


# Decidability of Graph Neural Networks via Logical Characterizations

Michael Benedikt ✉ 

University of Oxford, UK

Chia-Hsuan Lu ✉ 

University of Oxford, UK

Boris Motik ✉ 

University of Oxford, UK

Tony Tan ✉ 

University of Liverpool, UK

---

## Abstract

We present results concerning the expressiveness and decidability of a popular graph learning formalism, graph neural networks (GNNs), exploiting connections with logic. We use a family of recently-discovered decidable logics involving “Presburger quantifiers”. We show how to use these logics to measure the expressiveness of classes of GNNs, in some cases getting exact correspondences between the expressiveness of logics and GNNs. We also employ the logics, and the techniques used to analyze them, to obtain decision procedures for verification problems over GNNs. We complement this with undecidability results for static analysis problems involving the logics, as well as for GNN verification problems.

**2012 ACM Subject Classification** Theory of computation → Logic and verification

**Keywords and phrases** Logic, Graph Neural Networks

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2024.127

**Category** Track B: Automata, Logic, Semantics, and Theory of Programming

**Related Version** *Full Version*: <https://arxiv.org/abs/2404.18151>

**Funding** *Michael Benedikt*: This research was funded by EPSRC grant EP/T022124/1.

## 1 Introduction

Graph Neural Networks (GNNs) have become the most common model for learning functions that work on graph data. Like traditional neural networks, GNNs consist of a layered architecture where layer  $k + 1$  takes as input the output of layer  $k$ . Each layer computes a function from graph vertices to a vector of numerical values – the *feature vector*. Computation of the feature vector at layer  $k + 1$  for a node  $u$  is based on aggregating vectors for layer  $k$  of nodes  $v$  that are related to  $u$  in the source graph: for example aggregating vectors associated to nodes adjacent to  $u$  in the graph. In an aggregation, the vectors of the previous vectors may be transformed using linear functions. A layer can perform multiple aggregations – corresponding to different linear functions – and then combine them to get the feature vector for the next layer. The use of graph structure ensures that the computation of the network is *invariant*: depending only on the input graph and the node up to isomorphism. There are many variations of GNN. One key design choice is the kind of aggregation used - one can use “local aggregation”, over the neighbors of a node, or aggregation over all nodes in the graph. A second design choice is the kind of numerical functions that can be applied to vector components, in particular the kind of *activation functions* that can be applied at each layer: e.g. ReLU, sigmoid, piecewise linear functions.



© Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan;  
licensed under Creative Commons License CC-BY 4.0

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).

Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;

Article No. 127; pp. 127:1–127:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



An important issue in the study of graph learning is the *expressiveness* of a learning model. What kinds of computations can a given type of GNN express? The first results in this line were about the *separating power of a graph learning model*: what pairs of nodes can be distinguished using GNNs within a certain class. For example, it is known that the separating power of standard GNN models is limited by the Weisfeiler-Leman (WL) test [16].

A finer-grained classification would characterize the functions computed by GNNs within a certain class, in terms of some formalism that is easier to analyze. Such characterizations are referred to as *uniform expressiveness results* and there has been much less work in this area. [1] provides a classification of a class of GNNs in terms of *modal logic*. The main result in [1] is a characterization of the classifiers expressible in first-order logic that can be performed with a GNN having only *local aggregation* and *truncated ReLU activations* over *undirected graphs*. They also provide a lower bound on the expressiveness of GNNs having in addition a “global aggregator”, that sums over all nodes in the graph.

In this work we continue the line of work on uniform expressiveness. Our work improves on the state of the art in a number of directions:

- *From first order expressiveness to general expressiveness* In contrast to [1], we provide logical characterizations of *all* the functions that can be computed by certain GNN formalisms, not just the intersection with first-order logic. To do this we utilize logics that go beyond first order, but which are still amenable to analysis.
- *From expressiveness to verification* While we deal with GNNs that go beyond first-order logic, we can still obtain characterizations in a logic where the basic satisfiability problems are decidable. This provides us with decidability of a number of natural verification problems related to GNNs. In doing this, we show a surprising link between GNNs and recently-devised decidable logics going beyond first-order logic, so-called *Presburger logics*.
- *From undirected graphs to directed graphs* While prior work focused on undirected graphs, we explore how the expressiveness characterizations vary with GNNs that can recognize directionality of graph edges. The aim is to show that the logical characterizations and decidability are often independent of the restriction to undirected graphs.
- *From bounded to unbounded activations* We explore the impact of the activation functions. We begin with the case of *bounded activation functions*, like the truncated ReLU of [1], and establish characterizations and decidability results for GNNs using this function. We show both some contrasts and some similarity to the case of *unbounded activation functions*, including the standard ReLU. Here some, but not all, of the corresponding decidability results fail.

**Related work.** Logics have been used to characterize the separating power of GNN languages (“non-uniform expressiveness”) for a number of years: see [8] for an overview. The recent [9] provides logical characterizations of GNNs with piecewise linear activations. The logic is not decidable; indeed our undecidability results imply that one cannot capture such GNNs with a decidable logic.

We employ logical characterizations to gain insight on two basic verification problems – whether a given classification can be achieved on some nodes or on all nodes. There is prior work on verification of GNNs, but it focuses on more complex (but arguably more realistic) problems, adversarial robustness. The closest paper to ours is the recent [14], which formalizes a broad set of problems related to verifying that the output is in a certain region in Euclidean space. [14] provides both decidability and undecidability theorems, but they are incomparable to ours both in the results and in the techniques. For example Theorem 1 of [14] shows undecidability of a satisfiability problem where we verify that certain nodes

output a particular value, over GNNs which always distinguish a node from its neighbor. Theorem 2 of [14] shows a decidability result with a different kind of specification, where the degree of input graphs is bounded.

Recently, logics that combine uninterpreted relations with Presburger arithmetic have been applied to the analysis of transformers – transducers that process strings [4, 2]. Since this is outside of the context of general graphs, the details of the logics that are employed are a bit different than those we consider, and the focus is not on the decidability border.

**Organization.** We formalize our GNN model and the basic logics we study in Section 2. We present results on logical characterizations of GNNs with “bounded activation functions” – like the truncated ReLU of [1]. We apply these characterizations to get decidability results. Section 4 turns to the case of unbounded activation functions, which includes the traditional ReLU function. Here we provide lower bounds for expressiveness, and then turn to the implications for decidability. Section 5 gives conclusions and discusses several open issues and directions for future work. For space reasons, *many details of the constructions and proofs are deferred to the full version, which is available on arxiv.*

## 2 Preliminaries

Let  $\mathbb{N}$ ,  $\mathbb{N}^+$ ,  $\mathbb{Z}$ , and  $\mathbb{Q}$  be the set of natural numbers, positive natural numbers, integers, and rational numbers, respectively. For  $p, q \in \mathbb{Z}$  and  $p \leq q$ ,  $[p, q]$  is the set of integers between  $p$  and  $q$ , including  $p$  and  $q$ . For  $r \in \mathbb{Q}$ ,  $\lceil r \rceil$  is the smallest integer greater than or equal to  $r$ .

For a function  $f$  mapping from  $\mathbb{Q}$  to  $\mathbb{Q}$  and a vector  $b \in \mathbb{Q}^m$ ,  $f(b)$  denotes that  $f$  is applied to each entry of  $b$ .

► **Definition 1.** An  $n$ -graph is a tuple  $\langle V, E, \{U_c\}_{1 \leq c \leq n} \rangle$ , where  $n \in \mathbb{N}$  is the number of vertex colors;  $V$  is a nonempty finite set of vertices;  $E \subseteq V \times V$  is a set of edges; each  $U_c \subseteq V$  is the set of  $c$ -colored vertices.

Note that we allow self-loops in graphs, and a graph is by default a *directed graph*. For a graph  $\mathcal{G}$ , we say that  $\mathcal{G}$  is a *undirected graph* if for all  $v, u \in V$ ,  $(v, u) \in E$  if and only if  $(u, v) \in E$ . For a vertex  $v \in V$ , we let  $\mathcal{N}_{\text{out}, \mathcal{G}}(v) := \{u \mid (v, u) \in E\}$  and refer to this as the set of *out-neighbors* of  $v$ . The set of *in-neighbors* of  $v$ , denoted  $\mathcal{N}_{\text{in}, \mathcal{G}}(v)$  are defined analogously.

**Graph Neural Networks.** We use a standard notion of “aggregate-combine” graph neural networks with rational coefficients. The only distinction from the usual presentation is that we allow GNNs to work over directed graphs, with separate aggregations over incoming and outgoing edges, while traditional GNNs work on undirected graphs.

► **Definition 2.** An  $n$ -graph neural network (GNN) is a tuple

$$\left\langle \{d_\ell\}_{0 \leq \ell \leq L}, \{f^\ell\}_{1 \leq \ell \leq L}, \{C^\ell\}_{1 \leq \ell \leq L}, \{A_x^\ell\}_{\substack{1 \leq \ell \leq L \\ x \in \{\text{out}, \text{in}\}}}, \{R^\ell\}_{1 \leq \ell \leq L}, \{b^\ell\}_{1 \leq \ell \leq L} \right\rangle,$$

where  $L \in \mathbb{N}^+$  is the number of layers; each  $d_\ell \in \mathbb{N}^+$ , called the dimension of the  $\ell^{\text{th}}$  layer, requiring  $d_0 := n$ , the number of colors; each  $f^\ell : \mathbb{Q} \rightarrow \mathbb{Q}$ , the activation function of the  $\ell^{\text{th}}$  layer; each  $C^\ell, A_x^\ell, R^\ell \in \mathbb{Q}^{d_\ell \times d_{\ell-1}}$ , the coefficient matrices of the  $\ell^{\text{th}}$  layer; and  $b^\ell \in \mathbb{Q}^{d_\ell}$ , the bias vector of the  $\ell^{\text{th}}$  layer.

