Graph Neural Networks, Their Logic, and Formal Verification

Nicolas Troquard [1]

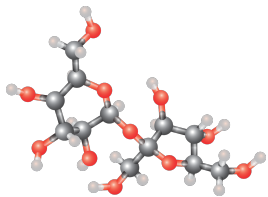
[1] Gran Sasso Science Institute, L'Aquila, Italy

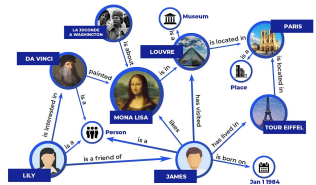
joint work with Artem Chernobrovkin, Marco Sälzer, François Schwarzentruber

Outline

Graph Neural Networks: neural networks processing graphs. [Scarselli et al. 2009][1]

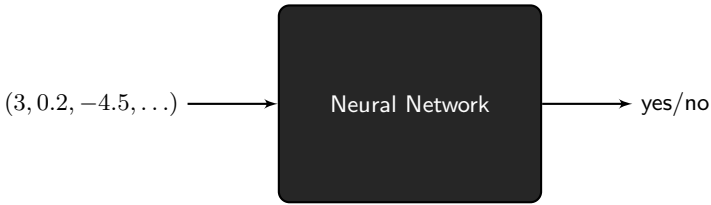
Applications in social recommendations, drug discovery, material science, financial networks anomaly detection, combinatorial optimization, etc.

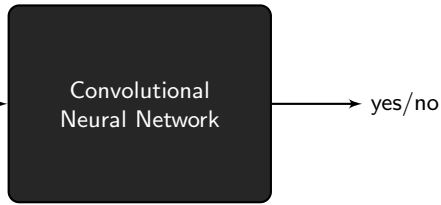


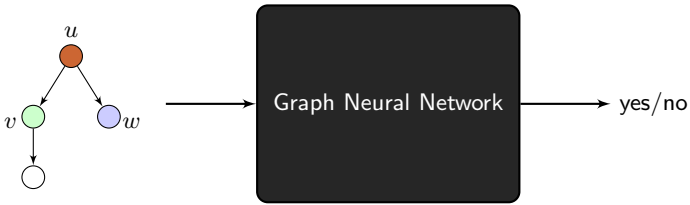


Implement a function $\tau(G, u) \in \mathbb{R}^m$, mapping a graph and one of its vertices into a m-dimensional Euclidean space; or simply a value in $\{0, 1\}$.

[1] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80.


$(3, 0.2, -4.5, \ldots)$
Neural Network
yes/no


Convolutional
Neural Network
yes/no


u
v
w
Graph Neural Network
yes/no

Research directions

- architecture restrictions/extensions: readout, attention, recurrent[2], higher-order[3], ...
- variations on numbers, aggregation, and activation functions: real/rationals/integers/fixed bitwidth[4],[5], bounded aggregation, eventually constant activations, ...
- expressivity: WL,[6],[7], relative to FO,[8], ..., descriptive complexity,[9] ...
- complexity: verification tasks
- ...

[2] Veeti Ahvonen, Damian Heiman, and Antti Kuusisto. "Graph neural networks and MSO". In: *CoRR* abs/2505.07816 (2025).

[3] Huan Luo and Jonni Virtema. "Unifying approach to uniform expressivity of graph neural networks". In: *CoRR* abs/2602.18409 (2026).

[4] Michael Benedikt et al. "Decidability of Graph Neural Networks via Logical Characterizations". In: *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024*. 2024.

[5] Veeti Ahvonen et al. "Logical characterizations of recurrent graph neural networks with reals and floats". In: *NeurIPS 2024*. 2024.

[6] Martin Grohe. "The Logic of Graph Neural Networks". In: *LICS 2021*. IEEE, 2021.

[7] Pablo Barceló et al. "The Logical Expressiveness of Graph Neural Networks". In: *8th International Conference on Learning Representations (ICLR 2020)*. 2020.

