Verifying Quantized Graph Neural Networks is PSPACE-complete

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

Abstract

In this paper, we investigate the verification of quantized Graph Neural Networks (GNNs), where some fixed-width arithmetic is used to represent numbers. We introduce the linear-constrained validity (LVP) problem for verifying GNNs properties, and provide an efficient translation from LVP instances into a logical language. We show that LVP is in PSPACE, for any reasonable activation functions. We provide a proof system. We also prove PSPACE-hardness, indicating that while reasoning about quantized GNNs is feasible, it remains generally computationally challenging.

1 Introduction

Graph Neural Networks (GNNs) are neural network models computing functions over graphs or graph-node pairs. Their ability to extract feature and structural information from graphs has made GNNs a promising candidate for tackling tasks in fields such as social network analysis, chemistry applications, and knowledge graphs. For a comprehensive survey of GNN applications, see Zhou *et al.* [2020], and for a survey of GNNs for knowledge graphs, see Ye *et al.* [2022].

The increasing use of GNNs (or neural network-based models in general) leads to a heightened necessity for reliable safety guarantees or explanations of their behavior, in order to meet legal requirements or client desiderata. Let us illustrate some guarantees we may expect.

Example 1. *Consider a GNN N used to identify bots in a fictitious social network* `NEWSNET` *(see Figure 1). The network forms a graph where any registered account is a node and the edges represent relationships between accounts. In such a setting, the GNN decides, based on account-specific features, such as the frequency of posts or time of activity, whether an account is a regular human user or a (malicious) bot.*

For the trustworthiness of `NEWSNET`*, it is essential that the GNN N comes with guarantees such as:*

- A *"Every account that spams* 100 *messages or more per minute is identified as a bot by N."*
- A' *"If there is a significant, humanly impossible activity of a user within a short amount of time, then N will identify it as a bot."*

Figure 1: Setting where a GNN N is used to identify whether an account is a human or a bot in a fictitious social network.

- B *"Every account whose friends send more than* 1000 *messages per minute in total is flagged as part of a bots' network by N."*
- B' *"If an account is a friend with active bots, then N will flag it as a bot by association."*

The properties A and B can be seen as safety properties, indicating that N is somewhat robust in its decision making. The properties A' and B' provide reasonable explanations for the decision-making process of N.

Unfortunately, like most neural network-based models, GNNs exhibit a black-box nature, making them particularly challenging to analyze. The ultimate goal in this context is *formal reasoning*, which involves using sound and complete procedures to interpret the behavior of GNNs, and to certify or falsify specific safety properties of GNNs. Unfortunately, recent works such as [Sälzer and Lange, 2023; Benedikt *et al.*, 2024] suggest that formal reasoning is practically intractable for highly expressive models like GNNs. Consequently, most of the research so far has focused on less rigorous types of reasoning, such as sound but incomplete or probabilistic procedures. For a comprehensive overview of non-formal verification procedures, see Günnemann [2022].

Several works have explored the expressivity and the verification of GNNs through logical frameworks ([Barceló *et al.*, 2020; Benedikt *et al.*, 2024; Nunn *et al.*, 2024; Ahvonen *et al.*, 2024; Cucala and Grau, 2024]). However, much of this work focuses on idealized GNNs with real-valued or unbounded integer parameters, whereas practical GNNs are typically *quantized*—their numerical parameters and internal computations are constrained by fixed-size representations. For example, the PyG library ([Fey and Lenssen, 2019]) uses 32- or 64-bit floating-point arithmetic, and recent research increasingly targets quantized models with reduced precision ([Gholami *et al.*, 2021; Zhu *et al.*, 2023]). In addition, most existing results focus on eventually constant activation functions such as truncated ReLU ([Barceló *et al.*, 2020; Nunn *et al.*, 2024]), or provide only partial results for standard ReLU ([Benedikt *et al.*, 2024]). Orthogonally, [Ahvonen *et al.*, 2024] investigates the expressivity of *recurrent* GNNs over floats, while [Cucala and Grau, 2024] addresses a weaker class of GNNs but considers a broader range of activation functions.

The contributions of the paper are as follows.

- We introduce a logic $\mathcal{L}_{\mathsf{quantGNN}}$ capturing a meaningful class of quantized (aggregate-combine) GNNs as well as linear input and output constraints on these GNNs (Section 3).
- We introduce the linear-constrained validity problem (LVP) that enables to model properties like the ones in Example 1. We show how to solve LVP by reducing it to the satisfiability problem of $\mathcal{L}_{\mathsf{quantGNN}}$ (Section 4).
- We provide a proof system that enables to create graph counterexamples. A prototype of the proof system is implemented in Python. Noticeably, the proof system gives a PSPACE upper bound for a large class of GNNs, with any reasonable activation function (Section 5).
- We also study the PSPACE lower bound and establish that LVP as well as satisfiability of $\mathcal{L}_{\mathsf{quantGNN}}$ are indeed PSPACE-complete (Section 6).

2 Fundamentals

Fixed-width Arithmetic We consider different sets of numbers, denoted by $\mathbb{K}$, represented in binary using a fixed number of bits. Depending on the underlying (fixed-width) arithmetic, like fixed- or floating-point arithmetic, we denote the respective variants of operations like $+, \cdot, \div$ or relations like $\leq, \geq, =$ by $+_{\mathbb{K}}, \leq_{\mathbb{K}}, \dots$ and so on. For mathematically rigorous definitions of such arithmetics, see, for example, Ercegovac and Lang; Goldberg [2004; 1991]. To keep the notation uncluttered, we use $\mathbb{K}$ to refer to the set of values as well as the underlying fixed-width arithmetic. In general, we assume $\mathbb{K}$ contains 0, 1 and -1.

Example 2. *16-bit fixed-point arithmetic may be used in micro-controllers. For instance, $\mathbb{K}$ may be the set of numbers, of the form $(-1)^s \frac{k}{10^4}$ with four decimal places precision, where $s \in \{0, 1\}$ is the sign, $k \in \{0, \dots, 2^{15} - 1\}$.*

Example 3. *In IEEE754 32-bit floating-point arithmetic, $\mathbb{K}$ exactly contains 0, $-\infty$, $+\infty$ and numbers that are of the form $(-1)^s 2^e (1 + \frac{k}{2^{23}})$ where $s \in \{0, 1\}$ is the sign, $e \in \{-128, -125, \dots, 127\}$, $k \in \{0, \dots, 2^{23} - 1\}$.*

Graphs We consider directed, vector-labelled graphs $G = (V, E, \ell)$ where V is a finite set of nodes, or vertices, E is a set of directed edges and $\ell : V \to \mathbb{R}^m$. We denote the class of all such graphs by $\mathcal{G}$. We call a pair (G, v) where v is a node of G a *pointed graph*. We denote the class of all pointed graphs by $\mathcal{G}_\bullet$. If we restrict the classes $\mathcal{G}$ and $\mathcal{G}_\bullet$ to vector labels with values from some fixed-width arithmetic $\mathbb{K}$, we denote this by $\mathcal{G}[\mathbb{K}]$ and $\mathcal{G}_\bullet[\mathbb{K}]$.

