Proceedings of the 23rd International Conference on Principles of Knowledge Representation and Reasoning
KR meets Machine Learning and Explanation

Verifying Quantized GNNs With Readout Is Decidable But Highly Intractable

Artem Chernobrovkin[1] , **Marco Sälzer**[2] , **François Schwarzentruber**[3] , **Nicolas Troquard**[1]
[1]Gran Sasso Science Institute (GSSI), Viale F. Crispi, 7 – 67100 L'Aquila, Italy
[2]Rheinland-Pfälzische Technische Universität (RPTU), Kaiserslautern, Germany
[3]ENS de Lyon, CNRS, Université Claude Bernard Lyon 1, Inria, LIP, UMR 5668, 69342, Lyon, France
{artem.chernobrovkin, nicolas.troquard}@gssi.it, marco.saelzer@rptu.de,
francois.schwarzentruber@ens-lyon.fr

Abstract

We introduce a logical language for reasoning about quantized aggregate-combine graph neural networks with global readout (ACR-GNNs). We provide a logical characterization and use it to prove that verification tasks for quantized GNNs with readout are (co)NEXPTIME-complete. This result implies that the verification of quantized GNNs is computationally intractable, prompting substantial research efforts toward ensuring the safety of GNN-based systems. We also experimentally demonstrate that quantized ACR-GNN models are lightweight while maintaining good accuracy and generalization capabilities with respect to non-quantized models.

1 Introduction

Graph neural networks (GNNs) are models used for classification and regression tasks on graph-structured data, including node-level and graph-level tasks. GNNs find applications in recommendation systems in social networks (Salamat, Luo, and Jafari 2021), knowledge graphs (Ye et al. 2022), chemistry (Reiser et al. 2022), drug discovery (Xiong et al. 2021), etc. Like several other machine learning models, GNNs are difficult to interpret, understand, and verify. This poses a significant issue for their adoption, morally and legally, with the enforcement of regulatory policies such as the EU AI Act (European Parliament 2024). Previous works lay the foundation for analyzing them using formal logic, as seen in (Barceló et al. 2020), (Nunn et al. 2024), or (Benedikt et al. 2024). However, many of these approaches consider *idealized* GNNs in which numbers are either arbitrarily large integers or rationals, whereas in real-world implementations, GNNs are *quantized*; numbers are represented as Standard IEEE 754 64-bit floats, INT8, or FP8 (Micikevicius et al. 2022). Verification of quantized GNNs without global readout has been addressed by (Sälzer, Schwarzentruber, and Troquard 2025). However, global readout is a crucial component of GNNs, particularly for graph classification (Xu et al. 2019).

In this paper, we consider three verification tasks for GNNs to check critical properties of graphs. Examples of such verification tasks are: "Is it the case that...

- ... any power plant serving over a thousand users is classified by the GNN as essential?"
- ... any power plant classified by the GNN as essential has at least 3 substations?"
- ... 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?"

They are respectively instances of *sufficiency*, *necessity*, and *consistency* tasks that will be formally defined in due time. Verifying critical properties of GNNs helps ensure the safety of GNN-based systems, analogous to how correctness proofs are used to guarantee the safety of traditional computer programs (Hoare 1969).

Contribution. The contribution is threefold. First, we show that verifying the three properties above for Aggregate-Combine Graph Neural Networks with global Readout (ACR-GNNs) is decidable and (co)NEXPTIME-complete. This contrasts with the PSPACE-completeness without global readout as demonstrated by (Sälzer, Schwarzentruber, and Troquard 2025). This implies that global readout makes quantized GNN verification highly intractable[1]. To prove this, we define the logic $q\mathcal{L}$, extending the one introduced by (Sälzer, Schwarzentruber, and Troquard 2025) to capture global readout. It is expressive enough to capture quantized ACR-GNNs with arbitrary activation functions. Moreover, $q\mathcal{L}$ can serve as a flexible graph property specification language reminiscent of modal logics (Blackburn, de Rijke, and Venema 2001). The following example illustrates the use of $q\mathcal{L}$ for expressing graph properties.

Example 1. *Assume a class of knowledge graphs (KGs) representing communities of people and animals, where each node corresponds to an individual. Each individual can be Animal, Human, Leg, Fur, White, Black, etc. These concepts can be encoded with features $x_0, x_1, \ldots, x_5, \ldots$ respectively, taking values 0 or 1. Edges in a KG represent a generic 'has' relationship: a human can have an animal (pet); an animal can have a human (owner), a leg, a fur; a fur can have a color; etc. Suppose that $\mathcal{A}$ is a GNN processing those KGs and is*

[1]As pointed out in (Jogl, Thiessen, and Gärtner 2023), any inference of a ACR-GNN can be simulated by a AC-GNN (Aggregate-Combine Graph Neural Networks *without* global Readout) where the global readout mechanism is replaced by an extra complete relation $V \times V$ where V is the set of vertices of the input graph. So global readout has no significant impact on the complexity of inference. On the contrary, verification tasks involve quantifications over the set of pointed graphs where this extra relation *must* be $V \times V$.

trained to supposedly recognize dogs. We can verify that the nodes recognized by $\mathcal{A}$ are animals—arguably a critical property of the domain—by checking the validity (i.e., the non-satisfiability of the negation) of $\varphi_{\mathcal{A}} \to x_0 = 1$ where $\varphi_{\mathcal{A}}$ is a $q\mathcal{L}$-formula corresponding to $\mathcal{A}$'s computation, true in exactly the pointed graphs accepted by $\mathcal{A}$. Ideally, $\mathcal{A}$ should not overfit the concept of dog as a perfect prototypical animal. For instance, three-legged dogs do exist. We can verify that $\mathcal{A}$ lets it be a possibility by checking the satisfiability of the formula $\varphi_{\mathcal{A}} \wedge \Diamond^{\leq 3}(x_2 = 1)$. More complex $q\mathcal{L}$ formulas can be written to express graph properties to be evaluated against an ACR-GNN, which will be formalized later in Example 2: 1. Has a human owner, whose pets are all two-legged. 2. A human in a community that has more than twice as many animals as humans, and more than five animals without an owner. [2] *3. An animal in a community where some animals have both white and black fur.*

Firstly, to prove the (co)NEXPTIME upper bound, we reuse a concept from mathematical logic called Hintikka sets (Blackburn, de Rijke, and Venema 2001), which are complete sets of subformulas that can be true at a given vertex of a graph. We then introduce a quantized variant of Quantifier-Free Boolean Algebra with Presburger Arithmetic (QFBAPA) logic (Kuncak and Rinard 2007), denoted by $\text{QFBAPA}_{\mathbb{K}}$, and prove that it is in NP, similar to the original QFBAPA on integers. We reduce the satisfiability problem of $q\mathcal{L}$ to that of $\text{QFBAPA}_{\mathbb{K}}$. On the other hand, (co)NEXPTIME-hardness is proven by reduction from a suitable tiling problem. Similarly, we also add global counting to the logic $K^\sharp$ previously introduced by (Nunn et al. 2024). We show that it corresponds to AC-GNNs over $\mathbb{Z}$ with global readout and truncated ReLU activation functions. We prove that the satisfiability problem is NEXPTIME-complete, partially addressing a problem left open in the literature, that is, for the case of integer values and truncated ReLU activation functions (Benedikt et al. 2024). Details are in (Chernobrovkin et al. 2026b) to keep the presentation concise.

Secondly, since NEXPTIME is highly intractable—it is provably distinct from NP (Seiferas, Fischer, and Meyer 1978)—we relax the satisfiability problem of $q\mathcal{L}$ and ACR-GNNs, focusing on finding graph counterexamples where the number of vertices is bounded. This problem is NP-complete, and an implementation is provided as a proof of concept and a baseline for future research.

Finally, we experimentally demonstrate that the quantization of GNNs results in minimal accuracy degradation. Our results confirm that quantized models retain robust predictive performance while achieving substantial reductions in model size and inference cost. These findings demonstrate the practical viability of quantized ACR-GNNs for deployment in resource-constrained environments.

Outline. We introduce quantized ACR-GNNs in Section 2 and present experimental results justifying their practical utility in Section 3. Section 4 defines the logic $q\mathcal{L}$, discusses its expressivity, and defines ACR-GNN verification tasks. Section 5 provides the (co)NEXPTIME membership for the satisfiability problem of $q\mathcal{L}$ and the ACR-GNN verification tasks. In Section 6, we show that these problems are (co)NEXPTIME-complete. Section 7 discusses the relaxation making the problems (co)NP-complete. We complement these results by reporting on the results of a prototype implementation (Chernobrovkin et al. 2026b, Appendix F.7) for verifying said GNNs in a bounded setting, highlighting the hardness of verifying quantized GNNs.

Related work. (Barceló et al. 2020) showed that ACR-GNNs are at least expressive as FOC_2, two-variable first-order logic with counting. (Hauke and Walega 2026) showed that there are ACR-GNNs which can be captured by first-order logic but not FOC_2. Further work has explored the logical expressiveness of GNN variants in more detail. Notably, (Nunn et al. 2024) and (Benedikt et al. 2024) introduced logics to exactly characterize the capabilities of different forms of GNNs. Similarly, (Cucala and Grau 2024) analyzed Max-Sum-GNNs through the lens of Datalog. (Sälzer, Schwarzentruber, and Troquard 2025) considered the expressivity of GNN with quantized parameters but without global readout. (Ahvonen et al. 2024) offered several logical characterizations of recurrent GNNs over floats and real numbers.

