A Logic for Reasoning about Aggregate-Combine Graph Neural Networks

Pierre Nunn[1], **Marco Sälzer**[2], **François Schwarzentruber**[1] and **Nicolas Troquard**[3]
[1]University of Rennes, IRISA, CNRS, France
[2]Theoretical Computer Science / Formal Methods, University of Kassel, Germany
[3]Gran Sasso Science Institute, L'Aquila, Italy
pierre.nunn@ens-rennes.fr, marco.saelzer@uni-kassel.de, francois.schwarzentruber@ens-rennes.fr, nicolas.troquard@gssi.it

Abstract

We propose a modal logic in which counting modalities appear in linear inequalities. We show that each formula can be transformed into an equivalent graph neural network (GNN). We also show that a broad class of GNNs can be transformed efficiently into a formula, thus significantly improving upon the literature about the logical expressiveness of GNNs. We also show that the satisfiability problem is PSPACE-complete. These results bring together the promise of using standard logical methods for reasoning about GNNs and their properties, particularly in applications such as GNN querying, equivalence checking, etc. We prove that such natural problems can be solved in polynomial space.

1 Introduction

Graph Neural Networks (GNNs) perform computations on graphs or on pairs of graphs and vertices, referred to as pointed graphs. As a prominent deep learning model, GNNs find applications in various domains such as social recommendations [Salamat *et al.*, 2021], drug discovery [Xiong *et al.*, 2021], material science and chemistry [Reiser *et al.*, 2022], knowledge graphs [Ye *et al.*, 2022], among others (see Zhou *et al.* [2020] for an overview). This growing adoption of GNNs comes with certain challenges. The use of GNNs in safety-critical applications sparks a significant demand for safety certifications, given by dependable verification methods. Furthermore, human understandable explanations for the behaviour of GNNs are needed in order to build trustworthy applications, coming with the need for a general understanding of the capabilities and limitations inherent in specific GNN models.

In general, there are two approaches enabling rigorous and formal reasoning about GNNs: formal verification [Huang *et al.*, 2020] and formal explanation methods [Marques-Silva and Ignatiev, 2022] for neural network models such as GNNs. Formal verification procedures are usually concerned with the sound and complete verification of properties like "Does GNN $\mathcal{A}$ produce an unwanted output y in some specified region Y?" (reachability) or "Does GNN $\mathcal{A}$ behave as expected on inputs from some specified region X?" (robustness). Formal explainability methods are concerned with giving answers for questions like "Is there a minimal, humanly interpretable reason that GNN $\mathcal{A}$ produces output y given some input x?" (abductive explanations). In both cases, formal verification and formal explanation, logical reasoning offers an all-purpose tool. For example, the following algorithm enables addressing correspondence questions: given some GNN $\mathcal{A}$, produce a logical formula φ such that $[[\mathcal{A}]] = [[\varphi]]$ where $[[\mathcal{A}]]$ is the class of (pointed) graphs recognized by the GNN $\mathcal{A}$, and $[[\varphi]]$ is the class of pointed graphs in which φ holds. Informally put, the goal is to compute a formula φ that completely characterizes the class of (pointed) graphs recognized by GNN $\mathcal{A}$. Given this, one can then investigate GNNs purely based on this logical characterization. Unfortunately, the synthesis of a formula of a logic, say first-order logic (FO) or modal logic (ML), that captures a semantic condition can be notoriously challenging (e.g., Pinchinat *et al.* [2022]).

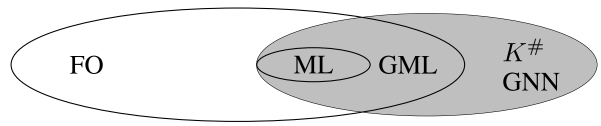

Figure 1: Expressivity of our logic $K^{\#}$ compared to modal logic (ML), graded modal logic (GML) and first-order logic (FO).

Barceló *et al.* [2020] have shown that any formula of Graded Modal Logic (GML) can be transformed into a GNN. Conversely, they have shown that given a GNN *that is expressible in first-order logic* (FO), there exists an equivalent formula in GML. Doing so, they also characterized GML as being the intersection of FO and GNNs. While their result is promising, there is no full logical characterization of what a GNN can express.

That is why we define a logic called $K^{\#}$ combining counting modalities and linear programming, which is expressive enough to capture a broad and natural class of GNNs. As pictured in Figure 1, it is more expressive than graded modal logic [de Rijke, 2000] and, thus, $K^{\#}$ captures a broader class of GNNs than previously identified in Barceló *et al.* [2020]. Furthermore, we show that the satisfiability problem of $K^{\#}$ is PSPACE-complete, leading to immediate complexity results for various formal verification and explainability problems as reachability, robustness, or producing abductive explanations.

Overview of the Main Contributions. We present the logic $K^{\#}$ which captures Aggregate-Combine Graph Neural Networks, also called Message Passing Neural Networks [Barceló *et al.*, 2020; Gilmer *et al.*, 2017], where the aggregation function is the sum, the combination and final classification functions are linear functions with integer parameters, and truncated ReLU ($max(0, min(1, x))$) is used for the activation function. We refer to these GNNs simply as GNNs in the paper. In particular, we show that:

- for each formula φ of $K^{\#}$ there is a GNN $\mathcal{A}$ recognizing exactly the same pointed graphs as φ (Theorem 1),
- for each GNN $\mathcal{A}$ there is a formula φ of $K^{\#}$ recognizing exactly the same pointed graphs as $\mathcal{A}$ (Theorem 2).

These results significantly extend the class of GNNs for which a logical characterization is known from Barceló *et al.* [2020]. Furthermore, we provide an algorithm for the satisfiability problem of $K^{\#}$, proving that the problem is PSPACE-complete (Theorem 3). This also provides algorithmic solutions for the following problems: given a GNN $\mathcal{A}$ and a logical formula φ, decide whether (P1) $[[\mathcal{A}]] = [[\varphi]]$, (P2) $[[\mathcal{A}]] \subseteq [[\varphi]]$, (P3) $[[\varphi]] \subseteq [[\mathcal{A}]]$, (P4) $[[\varphi]] \cap [[\mathcal{A}]] \neq \emptyset$.