Example 4. *Figure 2 shows a vector-labelled graph with three nodes. Here $\ell : V \to \mathbb{R}^2$ and $\ell(v) = \begin{pmatrix}1\\1\end{pmatrix}$ for instance.*

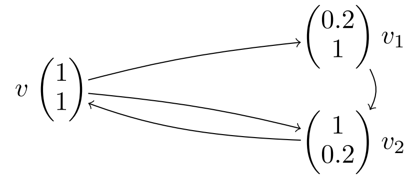

Figure 2: Example of a vector-labelled graph.

Graph Neural Networks A *Graph Neural Network (GNN)* N is a tuple $(\mathcal{L}_1, \dots, \mathcal{L}_m, \mathcal{L}_{out})$. Each layer $\mathcal{L}_i$ computes $\mathcal{L}_i(\boldsymbol{x}, M) = comb_i(\boldsymbol{x}, agg_i(M))$ where $\boldsymbol{x}$ is a real-valued vector, M is a multiset of vectors, agg is called an *aggregation function*, mapping a multiset of vectors onto a single vector, and $comb$ is called a *combination function*. Unless explicitly stated otherwise, agg denotes the sum. Layer $\mathcal{L}_{out}$ computes a function from vectors to vectors. The GNN N computes a function $\mathcal{G}_\bullet \to \mathbb{R}^n$, where n is the *output dimension of* N, in the following way. Let (G, v) where $G = (V, E, \ell)$ be a pointed graph. First, N computes a state $\boldsymbol{x}_u^m$ for all $u \in V$ in a layer-wise fashion by $\boldsymbol{x}_u^i = comb_i(\boldsymbol{x}_u^{i-1}, agg_i(\{\{\boldsymbol{x}_{u'}^{i-1} \mid (u, u') \in E\}\}))$ and $\boldsymbol{x}_u^0 = \ell(u)$. The overall output $N(G, v)$ is given by $\mathcal{L}_{out}(\boldsymbol{x}_v^m)$. If we restrict the computation of N to some fixed-width arithmetic, it computes a function $\mathcal{G}_\bullet[\mathbb{K}] \to \mathbb{K}^n$ and we assume that all numerical parameters of N as well as internal computations are carried out in $\mathbb{K}$. We call N *quantized (to $\mathbb{K}$)* in this case.

Example 5. *Consider a GNN $N = (\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_{out})$ with $\mathcal{L}_i(\boldsymbol{x}, M) = comb(\boldsymbol{x}, agg(M))$ where agg is the sum and with the same combination function $comb : \mathbb{R}^{2\cdot 2} \to \mathbb{R}^2$ in both layers where $comb((x, x'), (a, a'))$ is given by*

$$\begin{pmatrix}\alpha(x + 2x' - 3a + 4a' + 5)\\ \alpha(6x + 7x' + 8a - 9a' + 10)\end{pmatrix}$$

for some activation function α, such as ReLU. The output layer $\mathcal{L}_{out}$ computes $\mathbb{R}^2 \to \mathbb{R}$ and is given by $\alpha(x_1 + x_2 - 2)$.

3 Boolean Combinations Over Modal Expressions

To reason about broad classes of GNNs and overcome the limitations of existing logical characterizations, we propose the logic $\mathcal{L}_{\mathsf{quantGNN}}$, which enables reasoning about general but quantized GNNs with arbitrary fixed-width arithmetic and activation functions. Existing characterizations, such as graded modal logic ([Barceló *et al.*, 2020]), modal logic on linear inequalities over counting ([Nunn *et al.*, 2024]), or fragments of Presburger logic ([Benedikt *et al.*, 2024]), typically relate to GNNs with eventually constant activation functions, like truncated ReLU $x \mapsto \max(0, \min(1, x))$. In contrast, $\mathcal{L}_{\mathsf{quantGNN}}$ extends beyond these constraints using the sole assumption of a quantized setting.

3.1 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:

$$\xi ::= c \mid x_i \mid \alpha(\xi) \mid agg(\xi) \mid \xi + \xi \mid c \times \xi$$

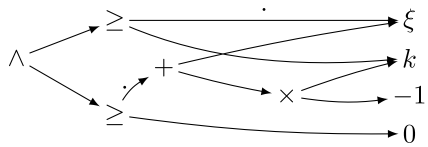

Figure 3: DAG representation for $(\xi \geq k) \wedge (-1{\times}\xi + k \geq 0)$.

where $c \in \mathbb{K}$, $x_i \in F$. A *formula* of the logic $\mathcal{L}_{\textsf{quantGNN}}$ is then a Boolean formula where the atomic formulas are of the form $\xi \geq k$ for some $k \in \mathbb{K}$. When there is no ambiguity and to lighten the notation, we sometimes use more standard expressions, like $-x$ instead of $-1 \times x$, or $c\xi$ instead of $c \times \xi$, etc. We use $\xi = k$ as an abbreviation for $(\xi \geq k) \wedge (-1 \times \xi + k \geq 0)$. Formulas are represented as DAGs. For example, in $(\xi \geq k) \wedge (-1 \times \xi + k \geq 0)$ the expressions ξ and k are *not* duplicated as shown in Figure 3. The symbol $\top$ denotes the tautology (any universally true formula, e.g. $x_1 - x_1 \geq 0$).

3.2 Semantics

Recall that $\mathbb{K}$ is a finite set of numbers, equipped with an addition $+_{\mathbb{K}}$, multiplication $\times_{\mathbb{K}}$ operation, and comparison operations $\leq_{\mathbb{K}}$, $<_{\mathbb{K}}$. We fix $[\![\alpha]\!]$ to be an activation function. For instance, $[\![\alpha]\!] = ReLU$, with $ReLU(x) = x$ if $x \geq_{\mathbb{K}} 0$ and 0 otherwise. Formulas are evaluated over the class of pointed, labelled graphs $G = (V, E, \ell)$ where $\ell : V \to \mathbb{K}^I$, defined as follows. The semantics $[\![\xi]\!]_u$ of an expression ξ with respect to a vertex $u \in V$ is inductively defined on ξ:

$$
\begin{aligned}
[\![c]\!]_u &= c & [\![c \times \xi]\!]_u &= c \times_{\mathbb{K}} [\![\xi]\!]_u \\
[\![x_i]\!]_u &= \ell(u)_i & [\![\alpha(\xi)]\!]_u &= [\![\alpha]\!]([\![\xi]\!]_u) \\
[\![\xi + \xi']\!]_u &= [\![\xi]\!]_u +_{\mathbb{K}} [\![\xi']\!]_u & [\![agg(\xi)]\!]_u &= \Sigma_{v|uEv} [\![\xi]\!]_v
\end{aligned}
$$

The semantics $[\![x_i]\!]_u$ is the value of i-th feature in vertex u. Note that $agg(\xi)$ is motivated by the message-aggregation process in a GNN and is the 'modal' part of $\mathcal{L}_{\textsf{quantGNN}}$. Its semantics $[\![agg(\xi)]\!]_u$ consists in summing up the values/semantics of ξ in the successors v of u. We write $G, u \models \varphi$ to say that the formula φ is true in the pointed graph (G, u), and we say that (G, u) is a *model* of φ. Given $\delta \in \mathbb{N} \cup \{+\infty\}$, a formula φ is *δ-satisfiable* (simply satisfiable if $\delta = +\infty$) if it has a model in which each vertex is of out-degree (number of successors) at most δ.

