$$F_{i+1}(\mathbf{a}) \coloneqq \begin{cases} [\![\theta]\!]^{(\mathcal{A}, F_i)}(\mathbf{a}) & \text{if } F_i(\mathbf{a}) = \perp, \\ F_i(\mathbf{a}) & \text{if } F_i(\mathbf{a}) \neq \perp, \end{cases}$$

  for all $\mathbf{a} \in A^k$.

- For $\mathbf{a}' \in A^k$ we let

$$[\![\eta]\!]^{\mathcal{A}}(\mathbf{a}') \coloneqq \begin{cases} \perp & \text{if } F_i(\mathbf{a}') = \perp \text{ for all } i \geq 0, \\ F_i(\mathbf{a}') & \text{for the least } i \text{ such that } F_i(\mathbf{a}') \neq \perp \text{ otherwise.} \end{cases}$$

Example: Evaluating FNNs

There is an IFP(SUM)*-term* eval-node(x) *evaluating an FNN at node* x.

Example: Evaluating FNNs

There is an IFP(SUM)*-term* eval-node(x) *evaluating an FNN at node* x*.*

$$\text{eval-node}(x) := \text{ifp}\Big(F(x) \leftarrow \text{if } \text{inp}(x) \neq \bot \text{ then } \text{inp}(x) \text{ else } \text{relu}\Big(\text{bias}(x) + \sum_{y:\text{edge}(y,x)} \text{wt}(y,x) \cdot F(y)\Big)\Big)(x)$$

Example: Evaluating FNNs

There is an IFP(SUM)*-term* eval-node(x) *evaluating an FNN at node* x*.*

$$\text{eval-node}(x) := \text{ifp}\Big(F(x) \leftarrow \text{if } \text{inp}(x) \neq \bot \text{ then } \text{inp}(x) \text{ else } \text{relu}\Big(\text{bias}(x) + \sum_{y:\text{edge}(y,x)} \text{wt}(y,x) \cdot F(y)\Big)\Big)(x)$$

We can also define a closed IFP(SUM)-term eval evaluating an FNN of output dimension 1.

Complexity

Rational Structures

When studying the complexity of query evaluation, we restrict our attention to weighted structures with weights in $\mathbb{Q}_\perp$. We call them rational weighted structures.

Rational Structures

When studying the complexity of query evaluation, we restrict our attention to weighted structures with weights in $\mathbb{Q}_\perp$. We call them rational weighted structures.

Observation

1. *Over rational weighted structures, all* FO(SUM) *and* IFP(SUM) *terms take rational values.*

Rational Structures

When studying the complexity of query evaluation, we restrict our attention to weighted structures with weights in $\mathbb{Q}_\perp$. We call them rational weighted structures.

Observation

1. *Over rational weighted structures, all* FO(SUM) *and* IFP(SUM) *terms take rational values.*
2. *FNNs (with* ReLU *activations) with rational weights, biases, and inputs, take rational values.*

Rational Structures

When studying the complexity of query evaluation, we restrict our attention to weighted structures with weights in $\mathbb{Q}_\perp$. We call them rational weighted structures.

Observation

1. *Over rational weighted structures, all* FO(SUM) *and* IFP(SUM) *terms take rational values.*
2. *FNNs (with* ReLU *activations) with rational weights, biases, and inputs, take rational values.*

We encode integers and rationals in binary. The size $\|\mathcal{A}\|$ of a rational weighted structure $\mathcal{A}$ is the length of a reasonable binary encoding of $\mathcal{A}$.

Rational Structures

When studying the complexity of query evaluation, we restrict our attention to weighted structures with weights in $\mathbb{Q}_\perp$. We call them rational weighted structures.

Observation

1. *Over rational weighted structures, all* FO(SUM) *and* IFP(SUM) *terms take rational values.*
2. *FNNs (with* ReLU *activations) with rational weights, biases, and inputs, take rational values.*

We encode integers and rationals in binary. The size $\|\mathcal{A}\|$ of a rational weighted structure $\mathcal{A}$ is the length of a reasonable binary encoding of $\mathcal{A}$.

We can assume that $\|\mathcal{A}\|$ is polynomial in $|A|$ and the bit sizes of all weights.

Complexity of FO(SUM)

Theorem

The data complexity of FO(SUM) *is in uniform* TC^0.

Complexity of FO(SUM)

Theorem

The data complexity of FO(SUM) *is in uniform* TC^0.

That is, for every closed FO(SUM) *expression* ξ *the value* $[\![\xi]\!]^{\mathcal{A}}$ *of* ξ *in a given structure* $\mathcal{A}$ *can be computed by a dlogtime uniform family of threshold circuits of bounded depth and polynomial size in* $\|A\|$.