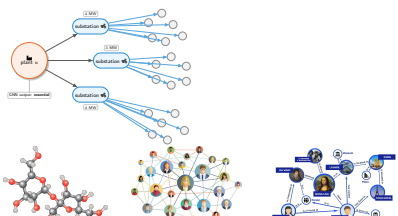
[8] Stan P. Hauke and Przemyslaw Andrzej Walega. "Aggregate-Combine-Readout GNNs Can Express Logical Classifiers Beyond the Logic C2". In: *AAAI 2026*. 2026.

[9] Martin Grohe. "The Descriptive Complexity of Graph Neural Networks". In: *TheoretiCS* 3, 25 (2024).

GNN verification tasks

Critical domains:

- infrastructure networks
- molecules and materials
- social and financial networks
- knowledge graphs



Typical verification questions

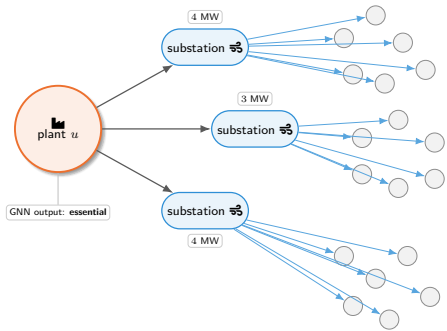
Given a (Boolean classifier) GNN $\mathcal{A}$ and a logical specification φ, decide whether:

- $[[\varphi]] \subseteq [[\mathcal{A}]]$ sufficiency
- $[[\mathcal{A}]] \subseteq [[\varphi]]$ necessity
- $[[\varphi]] \cap [[\mathcal{A}]] \neq \emptyset$ consistency

Example: essential power plants

Examples of verification: "Is it the case that...

- (sufficiency) ... any power plant serving over a 1000 users is classified by the GNN as essential?"
- (necessity) ... any power plant classified by the GNN as essential has at least 3 substations?"
- (consistency) ... a power plant classified by the GNN as essential can be a wind farm whose wind turbines have a total capacity of less than 12 MW?"



Model restrictions

We study aggregate-combine GNNs with deliberately discrete ingredients.

- finite directed labelled graphs;
- integer parameters in $\mathbb{Z}$;
- $\sum$ aggregation;
- linear combinations;
- truncated ReLU activation $\sigma(x) = \max(0, \min(1, x))$;
- linear Boolean classification.

This talk

1. provides some background on AC(R)-GNNs
2. studies the verification tasks of both AC-GNNs and ACR-GNNs.

Model	Capturing logic	Satisfiability / verification
AC-GNN	$K^{\#}$	PSPACE-complete
ACR-GNN	$K^{\#,\#_\forall}$	NEXPTIME-complete / (co)NEXPTIME-complete

Main ideas behind the translations from GNNs to logic and back already in [Barceló et al. 2020][10].

The main contributions are in

- identifying the 'right' logics, and demonstrate that they have
  - right expressivity
  - right succinctness
- and establishing their complexity
  - ... identifying the right already known logics to translate them into.

[10] Barceló et al., "The Logical Expressiveness of Graph Neural Networks".

Outline

Graphs and embeddings

A (labeled directed) graph G is a tuple (V, E, ℓ) such that V is a finite set of vertices, $E \subseteq V \times V$ a set of directed edges and ℓ is a mapping from V to a valuation over a set of atomic propositions.

We write $\ell(u)(p) = 1$ when the atomic proposition p is true in u, and $\ell(u)(p) = 0$ otherwise.

Given a graph G and a vertex $u \in V$, we call (G, u) a pointed graph.