Example 1. *Consider a setting where GNNs are used to classify users in a social network. Assume we have a GNN $\mathcal{A}$ which is* intended *to recommend exactly those users who have at least one friend who is a musician and at most one-third of their friends play the tuba. We call this the* few-tubas *property. Given our translation from $\mathcal{A}$ to a $K^{\#}$ formula, we can answer questions like*

A *"Is each recommended person a person that has the few-tubas property?"*

B *"Does any recommended person not have the few-tubas property?"*

C *"Is any person that befriends a musician but also too many tuba players recommended?"*

D *"Is it possible to recommend a person that does not have the few-tubas property?"*

by representing the corresponding properties as $K^{\#}$ formulas and then solving problem P1 *for* **A**, P2 *for* **B**, P3 *for* **C** *and* P4 *for* **D**. *Question* **A** *corresponds to giving an abductive explanation, as a positive answer would indicate that "at least one musician friend and not too many tuba playing friends" is a (minimal) reason for "recommended person". In the same manner,* **B** *and* **C** *are reachability properties and* **D** *is a robustness property.*

Outline. In Section 2 we recall the necessary preliminaries on graph neural networks. In Section 3, we define the logic $K^{\#}$. In Section 4, we study the correspondence between GNNs and $K^{\#}$. In Section 5, we discuss the satisfiability problem of $K^{\#}$. Section 6 addresses the complexity of the problems P1–P4. Section 7 is about the related work, and Section 8 concludes.

2 Background on GNNs

In this paper, we consider Aggregate-Combine GNNs (AC-GNN) [Barceló *et al.*, 2020], also sometimes called message passing neural networks [Gilmer *et al.*, 2017].

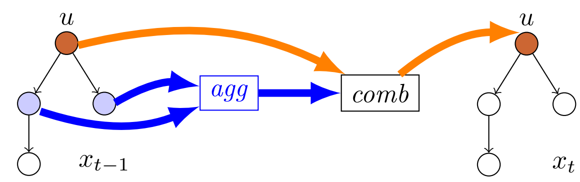

Figure 2: A layer in a GNN transforms the state x_{t-1} at step $t-1$ into the state x_t at time t. The figure shows how $x_t(u)$ is computed. First, the function agg is applied to the features in the successors of u. Then $comb$ is applied to that result and $x_{t-1}(u)$ to obtain $x_t(u)$.

An *aggregation function agg* is a function mapping finite multisets of vectors in $\mathbb{R}^d$ to vectors in $\mathbb{R}^d$. A *combination function comb* is a function mapping a vector in $\mathbb{R}^{2d}$ to vectors in $\mathbb{R}^{d'}$. An *AC-GNN layer* $\mathcal{L}$ of input dimension d is defined by an aggregation function agg and a combination function $comb$ of matching dimensions, meaning agg expects and produces vectors from $\mathbb{R}^d$ and $comb$ has input dimensionality $2d$. The output dimension of $\mathcal{L}$ is given by the output dimension of $comb$. Then, an *AC-GNN* is a tuple $(\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(L)}, cls)$ where $\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(L)}$ are L AC-GNN layers and $cls : \mathbb{R}^d \to \{0, 1\}$ 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.

Definition 1. *Here we call a* GNN *an AC-GNN $\mathcal{A}$ where all aggregation functions are given by $agg(X) = \sum_{x \in X} x$, all combination functions are given by $comb(x, y) = \vec{\sigma}(xC + yA + b)$ where $\vec{\sigma}(x)$ is the componentwise application of the* truncated ReLU $\sigma(x) = max(0, min(1, x))$, *where C and A are matrices of integer parameters and b is a vector of integer parameters, and where the classification function is $cls(x) = \sum_i a_i x_i \geq 1$, with a_i integers.*

A *(labeled directed) graph* G is a tuple (V, E, ℓ) such that V is a finite set of vertices, $E \subseteq V \times V$ a set of directed edges and ℓ is a mapping from V to a valuation over a set of atomic propositions. We write $\ell(u)(p) = 1$ when atomic proposition p is true in u, and $\ell(u)(p) = 0$ otherwise. Given a graph G and vertex $u \in V$, we call (G, u) a *pointed graph*.

Let $G = (V, E, \ell)$ be a graph. A *state* x is a mapping from V into $\mathbb{R}^d$ for some d. When applied to a graph G, the t-th GNN layer $\mathcal{L}^{(t)}$ transforms the previous state x_{t-1} into the next state x_t. Supposing that the atomic propositions occurring in G are $p_1, \ldots, p_k$, the *initial state* x_0 is defined by:

$$x_0(u) := (\ell(u)(p_1), \ldots, \ell(u)(p_k)) \in \mathbb{R}^d$$

for all $u \in V$. Then:

$$x_t(u) := comb(x_{t-1}(u), agg(\{\{x_{t-1}(v) | uv \in E\}\}))$$

where agg and $comb$ are respectively the aggregation and combination function of the t-th layer. In the above equation, note that the argument of agg is the multiset of the feature vectors of the successors of v. Thus, the same vector may occur several times in that multiset. Figure 2 explains how a layer works at each vertex.

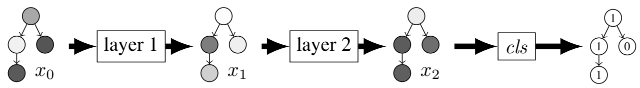

Figure 3: General idea of a GNN with 2 layers applied on a graph with 4 vertices.

Example 2. *Consider a layer defined by* $agg(X) := \sum_{x \in X} x$ *and*

$$comb((x, x'), (y, y')) := \begin{pmatrix} \sigma(x + 2x' + 3y + 4y' + 5) \\ \sigma(6x + 7x' + 8y + 9y' + 10) \end{pmatrix}.$$

Suppose as in Figure 2, that vertex u has two successors named v and w. Here we suppose that feature vectors are of dimension 2: $x_{t-1}(u), x_{t-1}(v), x_{t-1}(w) \in \mathbb{R}^2$. First, $agg(\{\{x_{t-1}(v), x_{t-1}(w)\}\}) = x_{t-1}(v) + x_{t-1}(w)$. Second, with our example of combination function we get:

$$x_t(u) := \begin{pmatrix} \sigma(x_{t-1}(u)_1 + 2x_{t-1}(u)_2 + 3y_1 + 4y_2 + 5) \\ \sigma(6x_{t-1}(u)_1 + 7x_{t-1}(u)_2 + 8y_1 + 9y_2 + 10) \end{pmatrix}$$

where $y_1 = x_{t-1}(v)_1 + x_{t-1}(w)_1$ and $y_2 = x_{t-1}(v)_2 + x_{t-1}(w)_2$.