On the verification side, (Henzinger, Lechner, and Zikelic 2021) studied the complexity of verification of quantized feedforward neural networks (FNNs), while (Sälzer and Lange 2021; Sälzer and Lange 2023) investigated reachability and reasoning problems for general FNNs and GNNs. Some verification methods have been proposed, via integer linear programming (ILP) by (Huang et al. 2024), and (Zhang et al. 2023), and via model checking by (Sena et al. 2021).

From a logical perspective, reasoning over structures involving arithmetic constraints is closely tied to several well-studied logics. Relevant work includes Kuncak and Rinard's decision procedures for QFBAPA (Kuncak and Rinard 2007), as well as further developments by (Demri and Lugiez 2010), (Baader, Bednarczyk, and Rudolph 2020), (Bednarczyk et al. 2021), and (Galliani, Kutz, and Troquard 2023). These logics form the basis for the characterizations established in (Nunn et al. 2024; Benedikt et al. 2024).

Quantization techniques in neural networks exist, with surveys such as (Gholami et al. 2022; Nagel et al. 2021) providing comprehensive overviews focused on maintaining model accuracy. Although most practical advancements target convolutional neural networks (CNNs), many of the underlying principles extend to GNNs as well (Zhou et al. 2020). NVIDIA has demonstrated hardware-ready quantization strategies (Wu et al. 2020), and frameworks like PyTorch (Ansel et al. 2024) support both post-training quantization and quantization-aware training (QAT), the latter simulating quantization effects during training to improve low-precision performance. QAT has been particularly effective in closing the gap between quantized and full-precision models, especially for highly compressed or edge-deployed systems (Jacob et al. 2018). In the context of GNNs, (Tai-

[2] Interestingly, $q\mathcal{L}$ goes beyond the capabilities of graded modal logic and even first-order logic (FOL). The property of Item 2 cannot be expressed in FOL.

lor, Fernandez-Marques, and Lane 2021) proposed Degree-Quant, incorporating node degree information to mitigate quantization-related issues. Based on this, (Zhu et al. 2023) introduced A^2Q, a mixed-precision framework that adapts bitwidths on graph topology to achieve high compression with minimal performance loss.

2 Background

Quantized numbers. Let $\mathbb{K}$ be a set of quantized numbers, and let n denote the *bitwidth* of $\mathbb{K}$, which is the number of bits required to represent a number in $\mathbb{K}$. The bitwidth n is written in unary; this is motivated by the fact that n is small and that we would, in any case, need to allocate n-bit consecutive memory for storing a number. Formally, we consider a sequence $\mathbb{K}_1, \mathbb{K}_2, \ldots$ corresponding to bitwidths 1, 2, etc., but we retain the notation $\mathbb{K}$ for simplicity. We suppose that $\mathbb{K}$ saturates; for example, if $x \geq 0$ and $y \geq 0$, then $x + y \geq 0$ (i.e., there is no modulo behavior like in `int` in C, for instance). We assume that $1 \in \mathbb{K}$.

We consider Aggregate-Combine Graph Neural Networks with global Readout (ACR-GNNs), a standard class of message-passing GNNs (Barceló et al. 2020; Gilmer et al. 2017).

Graphs. 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 $\mathbb{K}^m$, for some integer m, which is the dimension of the node embedding. Given a graph G and vertex $u \in V$, we call (G, u) a *pointed graph*.

ACR-GNNs. We define quantized Aggregate-Combine GNNs with Readout (ACR-GNN).

An *ACR-GNN layer* $\mathcal{L} = (comb, agg, agg_\forall)$ is a tuple where $comb : \mathbb{K}^{3m} \to \mathbb{K}^{m'}$ is a so-called *combination function*, agg is a so-called *local aggregation function*, mapping multisets of vectors from $\mathbb{K}^m$ to a single vector from $\mathbb{K}^m$, $agg_\forall$ is a so-called *global aggregation function*, also mapping multisets of vectors from $\mathbb{K}^m$ to a single vector from $\mathbb{K}^m$. We call m the *input dimension* of layer $\mathcal{L}$ and m' the *output dimension* of layer $\mathcal{L}$. Then, a *ACR-GNN* is a tuple $(\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(L)}, cls)$ where $\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(L)}$ are L ACR-GNN layers and $cls : \mathbb{K}^m \to \{0, 1\}$ is a *classification function*. We assume that all GNNs are well-formed in the sense that output dimension of layer $\mathcal{L}^{(i)}$ matches input dimension of layer $\mathcal{L}^{(i+1)}$ as well as output dimension of $\mathcal{L}^{(L)}$ matches input dimension of cls.

Let $G = (V, E, x_0)$ be a graph with initial vertex labeling $x_0 : V \to \mathbb{K}^m$, Then, the i-th layer of $\mathcal{A}$ computes an updated state $x_i(u)$ for all $u \in V$ by

$$x_i(u) := comb\Big(x_{i-1}(u),\ agg\big(\{\{x_{i-1}(v) \mid (u, v) \in E\}\}\big),\ agg_\forall\big(\{\{x_{i-1}(v) \mid v \in V\}\}\big)\Big)$$

where agg, $agg_\forall$, and $comb$ are respectively the local aggregation, global aggregation and combination function of the i-th layer. We call each x_i a *state of* G. Let (G, u) be a pointed graph. We write $\mathcal{A}(G, u)$ to denote the application of $\mathcal{A}$ to (G, u), which is formally defined as $\mathcal{A}(G, u) = cls(x_L(u))$ where x_L is the state of G computed by $\mathcal{A}$ after layer L. This corresponds informally to a binary classification of vertex u.

We concentrate on a specific subclass where both agg and $agg_\forall$ perform summation over vectors, and where $comb(x, y, z) = N(x, y, z)$ is a classical feedforward neural network (FNN) using activation function α. The classification function is a linear threshold: $cls(x) = \sum_i a_i x_i + b \geq 1$ with weights and bias $a_i, b \in \mathbb{K}$. We assume that all arithmetic operations are executed according to the arithmetic related to $\mathbb{K}$. It is assumed that the context makes clear the $\mathbb{K}$ and arithmetic being used. We use $[[\mathcal{A}]]$ to denote the set of pointed graphs (G, u) such that $\mathcal{A}(G, u) = 1$. An ACR-GNN $\mathcal{A}$ is satisfiable if $[[\mathcal{A}]]$ is non-empty. The *satisfiability problem* for ACR-GNNs is: Given a ACR-GNN $\mathcal{A}$, decide whether $\mathcal{A}$ is satisfiable.

3 Quantization Effects on Accuracy, Performance and Model Size

To begin on the right foot, we demonstrate that the quantized ACR-GNNs defined before are indeed of practical interest. To this aim, we investigate the application of dynamic Post-Training Quantization (PTQ) to ACR-GNNs. As a reference, we used the models described and analyzed in (Barceló et al. 2020), using their implementation (Barceló et al. 2021) as the baseline. Experiments used two datasets: one synthetic (Erdös–Rényi model) and one real-world dataset (Protein-Protein Interactions (PPI) benchmark by (Zitnik and Leskovec 2017)). In the original work, experiments were made with two activation functions: Rectified Linear Unit (ReLU) and truncated ReLU. Since the models of Section 2 can handle arbitrary activation functions, in our experiments, we used several types of activation function: Piecewise linear (ReLU, ReLU6, and trReLU), Smooth unbounded (GELU and SiLU), Smooth bounded (Sigmoid), and Smooth ReLU-like (Softplus and ELU). The quantization method is implemented in PyTorch (Ansel et al. 2024; PyTorch Team 2024), dynamic PTQ transforms a pre-trained floating-point model into a quantized version without requiring retraining. In this approach, model weights are statically quantized to INT8, while activations remain in floating-point format until they are dynamically quantized at compute time. This hybrid representation enables efficient low-precision computation using INT8-based matrix operations, thereby reducing the memory footprint and improving the inference speed. The PyTorch implementation applies per-tensor quantization to the weights and stores the activations as floating-point values between operations to strike a balance between accuracy and performance. Synthetic graphs (Table 3 of (Chernobrovkin et al. 2026b, Appendix F)) were generated using the dense Erdös–Rényi model, a classical approach to constructing random graphs. Each graph includes five initial node colors, encoded as one-hot feature vectors. Following (Barceló et al. 2020), labels were assigned using formulas from the logic fragment FOC_2. Specifically, a hierarchy of classifiers $\alpha_i(x)$ was defined as follows: $\alpha_0(x) := \mathrm{Blue}(x)$ and

$\alpha_{i+1}(x) := \exists^{[N,M]} y\,(\alpha_i(y) \wedge \neg E(x,y))$, where $\exists^{[N,M]}$ denotes the quantifier "there exist between N and M nodes" satisfying a given condition. Each classifier $\alpha_i(x)$ can be expressed within FOC_2, as the bounded quantifier can be rewritten using $\exists^{\geq N}$ and $\neg\exists^{\geq M+1}$. Each property p_i corresponds to a classifier α_i with $i \in \{1,2,3\}$. For the analysis, we collected training time, model size, and accuracy for both datasets. We list the principal findings of the analysis. More detailed statistics can be found in the (Chernobrovkin et al. 2026b, Appendix F). According to our experimental flow, we first examine the training time. For both datasets, we found that piecewise-linear activation functions consistently achieve the shortest training time (Table 4 and Table 14 of (Chernobrovkin et al. 2026b, Appendix F)). Moreover, for computational efficiency, we found that the model with the activation function (Smooth) was consistently the slowest, regardless of the datasets (Table 6 and Table 15 of (Chernobrovkin et al. 2026b, Appendix F)). We performed an analysis of the accuracy of both data across all activation families, the observed Δ_{acc} values are generally within $\pm 1\%$. In addition,we observed a drop in the accuracy of the baseline models after two layers.