Example 6. *The formula $x_1 + \alpha(x_2) \geq 0$ is true in the pointed graphs (G, u) where the sum of the first feature at u plus the result of the application of the activation function on the second feature at u is positive (i.e., $(\ell(u))_1 + [\![\alpha]\!](\ell(u))_2 \geq_{\mathbb{K}} 0$). The formula $agg(1) = 4$ is true in the pointed graphs (G, u) with u having exactly 4 successors. The formula $agg(3) = 10$ is unsatisfiable.*

4 Enabling Formal Reasoning About GNNs Using $\mathcal{L}_{\textsf{quantGNN}}$

Let us now formally define *linear-constrained validity problem (LVP)* for GNNs.

Definition 7. *The* linear-constrained validity problem (LVP) *is defined as follows:*

- *given a quantized GNN N with input dimensionality m and output dimensionality n, two systems of linear inequalities L_{in} over variables $x_1, \ldots, x_m$ and L_{out} over variables $y_1, \ldots, y_n$, and a bound $\delta \in \mathbb{N} \cup \{+\infty\}$,*
- *decide whether $(N, L_{in}, L_{out}, \delta)$ is* valid*, that is all pointed graphs (G, v) where the out-degree of each vertex is less or equal than δ, v has m features, and v satisfies system L_{in} imply that $N(G, v)$ satisfies system L_{out}.*

We write $(N, L_{\text{in}}, L_{\text{out}})$ instead of $(N, L_{\text{in}}, L_{\text{out}}, +\infty)$.

Example 1 (Continued)**.** *Recall the setting of Example 1 where we considered the safety property, "Every account that spams 100 messages or more per minute is identified as a bot by N."*

This property is expressed by the LVP instance (N, L_{in}, L_{out}) with $L_{in} := x_1 \geq 100$ and $L_{out} := y_1 \geq 0.6$ where x_1 is the feature corresponding to the number of messages sent by a user, and y_1 is the output of N corresponding to its classification as a bot ($y_1 \geq 0.6$) or a regular user ($y_1 < 0.6$). Note that in this simple example L_{in} and L_{out} are systems consisting of a single inequality.

We show that LVP over quantized GNNs, where aggregation is given by summation, and combination functions and output functions are realized by classical feedforward neural networks (FNNs) using some kind of activation α, is reducible to the satisfiability problem of $\mathcal{L}_{\textsf{quantGNN}}$.

We denote the size of the GNN N by $|N|$, with $|N| \in \mathcal{O}(|N_{comb_1}| + \cdots + |N_{comb_m}| + |N_{out}|)$ where m is the depth of N and $|N_{comb_i}|$ and $|N_{out}|$ are the sizes of the FNN used in N. The size of an FNN is simply the sum of the sizes of all its numerical parameters, namely weights and biases.

Theorem 8. *Let $I = (N, L_{in}, L_{out}, \delta)$ be an LVP instance. There is a $\mathcal{L}_{\textsf{quantGNN}}$ formula φ_I such that I is valid if and only if φ_I is not δ-satisfiable. Furthermore, φ_I can be computed from N, L_{in}, L_{out} in polynomial time with respect to $|N| + |L_{in}| + |L_{out}|$.*

Proof. Let $I = (N, L_{\text{in}}, L_{\text{out}})$ be an LVP instance where the GNN N has l layers $\mathcal{L}_i$, each having a combination function $comb_i$ represented by an FNN N_{comb_i} as well as aggregation function agg_i represented by the sum $\sum$ and output function represented by FNN N_{out}, L_{in} consists of linear inequalities $\varphi_1, \ldots, \varphi_{k_1}$ working over variables $x_1, \ldots, x_{n_1}$ and L_{out} consists of linear inequalities $\psi_1, \ldots, \psi_{k_2}$ working over variables $y_1, \ldots, y_{n_2}$. Formula φ_I is the conjunction $\varphi_1 \wedge \cdots \wedge \varphi_{k_1} \wedge \varphi_N \wedge (\neg\psi_1 \vee \cdots \vee \neg\psi_{k_2})$ where φ_N is constructed as follows. Formula φ_N is a conjunction $\xi_{N,y_1} - y_1 = 0 \wedge \cdots \wedge \xi_{N,y_n} - y_{n_2} = 0$ where ξ_{N,y_i} captures the computation of N corresponding to its i-th output as follows.

Consider layer $\mathcal{L}_1$ and assume that N_{comb_1} has input dimension $2i_1$ and output dimension o_1. We define the sub-expression $\xi_{\mathcal{L}_1,j} = \xi_{N_{comb_1},j}(z_1, \ldots, z_{i_1}, agg(z_1), \ldots, agg(z_{i_1}))$ for $j \leq o_1$ where $\xi_{N_{comb_1},j}$ is the straightforward unfolding of the function of N_{comb_1} corresponding to its j-th output using $c, c\xi, +$ and $\alpha(\xi)$ operators on the inputs $z_1, \ldots, agg(z_{i_1})$. Next, consider the layers $\mathcal{L}_h$ for $2 \leq h \leq l$. We assume that N_{comb_h} has input dimension $2o_{h-1}$ and output dimension o_h. We define the sub-expression $\xi_{\mathcal{L}_h,j}$ equal to $\xi_{N_{comb_h},j}(\xi_{\mathcal{L}_{h-i},1},\ldots,\xi_{\mathcal{L}_{h-1},o_{h-1}}, agg(\xi_{\mathcal{L}_{h-1},1}), \ldots, agg(\xi_{\mathcal{L}_{h-1},o_{h-1}}))$ for each $j \leq o_h$. Finally, we define $\xi_{N,x_j} = \xi_{N_{out},j}(\xi_{\mathcal{L}_m,1},\ldots,\xi_{\mathcal{L}_m,o_m})$ where $\xi_{N_{out},j}$ is again the unfolding of the function of N_{out} corresponding to its j-th output.

The correctness follows from the construction above, as we simply unfold the computation of the GNN N using operators available in $\mathcal{L}_{\text{quantGNN}}$. The polynomial bound is due to the fact that we represent formulas as DAGs and, thus, avoiding any duplication of subformulas. □

Example 9. *Consider the simple GNN N of Example 5. The formula equivalent to N in the sense of Theorem 8 is $\varphi_N := \alpha(\xi_2 + \xi_2') - y = 0$ with*

$$\xi_{i+1} = \alpha(\xi_i + 2\xi_i' - 3agg(\xi_i) + 4agg(\xi_i') + 5),$$
$$\xi_{i+1}' = \alpha(6\xi_i + 7\xi_i' + 8agg(\xi_i) - 9agg(\xi_i') + 10),$$

$\xi_0 = x$ and $\xi_0' = x'$. Assuming that $L_{in} := x + x' \geq 0$ and $L_{out} := y \geq 0.6$, the overall $\mathcal{L}_{\text{quantGNN}}$ formula would be $\varphi_I := x + x' \geq 0 \wedge \varphi_N \wedge \neg(y \geq 0.6)$.