All the coefficients are rational. In order to have an effective representation of a GNN, we will also *assume that the activation functions are computable.*

► **Definition 3.** For an  $n$ -GNN  $\mathcal{A}$  and an  $n$ -graph  $\mathcal{G}$ , the computation of  $\mathcal{A}$  on  $\mathcal{G}$  is a sequence of derived feature functions  $\{\xi_{\mathcal{G}}^{\ell} : V \rightarrow \mathbb{Q}^{d_{\ell}}\}_{0 \leq \ell \leq L}$  defined inductively: for  $\ell = 0$ , if  $v \in U_c$ , then the  $c^{\text{th}}$  entry of  $\xi_{\mathcal{G}}^0(v)$  is 1; otherwise, the entry is 0. For  $1 \leq \ell \leq L$ ,

$$\xi_{\mathcal{G}}^{\ell}(v) := f^{\ell} \left( C^{\ell} \xi_{\mathcal{G}}^{\ell-1}(v) + \sum_{x \in \{\text{out}, \text{in}\}} \left( A_x^{\ell} \sum_{u \in \mathcal{N}_{x, \mathcal{G}}(v)} \xi_{\mathcal{G}}^{\ell-1}(u) \right) + R^{\ell} \sum_{u \in V} \xi_{\mathcal{G}}^{\ell-1}(u) + b^{\ell} \right).$$

For  $v \in V$ ,  $\xi_{\mathcal{G}}^{\ell}(v)$  is called the  $\ell$ -feature vector of  $v$ , and  $\xi_{\mathcal{G}, i}^{\ell}(v)$  is the  $i^{\text{th}}$  entry of  $\xi_{\mathcal{G}}^{\ell}(v)$ .

That is, we compute the feature values of a node  $v$  at layer  $\ell + 1$  by adding several components. One component aggregates over the  $\ell$ -layer feature vector from the outgoing neighbors of  $v$ , and applies a linear transformation. Another component does the same for the incoming neighbors of  $v$ , a third does this for every node in the graph, while another applies a transformation to the  $\ell$ -layer feature vector of  $v$  itself. The linear transformation can be different for each component, and in particular can be a zero matrix that just drops that component. The final component of the sum is the bias vector.

When the graph  $\mathcal{G}$  is clear from the context, we omit it and simply write  $\xi^{\ell}(v)$  and  $\xi_i^{\ell}(v)$ , and similarly when the graph  $\mathcal{G}$  is clear from the context, write  $\mathcal{N}_{\text{out}}(v)$  and  $\mathcal{N}_{\text{in}}(v)$  for the in-neighbors and out-neighbors.

Note that in most presentations of GNNs, one deals with only undirected edges. The above definition degenerates in that setting to two aggregations per layer, with the aggregation over all nodes often referred to in the literature as the *global readout*.

In some presentations of GNNs, a *classification function*, which associates a final Boolean decision to a node, is included in the definition. In our case, we have separated out the classification function as an independent component in defining the expressiveness: see the last part of the preliminaries.

**Classes of activation functions.** Following prior work on analysis of GNNs, some of our results will deal with activation functions that are bounded in value:

► **Definition 4.** We say that the function  $f : \mathbb{Q} \rightarrow \mathbb{Q}$  is eventually constant, if there exists  $t_{\text{left}}, t_{\text{right}} \in \mathbb{Q}$  satisfying  $t_{\text{left}} < t_{\text{right}}$ , called the left and right thresholds of  $f$ , such that for every  $x \leq t_{\text{left}}$ ,  $f(x) = f(t_{\text{left}})$ ; for every  $x \geq t_{\text{right}}$ ,  $f(x) = f(t_{\text{right}})$ .

A standard eventually constant function is the *truncated ReLU function*, which is 0 for negative reals, 1 for  $x$  greater than 1, and  $x$  otherwise [1]. There are other eventually constant functions that are used in practice: for example, the linear approximation of standard bounded functions used in graph learning, like the Sigmoid activation function. We will be interested in functions that are defined on the reals, but which preserve the rationals. The definition of eventually constant extends to such a function in the obvious way.

For a GNN with eventually constant activation functions, we use  $\{t_{\text{left}}^{\ell}\}_{1 \leq \ell \leq L}$  and  $\{t_{\text{right}}^{\ell}\}_{1 \leq \ell \leq L}$  to denote the left and right thresholds of the GNN's activation functions.

We also consider *unbounded activation functions*, such as the *standard ReLU function*, which is  $x$  for non-negative reals and 0 for negative reals.

**Flavors of GNN.** For a GNN  $\mathcal{A}$ , we say that  $\mathcal{A}$  is *outgoing-only*, denoted by  $\mathcal{O}$ , if for every  $1 \leq \ell \leq L$ ,  $A_{\text{in}}^{\ell}$  is a zero matrix.  $\mathcal{A}$  is *bidirectional*, denoted by  $\mathcal{B}$ , if there is no restriction on  $A_{\text{in}}^{\ell}$ .  $\mathcal{A}$  is *local*, denoted by  $\mathcal{L}$ , if for every  $1 \leq \ell \leq L$ ,  $R^{\ell}$  is a zero matrix. In the usual GNN terminology, this would mean that *there is no global readout*.  $\mathcal{A}$  is

global, denoted by  $\mathcal{G}$ , if global readout is allowed.  $\mathcal{A}$  is *eventually constant*, denoted by  $\mathcal{C}$ , if for  $1 \leq \ell \leq L$ ,  $f^\ell$  is an eventually constant function. Our results outside of eventually constant will deal with either *piecewise linear* activations, denoted  $\mathcal{PW}$ , truncated ReLU activations, denoted  $\text{TrReLU}$ , or standard ReLU activations. We use the following naming,  $(\mathcal{O}|\mathcal{B})(\mathcal{L}|\mathcal{G})(\mathcal{C}|\mathcal{PW}|\text{TrReLU}|\text{ReLU})\text{-GNN}$ , for the set of GNNs satisfying constraints given by the prefix. For example,  $\mathcal{OLC}\text{-GNN}$  is the set of outgoing-only, local, and eventually constant GNNs;  $\mathcal{BGPW}\text{-GNN}$  is the set of GNNs allowing both incoming and outgoing, global readout, and piecewise linear activations.

**Classifiers and Boolean semantics.** Our GNNs define vector-valued classification functions on nodes. But for comparing with expressiveness and in defining verification problems, we will often use a derived function from nodes to Booleans. We do this by thresholding at the end – below we use .5 for convenience, but other choices do not impact the results.

► **Definition 5.** For a  $L$ -layer  $n$ -GNN  $\mathcal{A}$ , an  $n$ -graph  $\mathcal{G}$ , and a vertex  $v \in V$ , we say that  $\mathcal{A}$  accepts the tuple  $\langle \mathcal{G}, v \rangle$ , if  $\xi_{\mathcal{G},1}^L(v) \geq 0.5$ .

Note that the global readout component can interact with the activation functions  $f^\ell$ , which can behave very differently on translated values due to non-linearity – think of a typical  $f^\ell$  as a piecewise linear function. Global readout can also interact with the classification threshold, pushing some values above the threshold while leaving others below.

**Two-variable Modal Logic with Presburger Quantifiers.** We review logic with Presburger quantifiers. The basic idea is to combine a decidable logic on uninterpreted structures, like two-variable logic or guarded logic, with the ability to perform some arithmetic on the number of elements. There are several formalisms in the literature that combine Presburger arithmetic with a decidable uninterpreted logic, some originating many years ago [11]. We will rely on a recent logic from [3], but we will need several variations of the underlying idea here.

► **Definition 6.** A Presburger quantifier is of the form:

$$\mathcal{P}(x) := \sum_{i=1}^k \lambda_i \cdot \#_y[\varphi_i(x, y)] \otimes \delta,$$

where  $\delta \in \mathbb{Z}$ ; each  $\lambda_i \in \mathbb{Z}$ ; each  $\varphi_i(x, y)$  is a formula with free variables  $x$  and  $y$ ;  $\otimes$  is one of  $=, \neq, \leq, \geq, <, >$ . Note that  $\mathcal{P}(x)$  has one free variable  $x$ .

We give the semantics of these quantifiers inductively, assuming a semantics for  $\varphi_i(x, y)$ . Given a graph  $\mathcal{G}$  and a vertex  $v \in V$ , we say that  $\mathcal{P}(x)$  holds in  $\mathcal{G}, x/v$ , denoted by  $\mathcal{G} \models \mathcal{P}(v)$ , if the following (in)equality holds in  $\mathbb{Z}$ .

$$\sum_{i=1}^k \lambda_i \cdot |\{u \in V \mid \mathcal{G} \models \varphi_i(v, u)\}| \otimes \delta$$

► **Remark 7.** Note that each Presburger quantifier can be rewritten as a Boolean combination of expressions which *only use the inequality symbol  $\geq$  as  $\otimes$* . For example,  $(\#_y[\varphi(x, y)] = \delta)$  and  $(\#_y[\varphi(x, y)] \geq \delta) \wedge \neg(\#_y[\varphi(x, y)] \geq \delta + 1)$  are semantically equivalent. Therefore it is sufficient to consider Presburger quantifiers which only use the inequality symbol  $\geq$ .

## 127:6 Decidability of Graph Neural Networks via Logical Characterizations

► **Remark 8.** We will make use of Presburger quantifiers that allow for rational coefficients of the form:

$$\tilde{\mathcal{P}}(x) := \kappa_0 + \sum_{i=1}^k \kappa_i \cdot \#_y[\varphi_i(x, y)] \circledast \lambda_0 + \sum_{i=1}^{\ell} \lambda_i \cdot \#_y[\psi_i(x, y)],$$

where each  $\kappa_i, \lambda_i \in \mathbb{Q}$ . This is a shorthand for the Presburger quantifier:

$$\mathcal{P}(x) := \sum_{i=1}^k (D\kappa_i) \cdot \#_y[\varphi_i(x, y)] + \sum_{i=1}^{\ell} (-D\lambda_i) \cdot \#_y[\psi_i(x, y)] \circledast D(\lambda_0 - \kappa_0),$$

where  $D$  is the least common multiplier of the denominators of the coefficients in  $\tilde{\mathcal{P}}(x)$ .