Complexity of FO(SUM)

Theorem

The data complexity of FO(SUM) *is in uniform* TC^0.

That is, for every closed FO(SUM) *expression* ξ *the value* $[\![\xi]\!]^{\mathcal{A}}$ *of* ξ *in a given structure* $\mathcal{A}$ *can be computed by a dlogtime uniform family of threshold circuits of bounded depth and polynomial size in* $\|A\|$.

Remark

Gerarts, Steegmans, and Van den Bussche (2025) have implemented an FO(SUM) query engine using SQL in a relational database system and evaluated it empirically.

Example: The Zero Query

Corollary

Let $d, m \in \mathbb{N}$, and let φ be a $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ sentence.
Then the problem of computing $[\![\varphi]\!]^{\mathcal{N}}$ for an FNN $\mathcal{N} \in \mathbf{N}_d(m, 1)$ is in uniform TC^0.

Example: The Zero Query

Corollary

Let $d, m \in \mathbb{N}$, and let φ be a $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ *sentence.*
Then the problem of computing $[\![\varphi]\!]^{\mathcal{N}}$ for an FNN $\mathcal{N} \in \mathbf{N}_d(m, 1)$ is in uniform TC^0.

Complexity of the Zero Query

Recall that the query Zero_m asks if an FNN in $\mathbf{N}(m, 1)$ evaluates to 0 on all inputs.

Example: The Zero Query

Corollary

Let $d, m \in \mathbb{N}$, and let φ be a $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ *sentence.*
Then the problem of computing $[\![\varphi]\!]^{\mathcal{N}}$ for an FNN $\mathcal{N} \in \mathbf{N}_d(m, 1)$ is in uniform TC^0.

Complexity of the Zero Query

Recall that the query Zero_m asks if an FNN in $\mathbf{N}(m, 1)$ evaluates to 0 on all inputs.

Theorem (Wurm 2024, G., Standke, Steegmans, Van den Bussche 2026)

Zero_1 *is* co-NP*-complete.*

Example: The Zero Query

Corollary

Let $d, m \in \mathbb{N}$, and let φ be a $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$ *sentence.*
Then the problem of computing $[\![\varphi]\!]^{\mathcal{N}}$ for an FNN $\mathcal{N} \in \mathbf{N}_d(m, 1)$ is in uniform TC^0.

Complexity of the Zero Query

Recall that the query Zero_m asks if an FNN in $\mathbf{N}(m, 1)$ evaluates to 0 on all inputs.

Theorem (Wurm 2024, G., Standke, Steegmans, Van den Bussche 2026)

Zero_1 *is* co-NP*-complete.*

Since Zero_m is expressible in $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f^{(m)})$, it follows from our results that Zero_m is in uniform TC^0 on FNNs of bounded depth.

The Complexity of IFP(SUM)

Fixed-point logics typically have a data complexity in P. Unfortunately, IFP(SUM) does not.

The Complexity of IFP(SUM)

Fixed-point logics typically have a data complexity in P. Unfortunately, IFP(SUM) does not.

Example (Grädel and Gurevich 1998)

Let

$$\sigma(x) \coloneqq \mathsf{ifp}\left(F(x) \leftarrow \mathsf{if}\ \exists y\, \mathsf{edge}(y,x)\ \mathsf{then}\ \Big(\sum_{y:\mathsf{edge}(y,x)} F(y)\Big)\cdot\Big(\sum_{y:\mathsf{edge}(y,x)} F(y)\Big)\ \mathsf{else}\ 2\right)(x).$$

If we evaluate the term over an FNN $\mathcal{N} \in \mathbf{N}(1,1)$ that is just a path of length d for the output node v of $\mathcal{N}$ it holds that

$$[\![\sigma]\!]^{\mathcal{N}}(v) = 2^{2^d}.$$

The Scalar Fragment

We introduced the scalar fragment sIFP(SUM) of IFP(SUM) by restricting the use of "intensional" weight function symbols (bound by ifp-operators) in multiplications and division.

The Scalar Fragment

We introduced the scalar fragment sIFP(SUM) of IFP(SUM) by restricting the use of "intensional" weight function symbols (bound by ifp-operators) in multiplications and division.

Most natural IFP(SUM) queries are in sIFP(SUM).

The Scalar Fragment

We introduced the scalar fragment sIFP(SUM) of IFP(SUM) by restricting the use of "intensional" weight function symbols (bound by ifp-operators) in multiplications and division.

Most natural IFP(SUM) queries are in sIFP(SUM).

Theorem (G., Standke, Steegmans, Van den Bussche 2026)

The data complexity of sIFP(SUM) *is in* P.

Towards Capturing Polynomial Time

Theorem (G., Standke, Steegmans, Van den Bussche 2026)

1. *There is a model-agnostic query on* $\mathbf{N}(1,1)$ *that is decidable in polynomial time, but not expressible in* IFP(SUM) *even on FNNs where all weights are* 1 *or* 0.