Figure 3 explains how a GNN works overall: the state is updated at each layer; at the end the function cls says whether each vertex is recognized (output 1) or not (output 0).

Let $\mathcal{A} = (\mathcal{L}^{(1)}, \dots, \mathcal{L}^{(L)}, cls)$ be a GNN. We define the semantics $[[\mathcal{A}]]$ of $\mathcal{A}$ as the set of pointed graphs (G, u) such that $cls(x_L(u)) = 1$.

3 The Logic $K^\#$

In this section, we describe the syntax and semantics of $K^\#$ and its fragment $K^{\#,-1}$.

3.1 Syntax

Consider a countable set Ap of propositions. We define the language of logic $K^\#$ as the set of formulas generated by the following BNF:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \xi \geq 0$$
$$\xi ::= c \mid 1_\varphi \mid \#\varphi \mid \xi + \xi \mid c \times \xi$$

where p ranges over Ap, and c ranges over $\mathbb{Z}$. Atomic formulas are propositions p, inequalities and equalities of linear expressions. We consider linear expressions over 1_φ and $\#\varphi$. The number 1_φ is equal to 1 if φ holds in the current world and equal 0 otherwise. The number $\#\varphi$ is the number of successors in which φ hold. The language seems strict but we write $\xi_1 \leq \xi_2$ for $\xi_2 - \xi_1 \geq 0$, $\xi = 0$ for $(\xi \geq 0) \wedge (-\xi \geq 0)$, etc. Recall that modal logic itself extends propositional logic with a modal construction $\Box\varphi$ whose semantics is 'the formula holds in all successors'. Logic $K^\#$ is an extension of modal logic since we can define $\Box\varphi := (\#(\neg\varphi) \leq 0)$: the number of successors in which φ does not hold equals 0. We write $\top$ for a tautology.

Example 3. *The few-tuba property of Example 1, users (nodes) who have at least one friend who is a musician and at most one-third of their friends play the tuba, can be represented by the formula $(\#musician \geq 1) \wedge (\#\top \geq 3 \times \#tubaplayer)$.*

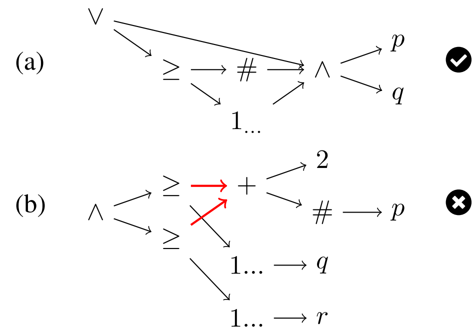

Figure 4: DAG representation of formulas. (a) We allow for reusing subformulas. (b) We disallow for reusing arithmetical expressions.

The set of subformulas, $sub(\varphi)$ is defined by induction on φ: $sub(p) = \{p\}$, $sub(\neg\varphi) = \{\neg\varphi\} \cup sub(\varphi)$, $sub(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup sub(\varphi) \cup sub(\psi)$, and $sub(\xi \geq 0) = \{\xi \geq 0\} \cup \bigcup\{sub(\psi) \mid 1_\psi \text{ or } \#\psi \text{ in } \xi\}$.

The modal depth of a formula, $md(\varphi)$ and the modal depth of an expression, $md(\xi)$ are defined by mutual induction on φ and ξ: $md(p) = md(c) = 0$, $md(\neg\varphi) = md(1_\varphi) = md(\varphi)$, $md(\xi \geq 0) = md(k \cdot \xi) = md(\xi)$, $md(\#\varphi) = md(\varphi) + 1$, and $md(\xi_1 + \xi_2) = \max(md(\xi_1), md(\xi_2))$. As in modal logic, modalities are organized in levels.

Example 4. $md(1_{p \wedge \#q \leq 4} \leq \#((\#p \geq 2) \leq 4)) = 2$. *The expressions $\#q$ and $\#(\#p \geq 2)$ are at the root level (level 1), while the expression $\#p$ is at level 2.*

In this paper, a formula is represented by a DAG (directed acyclic graph) instead of just a syntactic tree. DAGs, contrary to syntactic trees, allow the reuse of subformulas.

Example 5. *The DAG depicted in Figure 4(a), in which the subformula $p \wedge q$ is used thrice, represents the formula $(p \wedge q) \vee (\#(p \wedge q) \geq 1_{p \wedge q})$.*

However, we disallow DAGs reusing arithmetic expressions.

Example 6. *The formula* $(2 + \#p \geq 1_q) \wedge (2 + \#p \geq 1_r)$ cannot *be represented by the DAG shown in Figure 4(b) which refers to the expression $2 + \#p$ twice.*

Formally, the fact that arithmetic expressions are not reused in DAGs representing formulas is reflected by the simple property that the nodes representing arithmetical expressions have in-degree 1.

Definition 2. *A DAG of a formula is a graph in which nodes for c, 1_φ, $\#\varphi$, $\xi + \xi'$, $c \times \xi$ have in-degree 1.*

The reason for not allowing reusing arithmetical expressions in the DAG representation of formulas is technical. Firstly, as demonstrated by Theorem 2, the transformation of GNNs into formulas does not necessitate the reusing of arithmetical subexpressions but only subformulas. Second, we will need to efficiently transform formulas in DAG representation into formulas in tree representation (Lemma 1). For this result, we need DAGs wherein only subformulas are reused, excluding the reuse of arithmetical expressions. The size $|\varphi|$ of a $K^\#$ formula φ in DAG form is the number of bits needed to represent the DAG.

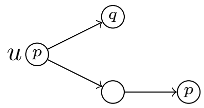

Figure 5: Example of a pointed graph G, u. We indicate true propositional variables at each vertex.

The logic $K^{\#,-1}$ is the syntactic fragment of $K^{\#}$ in which constructions 1_φ are disallowed.