► **Definition 9.** We give the syntax of two-variable modal logic with Presburger quantifiers ( $\text{MP}^2$ ) over vocabulary  $\tau$ . Formulas will have exactly one free variable, denoted  $x$  below:

- $\top$  is an  $\text{MP}^2$  formula.
- for a unary predicate  $U \in \tau$ ,  $U(x)$  is an  $\text{MP}^2$  formula.
- if  $\varphi(x)$  is an  $\text{MP}^2$  formula, then so is  $\neg\varphi(x)$ .
- if  $\varphi_1(x)$  and  $\varphi_2(x)$  are  $\text{MP}^2$  formulas, then so is  $\varphi_1(x) \wedge \varphi_2(x)$ .
- if  $\{\varphi_i(x)\}_{1 \leq i \leq k}$  is a set of  $\text{MP}^2$  formulas and  $\{\epsilon_i(x, y)\}_{1 \leq i \leq k}$  is a set of guard atoms, of form  $E(x, y)$ ,  $E(y, x)$ , or  $\top$ , then  $\left(\sum_{i=1}^k \lambda_i \cdot \#_y[\epsilon_i(x, y) \wedge \varphi_i(y)] \circledast \delta\right)$  is also an  $\text{MP}^2$  formula.  $\{\epsilon_i(x, y)\}_{1 \leq i \leq k}$  are the guards of the formula. Consistent with the restriction we announced on the logic, we consider the result as a formula with free variable  $x$ : if all  $\epsilon_i$  are  $\top$  it returns either every node or no node.

The semantics of the Boolean connectives is as usual, while the semantics of the Presburger quantifiers is given by Definition 6.

An  $\text{MP}^2$  formula  $\varphi(x)$  is an  $n$ -formula if its vocabulary consists of  $n$  unary predicates. We use abbreviations  $\vee$  and  $\rightarrow$  as usual. Note that the guarded universal quantifier  $\forall y E(x, y) \rightarrow \varphi(y)$  can be expressed as  $(1 \cdot \#_y[E(x, y) \wedge \neg\varphi(y)] = 0)$ , and the guarded existential quantifier  $\exists y E(x, y) \wedge \varphi(y)$  can be expressed as  $(1 \cdot \#_y[E(x, y) \wedge \varphi(y)] \geq 1)$ .

The logic  $\text{MP}^2$  combines Presburger arithmetic and quantification over the model. Thus one might worry that it has an undecidable satisfiability problem. And indeed, we will show this: see Theorem 29. An idea to gain decidability is to impose that the quantification is *guarded* – again, the underlying idea is from [3]. The logic  $\mathcal{L}\text{-MP}^2$  (or “local  $\text{MP}^2$ ”) is obtained by excluding  $\top$  as a guard. Analogously to what we did with GNNs, we use  $\mathcal{L}$  to indicate that quantification is “local”.

The logic  $\mathcal{L}\text{-MP}^2$  is contained in the following logic, defined in [3]:

► **Definition 10.** The syntax of the guarded fragment of two-variable logic with Presburger quantifiers ( $\text{GP}^2$ ) over colored graph vocabulary  $\tau$  starts with arbitrary atoms over the vocabulary, with the usual connective closure and the following rules for quantifiers:

- if  $\varphi(x)$  is a  $\text{GP}^2$  formula, then so are  $\forall x \epsilon(x) \rightarrow \varphi(x)$  and  $\exists x \epsilon(x) \wedge \varphi(x)$ , where  $\epsilon$  is either  $U(x)$  or  $x = x$  for some unary predicate  $U \in \tau$ .
- if  $\varphi(x, y)$  is a  $\text{GP}^2$  formula, then so are  $\forall x \epsilon(x, y) \rightarrow \varphi(x, y)$  and  $\exists x \epsilon(x, y) \wedge \varphi(x, y)$ , where  $\epsilon(x, y)$  is one of  $E(x, y)$  or  $E(y, x)$ .
- if  $\{\varphi_i(x, y)\}_{1 \leq i \leq k}$  is a set of  $\text{GP}^2$  formulas and  $\{\epsilon_i(x, y)\}_{1 \leq i \leq k}$  is a set of formulas, each of form  $E(x, y)$  or  $E(y, x)$ , then  $\left(\sum_{i=1}^k \lambda_i \cdot \#_y[\epsilon_i(x, y) \wedge \varphi_i(x, y)] \circledast \delta\right)$  is also a  $\text{GP}^2$  formula.

The main difference between the logic  $\mathcal{L}\text{-MP}^2$  and the logic above is that the former is “modal”, restricting to one-variable formulas, and allowing two variables only in the guards. While in the logic above we can build up more interesting two variable formulas, for example conjoining two guards.

We will make use of the following prior decidability result:

► **Theorem 11** ([3], Theorem 10). *The finite satisfiability problem of  $\text{GP}^2$  is decidable.*

From this we easily derive the decidability of  $\mathcal{L}\text{-MP}^2$ :

► **Corollary 12.** *The finite satisfiability problem of  $\mathcal{L}\text{-MP}^2$  is decidable.*

**Notions of expressiveness for GNNs and  $\text{MP}^2$  Formulas.** Recalling that we have a node-to-Boolean semantics available for both logical formulas and GNNs (via thresholding), we use the term *n-specification* for either a *n*-GNN or a *n*- $\text{MP}^2$  formula.

► **Definition 13.** *If  $S_1, S_2$  are *n*-GNNs, they are said to be equivalent if they accept the same nodes within *n*-graphs. If  $S_1$  is a GNN and  $S_2$  a node formula in some logic, we say  $S_1$  and  $S_2$  are equivalent if for every *n*-graph  $\mathcal{G}$  and vertex  $v \in V$ ,  $S_1$  accepts  $\langle \mathcal{G}, v \rangle$  if and only if  $\mathcal{G}, v$  satisfies  $S_2$ .*

The notions of two languages of specifications being equally expressive, or equally expressive over undirected graphs, is defined in the obvious way:

**Verification Problems for GNNs.** We focus on two verification problems. The first is the most obvious analog of satisfiability for GNNs, whether it accepts some node of some graph:

► **Definition 14.** *For an *n*-GNN  $\mathcal{A}$ , we say that  $\mathcal{A}$  is satisfiable, if there exist an *n*-graph  $\mathcal{G}$  and a vertex  $v \in V$ , such that  $\mathcal{A}$  accepts  $\langle \mathcal{G}, v \rangle$ .*

We will also consider a variation of the problem which asks whether a GNN accepts every node of some graph:

► **Definition 15.** *For an *n*-GNN  $\mathcal{A}$ , we say that  $\mathcal{A}$  is universally satisfiable, if there exist an *n*-graph  $\mathcal{G}$ , such that for every vertex  $v \in V$ ,  $\mathcal{A}$  accepts  $\langle \mathcal{G}, v \rangle$ .*

Two GNNs  $\mathcal{A}$  and  $\mathcal{B}$  are equivalent if they accept the same tuples. Note that, like satisfiability and unlike universal satisfiability, this does not require a quantifier alternation. For brevity we will not state results for equivalence, but *it can easily be seen that both our positive and negative results on satisfiability also apply to equivalence.*

### 3 Characterization and decidability of GNNs with eventually constant activation functions

In this section, we only consider GNNs with eventually constant activations. In Section 3.1, we establish a key tool to analyzing these GNNs: we show that the set of possible activation values is finite, and one can compute an overapproximation of this set. We use this for two purposes. First we give a decidability result for GNNs with eventually constant activations and only local aggregation, and then we show that even with global aggregation we get an equivalence of the GNNs in expressiveness with a logic.

In Section 3.2, we show that the finite satisfiability of  $\text{MP}^2$  is undecidable. Using the expressiveness characterization, this will imply that satisfiability problems for global GNNs are undecidable. These results were presented for GNNs and logics on directed graphs. In Section 3.3 we use the logical characterizations to show that they also apply to the standard setting for GNNs of undirected graphs.

### 3.1 Decidability of satisfiability problems for GNNs with eventually constant functions, via logic

We now come to one of the crucial definitions in the paper, the spectrum of a GNN.

► **Definition 16.** For a BGC-GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ , the  $\ell$ -spectrum of  $\mathcal{A}$ , denoted by  $\mathcal{S}^\ell$ , is the set  $\{\xi^\ell(v) \mid \text{for every } n\text{-graph } \mathcal{G} \text{ and vertex } v \in V\}$ .

That is, the  $\ell$ -spectrum is the range of the feature vectors computed at layer  $\ell$ , as we range over all input graphs and nodes. We show that the spectrum is actually finite, and a finite superset is computable:

► **Theorem 17.** For every BGC-GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ , the  $\ell$ -spectrum of  $\mathcal{A}$  is finite. We can compute a finite superset of the  $\ell$ -spectrum from the specification of  $\mathcal{A}$ .

We give some intuition for the proof. Our effective overapproximation of the spectrum will simulate the computation of the GNN, and will be defined inductively on the layers. Recall that a BLC-GNN is given by dimensions  $\{d_\ell\}_{0 \leq \ell \leq L}$ , activation functions  $\{f^\ell\}_{1 \leq \ell \leq L}$ , coefficient matrices for transforming the prior node value  $\{C^\ell\}_{1 \leq \ell \leq L}$ , coefficient matrices for local aggregation  $\{A_x^\ell\}_{\substack{1 \leq \ell \leq L \\ x \in \{\text{out}, \text{in}\}}}$ , coefficient matrices for global readout  $\{R^\ell\}_{1 \leq \ell \leq L}$ , and bias vectors  $\{b^\ell\}_{1 \leq \ell \leq L}$ .