These findings highlight the advantages of quantized ACR-GNN models of Section 2 with respect to non-quantized models, striking a remarkable balance between model size and accuracy. From the evaluation of 9,000 models on Erdös–Rényi and 80 models on real-world data, we selected the following hyperparameters: 1–3 layers, hidden dimension 32 for Erdös–Rényi experiments and 256 for PPI, and a learning rate of 0.01.

We evaluated ACR-GNNs (PPI data) across a range of bit-width configurations (32, 16, 8, 7, 6, 5, 4, 2) and also under finer-grained reductions from 8 to 4 bits. Across both datasets, 16- and 8-bit models preserved accuracy within a small margin of the full-precision baseline. For the PPI dataset (Tables 29 – 36 in (Chernobrovkin et al. 2026b, Appendix F)) exhibits strong robustness to dynamic post-training quantization: performance is preserved down to 6 bits across all activation functions, with only minor deviations (less than 3%) observed in the from 7 to 6 and from 6 to 5 bit transitions. The only marked degradation appears when moving into the very low-precision range (5 bits and below).

4 Logic $q\mathcal{L}$

We set up a logical framework called $q\mathcal{L}$ extending the logic introduced in (Sälzer, Schwarzentruber, and Troquard 2025) with global aggregation: it is a *lingua franca* to represent GNN computations and properties on graphs.

Syntax. Let F be a finite set of features and $\mathbb{K}$ be some finite-width arithmetic. We consider a set of *expressions* defined by the following grammar in Backus-Naur form:

$$\vartheta ::= c \mid x_i \mid \alpha(\vartheta) \mid agg(\vartheta) \mid agg_\forall(\vartheta) \mid \vartheta + \vartheta \mid c \times \vartheta$$

where c is a number in $\mathbb{K}$, x_i is a feature in F, α is a symbol for denoting the activation function, and agg and $agg_\forall$ denote the aggregation function for local and global readout respectively. We define the set of subexpression of ϑ, denoted by $sub(\vartheta)$ inductively by $sub(c) = \{c\}$, $sub(x_i) = \{x_i\}$, $sub(\alpha(\vartheta)) = \{\alpha(\vartheta)\} \cup sub(\vartheta)$, $sub(agg(\vartheta)) = \{agg(\vartheta)\} \cup sub(\vartheta)$, $sub(agg_\forall(\vartheta)) = \{agg_\forall(\vartheta)\} \cup sub(\vartheta)$, $sub(\vartheta+\vartheta') = \{\vartheta+\vartheta\} \cup sub(\vartheta) \cup sub(\vartheta')$, and $sub(c \times \vartheta) = \{c \times \vartheta\} \cup sub(c) \cup sub(\vartheta')$. A *formula* is defined as $\vartheta \geq k$ where ϑ is an expression and k is an element of $\mathbb{K}$. If $-1 \in \mathbb{K}$ we write $-\vartheta$ instead of $(-1) \times \vartheta$ for any expression ϑ. Other standard abbreviations can be used as well.

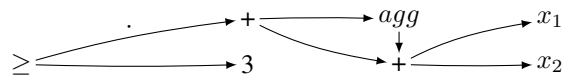

Figure 1: DAG representation of the formula $agg(x_1 + x_2) + (x_1 + x_2) \geq 3$.

Formulas are represented as directed acyclic graphs, aka circuits, meaning that we do not repeat the same expressions several times. For instance, the formula $agg(x_1 + x_2) + (x_1 + x_2) \geq 3$ can be represented as the DAG given in Figure 1. Formulas can also be represented by a sequence of assignments via new fresh intermediate variables. For instance: $y := x_1 + x_2, z := agg(y) + y, res := z \geq 3$.

Semantics. Consider a graph $G = (V, E, \ell)$, where vertices in V are labeled via a labeling function $\ell : V \to \mathbb{K}^{|F|}$ with feature values. The value of an expression ϑ in a vertex $u \in V$ is denoted by $[[\vartheta]]_{G,u}$ and is defined by induction on ϑ:

$$\begin{aligned}
[[c]]_{G,u} &= c,\\
[[x_i]]_{G,u} &= \ell(u)_i,\\
[[\vartheta + \vartheta']]_{G,u} &= [[\vartheta]]_{G,u} +_{\mathbb{K}} [[\vartheta']]_{G,u},\\
[[c \times \vartheta]]_{G,u} &= c \times_{\mathbb{K}} [[\vartheta]]_{G,u},\\
[[\alpha(\vartheta)]]_{G,u} &= [[\alpha]]\big([[\vartheta]]_{G,u}\big),\\
[[agg(\vartheta)]]_{G,u} &= \Sigma_{v|uEv}[[\vartheta]]_{G,v},\\
[[agg_\forall(\vartheta)]]_{G,u} &= \Sigma_{v \in V}[[\vartheta]]_{G,v}.
\end{aligned}$$

We define $[[\vartheta \geq k]] = \{G, u \mid [[\vartheta]]_{G,u} \geq_{\mathbb{K}} [[k]]_{G,u}\}$ (we write $\geq$ for the syntactic symbol and $\geq_{\mathbb{K}}$ for the relation defined in $\mathbb{K}$). A formula φ is satisfiable if $[[\varphi]]$ is non-empty. The *satisfiability problem* for $q\mathcal{L}$ is: Given a $q\mathcal{L}$-formula φ, decide whether φ is satisfiable.

Simulating a modal logic in the logic $q\mathcal{L}$. We show that extending $q\mathcal{L}$ with modal operators (Blackburn, de Rijke, and Venema 2001) does not increase the expressivity. We can even compute an equivalent $q\mathcal{L}$ without Boolean connectives and without modal operators in poly-time. It means that formulas like $\varphi_{\mathcal{A}_1} \to x_0 = 1$ or $\varphi_{\mathcal{A}_1} \wedge \Diamond^{\leq 3}(x_2 = 1)$ have poly-size equivalent formulas in $q\mathcal{L}$.

Let Atm_0 be the set of atomic formulas of $q\mathcal{L}$ of the form $\vartheta \geq 0$. We suppose that ϑ takes integer values. In general, $\vartheta \geq k$ is an atomic formula equivalent to $\vartheta - k \geq 0$. Without loss of generality, we thus assume that formulas of $q\mathcal{L}$ are over Atm_0. Let modal $q\mathcal{L}$ be the propositional logic on Atm_0 extended with modalities and a restricted variant of

graded modalities where number k in $\mathbb{K}$ in the following way:

$$\begin{aligned}
[[\Box\varphi]] &= \{G,u \mid G,v \in [[\varphi]] \text{ for every } v \text{ s.t. } uEv\}\\
[[\Box_g\varphi]] &= \{G,u \mid G,v \in [[\varphi]] \text{ for every } v \text{ in } V\}\\
[[\Diamond^{\geq k}\varphi]] &= \{G,u \mid |\{G,v \mid uEv, G,v \in [[\varphi]]\}| \geq_{\mathbb{K}} k\}\\
[[\Diamond_g^{\geq k}\varphi]] &= \{G,u \mid |[[\varphi]]| \geq_{\mathbb{K}} k\}
\end{aligned}$$

and modalities $\Diamond^{\leq k}\varphi$ and $\Diamond_g^{\leq k}\varphi$ defined the same way but with $\leq_{\mathbb{K}}$.

Example 2 (continuation of Example 1)**.** *We first define a few simple formulas to characterize the concepts of the domain. Let $\varphi_A := x_0 = 1$ (Animal), $\varphi_H := x_1 = 1$ (Human), $\varphi_L := x_2 = 1$ (Leg), $\varphi_F := x_3 = 1$ (Fur), $\varphi_W := x_4 = 1$ (White), and $\varphi_B := x_5 = 1$ (Black).:*

1. *Has a human owner, whose pets are all two-legged: $\Diamond(\varphi_H \wedge \Box(\varphi_A \rightarrow \Diamond^{=2}\varphi_L))$.*
2. *A human in a community that has more than twice as many animals as humans, and more than five animals without an owner: $\varphi_H \wedge (agg_\forall(x_0) - 2 \times agg_\forall(x_1) \geq 0) \wedge \Diamond_g^{\geq 5}(\varphi_A \wedge \Box(\neg\varphi_H))$.*
3. *An animal in a community where some animals have white and black fur: $\varphi_A \wedge \Diamond_g(\Diamond(\varphi_F \wedge \Diamond\varphi_W) \wedge \Diamond(\varphi_F \wedge \Diamond\varphi_B))$.*

We can see the boolean operator $\neg$, and the various modalities as functions from Atm_0 to Atm_0, and the boolean operator $\vee$ as a function from $\mathrm{Atm}_0 \times \mathrm{Atm}_0$ to Atm_0 defined as follows, where α is ReLU ($\alpha(\vartheta) = \vartheta$ if $\vartheta \geq_{\mathbb{K}} 0$ and $\alpha(\vartheta) = 0$ otherwise).

