A Logical View of GNN-Style Computation and the Role of Activation Functions

PABLO BARCELÓ, Institute for Mathematical and Computational Engineering,
Pontifical Catholic University of Chile & IMFD & CENIA, Chile
FLORIS GEERTS, Department of Computer Science, University of Antwerp, Belgium
MATTHIAS LANZINGER, Institute for Logic and Computation, TU Wien, Austria
KLARA PAKHOMENKO, Data Science Institute, Universiteit Hasselt, Belgium
JAN VAN DEN BUSSCHE, Data Science Institute, Universiteit Hasselt, Belgium

We study the numerical and Boolean expressiveness of MPLang, a declarative language that captures the computation of graph neural networks (GNNs) through linear message passing and activation functions. We begin with A-MPLang, the fragment without activation functions, and give a characterization of its expressive power in terms of walk-summed features. For bounded activation functions, we show that (under mild conditions) all eventually constant activations yield the same expressive power—numerical and Boolean—and that it subsumes previously established logics for GNNs with eventually constant activation functions but without linear layers. Finally, we prove the first expressive separation between unbounded and bounded activations in the presence of linear layers: MPLang with ReLU is strictly more powerful for numerical queries than MPLang with eventually constant activation functions, e.g., truncated ReLU. This hinges on subtle interactions between linear aggregation and eventually constant non-linearities, and it establishes that GNNs using ReLU are more expressive than those restricted to eventually constant activations and linear layers.

CCS Concepts: • **Computing methodologies** → **Machine learning**; • **Theory of computation** → **Database query languages (principles)**; • **Mathematics of computing** → **Graph coloring**.

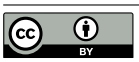
Additional Key Words and Phrases: Liouville number, symmetric function, simple function, graded modal logic

ACM Reference Format:
Pablo Barceló, Floris Geerts, Matthias Lanzinger, Klara Pakhomenko, and Jan Van den Bussche. 2026. A Logical View of GNN-Style Computation and the Role of Activation Functions . *Proc. ACM Manag. Data* 4, 2 (PODS), Article 118 (May 2026), 19 pages. https://doi.org/10.1145/3801914

1 Introduction

Context. The study of query languages for graph-structured data has traditionally focused on structural and navigational queries, where the goal is to select tuples of nodes and/or paths based on the topology of the underlying graph [3]. This line of work has produced a rich theory of languages such as *regular path queries* [13, 35], *conjunctive* regular path queries [11], and their extensions [6, 12, 34, 39], together with extensive results on expressive power, complexity of query evaluation, and static analysis [4, 15–20, 23, 40]. These foundational developments have also

Authors' Contact Information: Pablo Barceló, Institute for Mathematical and Computational Engineering, Pontifical Catholic University of Chile & IMFD & CENIA, Santiago, Chile, pbarcelo@uc.cl; Floris Geerts, Department of Computer Science, University of Antwerp, Antwerp, Belgium, floris.geerts@uantwerp.be; Matthias Lanzinger, Institute for Logic and Computation, TU Wien, Vienna, Austria, matthias.lanzinger@tuwien.ac.at; Klara Pakhomenko, Data Science Institute, Universiteit Hasselt, Hasselt, Belgium, klara.pakhomenko@uhasselt.be; Jan Van den Bussche, Data Science Institute, Universiteit Hasselt, Hasselt, Belgium, jan.vandenbussche@uhasselt.be.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 2836-6573/2026/5-ART118
https://doi.org/10.1145/3801914

influenced language design in practice, informing standards and systems such as SPARQL [30], G-CORE [2], Cypher [22], and, most recently, the ISO-standardized GQL [21].

A complementary perspective on querying graph data has emerged from the success of *graph neural networks* (GNNs) [26, 29, 33, 42]. Rather than selecting nodes or paths, GNNs compute numerical node embeddings via local *message passing*, combining linear aggregation with a (typically non-linear) activation function such as ReLU, truncated ReLU, sign, or bool. Each GNN therefore defines a *numerical* node query and, after thresholding, also a *Boolean* one. This computational pattern is captured by MPLang [25], a language influenced by modal logics for semiring-annotated databases [14]. It is a declarative language rooted in linear-algebraic frameworks such as MATLANG [10] and the tensor language TL for analysing GNN expressiveness [24]. MPLang expresses GNNs through linear combination, neighbourhood aggregation, and an activation function σ, providing a clean, logic-oriented setting for reasoning about embedding-based graph queries. This aligns with recent efforts to characterize the expressive power of GNN architectures [1, 5, 9, 28, 31, 38, 41, 44].