3.2 Semantics

As in modal logic, a formula φ is evaluated in a pointed graph (G, u) (also known as pointed Kripke model). We define the truth conditions $(G, u) \models \varphi$ (φ is true in u) by

$$\begin{array}{lll} (G,u) \models p & \text{if} & \ell(u)(p) = 1, \\ (G,u) \models \neg\varphi & \text{if} & \text{it is not the case that } (G,u) \models \varphi, \\ (G,u) \models \varphi \wedge \psi & \text{if} & (G,u) \models \varphi \text{ and } (G,u) \models \psi, \\ (G,u) \models \xi \geq 0 & \text{if} & [[\xi]]_{G,u} \geq 0, \end{array}$$

and the semantics $[[\xi]]_{G,u}$ (the value of ξ in u) of an expression ξ by mutual induction on φ and ξ as follows.

$$\begin{array}{ll} [[c]]_{G,u} & = c, \\ [[\xi_1 + \xi_2]]_{G,u} & = [[\xi_1]]_{G,u} + [[\xi_2]]_{G,u}, \\ [[c \times \xi]]_{G,u} & = c \times [[\xi]]_{G,u}, \\ [[1_\varphi]]_{G,u} & = \begin{cases} 1 & \text{if } (G,u) \models \varphi \\ 0 & \text{else,} \end{cases} \\ [[\#\varphi]]_{G,u} & = |\{v \in V \mid (u,v) \in E \text{ and } (G,v) \models \varphi\}|. \end{array}$$

We illustrate it in the next example.

Example 7. *Consider the pointed graph G, u shown in Figure 5. We have $G, u \models p \wedge (\#\neg p \geq 2) \wedge \#(\#p \geq 1) \leq 1$. Indeed, p holds in u, u has (at least) two successors in which $\neg p$ holds. Moreover, there is (at most) one successor which has at least one p-successor.*

We define $[[\varphi]]$ as the set of the pointed graphs G, u such that $G, u \models \varphi$. Furthermore, we say that φ is *satisfiable* when there exists a pointed graph G, u such that $G, u \models \varphi$.

Definition 3. *The* satisfiability problem *of $K^{\#}$ (resp. $K^{\#,-1}$) is: given a DAG representation of a formula φ in the language of $K^{\#}$ (resp. $K^{\#,-1}$), is φ satisfiable?*

3.3 Relationship with Other Logics

Logic $K^{\#}$ is also an extension of graded modal logic. Graded modal logic [Fattorosi-Barnaba and Caro, 1985] extends classical modal logic by offering counting modality constructions of the form $\Diamond^{\geq k}\varphi$ which means there are at least k successors in which φ holds. Logic $K^{\#}$ is more expressive than graded modal logic since $\Diamond^{\geq k}\varphi$ is rewritten in $k \leq \#\varphi$. In fact, the expressivity of $K^{\#}$ goes beyond FO.

Proposition 1. *There are some properties that can be expressed in $K^{\#}$ that cannot be expressed in FO.*

Proof. The property 'there are more p-successors than q-successors' can be expressed in logic $K^{\#}$ by the formula $\#p \geq \#q$, but cannot be expressed in first-order logic, and thus not in graded modal logic. This is proven via an Ehrenfeucht-Fraïssé game (see [Nunn *et al.*, 2024]). $\square$

4 Correspondence

In this section, we lay the foundations for expressing $K^{\#}$-formulas using GNNs, and vice versa.

4.1 From Logic to GNNs

We start by showing that each $K^{\#}$-formula is captured by some GNN. The proof follows the same line as Prop 4.1 in Barceló *et al.* [2020]). However, our result is a generalization of their result since $K^{\#}$ is more expressive than graded modal logic.

Theorem 1. *For every $K^{\#}$-formula φ, we can compute in polynomial time wrt. $|\varphi|$ a GNN $\mathcal{A}_\varphi$ such that $[[\varphi]] = [[\mathcal{A}_\varphi]]$.*

Proof. Let φ be a $K^{\#}$ formula with occurring propositions $p_1, \dots, p_m$. Let $(\varphi_1, \dots, \varphi_n)$ be an enumeration of the subformulas of φ such that $\varphi_i = p_i$ for all $i \leq m$, if $\varphi_i \in sub(\varphi_j)$ then $i \leq j$ and $\varphi_n = \varphi$. W.l.o.g. we assume that all $\xi \geq 0$ subformulas are of the form $\sum_{j \in J} k_j \times 1_{\varphi_j} + \sum_{j' \in J'} k_{j'} \times \#\varphi_{j'} - c \geq 0$ for some index sets $J, J' \subseteq \{1, \dots, n\}$. We build $\mathcal{A}_\varphi$ in a stepwise fashion. Since $\mathcal{A}_\varphi$ is a GNN it is completely defined by specifying the combination function $comb_i$ for each layer l_i with $i \in \{1, \dots, n\}$ of $\mathcal{A}_\varphi$ as well as the classification function.

The output dimensionality of each $comb_i$ is n. Let $comb_1$, namely the first combination in $\mathcal{A}_\varphi$, have input dimensionality $2m$ computing $comb_1(x, y) = (x, 0, \dots, 0)$. Informally, this ensures that state x_1 has dimensionality n where the first m dimensions correspond to the propositions p_i and all other are 0. Let $comb_i(x, y) = \vec{\sigma}(xC + yA + b)$ where C, A are $n \times n$ and b is n dimensional and specified as follows. All cells of C, A and b are zero except for the case $C_{ii} = 1$ if $i \leq m$, the case $C_{ji} = -1, b_i = 1$ if $\varphi_i = \neg\varphi_j$, the case $C_{ji} = C_{li} = 1$ if $\varphi_i = \varphi_j \vee \varphi_l$ and the case $C_{ji} = k_j, A_{j'i} = k_{j'}, b_i = -c + 1$ for all $j \in J, j' \in J'$ if $\varphi_i = \sum_{j \in J} k_j \times 1_{\varphi_j} + \sum_{j' \in J'} k_{j'} \times \#\varphi_{j'} \geq c$. This means that all $comb_i$ are equal. The classification function cls is given by $cls(x) = x_n \geq 1$.

The correctness of the construction is proven in [Nunn *et al.*, 2024]. $\square$

Furthermore, the number of layers of an equivalent GNN can be reduced to be proportional to $md(\varphi)$. In that case, however, the corresponding transformation is a priori not computable in poly-time (see [Nunn *et al.*, 2024]).

Example 8. *Consider the formula $\varphi = \neg(p \vee (8 \leq 3 \times \#q))$. We define the following GNN $\mathcal{A}$ with 2 layers which is equivalent to φ as follows. We first consider the following subformulas of φ in that order:*

φ_1	φ_2	φ_3	φ_4	φ_5
p	q	$8 \leq 3 \times \#q$	$\varphi_1 \vee \varphi_3$	$\neg\varphi_4$

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

$$C = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

and $b = \begin{pmatrix} 0 & 0 & -7 & 0 & 1 \end{pmatrix}$.