$$\begin{aligned}
f_\neg(\vartheta \geq 0) &:= -\vartheta - 1 \geq 0\\
f_\vee(\vartheta_1 \geq 0, \vartheta_2 \geq 0) &:= \vartheta_1 + \alpha(\vartheta_2 - \vartheta_1) \geq 0\\
f_\Box(\vartheta \geq 0) &:= agg(-\alpha(-\vartheta)) \geq 0\\
f_{\Diamond^{\geq k}}(\vartheta \geq 0) &:= agg(\alpha(\vartheta + 1) - \alpha(\vartheta)) - k \geq 0\\
f_{\Diamond^{\leq k}}(\vartheta \geq 0) &:= k - agg(\alpha(\vartheta + 1) - \alpha(\vartheta)) \geq 0
\end{aligned}$$

For the corresponding global modalities ($f_{\Box_g}(\vartheta \geq 0)$, $f_{\Diamond_g^{\geq k}}(\vartheta \geq 0)$, and $f_{\Diamond_g^{\leq k}}(\vartheta \geq 0)$), it suffices to use $agg_\forall$ in place of agg. The previous transformations can be generalized to arbitrary formulas of a modal version of $q\mathcal{L}$ as follows.

$$\begin{aligned}
mod2expr(\vartheta \geq 0) &:= \vartheta \geq 0\\
mod2expr(\neg\varphi) &:= f_\neg(mod2expr(\varphi))\\
mod2expr(\varphi_1 \vee \varphi_2) &:= f_\vee(mod2expr(\varphi_1), mod2expr(\varphi_2))\\
mod2expr(\boxplus\varphi) &:= f_\boxplus(mod2expr(\varphi)),
\end{aligned}$$

where $\boxplus \in \{\Box, \Box_g, \Diamond^{\geq k}, \Diamond_g^{\geq k}, \Diamond^{\leq k}, \Diamond_g^{\leq k}\}$.

We can show that formulas of modal $q\mathcal{L}$ can be captured by a single expression $\vartheta \geq 0$. This is a consequence of the following lemma [3], proven in (Chernobrovkin et al. 2026b).

Lemma 3. *Let φ be a formula of modal $q\mathcal{L}$. The formulas φ and $mod2expr(\varphi)$ are equivalent.*

[3]For simplicity, we do not present how to handle $\vartheta \geq 0$ when ϑ is not an integer. We could introduce several activation functions α in $q\mathcal{L}$, one of them could be interpreted as the Heavyside step function. In Definition 7, Point 4 is just repeated for each α.

Link between $q\mathcal{L}$ and ACR-GNN verification. We focus on the following decision problems in context of the verification of ACR-GNNs:

- (VT1, sufficiency) Given a GNN $\mathcal{A}$ and a $q\mathcal{L}$ formula φ, decide whether $[[\varphi]] \subseteq [[\mathcal{A}]]$.
- (VT2, necessity) Given a GNN $\mathcal{A}$ and a $q\mathcal{L}$ formula φ, decide whether $[[\mathcal{A}]] \subseteq [[\varphi]]$.
- (VT3, consistency) Given a GNN $\mathcal{A}$ and a $q\mathcal{L}$ formula φ, decide whether $[[\varphi]] \cap [[\mathcal{A}]] \neq \emptyset$.

Informally, these problems are described as follows: problem VT1 consists of verifying that pointed graphs satisfying the specification φ are classified positively by the GNN $\mathcal{A}$, while VT2 is the reverse, namely verifying that any pointed graphs positively classified by $\mathcal{A}$ satisfies φ. Problem VT3 consists in verifying whether $\mathcal{A}$ can classify some pointed graph satisfying φ.

Essential to our results is the straightforward insight that $q\mathcal{L}$ where α is given by ReLU and ACR-GNN are equally expressive.

Theorem 4. *Given an ACR-GNN $\mathcal{A}$, we can compute $q\mathcal{L}$-formula in poly-time in the size of $\mathcal{A}$ with $[[\mathcal{A}]] = [[\varphi_\mathcal{A}]]$. Vice-versa, with $\alpha = ReLU$, given a $q\mathcal{L}$ formula φ, we can compute an ACR-GNN $\mathcal{A}_\varphi$ in poly-time, with $[[\mathcal{A}_\varphi]] = [[\varphi]]$.*

Proof. We begin by arguing that for each $\mathcal{A}$ there is an equivalent $q\mathcal{L}$-formula. The key insight needed here is that $q\mathcal{L}$ is tailored to capture the computation of an ACR-GNN. Let $\mathcal{A} = (\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(L)}, cls)$ be an ACR-GNN. Let $\mathcal{L}^{(1)} = (comb^{(1)}, agg, agg_\forall)$ be the first layer of $\mathcal{A}$ computing $comb^{(1)}((x_1,\ldots,x_m),(y_1,\ldots,y_m),(z_1,\ldots,z_m))$ where $comb^{(1)}$ is represented by a feedforward neural network N_1 using activation function α. The vectors $(y_1,\ldots,y_m)$ and $(z_1,\ldots,z_m)$ correspond to the vectors aggregated by agg and $agg_\forall$, respectively. Let $v_{11}(x'_1,\ldots,x'_n) = \alpha(b_1 + \sum_{i=1}^n w_i x'_i)$ be a node of N_1 in the first layer where x'_i corresponds to either some x_i, y_i or z_i as determined by $comb^{(1)}$. We build an equivalent $q\mathcal{L}$-expression $\vartheta_{v_{11}}$ combining c, $\vartheta + \vartheta$, $c \times \vartheta$ and $\alpha(\vartheta)$ to represent the arithmetic and activation function, and we use x_i in the case that $x'_i = x_i$, we use $agg(x_i)$ in the case that $x'_i = y_i$, and we use $agg_\forall(x_i)$ in the case that $x'_i = z_i$. We do so for all nodes v_{1j} in the first layer of N_1. This results in a sequence of expressions $\vartheta_{\mathcal{L}^{(1)}11},\ldots,\vartheta_{\mathcal{L}^{(1)}1k_1}$ where k_1 is the layer size of the first layer of N_1. For subsequent layers, we use the same construction, but the expressions $\vartheta_{\mathcal{L}^{(1)}1j}$ as atomic. This results in a sequence of expressions $\vartheta_{\mathcal{L}^{(1)}m1},\ldots,\vartheta_{\mathcal{L}^{(1)}m_1k}$ where k is the size of the last layer m_1 of N_1, and also the output dimensionality of $\mathcal{L}^{(1)}$. We continue as before for GNN layer $\mathcal{L}^{(i)}$, but use $\varphi_{\mathcal{L}^{(i-1)}m_jj}$ as atomic expressions. The final classification condition cls is then given by the formula $\vartheta \geq 1$ where ϑ uses $\varphi_{\mathcal{L}^{(L)}m_Lj}$ as atomic as well as multiplicative parameters a_i and bias b determined by cls.

For the other direction, namely showing that for each $q\mathcal{L}$-formula there is an equivalent ACR-GNN, let φ be a $q\mathcal{L}$-formula $\vartheta \geq k$ over variables $x_1,\ldots x_m$. We remark

that the argument follows the same line of reasoning as in expressivity results of previous works such as (Sälzer, Schwarzentruber, and Troquard 2025; Nunn et al. 2024; Barceló et al. 2020). We note $sub(\vartheta)$ the set of subexpressions of ϑ. Let $\vartheta_1, \ldots, \vartheta_n$ be an enumeration of $sub(\vartheta)$ such that if $\vartheta_i \in sub(\vartheta_j)$ then we have $i \leq j$. We construct an ACR-GNN $\mathcal{A}_\varphi$ with n layers as follows. First, the input and output dimensionality of each layer of $\mathcal{A}$ is n, meaning that we have one dimension per subexpression. W.l.o.g. we assume that the first m dimensions correspond to the values of the basic variables $x_1, \ldots, x_m$ of ϑ. Furthermore, we assume for all layers $\mathcal{L}^{(i)}$ that they do not adjust any other dimension than the ith one of a state. Note that this can easily be realised using FNN with ReLU activation. Then, if $\vartheta_i = x_j$ layer $\mathcal{L}^{(i)}$ effectively does nothing in the sense that the FNN N_i representing the combination function simply represents the identity. In the cases $\vartheta_i = c$, $\vartheta_i = \vartheta + \vartheta$, $\vartheta_i = x \times \vartheta$, or $\vartheta_i = \alpha(\vartheta)$ we also only utilise FNN N_i to do the arithmetic operation or applying activation function α. Note that this involves the assumption that previous layers evaluated all necessary subexpressions already. Finally, for the cases $\vartheta_i = agg(\vartheta)$ and $\vartheta_i = agg_\forall(\vartheta)$ we simply use agg and $agg_\forall$ functions of layer $\mathcal{L}^{(i)}$ to resolve it. Then, in the *cls* of $\mathcal{A}_\varphi$ we evaluate $\varphi = \vartheta \geq k$ by choosing weights $a_i = 1$ and bias $b = 0$. □

The following example highlights the construction idea from ACR-GNNs to $q\mathcal{L}$ of Theorem 4.