Remark 10. *The translation in Theorem 8 can be straightforwardly extended beyond the capabilities of LVP. For instance, we can generalize to instances of the form (N, φ, ψ), where N is a GNN and φ, ψ are $\mathcal{L}_{\text{quantGNN}}$ formulas.*

Note that if an LVP instance I is not valid, then φ_I is satisfiable. A model of φ_I offers a counterexample for I.

5 Satisfiability of $\mathcal{L}_{\text{quantGNN}}$ Formulas

In this section, we tackle the following satisfiability problem, in which δ is a bound on the out-degrees of the pointed graph and n is the number of bits to represent an element of $\mathbb{K}$. For instance, for $n = 32$, $\mathbb{K}$ could be Example 3. We write $\mathbb{K}_n$ to emphasize that the numbers are encoded on n bits. In the sequel, we fix a sequence $\mathbb{K}_0, \mathbb{K}_1, \mathbb{K}_2 \ldots$ with the following reasonable poly-space assumption for each $\mathbb{K}_n$:

1. there is a uniform algorithm, taking n, k_1, k_2 and k in $\mathbb{K}_n$ as an input and checking whether $k_1{+}k_2{=}k$ in poly-space in n (same for whether $k_1{=}k_2$, $k{\times}k_1{=}k_2$).
2. given n written in unary, $k, k' \in \mathbb{K}_n$ written in binary, deciding whether $[\![\alpha]\!](k') = k$ can be performed in polynomial space in n. This assumption is weak and is true for a large class of activation functions $[\![\alpha]\!]$ (for instance $ReLU$, $truncatedReLU$ for fixed-point or floating-point arithmetics on n bits).

Definition 11 (satisfiability problem)**.** *The satisfiability problem of $\mathcal{L}_{\text{quantGNN}}$ is:*

- *input: an integer n written in unary, an integer δ (or $\delta = +\infty$) in unary or in binary, a formula Φ in $\mathcal{L}_{\text{quantGNN}}$*
- *question: does there exist a pointed graph with out-degree at most δ with features in a set $\mathbb{K}_n$ that satisfies φ?*

Note that n is written unary is reasonable since n is precisely of the same order of magnitude as the size of the representation of an element in $\mathbb{K}_n$, and we have to store the bits of numbers in $\mathbb{K}_n$ fully in memory anyway.

Bounding the out-degree δ is justified in applications such as chemistry. Generally, it is sufficient to represent δ in unary since the bound is often small. For instance, a carbon atom has 4 neighbors. Having δ in unary means also that we can keep all the successors/neighbors of a given vertex in memory at once. A graph can be exponential in δ; a tree may contain δ^{depth} nodes, making the search space exponential in the input size when the depth is unbounded. Using a binary representation for δ is also interesting since the search space is already exponential with a fixed depth. For instance, we could search for a pointed graph which is a tree of depth 1 where the root has an exponential number of children.

5.1 Tableau Method

We propose a tableau method ([Hähnle, 2001; Gasquet *et al.*, 2014]) to reason in our logic, i.e. to check whether a formula Φ is δ-satisfiable. For presentation purposes, we focus on the case where the degree bound δ is finite and given in unary. The cases where δ is written in binary or $\delta = +\infty$ are more involved, and are discussed in [Sälzer *et al.*, 2025].

The idea of tableau methods is to construct a model satisfying an initial formula by propagating constraints and creating new vertices when needed. We consider terms of the form $(w\ \varphi)$ where w is a word (that intuitively denotes a vertex in the graph in construction) and φ is a formula. The intuitive meaning of $(w\ \varphi)$ is that φ should be true in the vertex w.

We start the tableau method with the following term $(\epsilon\ \Phi)$, saying that Φ should be true in the initial vertex, denoted by the empty word ϵ. Figure 4 shows our tableau rules.

The rules for $(\neg \geq), (\geq), (+), (\alpha), (\times)$ exploit the fact that the set $\mathbb{K}_n$ is finite. For instance, rule $(\geq)$ non-deterministically chooses a value for ξ that is greater than k in $\mathbb{K}_n$. The rule $(+)$ on the term $(w\ \xi_1 + \xi_2 = k)$ (whose meaning is that the value of $\xi_1 + \xi_2$ should be equal to k) non-deterministically chooses values k_1 and k_2 for ξ_1 and ξ_2 respectively so that $k_1 + k_2 = k$. The rule $(clash_c)$ stops the current execution when the constant c is different from the value k to which it is constrained to be equal. Similarly, the rule $(clash_=)$ clashes when the feature x_i at the vertex denoted by w is supposed to be equal to k and k' with $k \neq k'$. The rules (α) and $(\times)$ work similarly to the rule $(+)$ but for the activation function and for multiplication by a scalar. The rule $(degree)$ non-deterministically guesses the out-degree of the vertex specified by w.

The rule (agg) is 'modal' and creates new successors $w1, \ldots, w\delta'$ where $w1$ is the word w on which we concatenated the letter 1, and so on. When the aggregation for the expression ξ is supposed to be equal to k and the out-degree is supposed to be δ', we create δ' successors and impose that the sum of the values of ξ taken in these successors is equal to k (to this aim, we non-deterministically choose values $k_1, \ldots, k_{\delta'}$ for ξ in these successors).

The proposed tableau method is a non-deterministic procedure that applies as long as possible the rules given in Figure 4 from the initial set $\{(\epsilon\ \Phi)\}$.The method either fails or

$$(\vee)\ \frac{(w\ \varphi \vee \psi)}{(w\ \varphi) \mid (w\ \psi)} \qquad (\neg\wedge)\ \frac{(w\ \neg(\varphi \wedge \psi))}{(w\ \neg\varphi) \mid (w\ \neg\psi)}$$

$$(\neg\vee)\ \frac{(w\ \neg(\varphi \vee \psi))}{(w\ \neg\varphi), (w\ \neg\psi)} \qquad (\neg\neg)\ \frac{(w\ \neg\neg\varphi)}{(w\ \varphi)}$$

$$(\wedge)\ \frac{(w\ \varphi \wedge \psi)}{(w\ \varphi), (w\ \psi)}$$

$$(\neg \geq)\ \frac{(w\ \neg(\xi \geq k))}{(w\ \xi = k') \text{ for some } k' \in \mathbb{K}_n \text{ with } k' <_{\mathbb{K}_n} k}$$

$$(+)\ \frac{(w\ \xi_1 + \xi_2 = k)}{(w\ \xi_1 = k_1), (w\ \xi_2 = k_2) \text{ for some } k_1, k_2 \in \mathbb{K}_n, \text{ with } k_1 +_{\mathbb{K}_n} k_2 = k}$$