Suppose that the atomic propositions occurring in G are $p_1, \ldots, p_k$. The initial embedding $\mathbf{x}_0$ is defined by:

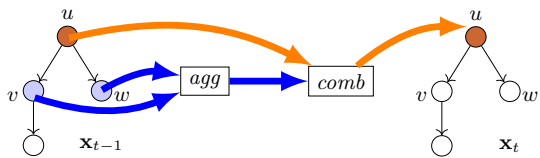
$$\mathbf{x}_0(u) := (\ell(u)(p_1), \ldots, \ell(u)(p_k)) \in \{0, 1\}^k$$

for all $u \in V$.

AC-GNN layer

At each layer, a vertex combines its current state with the sum of the states of its successors.

$$\mathbf{x}_t(u) := comb\Big(\mathbf{x}_{t-1}(u), agg(\{\{\mathbf{x}_{t-1}(v) \mid uv \in E\}\})\Big)$$



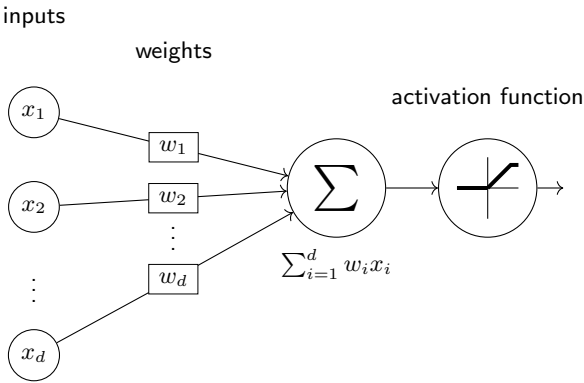
In this work:

$$agg(X) = \sum_{x \in X} x$$

$$comb(x, y) = \vec{\sigma}(xC + yA + b)$$

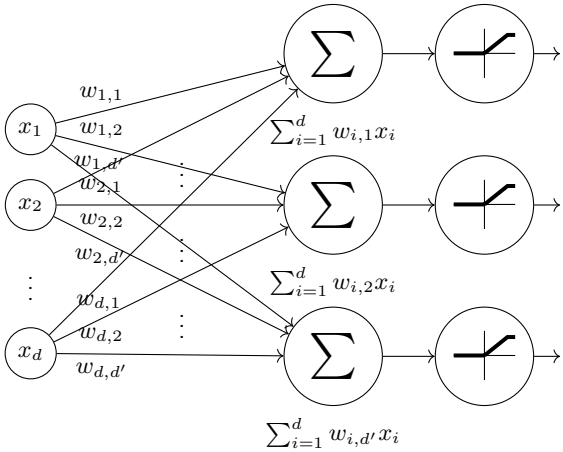
where A and C are matrices of integer parameters, and b an integer vector.

Intermission. The perceptron — a neuron with d inputs



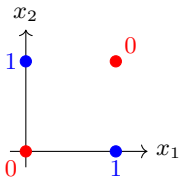
$$\vec{\sigma}\left((x_1, x_2, \ldots, x_d)\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{pmatrix}\right)$$

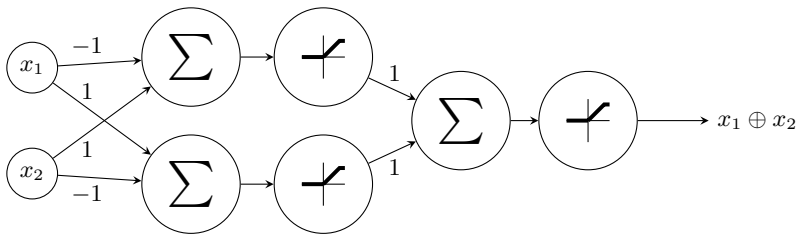
Intermission. The perceptron — a layer with d inputs and d' outputs



$$\vec{\sigma}\left((x_1, x_2, \ldots, x_d)\begin{pmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,d'} \\ w_{2,1} & \ddots & & \\ \vdots & & & \\ w_{d,1} & w_{d,2} & \ldots & w_{d,d'} \end{pmatrix}\right)$$