Problems studied. Our goal is to characterize the expressive power of MPLang-style languages in terms of their *numerical* and *Boolean* queries (the former assign real values to nodes, the latter values in $\{0, 1\}$). As a baseline for understanding the role of activation functions, we start with A-MPLang, the affine fragment of MPLang *without* activations. We analyse the queries definable in A-MPLang via the walks originating at the node under evaluation and compare their Boolean expressive power with that of modal logic (ML), a standard benchmark for GNN expressiveness [5, 9, 36]. We also study closure properties of A-MPLang under Boolean combinations.

We then consider the general case of σ-MPLang, which extends A-MPLang with expressions of the form $\sigma(e)$ applying an activation function σ to the embedding computed by e. Following standard distinctions, we treat *bounded* and *unbounded* activation functions separately [9, 28]. For bounded activations, we focus on *eventually constant* functions [9], a natural analogue of piecewise-linear and threshold operations such as truncated ReLU, sign, and bool. For such σ, we analyse σ-MPLang again in terms of walks and examine whether different eventually constant σ differ in expressive power. For unbounded activations, such as ReLU, we ask whether they yield strictly more expressive power than any eventually constant function—specifically, whether there exist queries definable in ReLU-MPLang that no σ-MPLang with eventually constant σ can express.

Our results. We consider *embedded* graphs, where each node has an associated feature vector. A central case is that of *coloured* graphs, whose embeddings are one-hot encodings of finitely many colours. Most results are stated for coloured graphs, with a few exceptions that extend to general embeddings. Crucially, all inexpressibility results already hold in the coloured setting.

- No activation function. We begin by analysing A-MPLang, the fragment of MPLang without activation functions. Numerical queries definable in A-MPLang correspond to affine combinations of walk-counts and walk-summed features (Theorem 4.2). Using this, we analyse how far the Boolean queries expressible by A-MPLang go as a Boolean query language and show that it is not closed under Boolean combinations, making it incomparable to Modal Logic (ML) on coloured graphs (Proposition 4.6). Since A-MPLang induces exactly the embeddings computed by GNNs with linear activations, this confirms that non-linearities are essential for capturing several meaningful graph properties, and in particular, that the ability of GNNs to match the expressive power of ML [5] crucially depends on them.
- Bounded activation functions. We show that, under mild assumptions, any eventually constant activation function σ yields the same expressive power for numerical queries in σ-MPLang over coloured graphs (Theorem 5.5), improving a related result in [9] on Boolean queries computed by GNNs without linear layers. This shows that σ-MPLang with any such

eventually constant σ is closed under Boolean operations. Our languages subsume the logics introduced by Benedikt et al. [9] and Nunn et al. [38] for Boolean GNN expressivity with eventually constant activations but without linear layers.

- Beyond bounded activation functions. We then turn to unbounded non-linear activation functions and compare their expressive power with that of arbitrary eventually constant ones. Our main technical result is that, for numerical queries over coloured graphs, ReLU-MPLang is strictly more expressive than MPLang with any eventually constant activation (Theorem 6.8). This is the most technically demanding part of the paper and shows that GNNs using both ReLU and linear layers surpass those relying only on truncated ReLU with linear layers. A related separation was shown by Benedikt et al. [9], but only for GNNs without linear layers. Incorporating identity activations is substantially more challenging, as linear layers interact delicately with non-linear transformations and isolating their contribution is essential [8, 32]. Overall, unbounded activations like ReLU permit numerical computations that bounded ones cannot reproduce, leading to different expressive capabilities.