$$(degree)\ \frac{(w\ \varphi) \text{ and no term } (w\ degree = ...)}{(w\ degree = \delta') \text{ for some } \delta' \leq \delta}$$

$$(clash_c)\ \frac{(w\ c = k) \text{ if } c \neq k}{\text{fail}}$$

$$(clash_=)\ \frac{(w\ x_i = k), (w\ x_i = k') \text{ if } k \neq k'}{\text{fail}}$$

$$(\alpha)\ \frac{(w\ \alpha(\xi) = k)}{(w\ \xi = k') \text{ for some } k' \in \mathbb{K}_n \text{ with } [\![\alpha]\!](k') = k}$$

$$(\geq)\ \frac{(w\ \xi \geq k)}{(w\ \xi = k') \text{ for some } k' \in \mathbb{K}_n \text{ with } k' \geq_{\mathbb{K}_n} k}$$

$$(\times)\ \frac{(w\ c\xi = k)}{(w\ \xi = k') \text{ for some } k' \in \mathbb{K}_n \text{ with } c \times_{\mathbb{K}_n} k' = k}$$

$$(agg)\ \frac{(w\ agg(\xi) = k) \qquad (w\ degree = \delta')}{(w1\ \xi = k_1), \dots, (w\delta'\ \xi = k_{\delta'}) \text{ for some } (k_u)_{u=1..\delta'}, \text{ with } k_1 +_{\mathbb{K}_n} \cdots +_{\mathbb{K}_n} k_{\delta'} = k}$$

Figure 4: Tableau rules for the satisfiability problem in $\mathcal{L}_{\mathsf{quantGNN}}$ when δ is written in unary.

stops successfully if no more rules are applicable.

A prototype implementation[1] of the proof system is presented in [Sälzer *et al.*, 2025].

5.2 Upper Bound

Theorem 12. *The satisfiability problem of $\mathcal{L}_{quantGNN}$ is in PSPACE, whether δ is given in unary, binary or is infinite.*

Proof. The proof is given for δ in unary. For the proof when δ in binary or infinite, we can adapt the proof system, see [Sälzer *et al.*, 2025]. We have to prove that Φ is satisfiable iff there is an accepting execution of the tableau method.

The $\boxed{\Rightarrow}$ direction is proven by taking a pointed graph G, v satisfying Φ and making the non-deterministic choices by letting oneself be guided by the evaluation of formulas in G. To do that, we start the execution of the tableau method with $(\epsilon\ \varphi)$ and we set $f(\epsilon) = v$. We then show that we can always continue the execution maintaining the invariant: the partial mapping f from the labels in the tableau to V is such that if $(w\ \varphi)$ appears in the tableau then $G, f(w) \models \varphi$; and if $(w\ degree = \delta')$ appears then $f(w)$ is of out-degree δ'.

More precisely, we make the non-deterministic choices so that we can build a partial mapping f from the labels in the tableau to V, starting from $f(\epsilon) = v$ so that if $(w\ \varphi)$ appears in the tableau then $G, f(w) \models \varphi$; and if $(w\ degree = \delta')$ appears then $f(w)$ is of out-degree δ'.

Let us explain the principle with the (agg) rule. Suppose $(w\ agg(\xi) = k)$ and $(w\ degree = \delta')$ appears in the tableau. By the invariant, it means that $G, f(w) \models agg(\xi) = k$ and that $f(w)$ is of out-degree δ'. But then $f(w)$ has δ' successors: $u_1, \dots, u_{\delta'}$. As $G, f(w) \models agg(\xi) = k$, there exists $k_1, \dots, k_{\delta'}$ with $k_1 +_{\mathbb{K}_n} \cdots +_{\mathbb{K}_n} k_{\delta'} = k$ with $G, u_i \models \xi = k_i$ for all $i = 1..\delta'$. We then set $f(w1) := u_1, \dots, f(w\delta') = u_{\delta'}$. We also apply the rule (agg) by choosing $(w1\ \xi = k_1), \dots, (w\delta'\ \xi = k_{\delta'})$. The invariant remains true.

By the invariant, the premises of rules $(clash_c)$ or $(clash_=)$ are never in the tableau. For instance, if some $(w\ \ c = k)$ with $c \neq k$ appears, we would $G, w \models c = k$ which is a contradiction.

The $\boxed{\Leftarrow}$ direction is proven by constructing a pointed graph G, v from the choices made in the accepting execution. The graph $G = (V, E)$ is defined with V the set of labels w and E is the set of edges of the form (w, wu) with w a label and $u \in \{1, \dots, \delta'\}$ with $(w\ \ degree = \delta')$ appearing in the tableau. The label is $\ell(w) = (v_1, \dots, v_m)$ with v_i such that $(w\ x_i = v_i)$ appears in the tableau, or v_i can be any number in $\mathbb{K}_n$ if no term $(w\ x_i = ...)$ appears.

We then prove that $G, \epsilon \models \varphi$. To do that, we prove that if $(w\ \ \psi)$ appears in the final tableau, then $G, w \models \psi$ by induction on ψ. We show it for $\psi =: agg(\xi) = k$. Suppose that $(w\ \ agg(\xi) = k)$ appears in the final tableau. Then it means that the rule (agg) has been applied on it. So $(w1\ \xi = k_1), \dots, (w\delta'\ \xi = k_{\delta'})$ for some $(k_u)_{u=1..\delta'}$, with $k_1 +_{\mathbb{K}_n} \cdots +_{\mathbb{K}_n} k_{\delta'} = k$ appear in the final tableau. By induction, it means that $G, wi \models \xi = k_i$. So $G, w \models agg(\xi) = k$. Finally, as $(\epsilon\ \ \varphi)$ is in the tableau, the induction hypothesis gives $G, \epsilon \models \varphi$.

It can be implemented in poly-space, in the same spirit as the tableau method for modal logic K ([Halpern and Moses, 1992]). The idea is to backtrack on rules (agg): first we try to add all the $(w1\ \xi = k_1)$ for all applications of (agg), then try to add the $(w2\ \xi = k_2)$ and so on. The algorithm is then a depth-first-search on the tree relying on the labels w (see Figure 5). We conclude by Savitch's theorem ([Savitch, 1970]) to obtain a deterministic algorithm. □

Now, Theorem 8 and 12 imply an upper bound for LVP. Actually, we consider a quantized variant of LVP. Extending Definition 7, that problem also takes n in unary and δ in unary or in binary (as in Definition 11).

Corollary 13. *LVP is in PSPACE.*

Note that when δ is in unary, the satisfiability problem is in NP when the number of nested agg operators in the input

[1]https://github.com/francoisschwarzentruber/ijcai2025-verifquantgnn

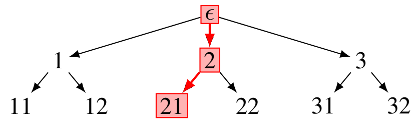

Figure 5: Depth-first search tree where each node contains terms of the form $(w \quad \ldots)$ for a given word w among $\epsilon, 1, 2, 3, 11, 12, 21, 22, 31, 32$. Only a single branch is in memory.

formula is bounded by a fixed integer i. Indeed, the certificate is a pointed labelled tree of depth i of out-degree at most δ (it contains $O(\delta^i)$ vertices). In consequence, LVP when we bound the number of layers by i is in coNP.