► **Definition 18.** For a BLC-GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ , the set  $\uparrow \mathcal{S}^\ell$  is defined as follows:

$$\begin{aligned} \uparrow \mathcal{S}^0 &:= \{0, 1\}^{d_0} \\ \uparrow \mathcal{S}_s^\ell &:= \left\{ f^\ell \left( C^\ell s + \sum_{x \in \{\text{out}, \text{in}\}} A_x^\ell \sum_{s' \in \uparrow \mathcal{S}^{\ell-1}} s' n_x^{A, s'} + R^\ell \sum_{s' \in \uparrow \mathcal{S}^{\ell-1}} s' n^{R, s'} + b^\ell \right) \mid n_x^{A, s'}, n^{R, s'} \in \mathbb{N} \right\} \\ \uparrow \mathcal{S}^\ell &:= \bigcup_{s \in \uparrow \mathcal{S}^{\ell-1}} \uparrow \mathcal{S}_s^\ell \end{aligned}$$

We show that the set  $\uparrow \mathcal{S}^\ell$  overapproximates the  $\ell$ -spectrum:

► **Lemma 19.** For every  $n$ -BLC-GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ , for every  $n$ -graph  $\mathcal{G}$  and vertex  $v \in V$ , there exists  $s \in \uparrow \mathcal{S}^\ell$ , such that  $\xi^\ell(v) = s$ .

It is quite straightforward to see that every element of the spectrum is captured. It is an overapproximation because different integers that we sum in an inductive step may not be realized in the same graph.

We can show by induction on the number of the layers that the set is finite – regardless of computability of the activation functions!

► **Lemma 20.** For every  $n$ -BLC-GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ ,  $\uparrow \mathcal{S}^\ell$  has finite size and can be computed.

In the inductive step, we have a finite set of rationals, thus some fixed precision. We take some integer linear combinations and we will obtain an infinite set of values, but only finitely many between the left and right thresholds of the eventually constant activations. Thus when we apply the activation functions to these values, we will get a finite set of rational values – since the activation functions map rationals to rationals.



► **Remark 21.** The restriction to rational coefficients is crucial in the argument. Consider the following 1-layer 1- $\mathcal{BLC}$ TrReLU-GNN. The dimensions are  $d_0 = d_1 = 1$ ; the coefficient matrix  $C^1$  is a zero matrix;  $(A_{\text{out}}^1)_{1,1} = \sqrt{2}$ ;  $(A_{\text{in}}^1)_{1,1} = -1$ ; the bias vector  $b^1$  is a zero vector. It is not difficult to see that its 1-spectrum is  $\{\text{TrReLU}(\sqrt{2}k_1 - k_2) \mid k_1, k_2 \in \mathbb{N}\}$ , whose size is infinite since  $\sqrt{2}$  is irrational.

► **Remark 22.** Even simple GNNs may have exponential size spectra. For example, let  $\mathcal{A}_k$  be a 1-layer 1- $\mathcal{BLC}$ TrReLU-GNN defined as follows: the dimensions are  $d_0 = d_1 = 1$ ; the coefficient matrices  $C^1$  and  $A_{\text{in}}^1$  are zero matrices;  $(A_{\text{out}}^1)_{1,1} = k^{-1}$ ; the bias vector  $b^1$  is a zero vector. By definition, its 1-spectrum is  $\{ik^{-1} \mid i \in [0, k]\}$ , whose size is  $k + 1$ . But the description of  $\mathcal{A}_k$  is only linear in  $\log k$ .

For GNNs with truncated ReLU and only local aggregation, there is a matching upper bound, as discussed after Theorem 25.

We now give several applications of the spectrum result. First we can use the finiteness of the spectrum to get a characterization of the expressiveness of  $\mathcal{BGC}$ -GNN and logic:

► **Theorem 23.** *For every  $n$ - $\mathcal{BGC}$ -GNN  $\mathcal{A}$ , there exists an  $n$ -MP<sup>2</sup> formula  $\Psi_{\mathcal{A}}(x)$ , effectively computable from the description of  $\mathcal{A}$ , such that  $\mathcal{A}$  and  $\Psi_{\mathcal{A}}(x)$  are equivalent. In the case we start with an  $n$ - $\mathcal{BLC}$ -GNN, the formula we obtain is in  $n$ - $\mathcal{L}$ -MP<sup>2</sup>.*

This expressiveness equivalence will be useful in getting further decidability results, as well as separations in expressiveness, for GNNs. The idea of the proof of the theorem is that we have only finitely many elements in the overapproximation set to worry about, so we can fix each in turn and write a formula for each.

Recall that finite satisfiability of  $\mathcal{L}$ -MP<sup>2</sup> is decidable by Corollary 12. Combining this with Theorem 23 we get decidability of satisfiability for  $\mathcal{BLC}$ -GNN:

► **Theorem 24.** *The satisfiability problem for  $\mathcal{BLC}$ -GNNs is decidable.*

A more realistic analysis of complexity requires stronger assumptions on the activation functions. For now we note only one special case, where everything is a truncated ReLU:

► **Theorem 25.** *For  $\mathcal{BLC}$ -GNNs with truncated ReLU activations, the satisfiability problem is PSPACE-complete. It is NP-complete when the number of layers is fixed.*

We briefly discuss the PSPACE upper bound argument. We can show that for an arbitrary input graph, there are only exponentially many activation values, each representable with a polynomial number of bits. We also show, via an “unravelling construction”, a common technique used in analysis of modal and guarded logics [12, 7], that a satisfying model can be taken to be a tree of polynomial depth and branching. These two facts immediately give an elementary bound, since we could guess the tree and the activation values. We can improve to PSPACE by exploring a satisfying tree-like model on-the-fly: again, this is in line with the PSPACE algorithm for modal logic [12].

The PSPACE lower bound is established by embedding the description logic  $\mathcal{ALC}$  into  $\mathcal{L}$ -MP<sup>2</sup>. PSPACE-hardness will follow from this, since concept satisfiability problem of  $\mathcal{ALC}$  with one role is PSPACE-hard [15]. The NP upper bound will use the same on-the-fly algorithm as in the PSPACE case, just observing that for fixed depth it can be implemented in NP. A direct encoding of SAT gives the lower bound.

The following converse to Theorem 23 shows that the logic is equally expressive as the GNN model:

## 127:10 Decidability of Graph Neural Networks via Logical Characterizations

► **Theorem 26.** *For every  $n$ - $\text{MP}^2$  formula  $\Psi(x)$ , there exists an  $n$ - $\text{BGTrReLU-GNN}$   $\mathcal{A}_\Psi$ , such that  $\Psi(x)$  and  $\mathcal{A}_\Psi$  are equivalent. If we start with an  $n$ - $\mathcal{L}\text{-MP}^2$  formula, we obtain an  $n$ - $\mathcal{BLTrReLU-GNN}$ .*

The idea of the proof is induction on the formula structure. For each subformula there will be an entry of a feature vector for the GNN which represents the subformula, in the sense that – for the final layer – its value is 1 if the subformula holds, or 0 otherwise. We will have an entry for each subformula at every iteration, but as we progress to later layers of the GNN, more of these entries will be correct with respect to the corresponding subformula. In an inductive case for a Presburger quantifier that uses some coefficients  $\lambda_i$ , the corresponding matrix will be multiplying certain quantifies by  $\lambda_i$ . Note that this translation is polynomial time, thus the size of the corresponding GNN is polynomial in the formula.

Putting together the two translation results, we have:

► **Corollary 27.** *The logic  $\text{MP}^2$  and  $\text{BGC-GNNs}$  are expressively equivalent, as are  $\mathcal{L}\text{-MP}^2$  and  $\mathcal{BLC-GNNs}$ .*

The translations also tell us that *the expressiveness of GNNs with truncated ReLU is the same as that of GNNs with arbitrary eventually constant activations* – provided we use the Boolean semantics based on thresholds.

Recall from Corollary 12 that finite satisfiability for the richer logic  $\text{GP}^2$ , allowing unguarded unary quantification and containing  $\mathcal{L}\text{-MP}^2$ , is decidable. Using this and the expressiveness characterization gives decidability of universal satisfiability for these GNNs:

► **Theorem 28.** *The universal satisfiability problem of  $\mathcal{BLC-GNNs}$  is decidable.*

### 3.2 Undecidability of $\text{MP}^2$ , and of GNNs with truncated ReLU and global readout

Note that we claimed that the spectrum is finite for GNNs with eventually constant activations, even when they have global readout. And we could compute a finite overapproximation of the spectrum. But in our decidability argument for  $\mathcal{BLC-GNN}$ , we required further the ability to decide membership in the spectrum for any fixed rational, and for this we utilized decidability of the logic. So what happens to decidability of the GNNs – or the corresponding logic – when global readout is allowed? We show undecidability of finite satisfiability for the logic  $\text{MP}^2$ , and of the corresponding GNN satisfiability problem. First for the logic:

► **Theorem 29.** *The finite satisfiability problem of  $\text{MP}^2$  is undecidable.*

For the proof we apply an approach based on ideas in [3], using a reduction from Hilbert’s tenth problem.

► **Definition 30.** *A simple equation system  $\varepsilon$  (with  $n$  variables and  $m$  equations) is a set of  $m$  equations of one of the forms  $v_{i_1} = 1$ ,  $v_{i_1} = v_{i_2} + v_{i_3}$ , or  $v_{i_1} = v_{i_2} \cdot v_{i_3}$ , where  $1 \leq i_1, i_2, i_3 \leq n$ . We say the system  $\varepsilon$  is solvable if it has a solution in  $\mathbb{N}$ .*

► **Lemma 31.** *For every simple equation system  $\varepsilon$  with  $n$  variables and  $m$  equations, there exists an  $(n + m)$ - $\text{MP}^2$  formula  $\Psi_\varepsilon(x)$  such that  $\varepsilon$  has a solution in  $\mathbb{N}$  if and only if  $\Psi_\varepsilon(x)$  is finitely satisfiable.*

Since the solvability (over  $\mathbb{N}$ ) of simple equation systems is undecidable, Theorem 29 follows. From the theorem and Corollary 27 we obtain undecidability of static analysis for GNNs with global readout:

► **Theorem 32.** *The satisfiability problem of BGTReLU-GNNs is undecidable.*

We give the reduction used in Lemma 31, leaving the verification for the reader. The vocabulary of the constructed formulas  $\Psi_\epsilon(x)$ , where  $\epsilon$  is the simple equation system, consists of unary predicates  $P_i$  and  $U_j$ , where  $1 \leq i \leq m$  and  $1 \leq j \leq n$ . For  $1 \leq i \leq m$ , we define  $\varphi_i(x)$  depending on the  $i^{\text{th}}$  equation in  $\epsilon$ .