Example 5. *To reason formally about ACR-GNNs, we represent their computations using $q\mathcal{L}$. Consider an ACR-GNN $\mathcal{A}$ with two layers of input and output dimension 2, using summation for aggregation, truncated ReLU as activation $\alpha(x) = \max(0, \min(1, x)) = [[\alpha]](x)$, and a classification function $2x_1 - x_2 \geq 1$. The combination functions are:*

$$comb_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) := \begin{pmatrix} \alpha(2x_1 + x_2 + 5y_1 - 3y_2 + 1) \\ \alpha(-x_1 + 4x_2 + 2y_1 + 6y_2 - 2) \end{pmatrix}^\intercal,$$

$$comb_2(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) := \begin{pmatrix} \alpha(3x_1 - y_1 + 2z_2) \\ \alpha(-2x_1 + 5y_2 + 4z_1) \end{pmatrix}^\intercal.$$

Note that this assumes that $\mathcal{A}$ operates over $\mathbb{K}$ with at least three bits. Then, the corresponding $q\mathcal{L}$ formula $\varphi_\mathcal{A}$ is given by:

$$\begin{aligned} \psi_1 &:= \alpha(2x_1 + x_2 + 5agg(x_1) - 3agg(x_2) + 1) \\ \psi_2 &:= \alpha(-x_1 + 4x_2 + 2agg(x_1) + 6agg(x_2) - 2) \\ \chi_1 &:= \alpha(3\psi_1 - agg(\psi_1) + 2(agg_\forall(\psi_2))) \\ \chi_2 &:= \alpha(-2\psi_1 + 5(agg(\psi_2)) + 4agg_\forall(\psi_1)) \\ \varphi_A &:= 2(\chi_1) - \chi_2 \geq 1. \end{aligned}$$

To sum up, given a GNN $\mathcal{A}$, we compute $q\mathcal{L}$-formula in poly-time in the size of $\mathcal{A}$ with $[[\mathcal{A}]] = [[\varphi_\mathcal{A}]]$. The vice-versa direction works analogously.

Finally, ACR-GNN verification tasks can be solved by reduction to the satisfiability problem of $q\mathcal{L}$:

- VT1 by checking that $\varphi \wedge \neg\varphi_\mathcal{A}$ is not satisfiable
- VT2 by checking that $\neg\varphi \wedge \varphi_\mathcal{A}$ is not satisfiable
- VT3 by checking that $\varphi \wedge \varphi_\mathcal{A}$ is satisfiable

5 Complexity Upper Bound

In this section, we prove the NEXPTIME membership of reasoning in modal quantized logic, and also of solving of ACR-GNN verification tasks (by reduction to the former).

Theorem 6. *Suppose that $[[\alpha]]$ is computable in exponential-time in the bit-width n of $\mathbb{K}$. The satisfiability problem of $q\mathcal{L}$ is decidable and in NEXPTIME, and so is* VT3. *Consequently,* VT1 *and* VT2 *are in coNEXPTIME.*

In order to prove Theorem 6, we borrow from the ideas behind the proof of the NEXPTIME membership of concept satisfiability in the description logic $\mathcal{ALCSCC}^{++}$ from (Baader, Bednarczyk, and Rudolph 2020) (which adapts a proof in (Baader and Ecke 2017) of the complexity of $\mathcal{ALC}$ ECBoxes consistency). We adapt it to the logic $q\mathcal{L}$. The difference resides in the definition of Hintikka sets and the treatment of quantization. The idea is to encode the constraints of a $q\mathcal{L}$-formula φ in a formula of exponential length of a quantized version of QFBAPA, that we prove to be in NP.

5.1 Hintikka Sets

Consider $q\mathcal{L}$-formula φ. Let $E(\varphi)$ be the set of subexpressions in φ. For instance, if φ is $agg(\alpha(x_2 + agg_\forall(x_1))) \geq 5$ then $E(\varphi) = \{agg(\alpha(x_2 + agg_\forall(x_1))), \alpha(x_2 + agg_\forall(x_1)), x_2, agg_\forall(x_1), x_1\}$. From now on, we consider equality formulas that are of the form $\vartheta{=}k$ where ϑ is a subexpression of φ and $k \in \mathbb{K}$.

Definition 7. *A Hintikka set H for φ is a subset of $\{\vartheta{=}k \mid \vartheta \in E(\varphi), k \in \mathbb{K}\}$, such that:*

1. *For all $\vartheta \in E(\varphi)$, there is a unique value $k \in \mathbb{K}$ such that $\vartheta = k \in H$*
2. *For all $c \in E(\varphi) \cap \mathbb{K}$, $c = c \in H$*
3. *For all $\vartheta_1{+}\vartheta_2 \in E(\varphi)$, if $\vartheta_1{=}k_1, \vartheta_2{=}k_2 \in H$ then $\vartheta_1{+}\vartheta_2{=}k' \in H$, where $k' = k_1{+}_\mathbb{K}k_2$*
4. *For all $c \times \vartheta \in E(\varphi)$, if $\vartheta = k \in H$ then $c \times \vartheta{=}k' \in H$ where $k' = c \times_\mathbb{K} k$*
5. *For all $\alpha(\vartheta) \in E(\varphi)$, $\vartheta{=}k \in H$ then $\alpha(\vartheta){=}k' \in H$ where $k' = [[\alpha]](k)$*

Informally, a *Hintikka set* contains equality subformulas obtained from a choice of a value for each subexpression of φ (point 1), provided that the set is consistent *at the current vertex* (points 1–5). The notion of Hintikka set does not take any constraints about agg and $agg_\forall$ into account since checking consistency of aggregation would require information about the neighbors or the whole graph.

Example 8. *If φ is $agg(\alpha(x_2 + agg_\forall(x_1))) \geq 5$, then an example of Hintikka set is: $\{agg(\alpha(x_2 + agg_\forall(x_1)) = 8, \alpha(x_2 + agg_\forall(x_1)) = 9, x_2 + agg_\forall(x_1) = 9, x_2 = 7, agg_\forall(x_1) = 2, x_1 = 5\}$.*

Proposition 9. *The number of Hintikka sets is bounded by $2^{n|\varphi|}$ where $|\varphi|$ is the size of φ, and n is the bitwidth of $\mathbb{K}$.*

5.2 Quantized Version of QFBAPA

A QFBAPA formula is a propositional formula where each atom is either an inclusion of sets or equality of sets or linear constraints ((Kuncak and Rinard 2007), and (Chernobrovkin

et al. 2026b, Appendix D.2)). Sets are denoted by Boolean algebra expressions, e.g., $(S \cup S') \setminus S''$, or $\mathcal{U}$ where $\mathcal{U}$ denotes the set of all points in some domain. Here S, S', etc., are set variables. Linear constraints are over $|S|$ denoting the cardinality of the set denoted by the set expression S. For instance, the QFBAPA-formula $(windfarm \subseteq powerplant) \wedge (|powerplant| + |\mathcal{U} \setminus windfarm| \geq 6) \wedge (|powerplant| < 2)$ is read as 'all wind farms are power plants, and the number of power plants plus the number of entities that are not wind farms is greater than 6 and the number of power plants is smaller than 2'.

We now introduce a *quantized* version QFBAPA$_{\mathbb{K}}$ of QFBAPA. It has the same syntax as QFBAPA except that numbers in expressions are in $\mathbb{K}$. Semantically, every arithmetic expression is interpreted in $\mathbb{K}$. For each set expression S, the interpretation of $|S|$ is not the cardinality c of the interpretation of S, but the result of the computation $1 + 1 + \ldots + 1$ in $\mathbb{K}$ with c occurrences of 1 in the sum.

We consider $\mathbb{K}$ that saturates, meaning that if $x + y$ exceeds the upper bound limit of $\mathbb{K}$, there is a special value denoted by $+\infty$ such that $x + y = +\infty$.

Proposition 10. *If the bitwidth n is in unary, and if $\mathbb{K}$ saturates, then satisfiability in QFBAPA$_{\mathbb{K}}$ is in NP.*

5.3 Reduction to QFBAPA$_{\mathbb{K}}$

Let φ be a formula of $q\mathcal{L}$. For each Hintikka set H, we introduce the set variable X_H that intuitively represents the H-vertices, i.e., the vertices in which all subformulas in H hold. Moreover, for each formula $\vartheta' = k$ in $\{\vartheta{=}k \mid \vartheta \in E(\varphi), k \in \mathbb{K}\}$, we introduce the set variable $X_{\vartheta'=k}$ that intuitively represents the vertices in which $\vartheta' = k$ holds. We capture this with two QFBAPA$_{\mathbb{K}}$-formulas. Equation (1) expresses that $\{X_H\}_H$ form a partition of the universe. Equation (2) makes the bridge between variables $X_{\vartheta'=k}$ and X_H.

$$\bigwedge_{H \neq H'} (X_H \cap X_{H'} = \emptyset) \wedge (\bigcup_{H} X_H = \mathcal{U}) \tag{1}$$

$$\bigwedge_{\vartheta' \in E(\varphi)} \bigwedge_{k \in \mathbb{K}} (X_{\vartheta'=k} = \bigcup_{H \mid \vartheta'=k \in H} X_H) \tag{2}$$