Types of embeddings and queries. Previous work on uniform logical expressivity of GNNs has largely focused on coloured graphs (Boolean node features) and Boolean queries [5, 9, 27]. We follow this tradition, but also study numerical queries and, where possible, extend results to general d-embeddings. As a guiding principle, our separation/inexpressibility results are proved already in the most restrictive setting (Boolean queries on colourings), while our containment results are stated in the most general setting we can obtain (often numerical queries over numerical embeddings). In the Conclusion we overview our results in Table 1. Finally, although applied GNNs typically start from real-valued attributes, discretization provides a direct link to our setting: continuous features can be quantized into finitely many bins and encoded as colours. This means our negative results remain informative under richer inputs, whereas extending some of our positive collapse/equivalence results beyond colourings is an open question.

Our techniques. Prior work has analysed GNNs through their distinguishing power [27], and also in non-uniform variants where the model (or an additional readout) is allowed to exploit the size of the input graph [5]. In contrast, we study *uniform* expressivity: a single expression/network architecture must work across all graph sizes. This perspective calls for different techniques. The analysis of the expressive power of the numerical query languages studied in this paper requires techniques that are less commonly encountered in classical database theory, which typically deals with query languages based on Boolean logic. Some of our results rely on traditional methods, e.g., establishing normal forms for expressions. Likewise, inexpressibility of a query can still be shown by exhibiting pairs of indistinguishable graphs on which the query yields different outcomes. However, the indistinguishability requirement becomes stronger, as we now need to show that all expressions return the same numerical value (or, for Boolean queries, values with the same sign). To this end, our proofs draw on notions from analysis, including properties of the rationals (as opposed to real numbers), alternating Liouville constructions and rational approximation of reals, limits, symmetric functions, and both simple and piecewise polynomial functions.

Organization of the paper. We present basic notions in Section 2, introduce MPLang and its connection to GNNs in Section 3. Then, in Section 4 we cover A-MPLang, and in Sections 5 and 6 we address bounded and unbounded activations, respectively. We conclude in Section 7 with remarks and open problems. Full proofs of the claims of this paper can be found in our technical report [7].

2 Preliminaries

Basics. For $k \in \mathbb{N}$, we set $[i, j] \coloneqq \{i, i+1, \ldots, j\}$ and $[k] \coloneqq \{1, \ldots, k\}$. A function $a : \mathbb{R}^k \to \mathbb{R}$ is *affine* if $a(x_1, \ldots, x_k) \coloneqq a_0 + \sum_{i=1}^{k} a_i x_i$ for $a_0, \ldots, a_k \in \mathbb{R}$. We define $\mathsf{ReLU} : \mathbb{R} \to \mathbb{R}$ by $\mathsf{ReLU}(x) = \max\{0, x\}$, and the *truncated* ReLU $\mathsf{TrReLU} : \mathbb{R} \to \mathbb{R}$ by $\mathsf{TrReLU}(x) = \max\{0, \min\{1, x\}\}$.

Graphs and embeddings. A graph G is a pair (V, E), where V is a finite set of nodes and $E \subseteq \big\{\{u, v\} \mid u, v \in V,\ u \neq v\big\}$ is a set of undirected edges. A *walk* of length $n \geq 0$ in G is a sequence of nodes $v_0, \ldots, v_n$ such that $\{v_k, v_{k+1}\} \in E$, for all $k \in [0, n-1]$. A walk of length 0 from a vertex v is the trivial walk (v).

For $d > 0$, a *d-embedding* of a graph $G = (V, E)$ is a map $\gamma : V \to \mathbb{R}^d$. The pair (G, γ) is called a *d-embedded* graph, and if v is a node in V, then the tuple (G, γ, v) is a *pointed d*-embedded graph.

Numerical queries. Our goal is to analyse the expressive power of query languages that compute and transform graph embeddings. A *numerical query* of type $d \to r$ maps a d-embedded graph (G, γ) to an r-embedding γ' of G. Since any such query can be decomposed into r numerical queries of type $d \to 1$, we restrict attention to queries with 1-dimensional output.

Boolean Queries. Following standard practice in the literature on the expressive power of graph embeddings computed by GNNs [5, 9], we associate with every numerical query Q of type $d \to 1$, for $d > 0$, a corresponding *Boolean* query $Q_{\mathbb{B}}$ defined by

$$Q_{\mathbb{B}}(G, \gamma)(v) \coloneqq \begin{cases} 1, & \text{if } Q(G, \gamma)(v) > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, a Boolean query is a numerical query of type $d \to 1$ whose output embedding assigns each node a value in $\{0, 1\}$. This provides a way to compare numerical query languages that compute and transform graph embeddings with purely Boolean ones (e.g., FO or ML, introduced below).