- If the equation is  $v_j = 1$ , then  $\varphi_i(x) := (\#_y[P_i(y) \wedge U_j(y)] = 1)$ .
- If the equation is  $v_{j_1} = v_{j_2} + v_{j_3}$ , then  $\varphi_i(x) := (\#_y[\psi_i(y)] - \#_y[\top] = 0)$  where

$$\begin{aligned} \psi_i(y) := & (P_i(y) \wedge (U_{j_2}(y) \vee U_{j_3}(y)) \rightarrow (\#_x[E(y, x) \wedge P_i(x) \wedge U_{j_1}(x)] = 1)) \wedge \\ & (P_i(y) \wedge U_{j_1}(y) \rightarrow (\#_x[E(x, y) \wedge P_i(x) \wedge (U_{j_2}(x) \vee U_{j_3}(x))] = 1)). \end{aligned}$$

- If the equation is  $v_{j_1} = v_{j_2} \cdot v_{j_3}$ , then  $\varphi_i(x) := (\#_y[\psi_i(y)] - \#_y[\top] = 0)$  where

$$\begin{aligned} \psi_i(y) := & (P_i(y) \wedge U_{j_2}(y) \rightarrow (\#_x[E(y, x) \wedge P_i(x) \wedge U_{j_1}(x)] - \#_x[P_i(x) \wedge U_{j_3}(x)] = 0)) \wedge \\ & (P_i(y) \wedge U_{j_1}(y) \rightarrow (\#_x[E(x, y) \wedge P_i(x) \wedge U_{j_2}(x)] = 1)). \end{aligned}$$

We now define  $\Psi_\epsilon(x)$ :

$$\begin{aligned} \psi^{\text{disj}}(x) & := \bigwedge_{1 \leq i_1 < i_2 \leq m} (\#_y[P_{i_1}(y) \wedge P_{i_2}(y)] = 0) \wedge \bigwedge_{1 \leq j_1 < j_2 \leq n} (\#_y[U_{j_1}(y) \wedge U_{j_2}(y)] = 0) \\ \psi^{\text{eq}}(x) & := \bigwedge_{\substack{1 \leq i_1 < i_2 \leq m \\ 1 \leq j \leq n}} (\#_y[P_{i_1}(y) \wedge U_j(y)] - \#_y[P_{i_2}(y) \wedge U_j(y)] = 0) \\ \Psi_\epsilon(x) & := \psi^{\text{disj}}(x) \wedge \psi^{\text{eq}}(x) \wedge \bigwedge_{1 \leq i \leq m} \varphi_i(x). \end{aligned}$$

Using a similar reduction, we obtain undecidability for universal satisfiability:

► **Theorem 33.** *The universal satisfiability problem of BGTReLU-GNNs is undecidable.*

### 3.3 Variations for the undirected case

Thus far we have been dealing with both logics and GNNs that work over directed graphs. We now show that all of the prior results apply to undirected graphs, the standard setting for GNNs.

We can enforce undirectedness within the larger decidable logic  $\text{GP}^2$  to obtain decidability:

► **Corollary 34.** *The finite satisfiability problem of  $\mathcal{L}\text{-MP}^2$  over undirected graphs is decidable.*

By reducing to decidability in the logic  $\mathcal{L}\text{-MP}^2$ , we can show that the satisfiability problem for GNNs on undirected graphs – that is, the standard notion of GNN – is decidable.

► **Theorem 35.** *The satisfiability problem of BLC-GNNs over undirected graphs is decidable.*

► **Theorem 36.** *The universal satisfiability problem of BLC-GNNs over undirected graphs is decidable.*

We can also revise our undecidability results for global GNNs to the undirected case, thus giving undecidability for the usual notion of GNN with global readout. This is done with the same reduction from solvability of simple equation systems to the finite satisfiability of  $\text{MP}^2$ , which we can show works over undirected graphs.

- **Theorem 37.** *The finite satisfiability problem of  $\text{MP}^2$  over undirected graphs is undecidable.*
- **Theorem 38.** *The satisfiability problem of  $\mathcal{BGT}\text{ReLU}$ -GNNs over undirected graphs is undecidable.*
- **Theorem 39.** *The universal satisfiability problem of  $\mathcal{BGT}\text{ReLU}$ -GNNs over undirected graphs is undecidable.*

## 4

 GNNs with unbounded activation functions

In this section, we consider GNNs with unbounded activations, such as the standard ReLU. Since we have already shown that global aggregation leads to undecidability even in the bounded case, in this section *we will only deal with GNNs having only local aggregation*. In Section 4.1 we show that the universal satisfiability problem of  $\mathcal{BL}\text{ReLU}$ -GNN is undecidable, a contrast to the case with eventually constant activation functions. In the process, we introduce a logic that also helps with understanding expressiveness of this class of GNNs.

In Section 4.2, we turn to the satisfiability problem, and give a partial positive result about decidability. Here we will not use the logic directly, but rather use components from decidability proofs for Presburger logics [3]. We will use the idea of representing the possible values of activations which was also used in the case of decidability for eventually constant activations. But in this case we will be representing an infinite set of values, using Presburger formulas.

### 4.1 (Un)decidability of GNNs with unbounded activation functions

We prove the undecidability of the universal satisfiability problem of  $\mathcal{BL}\text{ReLU}$ -GNN. Here we will use logic again. We will not obtain an expressiveness characterization, but merely a logic that embeds in  $\mathcal{BL}\text{ReLU}$ -GNNs: local two-variable modal logic with two-hop Presburger quantifiers ( $\mathcal{L}\text{-M2P}^2$ ), which is the extension of  $\text{MP}^2$  where the guards are conjunctions of at most two binary predicates.

► **Definition 40.** *The syntax of local two-variable modal logic with two-hop Presburger quantifiers ( $\mathcal{L}\text{-M2P}^2$ ) over vocabulary  $\tau$  is defined inductively:*

- $\top$  is a  $\mathcal{L}\text{-M2P}^2$  formula.
- for a unary predicate  $U \in \tau$ ,  $U(x)$  is a  $\mathcal{L}\text{-M2P}^2$  formula.
- if  $\varphi(x)$  is a  $\mathcal{L}\text{-M2P}^2$  formula, then so is  $\neg\varphi(x)$ .
- if  $\varphi_1(x)$  and  $\varphi_2(x)$  are  $\mathcal{L}\text{-M2P}^2$  formulas, then so is  $\varphi_1(x) \wedge \varphi_2(x)$ .
- if  $\{\varphi_i(x)\}_{1 \leq i \leq k} \cup \{\varphi'_i(x)\}_{1 \leq i \leq k'}$  is a set of  $\mathcal{L}\text{-M2P}^2$  formulas,  $\{\epsilon_i(x, z, y)\}_{1 \leq i \leq k}$  is a set of guard formulas, each of form  $E(x, z) \wedge E(z, y)$ ,  $E(x, z) \wedge E(y, z)$ ,  $E(z, x) \wedge E(z, y)$ , or  $E(z, x) \wedge E(y, z)$ , and  $\{\epsilon'_i(x, y)\}_{1 \leq i \leq k'}$  is another set of guard formulas, each of form  $E(x, y)$  or  $E(y, x)$ , then

$$\left( \sum_{i=1}^k \lambda_i \cdot \#_{z,y}[\epsilon_i(x, z, y) \wedge \varphi_i(y)] + \sum_{i=1}^{k'} \lambda'_i \cdot \#_y[\epsilon'_i(x, y) \wedge \varphi'_i(y)] \otimes \delta \right)$$

is also a  $\mathcal{L}\text{-M2P}^2$  formula. The numbers  $\delta, \lambda_i, \lambda'_i$ , and the comparison  $\otimes$  are as in the standard Presburger quantifier definition.

The idea is that we can still count a linear combination of cardinalities of the number of nodes satisfying a given lower-level formula that are one-hop away from the current node – as in  $\mathcal{L}\text{-MP}^2$ . Optionally, we can add on a linear combination of the number of two-hop paths that lead to a node satisfying other lower-level formulas.

The semantics of the formulas is given inductively, with the only step that is different from the usual cases being for the quantification, which is the obvious one. We call these *two-hop Presburger quantifiers*. We can apply a similar proof technique as in Theorem 26 to show that the  $\mathcal{L}$ -M2P<sup>2</sup> are expressible using  $\mathcal{B}\mathcal{L}\text{ReLU}$ -GNNs.

► **Theorem 41.** *For every  $n$ - $\mathcal{L}$ -M2P<sup>2</sup> formula  $\Psi(x)$ , there exists an  $n$ - $\mathcal{B}\mathcal{L}\text{ReLU}$ -GNN  $\mathcal{A}_\Psi$ , such that  $\Psi(x)$  and  $\mathcal{A}_\Psi$  are equivalent.*

Note that we do not claim an expressive equivalence here. Nevertheless this containment of the logic in the GNN class is useful, since we can show undecidability of the logic by reduction from the halting problem of two-counter machines, which is known to be undecidable [13].