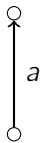
Towards Capturing Polynomial Time

Theorem (G., Standke, Steegmans, Van den Bussche 2026)

1. *There is a model-agnostic query on* $\mathbf{N}(1,1)$ *that is decidable in polynomial time, but not expressible in* IFP(SUM) *even on FNNs where all weights are* 1 *or* 0.
2. *For every* $b \geq 1$, *all polynomial-time decidable model-agnostic query on FNN of reduced weight at most* b *is expressible in* sIFP(SUM).

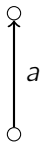
Theorem (G., Standke, Steegmans, Van den Bussche 2026)

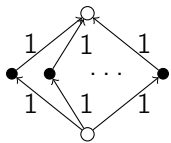
1. *There is a model-agnostic query on* $\mathbf{N}(1,1)$ *that is decidable in polynomial time, but not expressible in* IFP(SUM) *even on FNNs where all weights are* 1 *or* 0.
2. *For every* $b \geq 1$*, all polynomial-time decidable model-agnostic query on FNN of reduced weight at most* b *is expressible in* sIFP(SUM).



Theorem (G., Standke, Steegmans, Van den Bussche 2026)

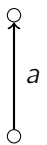
1. *There is a model-agnostic query on* $\mathbf{N}(1,1)$ *that is decidable in polynomial time, but not expressible in* IFP(SUM) *even on FNNs where all weights are* 1 *or* 0.
2. *For every* $b \geq 1$*, all polynomial-time decidable model-agnostic query on FNN of reduced weight at most* b *is expressible in* sIFP(SUM).

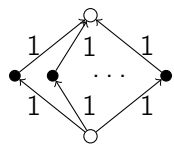


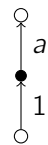


Theorem (G., Standke, Steegmans, Van den Bussche 2026)

1. *There is a model-agnostic query on* $\mathbf{N}(1,1)$ *that is decidable in polynomial time, but not expressible in* IFP(SUM) *even on FNNs where all weights are* 1 *or* 0.
2. *For every* $b \geq 1$*, all polynomial-time decidable model-agnostic query on FNN of reduced weight at most* b *is expressible in* sIFP(SUM).







Concluding Remarks

Conclusions

- Weight aggregation logics offer a principled way to query weighted structures, in particular, machine learning models.

Conclusions

- Weight aggregation logics offer a principled way to query weighted structures, in particular, machine learning models.
- On bounded depth FFNs, FO(SUM) is an expressive and robust logic.

Conclusions

- Weight aggregation logics offer a principled way to query weighted structures, in particular, machine learning models.
- On bounded depth FFNs, FO(SUM) is an expressive and robust logic.

  At the heart of the expressivity is the ability to define cylindrical cell decompositions in FO(SUM).

Conclusions

- Weight aggregation logics offer a principled way to query weighted structures, in particular, machine learning models.
- On bounded depth FFNs, FO(SUM) is an expressive and robust logic.

  At the heart of the expressivity is the ability to define cylindrical cell decompositions in FO(SUM).
- Beyond bounded depth, the situation is more brittle.

A Few Open Problems

- Can we extend $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs from linear to polynomial real arithmetic, that is, is $\mathrm{FO}(\mathcal{R}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs?

A Few Open Problems

- Can we extend $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs from linear to polynomial real arithmetic, that is, is $\mathrm{FO}(\mathcal{R}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs?
- Prove that $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \not\subseteq \mathrm{IFP}(\mathrm{SUM})$ on FNNs of unbounded depth.

A Few Open Problems

- Can we extend $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs from linear to polynomial real arithmetic, that is, is $\mathrm{FO}(\mathcal{R}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs?
- Prove that $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \not\subseteq \mathrm{IFP}(\mathrm{SUM})$ on FNNs of unbounded depth.
- What happens if we add other activation functions like the logistic function $x \mapsto (1 + e^{-x})^{-1}$ to the FNNs and the logics?

A Few Open Problems

- Can we extend $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs from linear to polynomial real arithmetic, that is, is $\mathrm{FO}(\mathcal{R}, f) \subseteq \mathrm{FO}(\mathrm{SUM})$ on bounded-depth FNNs?
- Prove that $\mathrm{FO}(\mathcal{R}_{\mathrm{lin}}, f) \not\subseteq \mathrm{IFP}(\mathrm{SUM})$ on FNNs of unbounded depth.
- What happens if we add other activation functions like the logistic function $x \mapsto (1 + e^{-x})^{-1}$ to the FNNs and the logics?
- Is there a logic with data complexity in P expressing all polynomial time model-agnostic queries on FNNs?