The role of the two first column coordinates are to keep the current values of p and q in each vertex of the graph (hence the identity submatrix in C wrt the first two columns and rows).

The third column is about subformula $8 \leq 3 \times \#q$. *The coefficient 3 in A is at the second row (corresponding to q). The third column of* $xC + yA + b$ *is equal to* $3 \times$ *number of q-successors*-7. *The third column of* $\vec{\sigma}(xC+yA+b)$ *equals* 1 *exactly when* $3\times$ *number of q-successors* $-7 \geq 1$, *i.e. exactly when* $8 \leq 3 \times \#q$ *should hold. The fourth column handles the disjunction of* $\varphi_1 \vee \varphi_3$, *hence the* 1 *in the first and third rows. The fourth column of* $\vec{\sigma}(xC+yA+b)$ *is* $\sigma(x_1+x_3)$ *and equals* 1 *iff* $x_1 = 1$ *or* $x_3 = 1$. *The last column handles the negation* $\neg\varphi_4$. *The last column of* $\vec{\sigma}(xC + yA + b)$ *is* $\sigma(1 - x_4)$.

4.2 From GNNs to Logic

Now, we shift our attention to show how to compute a $K^{\#}$-formula that is equivalent to a GNN. Note that this direction was already tackled by Barceló *et al.* [2020] for graded modal logic for the subclass of GNNs that are FO-expressible, but their proof is not constructive. Here we give an effective construction. Furthermore, the construction can be done in poly-time in $|\mathcal{A}|$. This point is crucial: it means that we can transform efficiently a GNN into a logical formula and then perform all the reasoning tasks in the logic itself.

Theorem 2. *Let* $\mathcal{A}$ *be a GNN. We can compute in polynomial time wrt.* $|\mathcal{A}|$ *a* $K^{\#}$*-formula* $\varphi_{\mathcal{A}}$, *represented as a DAG, such that* $[[\mathcal{A}]] = [[\varphi_{\mathcal{A}}]]$.

Proof. Let $\mathcal{A}$ be a GNN with layers $l_1, \ldots, l_k$ where $comb_i$ has input dimensionality $2m_i$, output dimensionality n_i and parameters C_i, A_i and b_i and $cls(x) = a_1x_1 + \cdots + a_{n_k}x_{n_k} \geq 1$. We assume that $m_i = n_{i-1}$ for $i \geq 2$, meaning a well defined GNN. We build a formula $\varphi_{\mathcal{A}}$ over propositions $p_1, \ldots, p_{m_1}$ inductively as follows. Consider layer l_1. We build formulas $\varphi_{1,j} = \sum_{k=1}^{m_1} 1_{p_k}(C_1)_{kj} + \#p_k(A_1)_{kj} + b_j \geq 1$ for each $j \in \{1, \ldots, n_1\}$. Now, assume that $\varphi_{i-1,1}, \ldots, \varphi_{i-1,n_{i-1}}$ are given. Then, we build formulas $\varphi_{i,j} = \sum_{k=1}^{m_i} 1_{\varphi_{i-1,k}} C_{kj} + \#\varphi_{i-1,k} A_{kj} + b_j \geq 1$. In the end, we set $\varphi_{\mathcal{A}} = 1_{\psi}$ where $\psi = a_1 1_{\varphi_{k,1}} + \cdots + a_{n_k} 1_{\varphi_{k,n_k}} \geq 1$.

Let G, v be some pointed graph. The correctness of our construction is straightforward: due to the facts that all parameters in $\mathcal{A}$ are from $\mathbb{Z}$, the value $x_0(u)$ for each u in G is a vector of 0 and 1 and that each $comb_i$ in $\mathcal{A}$ applies the truncated ReLU pointwise, we have for all $i \in \{1, \ldots, k\}$ that the value of $x_i(v)$ is again a vector of 0 and 1. Therefore, we can capture each x_i using a sequence of $K^{\#}$-formulas as done above. In combination, we have that $\varphi_{\mathcal{A}}$ exactly simulates the computation of $\mathcal{A}$.

The polynomial time of this inductive construction is straightforward: we represent from beginning to end the inductively built (sub)formulas in form of a single DAG. This implies that in intermediate steps we do not have to rewrite already built subformulas $\varphi_{i-1,k}$, but can simply refer to them. Thus, in each step of the inductive procedure we only need $n_i \cdot m_i$ many steps to build all corresponding subformulas. As all n_i, m_i and k are given by $\mathcal{A}$, we have that the procedure is polynomial in the size of $\mathcal{A}$ as long as we represent $\varphi_{\mathcal{A}}$ respectively its subformulas as a DAG. $\square$

Example 9. *The idea is to build formulas that characterize the state* x_t *(which is the output at layer* t *if* $t \geq 1$, *and also the input at layer* $t+1$*). Initially,* x_0 *is the state whose values comes the truth values of the atomic propositions. We suppose we have only two propositions* p_1, p_2. *Hence, the two formulas that represent state* x_0 *are* $\varphi_{01} = p_1$ *and* $\varphi_{02} = p_2$. *Suppose that the two layers are given by the same combination and aggregation functions:*

$$x_1(u) = comb(x_0(u), agg(\{\{x_0(v) | uv \in E\}\}))$$
$$x_2(u) = comb(x_1(u), agg(\{\{x_1(v) | uv \in E\}\}))$$

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

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

The formulas that represent the state x_1 *are formula* $\varphi_{11} = 1_{\varphi_{01}} + 2 \times 1_{\varphi_{02}} - 3\#\varphi_{01} + 4\#\varphi_{02} + 5 \geq 1$ *and formula* $\varphi_{12} = 6 \times 1_{\varphi_{01}} + 7 \times 1_{\varphi_{02}} + 8\#\varphi_{01} - 9\#\varphi_{02} + 10 \geq 1$. *The formulas that represent the state* x_2 *are formula* $\varphi_{21} = 1_{\varphi_{11}} + 2 \times 1_{\varphi_{12}} - 3\#\varphi_{11} + 4\#\varphi_{12} + 5 \geq 1$ *and formula* $\varphi_{22} = 6 \times 1_{\varphi_{11}} + 7 \times 1_{\varphi_{12}} + 8\#\varphi_{11} - 9\#\varphi_{12} + 10 \geq 1$.