► **Definition 42.** *A two-counter machine  $\mathcal{M}$  is a finite list  $d_1 \dots d_n$  of instructions having one of the forms **INC** ( $c_i$ ), **IF** ( $c_i = 0$ ) **GOTO** ( $j$ ), or **HALT**, where  $i \in \{0, 1\}$  and  $1 \leq j \leq n$ .*

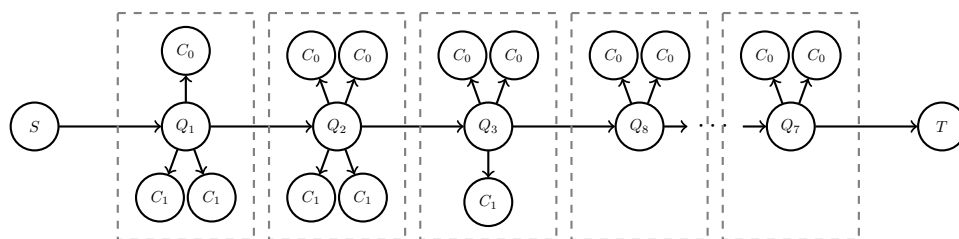
*A configuration is a tuple  $\langle q, c_0, c_1 \rangle$ , where  $1 \leq q \leq n$  and  $c_0, c_1 \in \mathbb{N}$ . We say  $\langle q', c'_0, c'_1 \rangle$  is the successor configuration of  $\langle q, c_0, c_1 \rangle$  if, letting  $d_q$  be the  $q^{\text{th}}$  instruction of the machine:*

- *If  $d_q$  is **INC** ( $c_i$ ), then  $q' = q + 1$ ,  $c'_i = c_i + 1$ , and  $c'_{1-i} = c_{1-i}$ .*
- *If  $d_q$  is **IF** ( $c_i = 0$ ) **GOTO** ( $j$ ), if  $c_i = 0$ , then  $q' = j$ ,  $c'_0 = c_0$ , and  $c'_1 = c_1$ ; otherwise,  $q' = q + 1$ ,  $c'_i = c_i - 1$ , and  $c'_{1-i} = c_{1-i}$ .*

*Note that if  $d_q$  is **HALT**, there is no successor. This configuration is called a halting configuration.*

*The computation of the machine is a (possibly infinite) sequence of configurations where the first is  $\langle 1, 0, 0 \rangle$ , consecutive pairs are in the successor relationship above, the last configuration is a halting configuration. The machine halts if its computation is a finite sequence.*

The reduction is by encoding the computation of a two-counter machine into the graph directly. We have illustrated it in Figure 1. Each configuration is encoded as a height 1 tree, which is denoted by a dashed box. Its line number is represented by the unary predicate  $Q_i$  realized by the root vertex, and the values of the counters are represented by the number of “labeled leaves” – those with predicate  $C_0$  or  $C_1$  being true. There are edges connected to the roots of each configuration, which encode the computation sequence. Then it is possible to assert the (in)equality between the number of leaves of some root and the root of the successor tree, which encodes the condition of a valid transition.



■ **Figure 1** An example of the encoding of the computation of the two-counter machines to directed graphs.

► **Lemma 43.** *For every two-counter machine  $\mathcal{M}$  with  $n$  instructions, there exists an  $(n + 5)$ - $\mathcal{L}$ -M2P<sup>2</sup> formula  $\Psi_{\mathcal{M}}(x)$  such that  $\mathcal{M}$  halts if and only if  $\forall x \Psi_{\mathcal{M}}(x)$  is finitely satisfiable.*

Since the halting problem of two-counter machines is undecidable, and  $\mathcal{L}$ -M2P<sup>2</sup> formulas can be translated to  $\mathcal{B}\mathcal{L}\text{ReLU}$ -GNNs, we obtain the undecidability of the universal satisfiability problem of  $\mathcal{B}\mathcal{L}\text{ReLU}$ -GNN, by reduction from  $\mathcal{L}$ -M2P<sup>2</sup>.

## 127:14 Decidability of Graph Neural Networks via Logical Characterizations

► **Theorem 44.** *The universal satisfiability problem of  $\mathcal{BLReLU}$ -GNNs is undecidable.*

We will later show that this holds also for undirected graphs: see Theorem 52 below.

We can also use the logic to get an expressiveness separation for GNNs: By Theorem 41 to show that  $\mathcal{BLReLU}$ -GNNs can do more than  $\mathcal{BLC}$ -GNNs, it is sufficient to show that there is a  $\mathcal{L}$ -M2P<sup>2</sup> formula that is not given by a  $\mathcal{BLC}$ -GNN:

► **Lemma 45.**  *$\mathcal{L}$ -M2P<sup>2</sup> is strictly more expressive than  $\mathcal{BLC}$ -GNN.*

The following results are direct consequences of the lemma above and the logical characterization in the prior section:

► **Corollary 46.**  *$\mathcal{L}$ -M2P<sup>2</sup> is strictly more expressive than  $\mathcal{L}$ -MP<sup>2</sup>.*

► **Corollary 47.**  *$\mathcal{BLReLU}$ -GNN is strictly more expressive than  $\mathcal{BLC}$ -GNN.*

We comment on the proof of Lemma 45. We claim that the property “the number of two-hop paths from the vertex  $v$  to the green vertices is the same as the number of two-hop paths from  $v$  to the blue vertices” gives the separation. It is easy to express in the two-hop logic. To show that no  $\mathcal{BLC}$ -GNN can express it, we construct a sequence of pairs of graphs, each with a special node, such that the property holds in the special node of the first graph and fails in the special node of the second, while for every  $\mathcal{BLC}$ -GNN  $\mathcal{A}$ , for any sufficiently large pairs of graphs in this sequence, the special nodes are indistinguishable by  $\mathcal{A}$ . The graphs are as follows:

► **Definition 48.** *For  $n_1, n_2 \in \mathbb{N}$ , the  $(n_1, n_2)$ -bipolar graph  $\langle V, E, U_1, U_2 \rangle$  is an undirected 2-graph defined as follows.*

$$U_1 := \{v_{1,i} \mid 1 \leq i \leq n_1\}$$

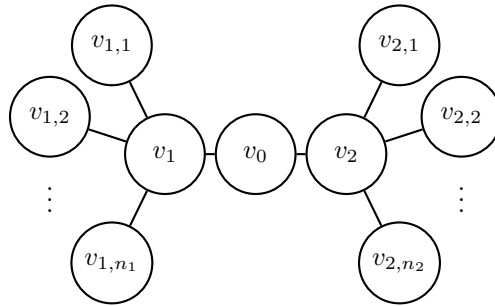
$$U_2 := \{v_{2,i} \mid 1 \leq i \leq n_2\}$$

$$V := U_1 \cup U_2 \cup \{v_0, v_1, v_2\}$$

$$\tilde{E} := \{(v_0, v_1), (v_0, v_2)\} \cup \{(v_1, v_{1,i}) \mid 1 \leq i \leq n_1\} \cup \{(v_2, v_{2,i}) \mid 1 \leq i \leq n_2\}$$

$$E := \tilde{E} \cup \left\{ (u, v) \mid (v, u) \in \tilde{E} \right\}$$

See Figure 2.



■ **Figure 2**  $(n_1, n_2)$ -bipolar graph.

It is easy to see that these graphs with the distinguished nodes  $v_0$  and  $v'_0$  disagree on the property, and we can also show that eventually no  $\mathcal{BLC}$ -GNN can distinguish them:

► **Lemma 49.** *For each 2-BLC-GNN  $\mathcal{A}$ , there exist a threshold  $n_{\mathcal{A}} \in \mathbb{N}$ , such that for every  $n_1, n_2 \geq n_{\mathcal{A}}$ , the following properties hold. Let  $\mathcal{G}$  be the  $(n_{\mathcal{A}}, n_{\mathcal{A}})$ -bipolar graph and  $\mathcal{G}'$  be the  $(n_1, n_2)$ -bipolar graph. For every  $0 \leq \ell \leq L$ ,*

- $\xi_{\mathcal{G}}^{\ell}(v_0) = \xi_{\mathcal{G}'}^{\ell}(v'_0)$ ,  $\xi_{\mathcal{G}}^{\ell}(v_1) = \xi_{\mathcal{G}'}^{\ell}(v'_1)$ , and  $\xi_{\mathcal{G}}^{\ell}(v_2) = \xi_{\mathcal{G}'}^{\ell}(v'_2)$ .
- for  $1 \leq i \leq n_{\mathcal{A}}$  and  $1 \leq j \leq n_1$ ,  $\xi_{\mathcal{G}}^{\ell}(v_{1,1}) = \xi_{\mathcal{G}}^{\ell}(v_{1,i}) = \xi_{\mathcal{G}'}^{\ell}(v'_{1,j})$ .
- for  $1 \leq i \leq n_{\mathcal{A}}$  and  $1 \leq j \leq n_2$ ,  $\xi_{\mathcal{G}}^{\ell}(v_{2,1}) = \xi_{\mathcal{G}}^{\ell}(v_{2,i}) = \xi_{\mathcal{G}'}^{\ell}(v'_{2,j})$ .

Above  $\xi_{\mathcal{G}}^{\ell}$  refers to the  $\ell^{\text{th}}$  derived feature function of the GNN  $\mathcal{A}$  over the graph  $\mathcal{G}$ .

Thus far the results in this section are stated for directed graphs. We explain briefly why the undecidability and expressiveness separation results on GNNs with unbounded activation functions apply also to undirected graphs. For the expressiveness results, note that the graphs that we constructed in the proof of Lemma 45 are undirected. Hence the expressiveness gap between BCLReLU-GNN and BLC-GNN still exists for the undirected case.

► **Theorem 50.** *BCLReLU-GNN is strictly more expressive over undirected graphs than BLC-GNN.*

To obtain the undecidability of the universal satisfiability problem over undirected graphs of BCLReLU-GNN, we again reduce from two-counter machines, but now with a modification to guarantee the direction of the transition.

► **Lemma 51.** *For every two-counter machine  $\mathcal{M}$  with  $n$  instructions, there exists an  $(n+8)$ - $\mathcal{L}$ -M2P<sup>2</sup> formula  $\Psi_{\mathcal{M}}(x)$  such that  $\mathcal{M}$  halts if and only if  $\forall x \Psi_{\mathcal{M}}(x)$  is finitely satisfiable over undirected graphs.*

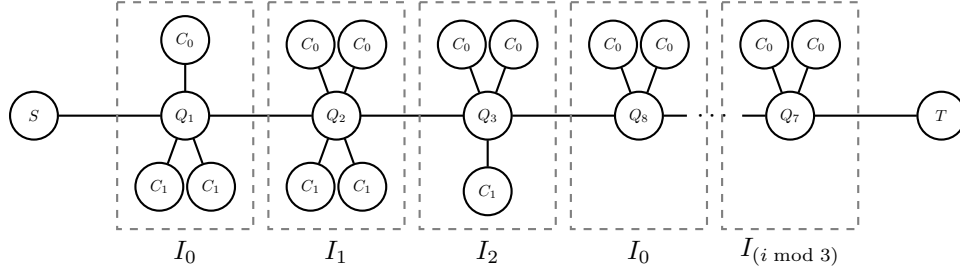
We cannot apply the exact encoding from Figure 1 and Lemma 43 here, because in that encoding we distinguished the predecessor and successor configurations by the direction of edges. Here, we sketch the trick that overcomes the lack of direction in the edges. We will utilize the predicates from the proof of Lemma 43: in particular we will have a predicate  $Q$  and an associated notion of  $Q$  vertex as in that proof.