Intermission. XOR with a multilayer perceptron

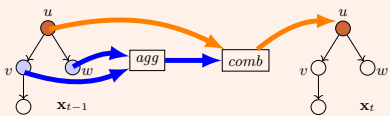




Example

Consider a layer $\mathcal{L}^{(t)}$ with input dimension 2 and output dimension 3, defined by:

- $agg(X) := \sum_{\mathbf{x} \in X} \mathbf{x}$,
- $C = \begin{pmatrix} -1 & -2 & 0 \\ 3 & 0 & -4 \end{pmatrix}$, $A = \begin{pmatrix} 5 & 0 & 0 \\ -6 & 7 & 0 \end{pmatrix}$, and $b = \begin{pmatrix} 8 & 9 & 10 \end{pmatrix}$



Suppose that vertex u has two successors v and w.

- $\mathbf{x}_{t-1}(u) = (1, 3)$, $\mathbf{x}_{t-1}(v) = (1, 1)$, $\mathbf{x}_{t-1}(w) = (0, 1)$

First, $\mathbf{y} = agg(\{\{\mathbf{x}_{t-1}(v), \mathbf{x}_{t-1}(w)\}\}) = \mathbf{x}_{t-1}(v) + \mathbf{x}_{t-1}(w) = (1, 1) + (0, 1) = (1, 2)$.

Second,

$$x_t(u) = comb(\mathbf{x}_{t-1}(u), \mathbf{y}) = \vec{\sigma}(\mathbf{x}_{t-1}(u)C + \mathbf{y}A + b)$$

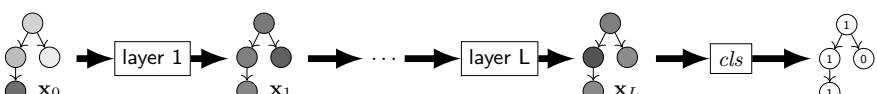
$$x_t(u) = \vec{\sigma}((1, 3) \cdot C + (1, 2) \cdot A + b) = (\sigma(9), \sigma(21), \sigma(-2)) = (1, 1, 0)$$

Semantics of a GNN

Let $\mathcal{A} = (\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(L)}, cls)$.

The semantics $[[\mathcal{A}]]$ is the set of pointed graphs (G, u) such that the classifier accepts the final state of u:

$$(G, u) \in [[\mathcal{A}]] \quad \text{iff} \quad cls(\mathbf{x}_L(u)) = 1.$$



Intermission. Relationship with the Weisfeiler-Leman test

Weisfeiler-Leman (WL) test: heuristic for the graph isomorphism problem [Weisfeiler, Leman 1968]

- color every vertex u of the graph with the same color $c(u) = 1$
- iteratively assign a new color $c(u) = (c(u), \{\{c(v) \mid uv \in E\}\})$, until the color partition stabilizes

If the WL test assigns different sets of colors to two graphs, they are not isomorphic.

Similar to AC-GNNs with injective aggregation and combination functions [Xu et al. 2019][11], [Morris et al. 2023][12]

If WL test assigns the same color to two nodes of a graph, then every AC-GNN classifies them identically.

[11] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *ICLR 2019*. 2019.

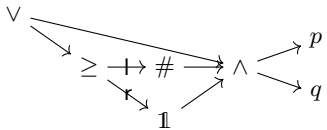
[12] Christopher Morris et al. "Weisfeiler and Leman go Machine Learning: The Story so far". In: *Journal of Machine Learning Research* 24.333 (2023).

The logic $K^{\#}$

The logic adds integer linear constraints over two kinds of quantities.

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \xi \geq 1$$
$$\xi ::= c \mid \mathbb{1}\varphi \mid \#\varphi \mid \xi + \xi \mid c \times \xi$$

Formulas as DAGs: E.g., $(p \wedge q) \vee (\#(p \wedge q) \geq \mathbb{1}(p \wedge q))$



Semantics:

- $[[\mathbb{1}\varphi]]_{G,u}$ is 1 if φ holds at vertex u, and 0 otherwise.
- $[[\#\varphi]]_{G,u}$ counts successors of u satisfying φ.