Equivalence of queries. Queries Q and Q' are *numerically equivalent* on a class C of embedded graphs if $Q(G, \gamma) = Q'(G, \gamma)$ for every $(G, \gamma) \in C$. Similarly, we say that Q and Q' are *Boolean equivalent* on C, if $Q_{\mathbb{B}}(G, \gamma) = Q'_{\mathbb{B}}(G, \gamma)$ for every $(G, \gamma) \in C$.

Comparisons of query languages. We compare query languages by containment of the numerical or Boolean queries they define on a class C of embedded graphs. Thus two languages may be *numerically equivalent* over C (when they express the same numerical queries on C), or one may be *Boolean strictly contained* in the other (when all its Boolean queries are expressible in the latter but not vice versa).

3 The language MPLang and GNNs

3.1 The language Σ-MPLang

For any finite set Σ of functions $\sigma : \mathbb{R} \to \mathbb{R}$, called *activation functions*, we define the query language *Σ-MPLang* over d-embeddings, for $d > 0$, by means of the following grammar:

$$e \ ::= \ 1 \mid P_i \mid ae \mid e + e \mid \oplus e \mid \sigma(e), \qquad \text{where } i \in [d],\ \sigma \in \Sigma, \text{ and } a \in \mathbb{R}.$$

Real constants a in expressions are often referred to as coefficients. Moreover, P_i can be understood as projecting on the i-th component.

Each σ-MPLang expression e over d-embeddings defines a numerical query of type $d \to 1$, which is inductively defined as follows on a d-embedded graph (G, γ):

- If $e = 1$, then $e(G, \gamma)(v) \coloneqq 1$.
- If $e = P_i$, then $e(G, \gamma)(v) \coloneqq \gamma(v)_i$, the i-th component of $\gamma(v)$.

- If $e = ae_1$, then $e(G,\gamma)(v) \coloneqq a \cdot e_1(G,\gamma)(v)$.
- If $e = e_1 + e_2$, then $e(G,\gamma)(v) \coloneqq e_1(G,\gamma)(v) + e_2(G,\gamma)(v)$.
- If $e = \oplus e_1$, then $e(G,\gamma)(v) \coloneqq \sum_{\{u,v\}\in E} e_1(G,\gamma)(u)$.
- If $e = \sigma(e_1)$, then $e(G,\gamma)(v) \coloneqq \sigma\big(e_1(G,\gamma)(v)\big)$.

For convenience, we write $e(G,\gamma,v) \coloneqq e(G,\gamma)(v)$ for pointed embedded graphs. We use $\oplus^i$ to denote i consecutive $\oplus$ symbols. If Σ contains only one function σ, we write σ-MPLang instead of $\{\sigma\}$-MPLang. A Σ-MPLang-expression is *rational* if all its coefficients are rational numbers; the set of such expressions is called *rational* Σ-MPLang.

3.2 Connections between MPLang and GNNs

We now examine the connection between MPLang and graph neural networks (GNNs).

GNNs. A *GNN layer* of type $d \to r$ is a tuple $L = (W_1, W_2, b, \sigma)$, where $W_1, W_2 \in \mathbb{R}^{r\times d}$, $b \in \mathbb{R}^r$, and $\sigma : \mathbb{R} \to \mathbb{R}$ is an activation function. Such a layer defines a numerical query of type $d \to r$ by mapping each d-embedded graph (G,γ) to:

$$L(G,\gamma)(v) \coloneqq \sigma\left(W_1 \cdot \gamma(v) + W_2 \cdot \sum_{\{u,v\}\in E} \gamma(u) + b\right), \qquad \text{for each } v \in V.$$

We abuse notation by writing $\sigma(x_1,\ldots,x_r)$, for $(x_1,\ldots,x_r) \in \mathbb{R}^r$, to denote $(\sigma(x_1),\ldots,\sigma(x_r))$.