We introduce three fresh unary predicates  $I_0$ ,  $I_1$ , and  $I_2$  to label the configuration's index modulo 3. We add an extra clause to the formula to guarantee that each element has exactly one of these three index labels. The elements in each 1-level tree will have the same index, in the sense of satisfying the same index predicates. Finally, for each  $Q$  vertex  $v$  with index  $i$ , there exists at most one  $Q$  vertex  $v'$  with index  $(i+1 \bmod 3)$ , such that  $v$  and  $v'$  are connected; there exists at most one  $Q$  vertex  $v''$  with index  $(i-1 \bmod 3)$ , such that  $v$  and  $v''$  are connected. Therefore we can modify the formula which identifies the successor and predecessor based on the index, rather than the direction of the edges, and show that the two-counter machine halts if and only if the modified formula is finitely satisfiable over undirected graphs.

► **Theorem 52.** *The universal satisfiability problem of BCLReLU-GNNs over undirected graphs is undecidable.*

## 4.2 Decidability of satisfiability for “modal” GNNs with unbounded activation functions

Thus the situation for universal satisfiability contrasts with the eventually constant case. What about the *satisfiability problem*? We do not know whether it is decidable for GNNs with piecewise linear activations, or even with just ReLU. We can see that even simple unbounded activation functions produce unbounded spectra, so the proof technique in the truncated case certainly will not work. For example, consider the following 1-layer 1-BCLReLU-GNN.



■ **Figure 3** An example of the encoding of the computation of a two-counter machines in undirected graphs.

The dimensions are  $d_0 = d_1 = 1$ ; the coefficient matrices  $C^1$  and  $A_{\text{in}}^1$  are zero matrices;  $(A_{\text{out}}^1)_{1,1} = 1$ ; the bias vector  $b^1$  is a zero vector. It is not difficult to see that the value of  $\xi_1^1(v)$  is the number of out-neighbors of  $v$ . Hence, the 1-spectrum of this GNN is the set of natural numbers.

We present a decidability result for the “modal version”: aggregation over nodes connected by outgoing edges only, within a directed graph:

► **Theorem 53.** *The satisfiability problem of OLPW-GNNs is decidable.*

Analogously to what we did in the eventually constant case, we describe all the possible values of a given activation function. Unlike in the eventually constant case, this will not be a finite set, but it will be *semi-linear*: that is, describable using a formula of Presburger arithmetic. We will first review the notion of semi-linear set that we use, where we modify the standard notion to deal with rational numbers. We then show that the set of all possible values output by a GNN is a semi-linear set.

For  $a_0 \in \mathbb{Q}^k$  and  $A = \{a_1, a_2, \dots, a_m\}$  a finite subset of  $\mathbb{Q}^k$ , we define:

$$\mathbb{N}\text{-Span}(a_0, A) := \left\{ a_0 + \sum_{1 \leq i \leq m} n_i a_i \mid n_i \in \mathbb{N} \right\}.$$

A set  $S \subseteq \mathbb{Q}^k$  is a *linear set*, if there is  $a_0 \in \mathbb{Q}^k$  and a finite set  $A \subseteq \mathbb{Q}^k$ , such that  $S$  is  $\mathbb{N}\text{-Span}(a_0, A)$ . The pair  $(a_0, A)$  is called *the basis of  $S$* . A *semi-linear set* is a finite union of linear sets. A basis of a semi-linear set  $\bigcup_{1 \leq i \leq k} \mathbb{N}\text{-Span}(a_0^k, A^k)$  is the set  $\{(a_0^1, A^1), (a_0^2, A^2), \dots, (a_0^k, A^k)\}$ .

For semi-linear sets  $S_1, S_2, S \subseteq \mathbb{Q}^k$ , we use the following operators:

$$T(S) := \{T(a) \mid a \in S\} \quad \text{where } T : \mathbb{Q}^k \rightarrow \mathbb{Q}^m \text{ is an affine transformation}$$

$$\text{KleeneStar}(S) := \left\{ \sum_{s \in S'} s \mid \text{For every finite multi-subset } S' \text{ of } S \right\}$$

We recall that in the context of integers, both operators are known to preserve semi-linearity and the basis of the resulting semi-linear set can be computed. See, e.g., [5, 10, 6]. The arguments adapt easily to our rational setting, thus *we assume below that we have an algorithm for pushing semi-linear representations through these operators*.

We consider piecewise linear functions, defined by a sequence  $((I_1, f_1), \dots, (I_p, f_p))$  where  $I_1 \cup \dots \cup I_p$  is a partition of  $\mathbb{Q}$  into  $p$  intervals and each  $f_i : \mathbb{Q} \rightarrow \mathbb{Q}$  is an affine function. The sequence  $((I_1, f_1), \dots, (I_p, f_p))$  defines a function where  $x$  is mapped to  $f_i(x)$  if  $x$  is in



the interval  $I_i$ . We apply a piecewise linear function on some fixed components of a vector, which is captured by the following notation. For a piecewise linear function  $f : \mathbb{Q} \rightarrow \mathbb{Q}$ , a rational vector  $a \in \mathbb{Q}^k$  and  $K \subseteq [1, k]$ , we write  $f_K(a)$  to denote the vector  $b \in \mathbb{Q}^k$  where  $b_i = f(a_i)$  for every  $i \in K$  and  $b_i = a_i$  for every  $i \notin K$ . In other words,  $f_K(a)$  only applies the function  $f$  on the components in  $K$  and the identity function on the components outside  $K$ . Similar to affine transformation and Kleene star, *piecewise linear functions also preserve semi-linearity and the basis of the resulting semi-linear sets can be computed*. Again, we can easily adapt the argument in [6] to our rational setting.

To prove Theorem 53, we need two more definitions. Let  $\mathcal{A}$  be a  $L$ -layer  $\mathcal{OLPW}$ -GNN. Let  $d_0, d_1, \dots, d_L$  be the dimension of the layers. We denote by  $\mathbb{Q}^{[d_0, d_1, \dots, d_{\ell-1}]}$  the Cartesian product  $\mathbb{Q}^{d_0} \times \mathbb{Q}^{d_1} \dots \times \mathbb{Q}^{d_{\ell-1}}$ . Given an element  $m$  of this product, the  $i^{\text{th}}$  component of  $m$ , denoted by  $m[i]$ , is the projection of  $m$  to  $\mathbb{Q}^{d_i}$ .

For a graph  $\mathcal{G}$ , vertex  $v \in V$ , and  $0 \leq \ell \leq L$ , the  $\ell$ -history of  $v$  in  $\mathcal{G}$  (w.r.t.  $\mathcal{A}$ ), denoted by  $\text{hist}_{\mathcal{G}}^{\ell}(v) \in \mathbb{Q}^{[d_0, d_1, \dots, d_{\ell}]}$ , is the tuple that collects the first  $(\ell + 1)$  feature vectors of  $v$ . Formally, for  $0 \leq i \leq \ell$ ,  $(\text{hist}_{\mathcal{G}}^{\ell}(v))[i] = \xi_{\mathcal{G}}^i(v)$ . When the graph  $\mathcal{G}$  is clear from the context, we omit it and simply write  $\text{hist}^{\ell}(v)$ . The  $\ell$ -history-space of  $\mathcal{A}$  is the set of all possible histories.

We now state our representation theorem, which immediately implies Theorem 53:

► **Theorem 54.** *For every  $\mathcal{OLPW}$ -GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ , the  $\ell$ -history-space is semi-linear, and its basis can be effectively computed.*

We contrast the theorem with Theorem 17. There we could only overapproximate the spectrum, because we could not determine which numbers from previously layers were simultaneously realizable. By inductively maintaining the entire history at each node, we have enough information to resolve these questions of consistency, and compute an *exact* representation of the semantic object, not just an overapproximation.

The rest of this section is devoted to the proof of Theorem 54. We will first explain the intuition behind it. Let  $\mathcal{A}$  be a  $L$ -layer  $\mathcal{OLPW}$ -GNN, as in Definition 2. Let  $\mathcal{G}$  be a graph and  $v$  be a vertex. Recall that for every  $1 \leq \ell \leq L$ , the  $\ell$ -feature vector of  $v$  is:

$$\xi^{\ell}(v) := f^{\ell} \left( C^{\ell} \xi^{\ell-1}(v) + A_{\text{out}}^{\ell} \sum_{u \in \mathcal{N}_{\text{out}}(v)} \xi^{\ell-1}(u) + b^{\ell} \right).$$

We can rewrite it in terms of history:

$$\text{hist}^{\ell}(v)[0] = \xi^0(v), \tag{1}$$

and for each  $1 \leq i \leq \ell$ :

$$\text{hist}^{\ell}(v)[i] = f^i \left( C^i \cdot \text{hist}^{\ell}(v)[i-1] + A_{\text{out}}^i \cdot \left( \sum_{u \in \mathcal{N}_{\text{out}}(v)} \text{hist}^{\ell-1}(u) \right)[i-1] + b^i \right). \tag{2}$$

Thus, the  $\ell$ -history of  $v$  can be computed by applications of sum, affine transformations, and piecewise linear functions on the sum of the history of its out-neighbors.

We formalise this intuition in the following paragraphs. For each  $0 \leq \ell \leq L$ , we define the set  $\mathcal{H}^{\ell}$ :

$$\begin{aligned} \mathcal{H}^0 &:= \{0, 1\}^{d_0} \\ \mathcal{H}^{\ell} &:= \bigcup_{e \in \{0, 1\}^{d_0}} \text{proj}_{\ell} \circ T_{\ell} \circ T_{\ell-1} \circ \dots \circ T_{0,e} \circ \text{KleeneStar}(\mathcal{H}^{\ell-1}), \end{aligned}$$

where the definition and intuition of each  $\text{proj}_{\ell}, T_{\ell}, \dots, T_1, T_{0,e}$  is as follows.