5.3 Usage Example

We assume that $\mathbb{K}$ is signed 32-bit fixed-point numbers, with four decimal places precision, similar to Example 2. The number $\frac{1}{125}$ is stored as (the binary encoding of the decimal number) 80, $\frac{1}{1000}$ is stored as 10, 0.9 is stored as 9000, 1000 is stored as 10000000, 250 is stored as 2500000, etc.

Elaborating on Example 1, consider the GNN $N = (\mathcal{L}_1, \mathcal{L}_{out})$ where $\mathcal{L}_1 = (agg_1, comb_1)$, $agg_1 = \sum$, $comb_1((x_1),(a_1)) = \begin{pmatrix} ReLU(+\frac{1}{125}x_1 + 0a_1 + 0) \\ Id(+0x_1 + \frac{1}{1000}a_1 + 0) \end{pmatrix}$. So N has only one layer, with input dimension 1 (the value of x_1 that labels a node indicates the number of messages per minute sent by the individual), and output dimension 2. The output layer is the identity. Consider again the LVP $I = (N, L_{\text{in}}, L_{\text{out}})$ where $L_{\text{in}} = x_1 \geq 100$ and $L_{\text{out}} = y_1 \geq 0.6$.

$\mathcal{L}_{\text{quantGNN}}$ formula. As per Theorem 8 and its proof, the LVP $I = (N, L_{\text{in}}, L_{\text{out}})$ is valid iff $\varphi_I = \varphi_{\text{in}} \wedge \varphi_N \wedge \neg\psi_{\text{out}}$ is not satisfiable, where:

- $\varphi_{\text{in}} := x_1 \geq 100$
- $\varphi_N := ReLU(\frac{1}{125}x_1) - y_1 = 0 \wedge \frac{1}{1000}agg(x_1) - y_2 = 0$
- $\psi_{\text{out}} := y_1 \geq 0.6$

Automated reasoning. We could show that $(agg(x_1) \geq 1000) \wedge \varphi_N \wedge \neg(y_2 \geq 1.0)$ is not satisfiable, establishing that N comes with the guarantee of property B. Here we focus on property A. If the pointed graph (G, v) satisfies L_{in}, then $N(G, v)$ will satisfy L_{out}. We could use the tableau method to automatically show that φ_I is not satisfiable and therefore that I is valid. Unfortunately, a complete formal proof cannot be reasonably presented here or even done by hand. Hence, to illustrate how the reasoning method can be used for the verification of quantized GNNs, we slightly modify the LVP I with a different output constraint.

Let I' be the LVP $(N, L_{\text{in}}, L'_{\text{out}}, \delta)$, where N and L_{in} are as before, $L'_{\text{out}} := y_1 \geq 0.9$, and $\delta = 5$. We are going to show that the new output constraint is not always satisfied. I' is valid iff $\varphi_{I'} := \varphi_{\text{in}} \wedge \varphi_N \wedge \neg\psi'_{\text{out}}$ is not satisfiable, where:

- $\psi'_{\text{out}} := y_1 \geq 0.9$

We employ our tableau method to prove that $\varphi_{I'}$ is satisfiable and therefore that I' is not valid. We use ✓ to indicate that we have reached a term for which there are no rules to apply, except possibly $(clash_=)$. If in the end $(clash_=)$ does not apply, then we can read a model of $\varphi_{I'}$ where the ✓s appear. We start the tableau method by applying the rule $(\wedge)$ and resolve inequalities with rules $(\geq)$ and $(\neg \geq)$ (we write (Δ) when we just use the definition of a formula):

$$(\wedge)\ \frac{(\epsilon\ \varphi_{\text{in}} \wedge \varphi_N \wedge \neg\psi'_{\text{out}})}{(\epsilon\ \varphi_{\text{in}}), (\epsilon\ \varphi_N), (\epsilon\ \neg\psi'_{\text{out}})}$$

$$(\Delta)\ \cfrac{(\epsilon\ \varphi_{\text{in}})}{(\geq)\ \cfrac{(\epsilon\ x_1 \geq 100)}{(\epsilon\ x_1 = 100)\ ✓}} \quad (\Delta)\ \cfrac{(\epsilon\ \varphi_N)}{(\wedge)\ \cfrac{(\epsilon\ C_1 \wedge C_2)}{(\epsilon\ C_1), (\epsilon\ C_2)}} \quad (\Delta)\ \cfrac{(\epsilon\ \neg\psi'_{\text{out}})}{(\neg\geq)\ \cfrac{(\epsilon\ \neg y_1 \geq 0.9)}{(\epsilon\ y_1 = 0.8)\ ✓}}$$

where we write C_1 for the conjunct $ReLU(\frac{1}{125}x_1) - y_1 = 0$ and C_2 for the conjunct $\frac{1}{1000}agg(x_1) - y_2 = 0$.

We handle the conjunct C_1 in φ_N.

$$(+)\ \frac{(\epsilon\ ReLU(\frac{1}{125}x_1) + (-1)y_1 = 0)}{(\epsilon\ ReLU(\frac{1}{125}x_1) = 0.8), (\epsilon\ (-1)y_1 = -0.8)}$$

$$(\alpha)\ \cfrac{(\epsilon\ ReLU(\frac{1}{125}x_1) = 0.8)}{(\times)\ \cfrac{(\epsilon\ \frac{1}{125}x_1 = 0.8)}{(\epsilon\ x_1 = 100)\ ✓}} \quad (\times)\ \frac{(\epsilon\ (-1)y_1 = -0.8)}{(\epsilon\ y_1 = 0.8)\ ✓}$$

Finally, we handle the conjunct C_2 in φ_N.

$$(+)\ \frac{(\epsilon\ \frac{1}{1000}agg(x_1) + (-1)y_2 = 0)}{(\epsilon\ \frac{1}{1000}agg(x_1) = 1), (\epsilon\ (-1)y_2 = -1)}$$

$$(\times)\ \cfrac{(\epsilon\ \frac{1}{1000}agg(x_1) = 1)}{(degree)\ \cfrac{(\epsilon\ agg(x_1) = 1000)}{(\epsilon\ degree = 4)\ ✓}} \quad (\times)\ \frac{(\epsilon\ (-1)y_2 = -1)}{(\epsilon\ y_2 = 1)\ ✓}$$

$$(agg)\ \frac{(\epsilon\ agg(x_1) = 1000) \qquad (\epsilon\ degree = 4)}{(1\ x_1 = 250), \ldots, (4\ x_1 = 250)\ ✓}$$

Obtaining a model. We obtain a counterexample (G, v) for the formula $\varphi_{I'}$, with the node v labelled $(x_1) = (100)$, which has 4 successors, all labelled $(x_1) = (250)$. So, by Theorem 8, I' is not valid. Indeed, G, v satisfies L_{in}. But $N(G, v)$ is $(y_1, y_2) = (0.8, 1)$ which does not satisfy L'_{out}.