Finally, suppose that the classification function is given by $cls(x, x') = 5x - 3x' \geq 1$. *So the formula* $tr(\mathcal{A})$ *that represents the GNN* $\mathcal{A}$ *is* $5 \times 1_{\varphi_{21}} - 3 \times 1_{\varphi_{22}} \geq 1$.

5 Complexity of the Logic

In this section, we address the complexity of the satisfiability problem of the logic $K^{\#}$. Specifically, we prove that it is PSPACE-complete (Theorem 3). Additional details are available in [Nunn *et al.*, 2024].

We know from Ladner [1977] that the satisfiability problem of the standard modal logic K is PSPACE-hard, and we observed before that $K^{\#}$ is an extension of logic K since we can define $\Box\varphi := (\#(\neg\varphi) \leq 0)$ and we are working with graphs whose relation (represented by the set E of edges) is unconstrained. Hence, we have:

Proposition 2. *The satisfiability problem of the logic* $K^{\#}$ *is* PSPACE*-hard.*

To show that the satisfiability problem of $K^{\#}$ is also in PSPACE, we are going to follow a strategy illustrated in Figure 6:

1. First we show that the problem can be reduced efficiently to the satisfiability problem of $K^{\#}$ with the formulas represented as trees (and not arbitrary DAGs). Let us call this problem $K^{\#}_{tree}$-sat. This will be Lemma 1.

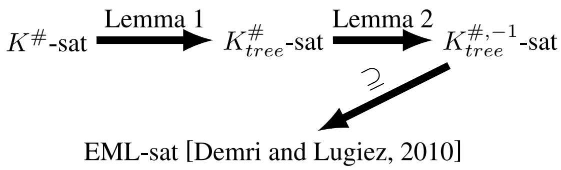

Figure 6: Schema of the proof to establish the PSPACE upper bound of the satisfiability problem of $K^{\#}$. Arrows are poly-time reductions.

2. Second, we show that $K^{\#}_{tree}$-sat can be reduced efficiently to the satisfiability problem of $K^{\#,-1}$ with formulas represented as trees. This will be Lemma 2.
3. Third, we conclude by simply observing that $K^{\#,-1}$ can be seen as fragment of Extended Modal Logic (EML) introduced by Demri and Lugiez [2010], whose satisfiability problem is in PSPACE.

It is sufficient to establish an upper-bound on the complexity of $K^{\#}$, which is stated in the next proposition.

Proposition 3. *The satisfiability problem of the logic $K^{\#}$ is in* PSPACE.

The initial step in both proofs of Lemma 1 and Lemma 2 shares the same idea. Given a subformula φ or arithmetic subexpression 1_φ, we introduce a fresh propositional variable p_φ. The variable can then be used as a shortcut, or used *in* a shortcut, to refer to the truth (resp. value) of the original subformula (resp. arithmetic subexpression). An adequate formula is then added to 'factorize' the original subformula or arithmetic subexpression, and enforced at every 'relevant' world of a model to capture the intended properties of the transformations. To this end, we can rely on the modality $\square^{\to m}\varphi$, simply defined as $\square^{\to m}\varphi := \bigwedge_{0\leq i\leq m} \square^i\varphi$, with m being the modal depth of the original formula.

To prove Proposition 3, we first show that we can efficiently transform a $K^{\#}$ formula (represented as a DAG) into an equi-satisfiable $K^{\#}$ formula which is represented as a tree.

For every node of the DAG that is a formula, we introduce a fresh proposition. Starting from the leafs of the DAG, we replace every subformula ψ with its corresponding proposition p_ψ which will simulate the truth of ψ. As we go towards the root, we replace ψ with p_ψ, and we add a formula of the form $\square^{\to m}(p_\psi \leftrightarrow \psi')$, where ψ' is a syntactic variant of ψ, using the previously introduced fresh variables in place of the original formulas. Since the root is a formula (as opposed to an arithmetic expression), we end with a conjunction of $\square^{\to m}(p_\psi \leftrightarrow \psi')$ formulas, and a DAG consisting of only one node with the proposition p_φ which simulates the original formula. Since we are not duplicating arbitrary formulas, but only propositional variables, this process ensures that the size of the new formula (as a string or as a tree) remains polynomial in the size of the DAG. The idea is similar to Tseitin transformation [Tseitin, 1983].

Example 10. *Consider the DAG depicted on Figure 4(a). We introduce the propositional variables p_p, p_q, $p_\wedge$, $p_\leq$, and $p_\vee$, each corresponding to a node of the DAG denoting a subformula. The formula is equi-satisfiable with the formula containing the following conjuncts: $p_\vee$, $\square^{\to 1}(p_p \leftrightarrow p)$, $\square^{\to 1}(p_q \leftrightarrow q)$, $\square^{\to 1}(p_\wedge \leftrightarrow p_p \wedge p_q)$, $\square^{\to 1}(p_\geq \leftrightarrow \#p_\wedge \leq 1_{p_\wedge})$, and $\square^{\to 1}(p_\vee \leftrightarrow p_\wedge \vee p_\geq)$. For the formula to be true, $p_\vee$ must be true, and so must be $p_\wedge$ or $p_\geq$, and so on.*

The previous example explains the idea of the proof in order to efficiently transform formulas represented as arbitrary DAGs into equi-satisfiable formulas represented as tree. The following lemma states that result as a reduction.

Lemma 1. *The satisfiability problem of the logic $K^{\#}$ reduces to $K^{\#}_{tree}$-sat in poly-time.*

Now, we show that 1_χ expressions can be removed efficiently and in a satisfiability-preserving way. Technically, 1_χ expressions are replaced by a counting modality, counting successors with a fresh proposition p_χ, which artificially simulates the truth of χ.

Example 11. *We consider the formula $1_\chi \geq 1$. We introduce a fresh propositional variable p_χ and replace the 1_χ with a counting modality $\#p_\chi$. Hence, we simply rewrite the $K^{\#}$ formula into the $K^{\#,-1}$ formula $\#p_\chi \geq 1$. We then add a subformula that characterizes the fact that counting p_χ-worlds simulates the value of 1_χ. For that, we say that if χ holds, then there must be exactly one p_χ-successor and if χ does not hold then there must be no p_χ-successor. At the end, the formula $1_\chi \geq 1$ is rewritten into the $K^{\#,-1}$ formula:*

$$(\#p_\chi \geq 1) \wedge \square^{\to 0}((\chi \to \#p_\chi = 1) \wedge (\neg\chi \to \#p_\chi = 0)).$$

The previous example explains the idea of the proof in order to get rid of 1_χ expressions. The following lemma states that result as a reduction.