We introduce also a variable S_H that denotes the set of all successors of some H-vertex. If there is no H-vertex then the variable S_H is just irrelevant. The QFBAPA$_{\mathbb{K}}$-formula in Equation (3) encodes the semantics of $agg(\vartheta)$. More precisely, it says that for all subexpressions $agg(\vartheta)$, for all values k, for all Hintikka sets H containing formula $agg(\vartheta){=}k$, if there is some H-vertex (i.e., some vertex in X_H), then the aggregation obtained by summing over the successors of some H-vertex is k:

$$\bigwedge_{agg(\vartheta) \in E(\varphi)} \bigwedge_{k \in \mathbb{K}} \bigwedge_{H \mid agg(\vartheta)=k \in H} ((X_H \neq \emptyset) \rightarrow \sum_{k' \in \mathbb{K}} |S_H \cap X_{\vartheta=k'}| \times k' = k) \tag{3}$$

In the previous sum, we partition S_H into subsets $S_H \cap X_{\vartheta=k'}$ for all possible values k'. Each contribution for a successor in $S_H \cap X_{\vartheta=k'}$ is k'. We rely here on the fact[4] that $(1 + 1 + \ldots + 1) \times k' = k' + k' + \ldots + k'$. We also fix a specific order over the values k' in the summation (it means that $agg(\vartheta)$ is computed as follows: first order the successors according to the taken values of ϑ in that specific order, then perform the summation). Finally, the semantics of $agg_\forall$ is captured by the QFBAPA$_{\mathbb{K}}$-formula:

$$\bigwedge_{agg_\forall(\vartheta) \in E(\varphi)} \bigwedge_{k \in \mathbb{K}} X_{agg_\forall(\vartheta)=k} \neq \emptyset \rightarrow \sum_{k' \in \mathbb{K}} |X_{\vartheta=k'}| \times k' = k \tag{4}$$

Note that intuitively Equation (4) implies that for $X_{agg_\forall(\vartheta)=k}$ is interpreted as the universe, for the value k which equals the semantics of $\sum_{k' \in \mathbb{K}} |X_{\vartheta=k'}| \times k'$.

Finally, given $\varphi = \vartheta \geq k$, we define

$$tr(\varphi) := \psi_{1\text{–}4} \wedge \bigvee_{k' \geq_{\mathbb{K}} k} X_{\vartheta=k'} \neq \emptyset$$

where $\psi_{1\text{–}4}$ is the conjunction of Formulas 1–4. The function tr requires to compute all the Hintikka sets. In particular, all of them need to satisfy point 4 of Definition 7. Therefore, we get the following when $[[\alpha]]$ is computable in time exponential in n.

Proposition 11. *$tr(\varphi)$ is of exponential size and is computable in time exponential in $|\varphi|$ and n.*

Crucially, the translation $tr(\ldots)$ preserves satisfiability.

Proposition 12. *Let φ be a formula of $q\mathcal{L}$. φ is satisfiable iff $tr(\varphi)$ is QFBAPA$_{\mathbb{K}}$ satisfiable.*

Proof sketch. From left to right, suppose that the $q\mathcal{L}$ formula φ is satisfiable. Then there exist a graph $G = (V, E, \ell)$ and $u \in V$ such that $G, u \models \varphi$. We define the QFBAPA$_{\mathbb{K}}$-substitution σ as follows:

- $\sigma(\mathcal{U}) := V$,
- $\sigma(X_{\vartheta=k}) := \{w \in V \mid [[\vartheta]]_{G,w} = k\}$,
- $\sigma(X_H) := \{w \in V \mid G, w \models \bigwedge H\}$,
- $\sigma(S_H) := \{w \mid \exists v \in \sigma(X_H) \text{ and } (v, w) \in E\}$.

We then show that σ satisfies $tr(\varphi)$.

Every vertex induces a unique Hintikka set. Hence the sets $\sigma(X_H)$ form a partition of V, yielding Equation (1). By definition of $\sigma(X_{\vartheta=k})$ and the construction of Hintikka sets, Equation (2) also holds.

Consider $agg(\vartheta) = k \in H$. If $\sigma(X_H) = \emptyset$, then the implication in Equation (3) is trivial. Otherwise, let $v \in \sigma(X_H)$. Then $\sigma(\sum_{k' \in \mathbb{K}} |S_H \cap X_{\vartheta=k'}| \times k')$ counts the contribution of all successors of v, grouped by their ϑ-value. This coincides with $\sum_{w \mid vEw} [[\vartheta]]_{G,w}$, hence equals k. So Equation (3) is satisfied by σ. Equation (4) is also satisfied by σ by the same reasoning.

Finally, as $G, u \models \varphi$ and φ is of the form $\vartheta \geq k$, there exists $k' \geq_{\mathbb{K}} k$ with $u \in \sigma(X_{\vartheta=k'})$. Thus the last disjunct of $tr(\varphi)$ holds and σ satisfies $tr(\varphi)$.

[4] This is true for some fixed-point arithmetics but not for floating-point arithmetics. See (Chernobrovkin et al. 2026b, Appendix F.8).

Conversely, from right to left, let σ be a QFBAPA$_{\mathbb{K}}$-substitution that satisfies $tr(\varphi)$. Using the definition of $tr(\varphi)$ and Definition 7, we can prove that σ also satisfies the following formulas:

$$\bigwedge_{\vartheta} \bigwedge_{k \neq_{\mathbb{K}} k'} (X_{\vartheta=k} \cap X_{\vartheta=k'} = \emptyset) \wedge (\bigcup_{k} X_{\vartheta=k} = \mathcal{U}) \quad (5)$$

$$\bigwedge_{c} X_{c=c} = \mathcal{U} \quad (6)$$

$$\bigwedge_{\vartheta_1+\vartheta_2 \in E(\varphi)} \bigwedge_{k_1,k_2} X_{\vartheta_1=k_1} \cap X_{\vartheta_2=k_2} \subseteq X_{\vartheta_1+\vartheta_2=k_1+_{\mathbb{K}}k_2} \quad (7)$$

$$\bigwedge_{c\times\vartheta \in E(\varphi)} \bigwedge_{k} X_{\vartheta=k} = X_{c\times\vartheta=c\times_{\mathbb{K}}k} \quad (8)$$

$$\bigwedge_{\alpha(\vartheta) \in E(\varphi)} \bigwedge_{k} X_{\vartheta=k} = X_{\alpha(\vartheta)=[[\alpha]](k)} \quad (9)$$

(It is presented as Lemma 18 in (Chernobrovkin et al. 2026b).)

We construct a graph $G = (V, E, \ell)$ such that:

- $V := \sigma(\mathcal{U})$,
- $E := \{(u, v) \mid \exists H, u \in \sigma(X_H) \text{ and } v \in \sigma(S_H)\}$,
- $\ell(v)_i := k$ where $v \in \sigma(X_{x_i=k})$.

The set of vertices is the (QFBAPA$_{\mathbb{K}}$) universe, and we add an edge between any H-vertex u and a vertex $v \in \sigma(S_H)$. The labeling is well defined because of Equation (5).

The substitution σ satisfies $tr(\varphi)$, and therefore the formula $\bigvee_{k' \geq_{\mathbb{K}} k} X_{\vartheta=k'} \neq \emptyset$. So there is $k' \geq_{\mathbb{K}} k$ and $u \in \mathcal{U}$ such that $u \in \sigma(X_{\vartheta=k'})$. that $G, u \models \varphi$.

We can then prove by structural induction on ϑ that if $u \in \sigma(X_{\vartheta=k})$ then $[[\vartheta]]_{G,u} = k$. For the base cases, we use Equation (6) for constants, while the definition of ℓ is used directly for variables. The cases of sum, product, and activation functions use Equation (7), Equation (8), and Equation (9) respectively. The cases of aggregations exploit Equation (3) and Equation (4).

□

Finally, in order to check whether a $q\mathcal{L}$-formula φ is satisfiable, we construct a QFBAPA$_{\mathbb{K}}$-formula $tr(\varphi)$ of exponential size in exponential time. As the satisfiability problem of QFBAPA$_{\mathbb{K}}$ is in NP, we obtain that the satisfiability problem of $q\mathcal{L}$ is in NEXPTIME. We proved Theorem 6.

Remark 13. *Our methodology can be generalized to reason in subclasses of graphs. For instance, we may tackle the problem of satisfiability in a graph where vertices are of bounded degree bounded by d. To do so, we add the constraint $\bigwedge_H |S_H| \leq d$.*

6 Complexity Lower Bound

The NEXPTIME upper-bound is tight. Having defined modalities in $q\mathcal{L}$ and stated Lemma 3, Theorem 14 is proven by adapting the proof of NEXPTIME-hardness of deciding the consistency of $\mathcal{ALCQ}$-T_CBoxes presented in (Tobies 2000).

NEXPTIME-hardness is proven via a reduction from the tiling problem by Wang tiles of a torus of size $2^n \times 2^n$.

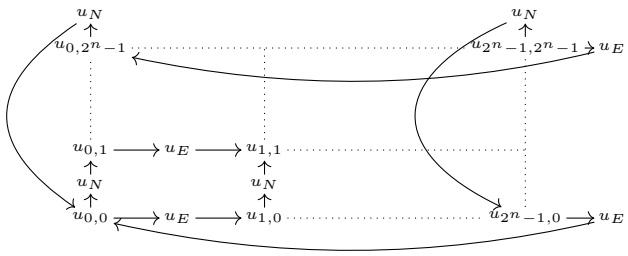

Figure 2: Encoding a torus of exponential size with (modal) $q\mathcal{L}$ formulas. Vertices $u_{x,y}$ correspond to locations (x, y) in the torus while u_N and u_E denote intermediate vertices indicating the direction (resp., north and east).