This illustrates how the proof system serves to verify the properties of quantized GNN, but also to exhibit counterexamples when they exist.

6 Lower Bounds

If not said otherwise, we suppose now the activation function is truncated ReLU; it corresponds to the settings in many works ([Barceló *et al.*, 2020; Nunn *et al.*, 2024; Benedikt *et al.*, 2024]). Furthermore, to obtain stronger PSPACE lower bound results, we will consider two restrictions:

- Case$_{\{0,1,2\},\delta=2}$: $\mathbb{K}$ contains 0, 1, 2 and the out-degree is bounded by δ=2;
- Case$_{\{-2..3\},satu}$: $\mathbb{K}$ includes $\{-2, -1, 0, 1, 2, 3\}$ and moreover, $\mathbb{K}$ saturates in the following sense: for all integers x, $x \geq 3$ implies $x_{\mathbb{K}} \geq_{\mathbb{K}} 3_{\mathbb{K}}$. In particular, it means that we have no modulo behavior in $\mathbb{K}$ (e.g., $\mathbb{K} = \mathbb{Z}_{10}$ would not saturate since despite $10 \geq 9$ as integers, we would not have $0 = 10 \geq_{\mathbb{K}} 9$ modulo 10).

In both cases, $n = 3$ bits are enough to represent the numbers. Our lower bound results rely on *graded modal logic* which extends modal logic with constructions $\exists^{\geq k}\varphi$ whose truth conditions is: $G, v \models \exists^{\geq k}\varphi$ iff there are at least k successors u of v such that $G, u \models \varphi$. We call GML_K the syntactic fragment where occurrences of $\exists^{\geq k}$ are such that $k \leq K$.

Lemma 14. *The satisfiability of GML_1 on graphs wit out-degree at most 2, and the satisfiability of GML_3 is PSPACE-hard.*

Theorem 15. *LVP is PSPACE-complete. PSPACE-hardness holds already for $Case_{\{0,1,2\},\delta=2}$ or $Case_{\{-2..3\},satu}$.*

Proof. PSPACE-membership is stated in Corollary 13. The lower bound relies on the poly-time transformation τ of a GML-formula φ into an equivalent GNN $N = \tau(\varphi)$ in the sense that $N(G, v)$ satisfies $x_1 \geq 1$, given in Barceló *et al.* [2020, proof of Proposition 4.1]. More specifically, we carefully analyze the GNN N when the formula φ is in GML_K.

In N, the activation function is truncated ReLU. The layers in $\tau(\varphi)$ are all the same. The matrices C contain coefficients -1, 0, 1 and all coefficients on a column are 0 except two at most. The matrices A only contain 0 and 1. On each column of A, all coefficients are 0 except one which may be 1. Coefficients in the biases b are in $\{-K+1, -K+2, \ldots, 1\}$.

Propositional variables in φ are among $p_1, \ldots, p_J$, $N(G, v)$ starts with the state $(\boldsymbol{x}_u^0)_j = 1$, if $j \in \{1, \ldots, J\}$ and p_j holds in u, and $(\boldsymbol{x}_u^0)_j = 0$ otherwise. As the activation function is truncated ReLU, all vectors $\boldsymbol{x}_u^i$ are 0–1-vectors.

Given the coefficients in the matrices, intermediate values are all integers.We end the proof of our Theorem 15 by stating the reductions and arguing that the computation is *safe* when done in $\mathbb{K}$, meaning that there is no overflow and that the computation in $\mathbb{K}$ and in $\mathbb{Z}$ yields the same result.

Proof for Case$_{\{0,1,2\},\delta=2}$ We reduce the satisfiability of GML_1 on graphs of out-degree at most 2 which is PSPACE-hard (Lemma 14) into the dual problem $\overline{\mathrm{LVP}}$ of LVP with δ=2: from φ a GML_1-formula, we compute the $\overline{\mathrm{LVP}}$-instance $(\tau(\varphi), \top, y_1 \leq 0, \delta{=}2)$, and φ is satisfiable on graphs of out-degree 2 iff $(\tau(\varphi), \top, y_1 \leq 0, \delta{=}2)$ is a negative instance of LVP. $\overline{\mathrm{LVP}}$ is PSPACE-hard, so is LVP. The computation is safe because there are at most two successors, so we never exceed 2 in a computation in $\mathbb{K}$.

Proof for Case$_{\{-2..3\},satu}$ We reduce the satisfiability of GML_3 which is PSPACE-hard (Lemma 14) into $\overline{\mathrm{LVP}}$: given φ a GML_3-formula, we compute the $\overline{\mathrm{LVP}}$-instance $(\tau(\varphi), \top, y_1 \leq 0, \delta{=}{+}\infty)$. We have that φ is GML_3-satisfiable iff there exists G, v s.t. $G, v \models \varphi$ iff $(\tau(\varphi), \top, y_1 \leq 0, \delta{=}{+}\infty)$ is a negative instance of LVP. The computation is safe because, even if there are an arbitrary large number x of successors, if $x \geq 3$, we also have $x_{\mathbb{K}} \geq_{\mathbb{K}} 3_{\mathbb{K}}$ because $\mathbb{K}$ saturates. ☐

Corollary 16. *The satisfiability problem of $\mathcal{L}_{\mathsf{quantGNN}}$ is PSPACE-complete, and already PSPACE-hard for $Case_{\{0,1,2\},\delta=2}$ or $Case_{\{-2..3\},satu}$, and also when the activation function is ReLU.*

7 Related Work

Starting with Graded Modal Logic ([Fine, 1972]), there are numerous logics that capture modal aspects of graphs and express arithmetic constraints, ([Demri and Lugiez, 2010; Baader, 2017; Bednarczyk *et al.*, 2021; Galliani *et al.*, 2023; van Benthem and Icard, 2023]).

Previous research has already established several correspondences between logic and GNNs. For instance, Barceló *et al.* [2020] explore the relationship between graded modal logic and GNNs, while Nunn *et al.* [2024] examine a modal logic over linear inequalities with counting and its connection to GNNs. Additionally, Benedikt *et al.* [2024] investigates fragments of Presburger logic in the context of GNNs. However, these existing works focus on GNNs with specific activation functions and do not consider the broader class of quantized GNNs. In particular, decidability in PSPACE has been established only for cases where the activation function is either a truncated ReLU ([Nunn *et al.*, 2024]) or eventually constant functions ([Benedikt *et al.*, 2024]). Ahvonen *et al.* [2024] offer several logical characterizations of recurrent GNNs over floats and real numbers. Also related, is the work of Grohe [2023], which establishes that the graph queries computable by a polynomial-size, bounded-depth family of GNNs are precisely those definable in the guarded fragment of first-order logic with counting and built-in relations.

Henzinger *et al.* [2021] also address the verification of quantized neural networks, but in contrast to this paper they focus on FNN. The authors establish a PSPACE-hardness result, relying on a binary representation of the number of bits. In contrast, if the number of bits were unary, their problem is presumably in NP (using a guess and check argument, relying on similar arguments as used by [Sälzer and Lange, 2021]).