Lemma 2. *When formulas are represented as trees, the satisfiability problem of the logic $K^{\#}$ reduces to the satisfiability problem of $K^{\#,-1}$ in poly-time.*

The logic $K^{\#,-1}$ can be seen as a fragment of the logic EML introduced by Demri and Lugiez [2010], where formulas are represented as strings (which have the same size as their syntactic tree representation), and whose satisfiability problem is in PSPACE. Together with Lemma 1 and Lemma 2, this proves Proposition 3. Proposition 2 and Proposition 3 allow us to conclude.

Theorem 3. *The satisfiability problem of the logic $K^{\#}$ is* PSPACE*-complete.*

Since coPSPACE = PSPACE, the validity and unsatisfiability problems of $K^{\#}$ formulas are also PSPACE-complete. This will be instrumental in the next section.

6 Complexity of Reasoning about GNNs

We are now ready to wrap up algorithmic results for reasoning about GNNs.

Corollary 1. *When considering GNNs, the problems* P1–P4 *are in* PSPACE.

Proof. Let us prove it for P1. For the other problems, the principle is similar. Given $\mathcal{A}$, φ, we can check that $[[\mathcal{A}]] = [[\varphi]]$, by computing the $K^{\#}$-formula $tr(\mathcal{A}) \leftrightarrow \varphi$ and then check that $tr(\mathcal{A}) \leftrightarrow \varphi$ is valid. In order to check for validity, we check that $\neg(tr(\mathcal{A}) \leftrightarrow \varphi)$ is not $K^{\#}$-satisfiable. This can be done in poly-space via Theorem 3. $\square$

Theorem 4. *Problems* P1–P4 *are* PSPACE*-complete.*

Proof. Membership comes from Corollary 1. Hardness follow directly from the PSPACE-hardness of the satisfiability/unsatisfiability/validity of a $K^{\#}$-formula (cf. Theorem 3). Consider $\mathcal{A}_{all}$ a GNN that accepts all graphs, and $\mathcal{A}_{none}$ a GNN that rejects all graphs. Consider the following poly-time reductions:

- From validity of φ to P1: $[[\mathcal{A}_{all}]] = [[\varphi]]$.
- From validity of φ to P2: $[[\mathcal{A}_{all}]] \subseteq [[\varphi]]$.
- From unsatisfiability of φ to P3: $[[\varphi]] \subseteq [[\mathcal{A}_{none}]]$.
- From satisfiability of φ to P4: $[[\varphi]] \cap [[\mathcal{A}_{all}]] \neq \emptyset$.

$\square$

7 Related Work

The links between graded modal logic [Fattorosi-Barnaba and Caro, 1985] and GNNs have already been observed in the literature [Barceló *et al.*, 2020; Grohe, 2021]. We know from Barceló *et al.* [2020] that a GNN expressible in first-order logic (FO) is also captured by a formula in graded modal logic. Nonetheless, graded modal logic has significant limitations as it cannot represent fundamental arithmetic properties essential to GNN computations, especially those that are not expressible in first-order logic (FO). In a similar vein of finding logical counterparts of GNNs, Cucala *et al.* [2023] identify a class of GNNs that corresponds to Datalog. Many works combine modal/description logic and quantitative aspects: counting (see, [Areces *et al.*, 2010; Demri and Lugiez, 2010; Hampson, 2016; Baader *et al.*, 2020]), or probabilities [Shirazi and Amir, 2007]. Galliani *et al.* [2023] extend the basic description logic $\mathcal{ALC}$ with a new operator to define concepts by listing features with associated weights and by specifying a threshold. They prove that reasoning wrt to a knowledge-base is EXPTIME-complete. Linear programming and modal logic have already been combined to solve the satisfiability problem of graded/probabilistic modal logic [Snell *et al.*, 2012]. Our logic $K^{\#,-1}$ can be seen as a 'recursification' of the logic used in Sälzer and Lange [2022]. They allow for counting successors satisfying a given feature, and not any subformula. Interestingly, they allow for counting also among all vertices in the graph (sort of counting universal modality). Their logic is proven to be undecidable by reduction from the Post correspondence problem. Contrary to our setting, they use their logic only to characterize labeled graphs, but not to give a back and forth comparison with the GNN machinery itself.

There are different ways to address explanation and verification of GNNs. One approach is to use GNNs, which are explainable by design as considered by Müller *et al.* [2022]. This is of course a deep debate: using models easy to use for learning, versus interpretable models [Rudin, 2019]. The choice depends on the target application. Yuan *et al.* [2023] provide a survey on methods used to provide explanations for GNNs by using black-box techniques. Instance-level explanations explain why a graph has been recognized by an GNN; model-level ones how a given GNN works. There are also methods based on Logic Explained Networks and variants to generate logical explanation candidates φ [Azzolin *et al.*, 2022]. Once a candidate is generated we could imagine to use our problem P1 (given in the introduction) to check whether $[[\mathcal{A}]] = [[\varphi]]$, and thus being able to fully synthesize a trustworthy explanation. The logic $K^{\#}$ and the results presented in this paper are thus precious tools to assist in the search of model-level explanations of GNNs.

8 Conclusion and Outlook

We presented logic $K^{\#}$, capturing a broad and natural class of GNNs. Furthermore, we proved that the satisfiability problem of $K^{\#}$ is PSPACE-complete, leading to direct practical perspectives to solve formal verification and explanation problems regarding this class of GNNs.

There are several directions to go from here. First, we aim to consider a larger class of GNNs. This will require to augment the expressivity of the logic, for instance by adding other activation functions like ReLU in an extension of $K^{\#}$. Fortunately, SMT solvers have been extended to capture ReLU (for example see Katz *et al.* [2017]), but it is open how a well-defined extension of our logic looks like. Similarly, it would be useful to allow for more flexibility in the combination or aggregation functions of the considered GNNs. For example, allowing for arbitrary feedforward neural networks in the combination functions or considering the mean function as aggregation. Another idea is to allow for global readouts in the considered GNNs. A GNN with global readout does not only rely on local messages passing from neighbors to neighbors, and can instead consider the whole graph. Presumably, an extension of $K^{\#}$ would need something akin to a universal modality to captures such behaviour.