A Wang tile is a square with colors, e.g., , , etc. The problem takes as input a number n in unary, and Wang tile types, and an initial condition—let us say the bottom row is already given. The objective is to decide whether the torus of $2^n \times 2^n$ can be tiled while colors of adjacent Wang tiles match. A slight difficulty resides in adequately capturing a two-dimensional grid structure—as in Figure 2—with only a single relation. To do that, we introduce special formulas φ_E and φ_N to indicate the direction (east or north). In the formula computed by the reduction, we also need to bound the number of vertices corresponding to tile locations by $2^n \times 2^n$. Thus $\mathbb{K}$ needs to encode $2^n \times 2^n$. We need a bit-width of at least $2n$.

Theorem 14. *The satisfiability problem in $q\mathcal{L}$ is NEXPTIME-hard, and so is the ACR-GNN verification task* VT3. *The ACR-GNN verification tasks* VT1 *and* VT2 *are coNEXPTIME-hard. The results already hold when $\alpha = ReLU$.*

Remark 15. *It turns out that the verification task only needs the fragment of $q\mathcal{L}$ where agg is applied directly on an expression $\alpha(\vartheta)$. Indeed, this is the case when we represent a GNN in $q\mathcal{L}$ or when we translate logical formulas in $q\mathcal{L}$ (Lemma 3). Reasoning about $q\mathcal{L}$ when $\mathbb{K} = \mathbb{Z}$ and the activation function is truncated ReLU is also NEXPTIME-complete (see (Chernobrovkin et al. 2026b, Appendix E)).*

7 Bounding the Number of Vertices

The satisfiability problem of $q\mathcal{L}$ is NEXPTIME-complete, thus far from tractable. The high complexity arises because counterexamples can be arbitrary large graphs. However, one usually looks for small counterexamples. Let $\mathcal{G}^{\leq N}$ be the set of pointed graphs with at most N vertices. We consider the $q\mathcal{L}$ and ACR-GNN *satisfiability problem with a bound on the number of vertices* and ACR-GNNs verification tasks: given a number N given in unary, 1. given a $q\mathcal{L}$-formula φ, is it the case that $[[\varphi]] \cap \mathcal{G}^{\leq N} \neq \emptyset$, 2. given an ACR-GNN $\mathcal{A}$, is it the case that $[[\mathcal{A}]] \cap \mathcal{G}^{\leq N} \neq \emptyset$. In the same way, we introduce the following verification tasks. Given a GNN $\mathcal{A}$, a $q\mathcal{L}$ formula φ, and a number N in unary: (VT'1, sufficiency) Do we have $[[\varphi]] \cap \mathcal{G}^{\leq N} \subseteq [[\mathcal{A}]] \cap \mathcal{G}^{\leq N}$? (VT'2, necessity) Do we have $[[\mathcal{A}]] \cap \mathcal{G}^{\leq N} \subseteq [[\varphi]] \cap \mathcal{G}^{\leq N}$? (VT'3, consistency) Do we have $[[\varphi]] \cap [[\mathcal{A}]] \cap \mathcal{G}^{\leq N} \neq \emptyset$?

Theorem 16. *The satisfiability problems with bounded number of vertices are NP-complete, so is ACR-GNN verification task* VT'3*, while the verification tasks* VT'1 *and* VT'2 *are coNP-complete.*

The satisfiability NP upper bound is obtained by guessing a graph with at most N vertices and then check that φ holds. NP-hardness already holds for agg-free formulas by reduction from SAT for propositional logic using the reduction of Lemma 3. The complexity of the bounded verification tasks follow from simple reductions.

It is possible to extend the methodology proposed in (Sena et al. 2021) to the verification of Graph Neural Networks. However, an efficient SMT encoding for GNN verification tasks would constitute a substantial contribution in itself. In this work, we therefore restrict ourselves to a proof of concept that establishes a baseline for future research (see (Chernobrovkin et al. 2026b, Appendix F.7) for performance details). For the verification task, we employ ESBMC (Efficient SMT-based Context-Bounded Model Checker) (Menezes et al. 2024). The verification workflow (Figure 7 in (Chernobrovkin et al. 2026b, Appendix F)) translates a trained PyTorch model into C code that can be analyzed by ESBMC, embedding the corresponding preconditions and postconditions.

After evaluating the verification workflow, we identified two principal limitations affecting the formal verification phase. First, the process is constrained by SMT encodability: only activation functions that admit efficient logical representations can be supported, thereby restricting the architectural design space of GNNs (see Table 2 in (Chernobrovkin et al. 2026b, Appendix F)). Second, scalability is governed by the size of the generated SMT formula, which increases with both the number of graph vertices and the model dimensionality. Our prototype evaluation confirms that even small increases in these parameters lead to substantial verification overhead due to the expansion of symbolic variables and constraints.

In particular, results on the Erdös–Rényi dataset (Table 11 in (Chernobrovkin et al. 2026b, Appendix F)) empirically illustrate the state-space explosion inherent in SMT-based verification of GNNs. Increasing the graph size from one to two vertices causes a dramatic rise in verification time, highlighting the poor scalability of naive full symbolic encodings for graph-structured neural models. This behavior arises because each additional vertex introduces new symbolic variables for node features, aggregation outputs, linear transformations, and activation constraints; furthermore, ReLU activations induce case distinctions that significantly increase the number of feasible solver paths, thereby aggravating solver complexity.

8 Conclusion and Future Work

The main result is the (co)NEXPTIME-completeness of verification tasks for GNNs with global readout. It helps to understand the inherent complexity, and demonstrates that the verification of ACR-GNNs is highly intractable. To further guide future research, we provide extensive experiments demonstrating that quantised GNNs are essential. However, naive approaches to verifying GNNs, even over a set of graphs with a bounded number of vertices, quickly reach their limits, as expected, given the inherently high complexity. This prompts significant efforts of the research community towards ensuring the safety of quantised GNN-based systems.

There are many directions to go. First, characterizing the modal flavor of $q\mathcal{L}$—a powerful graph property specification language—for other activation functions than ReLU. New extensions of $q\mathcal{L}$ could also be proposed to tackle other classes of GNNs. Verification of neural networks is challenging and is currently tackled by the verification community (Cordeiro et al. 2025). So it will be for GNNs as well. Our verification tool with a bound on the number of vertices is still preliminary and a mere baseline for future research. One obvious path would be to improve it, to compare different approaches (bounded model checking vs. linear programming as in (Huang et al. 2024)) and to apply it to real GNN verification scenarios. Designing a practical verification procedure in the general case (without any bound on the number of vertices) and overcoming the high computational complexity is an exciting challenge for future research towards the verification of GNNs.

Limitations. Section 5 and 6 reflect theoretical results. Some practical implementations of GNNs may not fully align with them. In particular, the order in the (non-associative) summation over values in $\mathbb{K}$ is fixed in formulas (3) and (4). It means that we suppose that the aggregation $agg(\vartheta)$ is computed in that order too (we sort the successors of a vertex according to the values of ϑ and then perform the summation).

Reproducibility. The full proofs are provided in (Chernobrovkin et al. 2026b).

To ensure reproducibility, we provide a replication package containing the implementation, trained models, and verification scripts. The package is publicly available via a GitHub repository (Chernobrovkin et al. 2026a).

The implementation of the verification prototype is included in the supplementary material and can be found in the folder 'src_verificationtool'.

The replication package for the experimental evaluation of ACR-GNN quantization is provided in the folder code_notebooks_csv. The Code subfolder contains the Python implementation, accompanied by a README.md file that documents the project structure and provides detailed instructions for reproducing the experiments. The Notebooks subfolder includes the analysis scripts, the corresponding .csv files, and additional documentation describing their usage.

AI Declaration

During the preparation of this work, some authors used Grammarly for grammar and spelling correction. No content was generated by artificial intelligence tools, and all scientific contributions, analyses, and conclusions are the sole responsibility of the authors.

Acknowledgements

Troquard acknowledges the support of the MUR (Italy) Department of Excellence 2023–2027 for GSSI.

References

Ahvonen, V.; Heiman, D.; Kuusisto, A.; and Lutz, C. 2024. Logical characterizations of recurrent graph neural networks with reals and floats. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Ansel, J.; Yang, E.; He, H.; Gimelshein, N.; Jain, A.; Voznesensky, M.; Bao, B.; Bell, P.; Berard, D.; Burovski, E.; Chauhan, G.; Chourdia, A.; Constable, W.; Desmaison, A.; DeVito, Z.; Ellison, E.; Feng, W.; Gong, J.; Gschwind, M.; Hirsh, B.; Huang, S.; Kalambarkar, K.; Kirsch, L.; Lazos, M.; Lezcano, M.; Liang, Y.; Liang, J.; Lu, Y.; Luk, C.; Maher, B.; Pan, Y.; Puhrsch, C.; Reso, M.; Saroufim, M.; Siraichi, M. Y.; Suk, H.; Suo, M.; Tillet, P.; Wang, E.; Wang, X.; Wen, W.; Zhang, S.; Zhao, X.; Zhou, K.; Zou, R.; Mathews, A.; Chanan, G.; Wu, P.; and Chintala, S. 2024. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM.