Simulating (graded) modal logic

$$\Box\varphi := (-1) \times \#(\neg\varphi) + 1 \geq 1$$

$$\Diamond^{\geq k}\varphi := \#\varphi + (-1) \times k + 1 \geq 1$$

Some expressivity beyond FOL

"There are at least as many p-successors as q-successors" is $\#p \geq \#q$, or:

$$\#p + (-1) \times \#q + 1 \geq 1$$

Equivalence: AC-GNNs and $K^{\#}$

Theorem

For every $K^{\#}$-formula φ, one can compute in polynomial time a GNN $\mathcal{A}_\varphi$ such that $[[\varphi]] = [[\mathcal{A}_\varphi]]$.

Theorem

For every AC-GNN $\mathcal{A}$, one can compute in polynomial time a $K^{\#}$-formula $\varphi_\mathcal{A}$ such that $[[\mathcal{A}]] = [[\varphi_\mathcal{A}]]$.

$K^{\#} \iff$ AC-GNN Boolean classifiers

Translation idea: logic → GNN

Enumerate subformulas of φ: $\varphi_1 = p_1, \ldots, \varphi_m = p_m, \ldots, \varphi_n = \varphi$.

Build n identical layers, with $n \times n$ matrices A and C, such that:

- $C_{ii} = 1$ if $i \leq m$,
- $C_{ji} = -1, b_i = 1$ if $\varphi_i = \neg\varphi_j$,
- $C_{ji} = C_{li} = 1$ if $\varphi_i = \varphi_j \vee \varphi_l$, and
- $C_{ji} = k_j, A_{j'i} = k'_{j'}, b_i = c$
  for all $j \in J, j' \in J'$ if $\varphi_i = \sum_{j \in J} k_j \times \mathbb{1}\varphi_j + \sum_{j' \in J'} k'_{j'} \times \#\varphi_{j'} + c \geq 1$.
- $C_{ji} = A_{ji} = 0$ otherwise

We use the classification function $cls(x) = x_n \geq 1$.

From logic to GNN (example)

Example

Consider the formula $\varphi = \neg(p \vee (8 \leq 3 \times \#q))$. We define the following GNN $\mathcal{A}$ which is equivalent to φ as follows. We first consider the following subformulas of φ in that order:

φ_1	φ_2	φ_3	φ_4	φ_5
p	q	$8 \leq 3 \times \#q \equiv 3 \times \#q + (-7) \geq 1$	$\varphi_1 \vee \varphi_3$	$\neg\varphi_4$

The combination function for all layers is $comb(x, y) = \vec{\sigma}(xC + yA + b)$ where

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \text{ and } b = \begin{pmatrix} 0 & 0 & -7 & 0 & 1 \end{pmatrix}.$$

Translation idea: GNN → logic

For each layer t and output coordinate j, build a formula $\varphi_{t,j}$ that says:

"output coordinate j of vertex u after layer t is 1".

The formula mirrors the layer equation:

$$\varphi_{t,j} = \sum_{k=1}^{d_t} C_{kj} \times \mathbb{1}\varphi_{t-1,k} + \sum_{k=1}^{d_t} A_{kj} \times \#\varphi_{t-1,k} + b_j \geq 1.$$

In the end, we set $\varphi_{\mathcal{A}} = a_1 \times \mathbb{1}\varphi_{L,1} + \cdots + a_{d'_L} \times \mathbb{1}\varphi_{L,d'_L} \geq 1$.

From GNN to logic (example)

Example

- Suppose we have two propositions p_1, p_2: the two formulas that represent state $\mathbf{x}_0$ are $\varphi_{01} = p_1$ and $\varphi_{02} = p_2$.