A second interesting direction would be to consider other classes of graphs. For instance, reflexive, transitive graphs. Restricted types of graphs lead to different modal logics: KT (validities on reflexive Kripke models), KD (on serial models), $S4$ (reflexive and transitive models), KB (models where relations are symmetric), $S5$ (models where relations are equivalence relations), etc. [Blackburn *et al.*, 2001]. The logic $K^{\#}$ defined in this paper is the counterpart of modal logic K with linear programs. In the same way, we could define $KT^{\#}$, $S4^{\#}$, $S5^{\#}$, etc. For instance, $KB^{\#}$ would be the set of validities of $K^{\#}$-formulas over symmetric models; $KB^{\#}$ would be the logic used when GNNs are only used to recognize undirected pointed graphs (for instance persons in a social network where friendship is undirected).

A third direction of research would be to build a tool for solving verification and explainability issues of GNNs using $K^{\#}$ as described in Section 1. The satisfiability problem of $K^{\#}$ is in PSPACE. Many practical problems are in PSPACE (model checking against a linear temporal logic property just to cite one [Sistla and Clarke, 1985]). This means that we may rely on heuristics to guide the search, but this needs thorough investigations.

Ethical Statement

There are no ethical issues.

Acknowledgments

This work was supported by the ANR EpiRL project ANR-22-CE23-0029, and by the MUR (Italy) Department of Excellence 2023–2027. We would like to thank Stéphane Demri for discussions.

References

[Areces *et al.*, 2010] Carlos Areces, Guillaume Hoffmann, and Alexandre Denis. Modal logics with counting. In Anuj Dawar and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information and Computation, 17th International Workshop, WoLLIC 2010, Brasilia, Brazil, July 6-9, 2010. Proceedings*, volume 6188 of *Lecture Notes in Computer Science*, pages 98–109. Springer, 2010.

[Azzolin *et al.*, 2022] Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of GNNs via logic combination of learned concepts. *CoRR*, abs/2210.07147, 2022.

[Baader *et al.*, 2020] Franz Baader, Bartosz Bednarczyk, and Sebastian Rudolph. Satisfiability and query answering in description logics with global and local cardinality constraints. In *24th European Conference on Artificial Intelligence (ECAI 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 616–623. IOS Press, 2020.

[Barceló *et al.*, 2020] Pablo Barceló, Egor V Kostylev, Mikaël Monet, Jorge Pérez, Juan Reutter, and Juan-Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*, Virtual conference, Ethiopia, April 2020.

[Blackburn *et al.*, 2001] Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[Cucala *et al.*, 2023] David Tena Cucala, Bernardo Cuenca Grau, Boris Motik, and Egor V. Kostylev. On the correspondence between monotonic max-sum GNNs and Datalog. 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 658–667, 2023.

[de Rijke, 2000] Maarten de Rijke. A note on graded modal logic. *Studia Logica*, 64(2):271–283, 2000.

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

[Fattorosi-Barnaba and Caro, 1985] Maurizio Fattorosi-Barnaba and F. Caro. Graded modalities. I. *Studia Logica*, 44(2):197–221, 1985.

[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.

[Gilmer *et al.*, 2017] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *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*, pages 1263–1272. PMLR, 2017.

[Grohe, 2021] Martin Grohe. The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–17. IEEE, 2021.

[Hampson, 2016] Christopher Hampson. Decidable first-order modal logics with counting quantifiers. In Lev D. Beklemishev, Stéphane Demri, and András Maté, editors, *Advances in Modal Logic 11, proceedings of the 11th conference on "Advances in Modal Logic," held in Budapest, Hungary, August 30 - September 2, 2016*, pages 382–400. College Publications, 2016.

[Huang *et al.*, 2020] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Comput. Sci. Rev.*, 37:100270, 2020.

[Katz *et al.*, 2017] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In Rupak Majumdar and Viktor Kuncak, editors, *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, volume 10426 of *Lecture Notes in Computer Science*, pages 97–117. Springer, 2017.

[Ladner, 1977] Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal on Computing*, 6(3):467–480, 1977.

[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.

[Müller *et al.*, 2022] Peter Müller, Lukas Faber, Karolis Martinkus, and Roger Wattenhofer. DT+GNN: A fully explainable graph neural network using decision trees. *CoRR*, abs/2205.13234, 2022.

[Nunn *et al.*, 2024] Pierre Nunn, Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. A Logic for Reasoning About Aggregate-Combine Graph Neural Networks. *CoRR*, abs/2405.00205, 2024.

[Pinchinat *et al.*, 2022] Sophie Pinchinat, Sasha Rubin, and François Schwarzentruber. Formula synthesis in propositional dynamic logic with shuffle. 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 9902–9909. AAAI Press, 2022.

[Reiser *et al.*, 2022] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, and Pascal Friederich. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(93), 2022.

[Rudin, 2019] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019.

[Salamat *et al.*, 2021] Amirreza Salamat, Xiao Luo, and Ali Jafari. Heterographrec: A heterogeneous graph-based neural networks for social recommendations. *Knowl. Based Syst.*, 217:106817, 2021.

[Sälzer and Lange, 2022] Marco Sälzer and Martin Lange. We cannot guarantee safety: The undecidability of graph neural network verification. *CoRR*, abs/2206.05070, 2022.

[Shirazi and Amir, 2007] Afsaneh Shirazi and Eyal Amir. Probabilistic modal logic. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 489–495. AAAI Press, 2007.

[Sistla and Clarke, 1985] A. Prasad Sistla and Edmund M. Clarke. The complexity of propositional linear temporal logics. *J. ACM*, 32(3):733–749, 1985.

[Snell *et al.*, 2012] William Snell, Dirk Pattinson, and Florian Widmann. Solving graded/probabilistic modal logic via linear inequalities (system description). In Nikolaj S. Bjørner and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings*, volume 7180 of *Lecture Notes in Computer Science*, pages 383–390. Springer, 2012.

[Tseitin, 1983] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483, 1983.

[Xiong *et al.*, 2021] Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021.

[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.

[Yuan *et al.*, 2023] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(5):5782–5799, 2023.

[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.