## 127:18 Decidability of Graph Neural Networks via Logical Characterizations

- Intuitively KleeneStar ( $\mathcal{H}^{\ell-1}$ ) captures the term  $\sum_{u \in \mathcal{N}_{\text{out}}(v)} \text{hist}^{\ell-1}(u)$  in Equation (2).
- $T_{0,e} : \mathbb{Q}^{[d_0, \dots, d_{\ell-1}]} \rightarrow \mathbb{Q}^{[d_0, \dots, d_{\ell-1}, d_0]}$  is an affine transformation that maps  $a$  to  $(a, e)$ , i.e., it simply “pads”  $e$  into  $a$ .
- For each  $1 \leq i \leq \ell$ , the transformation  $T_i : \mathbb{Q}^{[d_0, \dots, d_{\ell-1}, d_0, \dots, d_{i-1}]} \rightarrow \mathbb{Q}^{[d_0, \dots, d_{\ell-1}, d_0, \dots, d_{i-1}, d_i]}$  computes the vector  $\text{hist}^{\ell}(v)[i]$  defined in Equation (2) and pads it at the end. Formally,  $T_i$  maps  $a$  to  $(a, c)$  where  $c = f^i(C^i a[\ell + i - 1] + A_{\text{out}}^i a[i - 1] + b^i)$ .
- Finally,  $\text{proj}_{\ell} : \mathbb{Q}^{[d_0, \dots, d_{\ell-1}, d_0, \dots, d_{\ell}]} \rightarrow \mathbb{Q}^{[d_0, \dots, d_{\ell}]}$  is a projection that projects out the first  $\ell$  components.

We can show that  $\mathcal{H}^{\ell}$  is a semi-linear set, and this captures the  $\ell$ -history-space, as stated formally in Lemma 55. Note that Theorem 53 follows easily from the lemma and the computability of the basis of  $\mathcal{H}^{\ell}$ .

► **Lemma 55.** *For every  $\mathcal{OLPW}$ -GNN  $\mathcal{A}$  and  $0 \leq \ell \leq L$ ,*

1.  $\mathcal{H}^{\ell}$  is a semi-linear set.
2. For every  $s \in \mathbb{Q}^{[d_0, d_1, \dots, d_{\ell}]}$ , the following are equivalent.
  - $h \in \mathcal{H}^{\ell}$
  - There exists a graph  $\mathcal{G}$  and vertex  $v \in V$  such that  $\text{hist}^{\ell}(v) = h$ .

**Proof.** The first item follows immediately from the fact that  $\mathcal{H}^0$  is semi-linear and the operators Kleene star, affine transformations and piecewise linear functions all preserve semi-linearity.

We now prove the second item by induction on  $\ell$ . The base case  $\ell = 0$  is trivial.

For the induction hypothesis, we assume that the lemma holds for  $\ell - 1$ . We show that  $h \in \mathcal{H}^{\ell}$  if and only if there is a graph  $\mathcal{G}$  and a vertex  $v$  such that  $\text{hist}^{\ell}(v) = h$ .

We start with the “only if” direction. Suppose  $h \in \mathcal{H}^{\ell}$ . By definition, there is  $e \in \{0, 1\}^{d_0}$  and a finite multi-subset  $\{\{h_1, h_2, \dots, h_k\}\}$  of  $\mathcal{H}^{\ell-1}$  such that:

$$h = \text{proj}_{\ell} \circ T_{\ell} \circ T_{\ell-1} \circ \dots \circ T_{0,e} (h_1 + h_2 + \dots + h_k)$$

By the induction hypothesis, there exist graphs  $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_k$  and vertices  $v_1, v_2, \dots, v_k$  such that  $\text{hist}_{\mathcal{G}_i}^{\ell-1}(v_i) = h_i$  for every  $1 \leq i \leq k$ .

Let  $\mathcal{G}$  be the graph obtained by taking the disjoint union of  $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_k$  and adding a fresh vertex  $v$ . Recalling that  $\xi_{\mathcal{G}}^0(v)$  can achieve an arbitrary combination of  $\{0, 1\}$  vectors, based on the colors of  $v$ , we set the colors so that  $\xi_{\mathcal{G}}^0(v) = e$ . We have an outgoing edge from  $v$  to  $v_i$  for each  $1 \leq i \leq k$ . It is routine to verify that the  $\ell$ -history of  $v$  is precisely  $h$ . Note that *because  $\mathcal{A}$  is outgoing-only, the edge from  $v$  to  $v_i$  has no effect on the  $(\ell - 1)$ -history of  $v_i$* . Thus  $\text{hist}_{\mathcal{G}}^{\ell-1}(v_i) = \text{hist}_{\mathcal{G}_i}^{\ell-1}(v_i)$ .

For the “if” direction, let  $\mathcal{G}$  be a graph and  $v$  be a vertex. Let  $v_1, \dots, v_k$  be the out-neighbors of  $v$ . By definition, for each  $1 \leq i \leq \ell$ :

$$\text{hist}_{\mathcal{G}}^{\ell}(v)[i] = f^i \left( C^i \cdot \text{hist}_{\mathcal{G}}^{\ell}(v)[i-1] + A_{\text{out}}^i \cdot \left( \sum_{u \in \mathcal{N}_{\text{out}}(v)} \text{hist}_{\mathcal{G}}^{\ell-1}(u) \right) [i-1] + b^i \right).$$

It is routine to verify that:

$$\text{hist}_{\mathcal{G}}^{\ell}(v) = \text{proj}_{\ell} \circ T_{\ell} \circ T_{\ell-1} \circ \dots \circ T_{0,e} \left( \text{hist}_{\mathcal{G}}^{\ell-1}(v_1) + \text{hist}_{\mathcal{G}}^{\ell-1}(v_2) + \dots + \text{hist}_{\mathcal{G}}^{\ell-1}(v_k) \right),$$

where  $e = \xi_{\mathcal{G}}^0(v)$ . Therefore,  $\text{hist}_{\mathcal{G}}^{\ell}(v) \in \mathcal{H}^{\ell}$ . ◀

## 5 Discussion

This work extends the exploration of the relationship between aggregate-combine GNNs and logic, with exact characterizations of expressiveness for GNNs with eventually constant activation functions, and embedding a logic into the GNNs with standard ReLU activations. We also obtain both decidability and undecidability results, some using the logical characterizations and some by porting the techniques used for decidability of the logics to apply directly on the GNNs. Perhaps the main take-away, echoing the theme of [1], is that Presburger logics and the techniques for analyzing them can be relevant to GNNs.

We have left open one major technical problem: the decidability of satisfiability for standard GNNs using the ReLU activation function. Here we have proven decidability only for the “outgoing-only” variant. We also do not know whether the undecidability results we have proven – e.g. for standard GNNs with global readout – still hold for the variants with outgoing-only aggregation. Thus, for all we know, the most crucial dividing line for decidability could revolve around outgoing-only vs bidirectional aggregation, rather than (e.g.) local vs global aggregation or truncation vs non-truncation in the activation function.

Looking at broader open issues, we focused here on some very basic verification problems on GNNs: can a certain classification be achieved? But it is clear that our techniques apply to many other logic-based verification problems; for example, it can be applied to determine whether a GNN can achieve a certain classification on a graph satisfying a certain sentence – provided that the sentence is also in one of our decidable logics.

We have not focused on complexity in this paper. Of course, for the broad class of GNNs with eventually constant activation functions, it is difficult to talk about complexity bounds. For GNNs based on truncated ReLU and local aggregation, we have shown satisfiability is PSPACE-complete, and is NP-complete for a fixed number of layers. The finer-grained complexity analysis for other decidability results is left for future work.

Our work provides motivation for exploring the properties of Presburger logics over relational structures and their connections with GNNs beyond the setting here, which considers only graphs with discrete feature values from a fixed set. In our ongoing work we are adapting our techniques to deal with GNNs whose feature values are *unbounded integers*, specified by an initial semi-linear set.

---

### References

- 1 Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The Logical Expressiveness of Graph Neural Networks. In *ICLR*, 2020.
- 2 Pablo Barceló, Alexander Kozachinskiy, Anthony Wijdada Lin, and Vladamir Podolskii. Logical languages accepted by transformer encoders with hard attention. In *ICLR*, 2024.
- 3 Bartosz Bednarczyk, Maja Orłowska, Anna Pacanowska, and Tony Tan. On classical decidable logics extended with percentage quantifiers and arithmetics. In *FSTTCS*, 2021.
- 4 David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer encoders. In *ICML*, 2023.
- 5 Dmitry Chistikov and Christoph Haase. The taming of the semi-linear set. In *ICALP*, 2016.
- 6 Seymour Ginsburg and Edwin H. Spanier. Semigroups, Presburger formulas, and languages. *Pacific J. Math.*, 16(2):285–296, 1966.
- 7 Valentin Goranko and Martin Otto. Model theory of modal logic. In Blackburn, van Benthem, and Wolter, editors, *Handbook of Modal Logic*. North-Holland, 2007.
- 8 Martin Grohe. The logic of graph neural networks. In *LICS*, 2021.
- 9 Martin Grohe. The descriptive complexity of graph neural networks. In *LICS*, 2023.

## 127:20 Decidability of Graph Neural Networks via Logical Characterizations

- 10 C. Haase and Georg Zetsche. Presburger arithmetic with stars, rational subsets of graph groups, and nested zero tests. *LICS*, 2019.
- 11 Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. An Algorithm for Deciding BAPA: Boolean Algebra with Presburger Arithmetic. In *CADE*, 2005.
- 12 Richard E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6(3):467–480, 1977.
- 13 Marvin L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Inc., 1967.
- 14 Marco Sälzer and Martin Lange. Fundamental limits in formal verification of message-passing neural networks. In *ICLR*, 2023.
- 15 Manfred Schmidt-Schaubß and Gert Smolka. Attributive concept descriptions with complements. *Artif. Intell.*, 48(1):1–26, 1991.
- 16 Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks. In *ICLR*, 2019.