- Suppose we have two layers, given by the same combination and aggregation functions:

$$\mathbf{x}_1(u) = comb(\mathbf{x}_0(u), agg(\{\{\mathbf{x}_0(v) | uv \in E\}\})) \qquad \mathbf{x}_2(u) = comb(\mathbf{x}_1(u), agg(\{\{\mathbf{x}_1(v) | uv \in E\}\}))$$

with aggregation function $agg(X) = \sum_{x \in X} x$, and combination function

$$comb((x, x'), (y, y')) = \vec{\sigma}\left(\begin{pmatrix} x & x' \end{pmatrix} \begin{pmatrix} 1 & 6 \\ 2 & 7 \end{pmatrix} + \begin{pmatrix} y & y' \end{pmatrix} \begin{pmatrix} -3 & 8 \\ 4 & 9 \end{pmatrix} + \begin{pmatrix} 5 & 10 \end{pmatrix} \right)$$

The formulas that represent the state $\mathbf{x}_1$ are formula
$\varphi_{11} = 1 \times \mathbb{1}\varphi_{01} + 2 \times \mathbb{1}\varphi_{02} - 3\#\varphi_{01} + 4\#\varphi_{02} + 5 \geq 1$ and formula
$\varphi_{12} = 6 \times \mathbb{1}\varphi_{01} + 7 \times \mathbb{1}\varphi_{02} + 8\#\varphi_{01} - 9\#\varphi_{02} + 10 \geq 1$.
The formulas that represent the state $\mathbf{x}_2$ are formula
$\varphi_{21} = 1 \times \mathbb{1}\varphi_{11} + 2 \times \mathbb{1}\varphi_{12} - 3\#\varphi_{11} + 4\#\varphi_{12} + 5 \geq 1$ and formula
$\varphi_{22} = 6 \times \mathbb{1}\varphi_{11} + 7 \times \mathbb{1}\varphi_{12} + 8\#\varphi_{11} - 9\#\varphi_{12} + 10 \geq 1$.

- Suppose that the classification function is given by $cls(x, x') = 5x - 3x' \geq 1$.
The formula $\varphi_{\mathcal{A}}$ that represents the GNN $\mathcal{A}$ is $5 \times \mathbb{1}\varphi_{21} - 3 \times \mathbb{1}\varphi_{22} \geq 1$.

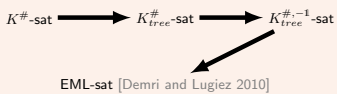
Complexity of $K^{\#}$ satisfiability

Theorem

The satisfiability problem of $K^{\#}$ is PSPACE*-complete.*

Upper bound intuition

Reduce $K^{\#}$ satisfiability to Extended Modal Logic satisfiability [Demri and Lugiez 2010][a].



[a] Stéphane Demri and Denis Lugiez. "Complexity of modal logics with Presburger constraints". In: *J. Appl. Log.* (2010).

Lower bound intuition

Modal logic K embeds into $K^{\#}$, and K-satisfiability is PSPACE-hard.

Complexity of AC-GNN verification

Corollary

The problems sufficiency, necessity and consistency are in PSPACE.

Proof.

E.g., for necessity: given $\mathcal{A}$, φ, we can check that $[[\mathcal{A}]] \subseteq [[\varphi]]$, by computing the $K^{\#}$-formula $\varphi_{\mathcal{A}} \rightarrow \varphi$ and then check that $\varphi_{\mathcal{A}} \rightarrow \varphi$ is valid. □

Theorem

The problems sufficiency, necessity and consistency are PSPACE*-complete.*

Proof.

Poly-time reductions from the satisfiability/unsatisfiability/validity of a $K^{\#}$-formula. Consider $\mathcal{A}_{all}$ a GNN that accepts all graphs, and $\mathcal{A}_{none}$ a GNN that rejects all graphs.