A *GNN* over d-embeddings is a sequence $\mathcal{S} = L_1,\ldots,L_n$ of layers, where L_1 has type $d \to s_1$, each L_i (for $i \in [2, n-1]$) has type $s_{i-1} \to s_i$, and L_n has type $s_{n-1} \to 1$ (with $s_1 = 1$ if $n = 1$). This GNN defines a numerical query of type $d \to 1$ on (G,γ) by iteratively applying the layers, i.e.,

$$\mathcal{S}(G,\gamma) \;=\; (L_n \circ \cdots \circ L_1)(G,\gamma).$$

Let Σ be a finite set of functions $\sigma : \mathbb{R} \to \mathbb{R}$. We denote by Σ-GNNs the class of GNNs $\mathcal{S}$, such that each GNN layer in $\mathcal{S}$ is of the form (W_1, W_2, b, σ) for some $\sigma \in \Sigma$. If $\Sigma = \{\sigma\}$, then we simply write σ-GNNs instead of $\{\sigma\}$-GNNs.

The connection. As implicit in [24], there is a clean correspondence between queries expressible in σ-MPLang and those definable by σ-GNNs augmented with identity (id) layers.

Proposition 3.1. [24] *Let Σ be a finite set of functions $\sigma : \mathbb{R} \to \mathbb{R}$ and $d > 0$. Then Σ-MPLang and $(\Sigma \cup \{\mathsf{id}\})$-GNNs, both over d-embeddings, are numerically equivalent on all d-embedded graphs.*

Remark 3.2. Further inspecting the proof in [24], one obtains an additional correspondence: if σ is idempotent and the identity function id can be expressed from σ using affine transformations, then the queries definable by σ-GNNs over d-embeddings coincide with those definable in σ-MPLang over d-embeddings. Notably, ReLU satisfies this property: $\mathsf{id}(x) = \mathsf{ReLU}(x) - \mathsf{ReLU}(-x)$. □

3.3 Traditional query languages on graphs.

Classical query languages for graphs operate on node-coloured structures and define Boolean queries by selecting nodes. We briefly introduce these languages and compare them with GNNs.

Coloured graphs. A *k-colouring* of a graph $G = (V,E)$ is a k-embedding $c : V \to \{0,1\}^k$, where $c_i(v) = 1$ indicates that v has colour i. A *k-coloured* graph is a pair (G,c), and a *pointed* one is (G,c,v) with $v \in V$. Since colourings are special embeddings, MPLang expressions and GNNs apply directly to coloured graphs. For readability, we use colour names (e.g. red, blue) in place of P_1, P_2.

Traditional query languages. First-Order Logic (FO) and Modal Logic (ML) are standard formalisms for Boolean queries on coloured graphs. FO is interpreted over k-coloured graphs with adjacency E and unary predicates P_i, for $i \in [k]$, indicating colours. ML evaluates formulas at nodes using propositional symbols P_i, Boolean connectives, and the modality $\Diamond$:

$$(\Diamond\varphi)(G,c)(v) = 1 \quad \text{iff} \quad \exists u\,(\{u,v\} \in E \wedge \varphi(G,c)(u) = 1).$$

Graded Modal Logic (GML) extends ML with counting modalities $\Diamond^{\geq n}$, where

$$(\Diamond^{\geq n}\varphi)(G,c)(v) = 1 \quad \text{iff} \quad |\{u : \{u,v\} \in E,\ \varphi(G,c)(u) = 1\}| \geq n.$$

Boolean GML queries over coloured graphs are also expressible in FO, but not vice versa.

Boolean expressiveness of GNNs. GNNs subsume GML for Boolean queries on coloured graphs.

Theorem 3.3. [5, 9] *Fix $k > 1$. We have that GML is Boolean strictly contained in σ-MPLang, for $\sigma \in \{ReLU, TrReLU\}$, over the class of k-coloured graphs.*

4 A-MPLang: No Activation Function

To set a baseline for analysing the role of activation functions in the expressivity of MPLang, we first look at the purely *affine* fragment, i.e., *without* activation functions: A-MPLang. Since it is possible to write any affine function in MPLang without the use of activation functions, intuitively id-MPLang is equivalent to A-MPLang (where A is the set of all affine functions), hence the name. By Proposition 3.1, this fragment captures GNNs that use only the identity activation id. As we observe, even this seemingly weak setting can express Boolean queries beyond FO.