There are other aggregation functions, like weighted sums in graph attention networks [Velickovic *et al.*, 2018], maximum in Max-GNNs [Cucala and Grau, 2024], or average. $\mathcal{L}_{\mathsf{quantGNN}}$ and the tableau method can be adapted to capture these aggregation functions, too (see [Sälzer *et al.*, 2025]).

On the practical side, Huang *et al.* [2024] presents a solution to the verification of quantized FNNs using heuristic search, and outperforming the approach of Zhang *et al.* [2023] based purely on integer linear programming.

8 Conclusion and Perspectives

We introduced a method for explicitly verifying and reasoning about properties of a practical class of quantized Graph Neural Networks. It allows us to establish new foundational results about the computational complexity of GNNs' tasks.

We proposed a tableau method for reasoning about quantized GNNs using the new logic $\mathcal{L}_{\mathsf{quantGNN}}$. As the domain for numbers is finite, we are able to work out our details using a parameterized class of quantized GNNs, allowing us to consider various activation functions. We showed that for such classes of GNNs using FNNs in their internal parts, we can answer basic formal reasoning questions using a reduction to the satisfiability problem of $\mathcal{L}_{\mathsf{quantGNN}}$. We showed that satisfiability of $\mathcal{L}_{\mathsf{quantGNN}}$ is solvable in PSPACE.

This work is a first step to further develop concrete methods for reasoning about quantized GNNs and other neural network models. Now, it will be interesting to implement our approach on a broader scale in order to tackle tasks such as certifying (adversarial) robustness properties ([Günnemann, 2022]) or giving formal explanations ([Marques-Silva and Ignatiev, 2022]) for certain GNN behaviours.

Ethical Statement

There are no ethical issues.

Acknowledgments

This work was supported by the ANR EpiRL project ANR22-CE23-0029, and by the MUR (Italy) Department of Excellence 2023–2027.

References

[Ahvonen *et al.*, 2024] Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. Logical characterizations of recurrent graph neural networks with reals and floats. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[Baader, 2017] Franz Baader. A new description logic with set constraints and cardinality constraints on role successors. In Clare Dixon and Marcelo Finger, editors, *Frontiers of Combining Systems - 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings*, volume 10483 of *Lecture Notes in Computer Science*, pages 43–59. Springer, 2017.

[Barceló *et al.*, 2020] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. 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, 2020.

[Bednarczyk *et al.*, 2021] Bartosz Bednarczyk, Maja Orlowska, Anna Pacanowska, and Tony Tan. On classical decidable logics extended with percentage quantifiers and arithmetics. In Mikolaj Bojanczyk and Chandra Chekuri, editors, *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*, pages 36:1–36:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Benedikt *et al.*, 2024] Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan. Decidability of graph neural networks via logical characterizations. In Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 127:1–127:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[Cucala and Grau, 2024] David J. Tena Cucala and Bernardo Cuenca Grau. Bridging max graph neural networks and Datalog with negation. In Pierre Marquis, Magdalena Ortiz, and Maurice Pagnucco, editors, *Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning, KR 2024, Hanoi, Vietnam. November 2-8, 2024*, 2024.

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

[Ercegovac and Lang, 2004] Miloš D. Ercegovac and Tomás Lang. *Digital Arithmetic*. Morgan Kaufmann, San Francisco, CA, 2004.

[Fey and Lenssen, 2019] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[Fine, 1972] Kit Fine. In so many possible worlds. *Notre Dame J. Formal Log.*, 13(4):516–520, 1972.

[Galliani *et al.*, 2023] Pietro Galliani, Oliver Kutz, and Nicolas Troquard. Succinctness and complexity of ALC with counting perceptrons. In Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 291–300, 2023.

[Gasquet *et al.*, 2014] Olivier Gasquet, Andreas Herzig, Bilal Said, and François Schwarzentruber. *Kripke's Worlds - An Introduction to Modal Logics via Tableaux*. Studies in Universal Logic. Birkhäuser, 2014.

[Gholami *et al.*, 2021] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *CoRR*, abs/2103.13630, 2021.

[Goldberg, 1991] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.*, 23(1):5–48, March 1991.

[Grohe, 2023] Martin Grohe. The descriptive complexity of graph neural networks. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2023, Boston, MA, USA, June 26-29, 2023*, pages 1–14. IEEE, 2023.

[Günnemann, 2022] Stephan Günnemann. Graph neural networks: Adversarial robustness. In *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 149–176. Springer Nature Singapore, Singapore, 2022.

[Hähnle, 2001] Reiner Hähnle. Tableaux and related methods. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 100–178. Elsevier and MIT Press, 2001.

[Halpern and Moses, 1992] Joseph Y. Halpern and Yoram Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(2):319–379, 1992.

[Henzinger *et al.*, 2021] Thomas A. Henzinger, Mathias Lechner, and Dorde Zikelic. 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*, pages 3787–3795. AAAI Press, 2021.

[Huang *et al.*, 2024] Pei Huang, Haoze Wu, Yuting Yang, Ieva Daukantas, Min Wu, Yedi Zhang, and Clark W. Barrett. Towards efficient verification of quantized neural networks. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *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*, pages 21152–21160. AAAI Press, 2024.

[Marques-Silva and Ignatiev, 2022] João Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 12342–12350. AAAI Press, 2022.

[Nunn *et al.*, 2024] Pierre Nunn, Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. A logic for reasoning about aggregate-combine graph neural networks. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, 3-9 August 2024*. ijcai.org, 2024.

[Sälzer and Lange, 2021] Marco Sälzer and Martin Lange. Reachability is NP-complete even for the simplest neural networks. In Paul C. Bell, Patrick Totzke, and Igor Potapov, editors, *Reachability Problems - 15th International Conference, RP 2021, Liverpool, UK, October 25-27, 2021, Proceedings*, volume 13035 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2021.

[Sälzer and Lange, 2023] Marco Sälzer and Martin Lange. 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, 2023.

[Sälzer *et al.*, 2025] Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. Verifying quantized graph neural networks is pspace-complete. *CoRR*, abs/2502.16244, 2025.

[Savitch, 1970] Walter J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[van Benthem and Icard, 2023] Johan van Benthem and Thomas Icard. Interleaving logic and counting. *Bull. Symb. Log.*, 29(4):503–587, 2023.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[Ye *et al.*, 2022] Zi Ye, Yogan Jaya Kumar, Goh Ong Sing, Fengyan Song, and Junsong Wang. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741, 2022.

[Zhang *et al.*, 2023] Yedi Zhang, Zhe Zhao, Guangke Chen, Fu Song, Min Zhang, Taolue Chen, and Jun Sun. 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, 2023. Association for Computing Machinery.

[Zhou *et al.*, 2020] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[Zhu *et al.*, 2023] Zeyu Zhu, Fanrong Li, Zitao Mo, Qinghao Hu, Gang Li, Zejian Liu, Xiaoyao Liang, and Jian Cheng. $\mathrm{A}^2\mathrm{Q}$: Aggregation-aware quantization for graph neural networks. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.