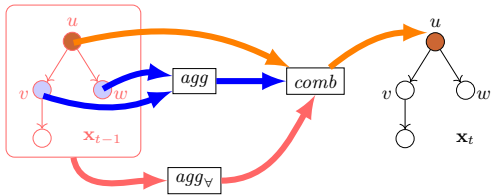
- sufficiency: $[[\varphi]] \subseteq [[\mathcal{A}_{none}]]$ iff φ is unsatisfiable
- necessity: $[[\mathcal{A}_{all}]] \subseteq [[\varphi]]$ iff φ is valid
- consistency: $[[\varphi]] \cap [[\mathcal{A}_{all}]] \neq \emptyset$ iff φ is satisfiable

□

Outline

Adding readout: ACR-GNNs

An ACR-GNN layer adds a global aggregation/readout term.



$$\mathbf{x}_t(u) := comb\Big(\mathbf{x}_{t-1}(u), agg(\{\{\mathbf{x}_{t-1}(v) \mid uv \in E\}\}), agg_\forall(\{\{\mathbf{x}_{t-1}(v) \mid v \in V\}\})\Big)$$

$$agg_\forall(X) = \sum_{x \in X} x$$

$$comb(x, y, z) = \vec{\sigma}(xC + yA + zR + b)$$

The logic $K^{\#,\#_\forall}$

Extend $K^{\#}$ with global counting:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \xi \geq 1$$
$$\xi ::= c \mid \mathbb{1}\varphi \mid \#\varphi \mid \#_\forall\varphi \mid \xi + \xi \mid c \times \xi.$$

$$[[\#_\forall\varphi]]_{G,u} = |\{v \in V \mid (G, v) \models \varphi\}|.$$

Example

"a windfarm who serves at least half of all users":

$$windfarm \wedge (2 \times \#user - \#_\forall user + 1 \geq 1)$$

Equivalence: ACR-GNNs and $K^{\#,\#_\forall}$

Theorem

For every ACR-GNN $\mathcal{A}$, one can compute in polynomial time a $K^{\#,\#_\forall}$-formula $\varphi_{\mathcal{A}}$ such that $[[\mathcal{A}]] = [[\varphi_{\mathcal{A}}]]$.

Theorem

For every $K^{\#,\#_\forall}$-formula φ, one can compute in polynomial time an ACR-GNN $\mathcal{A}_\varphi$ such that $[[\varphi]] = [[\mathcal{A}_\varphi]]$.

$K^{\#,\#_\forall}$ $\iff$ ACR-GNN Boolean classifiers

Complexity of ACR-GNNs verification

Theorem

The satisfiability problem of $K^{\#,\#_\forall}$ *is* NEXPTIME*-complete.*

Upper bound intuition

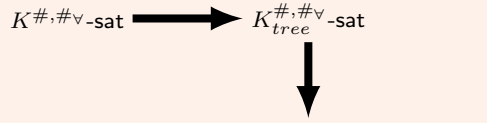
Reduce $K^{\#,\#_\forall}$ satisfiability to $\mathcal{ALCSCC}^{++}$ satisfiability [Baader, Bednarczyk, Rudolph 2020].

A QFBAPA formula is a Boolean combination of:

- set constraints: e.g., $Windfarm \subseteq Powerplant$
- cardinality constraints: e.g., $|Windfarm \cup SolarPlant| \leq |ThermalPlant|$

$\mathcal{ALCSCC}^{++}$ extends $\mathcal{ALC}$ with concepts such as::

- $Windfarm \sqcap \mathsf{sat}(|User| \leq 2 \times |serves \cap User|)$



Lower bound intuition

Reduce from $\mathcal{ALCQ}$-T_CBox consistency [Tobies 2000][a].

[a] Stephan Tobies. "The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics". In: *J. Artif. Intell. Res.* 12 (2000).

Outline

Verification of GNNs

There is hope for the formal verification of GNNs.

The logical equivalences turn GNN verification into satisfiability checking.