Baader, F., and Ecke, A. 2017. Extending the description logic ALC with more expressive cardinality constraints on concepts. In Benzmüller, C.; Lisetti, C. L.; and Theobald, M., eds., *GCAI 2017, 3rd Global Conference on Artificial Intelligence, Miami, FL, USA, 18-22 October 2017*, volume 50 of *EPiC Series in Computing*, 6–19. EasyChair.

Baader, F.; Bednarczyk, B.; and Rudolph, S. 2020. Satisfiability and query answering in description logics with global and local cardinality constraints. In Giacomo, G. D.; Catalá, A.; Dilkina, B.; Milano, M.; Barro, S.; Bugarín, A.; and Lang, J., eds., *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 616–623. IOS Press.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J. L.; and Silva, J. P. 2020. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J. L.; and Silva, J. P. 2021. GNN-logic. `https://github.com/juanpablos/GNN-logic.git`.

Bednarczyk, B.; Orlowska, M.; Pacanowska, A.; and Tan, T. 2021. On classical decidable logics extended with percentage quantifiers and arithmetics. In Bojanczyk, M., and Chekuri, C., eds., *41st IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2021, December 15-17, 2021, Virtual Conference*, volume 213 of *LIPIcs*, 36:1–36:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Benedikt, M.; Lu, C.; Motik, B.; and Tan, T. 2024. Decidability of graph neural networks via logical characterizations. In Bringmann, K.; Grohe, M.; Puppis, G.; and Svensson, O., eds., *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, 127:1–127:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Blackburn, P.; de Rijke, M.; and Venema, Y. 2001. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press.

Chernobrovkin, A.; Sälzer, M.; Schwarzentruber, F.; and Troquard, N. 2026a. Replication Package for Verifying Quantized GNNs With Readout Is Decidable But Highly Intractable. `https://github.com/francoisschwarzentruber/kr2026-Verifying-Quantized-GNNs-With-Readout-Is-Decidable-But-Highly-Intractable`. GitHub repository.

Chernobrovkin, A.; Sälzer, M.; Schwarzentruber, F.; and Troquard, N. 2026b. Verifying Quantized GNNs With Readout Is Decidable But Highly Intractable. *arXiv preprint arXiv:2510.08045*.

Cordeiro, L. C.; Daggitt, M. L.; Girard-Satabin, J.; Isac, O.; Johnson, T. T.; Katz, G.; Komendantskaya, E.; Lemesle, A.; Manino, E.; Sinkarovs, A.; and Wu, H. 2025. Neural network verification is a programming language challenge. *arXiv preprint arXiv.2501.05867*.

Cucala, D. J. T., and Grau, B. C. 2024. Bridging max graph neural networks and Datalog with negation. In Marquis, P.; Ortiz, M.; and Pagnucco, M., eds., *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024*.

Demri, S., and Lugiez, D. 2010. Complexity of modal logics with presburger constraints. *J. Appl. Log.* 8(3):233–252.

European Parliament. 2024. Artificial Intelligence Act.

Galliani, P.; Kutz, O.; and Troquard, N. 2023. Succinctness and complexity of ALC with counting perceptrons. In Marquis, P.; Son, T. C.; and Kern-Isberner, G., eds., *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, 291–300.

Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M. W.; and Keutzer, K. 2022. A survey of quantization methods for efficient neural network inference. In *Low-power computer vision*. Chapman and Hall/CRC. 291–326.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In Precup, D., and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70

of *Proceedings of Machine Learning Research*, 1263–1272. PMLR.

Hauke, S. P., and Walega, P. A. 2026. Aggregate-Combine-Readout GNNs Can Express Logical Classifiers Beyond the Logic C2. In Koenig, S.; Jenkins, C.; and Taylor, M. E., eds., *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2026, Singapore, January 20-27, 2026*, 21594–21601. AAAI Press.

Henzinger, T. A.; Lechner, M.; and Zikelic, D. 2021. Scalable verification of quantized neural networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 3787–3795. AAAI Press.

Hoare, C. 1969. An axiomatic basis for computer programming. *Communications of the ACM* 12(10):576–580.

Huang, P.; Wu, H.; Yang, Y.; Daukantas, I.; Wu, M.; Zhang, Y.; and Barrett, C. W. 2024. Towards efficient verification of quantized neural networks. In Wooldridge, M. J.; Dy, J. G.; and Natarajan, S., eds., *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, 21152–21160. AAAI Press.

Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; and Kalenichenko, D. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2704–2713.

Jogl, F.; Thiessen, M.; and Gärtner, T. 2023. Expressivity-preserving GNN simulation. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Kuncak, V., and Rinard, M. 2007. Towards efficient satisfiability checking for boolean algebra with presburger arithmetic. In Pfenning, F., ed., *Automated Deduction – CADE-21*, 215–230. Berlin, Heidelberg: Springer Berlin Heidelberg.

Menezes, R.; Aldughaim, M.; Farias, B.; Li, X.; Manino, E.; Shmarov, F.; Song, K.; Braußе, F.; Gadelha, M. R.; Tihanyi, N.; Korovin, K.; and Cordeiro, L. C. 2024. ESBMC 7.4: Harnessing the Power of Intervals. In 30^{th} *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'24)*, volume 14572 of *Lecture Notes in Computer Science*, 376–380. Springer.

Micikevicius, P.; Stosic, D.; Burgess, N.; Cornea, M.; Dubey, P.; Grisenthwaite, R.; Ha, S.; Heinecke, A.; Judd, P.; Kamalu, J.; Mellempudi, N.; Oberman, S. F.; Shoeybi, M.; Siu, M. Y.; and Wu, H. 2022. FP8 Formats for Deep Learning. *arXiv preprint arXiv.2209.05433*.

Nagel, M.; Fournarakis, M.; Amjad, R. A.; Bondarenko, Y.; van Baalen, M.; and Blankevoort, T. 2021. A white paper on neural network quantization. *arXiv preprint arXiv.2106.08295*.

Nunn, P.; Sälzer, M.; Schwarzentruber, F.; and Troquard, N. 2024. A logic for reasoning about aggregate-combine graph neural networks. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024*, 3532–3540. International Joint Conferences on Artificial Intelligence Organization.

PyTorch Team. 2024. Quantization — PyTorch 2.x Documentation. `https://pytorch.org/docs/stable/quantization.html`. Accessed: 2025-05-16.

Reiser, P.; Neubert, M.; Eberhard, A.; Torresi, L.; Zhou, C.; Shao, C.; Metni, H.; van Hoesel, C.; Schopmans, H.; Sommer, T.; and Friederich, P. 2022. Graph neural networks for materials science and chemistry. *Communications Materials* 3(93).

Salamat, A.; Luo, X.; and Jafari, A. 2021. Heterographrec: A heterogeneous graph-based neural networks for social recommendations. *Knowl. Based Syst.* 217:106817.

Sälzer, M., and Lange, M. 2021. Reachability is NP-complete even for the simplest neural networks. In Bell, P. C.; Totzke, P.; and Potapov, I., eds., *Reachability Problems - 15th International Conference, RP 2021, Liverpool, UK, October 25-27, 2021, Proceedings*, volume 13035 of *Lecture Notes in Computer Science*, 149–164. Springer.

Sälzer, M., and Lange, M. 2023. Fundamental limits in formal verification of message-passing neural networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Sälzer, M.; Schwarzentruber, F.; and Troquard, N. 2025. Verifying Quantized Graph Neural Networks is PSPACE-complete. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025*, 4660–4668. International Joint Conferences on Artificial Intelligence Organization.

Seiferas, J. I.; Fischer, M. J.; and Meyer, A. R. 1978. Separating nondeterministic time complexity classes. *J. ACM* 25(1):146–167.

Sena, L. H.; Song, X.; da S. Alves, E. H.; Bessa, I.; Manino, E.; and Cordeiro, L. C. 2021. Verifying Quantized Neural Networks using SMT-Based Model Checking. *arXiv preprint arXiv.2106.05997*.

Tailor, S. A.; Fernandez-Marques, J.; and Lane, N. D. 2021. Degree-quant: Quantization-aware training for graph neural networks. In *International Conference on Learning Representations*.

Tobies, S. 2000. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Intell. Res.* 12:199–217.

Wu, H.; Judd, P.; Zhang, X.; Isaev, M.; and Micikevicius, P. 2020. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv.004.09602*.

Xiong, J.; Xiong, Z.; Chen, K.; Jiang, H.; and Zheng, M. 2021. Graph neural networks for automated de novo drug design. *Drug Discovery Today* 26(6):1382–1393.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ye, Z.; Kumar, Y. J.; Sing, G. O.; Song, F.; and Wang, J. 2022. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access* 10:75729–75741.

Zhang, Y.; Zhao, Z.; Chen, G.; Song, F.; Zhang, M.; Chen, T.; and Sun, J. 2023. Qvip: An ILP-based formal verification approach for quantized neural networks. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22. New York, NY, USA: Association for Computing Machinery.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI open* 1:57–81.

Zhu, Z.; Li, F.; Mo, Z.; Hu, Q.; Li, G.; Liu, Z.; Liang, X.; and Cheng, J. 2023. A^2Q: Aggregation-aware quantization for graph neural networks. In *The Eleventh International Conference on Learning Representations*.

Zitnik, M., and Leskovec, J. 2017. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics* 33(14):i190–i198.