Question	Logical test	AC-GNN	ACR-GNN
Sufficiency	$\varphi \to \varphi_{\mathcal{A}}$ valid	PSPACE	coNEXPTIME
Necessity	$\varphi_{\mathcal{A}} \to \varphi$ valid	PSPACE	coNEXPTIME
Consistency	$\varphi \wedge \varphi_{\mathcal{A}}$ satisfiable	PSPACE	NEXPTIME

AC(R)-GNNs with quantized types

Real neural-network implementations use finite computer number formats rather than mathematical integers.

Proposition

The reasoning problems for AC-GNNs over finite computer number formats such as FP32 or INT8 remain PSPACE*-complete.*

Proposition

*The reasoning problems for ACR-GNNs over finite computer number formats such as FP32 or INT8 remain (co)*NEXPTIME*-complete.*

Related work

- Extended Modal Logic: [Demri and Lugiez 2010][13]
- Description logics with cardinality and set constraints: [Baader, Bednarczyk, Rudolph 2020][14]
- QFBAPA: [Kuncak and Rinard 2007][15]
- Similar and more general GNN/logical characterizations: [Benedikt, Lu, Motik, and Tan 2024][16]; [Benedikt, Lu, and Tan 2026][17]
- $K^{\#}$-like logic for transformers: [Yang and Chiang 2024][18]

[13] Demri and Lugiez, "Complexity of modal logics with Presburger constraints".

[14] Franz Baader, Bartosz Bednarczyk, and Sebastian Rudolph. "Satisfiability and Query Answering in Description Logics with Global and Local Cardinality Constraints". In: *ECAI 2020*. 2020.

[15] Viktor Kuncak and Martin Rinard. "Towards Efficient Satisfiability Checking for Boolean Algebra with Presburger Arithmetic". In: *CADE-21*. 2007.

[16] Benedikt et al., "Decidability of Graph Neural Networks via Logical Characterizations".

[17] Michael Benedikt, Chia-Hsuan Lu, and Tony Tan. "Decidability of Graph Neural Networks via Logical Characterizations". In: *ACM Trans. Comput. Logic* (2026).

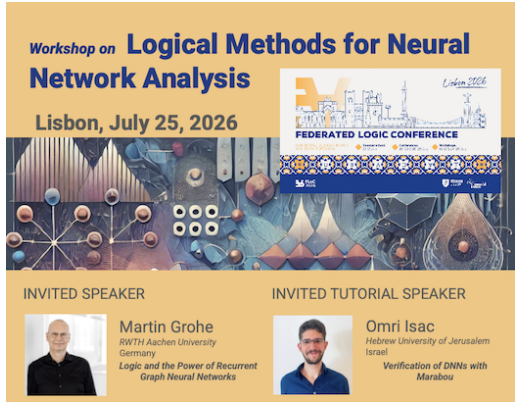
[18] Andy Yang and David Chiang. "Counting Like Transformers: Compiling Temporal Counting Logic Into Softmax Transformers". In: *COLM*. 2024.

Papers

- AC-GNNs:
  - Pierre Nunn et al. "A Logic for Reasoning about Aggregate-Combine Graph Neural Networks". In: *IJCAI 2024*. 2024
  - Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. "Verifying Quantized Graph Neural Networks is PSPACE-complete". In: *IJCAI 2025*. 2025
- ACR-GNNs: Artem Chernobrovkin et al. "Verifying Quantized GNNs With Readout Is Decidable But Highly Intractable". In: *KR 2026*. 2026

GNNs and logic at FLoC

- invited talk at KR (Carsten Lutz – *The Logical Expressive Power of Graph Neural Networks*)
- invited talk at OVERLAY (Przemysław Wałęga – *Preservation Theorems: From Tarski to Graph Neural Networks*)
- invited talk at LOGICNN (Martin Grohe – *Logic and the Power of Recurrent Graph Neural Networks*)
- Several papers at KR
- Several papers at LOGICNN



Graph Neural Networks, Their Logic, and Formal Verification

Nicolas Troquard [1]

[1] Gran Sasso Science Institute, L'Aquila, Italy

joint work with Artem Chernobrovkin, Marco Sälzer, François Schwarzentruber