Example 4.1. Consider coloured graphs with node colours red and blue. The A-MPLang expression $(\oplus\text{red} - \oplus\text{blue})$ defines the Boolean query that holds at a node iff it has more red than blue neighbours. This query is not definable in FO, and thus neither in ML. □

This illustrates that the expressive power of A-MPLang is subtler than it might initially appear. In fact, a useful way to think about the Boolean fragment of A-MPLang is to consider numerical A-MPLang expressions with one final bounded activation that implements thresholding.

4.1 Numerical expressivity

We show that A-MPLang can only compute numerical functions determined by walk counts and walk-summed features (over the class of all embedded graphs). To this aim, we derive a normal form of A-MPLang expressions in terms of walk counts using the notion of $\oplus$-*depth* of an expression:

- The $\oplus$-depth of the expressions 1 and P_i are 0.
- Given two Σ-MPLang expressions e, e' with $\oplus$-depths n and n', respectively, the $\oplus$-depth of ae and $\sigma(e)$ is n, the $\oplus$-depth of $e + e'$ is $\max\{n, n'\}$ and the $\oplus$-depth of $\oplus e$ is $n + 1$.

Of particular interest for the normal form are expressions of the form $\oplus^i 1$ and $\oplus^i P_j$. When evaluated on a pointed embedded graph (G, γ, v), these expressions correspond, respectively, to the number of walks of length i starting at v, and to the sum, over all walks of length i starting at v, of the j-th coordinate of the embedding at the endpoint of each walk. The normal form below shows that A-MPLang can extract information only through a simple form of walk statistics.

Theorem 4.2. *Let e be an A-MPLang-expression over d-embeddings of $\oplus$-depth n. Then, over the class of d-embedded graphs, e is numerically equivalent to an expression of the form*

$$\sum_{i=0}^{n} \Big(c_i \oplus^i 1 + \sum_{j=1}^{d} c_{i,j} \oplus^i P_j\Big).$$

This normal form can be proven by induction on the structure of A-MPLang expressions. Apart from delineating what type of graph structural information A-MPLang can extract, Theorem 4.2 can also be used to show that certain numerical functions cannot be expressed in A-MPLang. In particular, Theorem 4.2 implies the following equivalence.

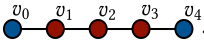
Corollary 4.3. *The following are equivalent for pointed d-embedded graphs (G_1, γ_1, v_1) and (G_2, γ_2, v_2):*

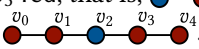
- *$e(G_1, \gamma_1)(v_1) = e(G_2, \gamma_2)(v_2)$, for every A-MPLang expression e of $\oplus$-depth n over d-embeddings.*
- *$(\oplus^i 1)(G_1, \gamma_1)(v_1) = (\oplus^i 1)(G_2, \gamma_2)(v_2)$ and $(\oplus^i P_j)(G_1, \gamma_1)(v_1) = (\oplus^i P_j)(G_2, \gamma_2)(v_2)$, for all $i \in [n]$ and $j \in [d]$.*

We refer to the condition in the second item by saying that (G_1, v_1, γ_1) and (G_2, v_2, γ_2) are *n-walk equivalent*. Moreover, they are called *walk equivalent* if they are n-walk equivalent for all $n \in \mathbb{N}$. Hence, to show that a numerical query f is not expressible in A-MPLang, we need to find two walk-equivalent pointed embedded graphs (G_1, γ_1, v_1) and (G_2, γ_2, v_2) with $f(G_1, \gamma_1)(v_1) \neq f(G_2, \gamma_2)(v_2)$.

By Theorem 4.2, every A-MPLang-expression is a linear combination of walk statistics of the form $\oplus^i 1$ and $\oplus^i P_j$, and hence can inspect the colours/features only at the *endpoints* of walks of length i from v. The next example exploits this: the query depends on a property of an *intermediate* node, which is invisible to endpoint-only walk sums and therefore inexpressible in A-MPLang.

Example 4.4. Consider the path P_5 on five nodes v_0, v_1, v_2, v_3, v_4 with undirected edges $\big\{\{v_0, v_1\}, \{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}\big\}$, and let the distinguished node be v_1. We consider two 2-colourings (with colours red and blue) of this graph:

- Colouring c colours the nodes v_0 and v_4 blue, and v_1, v_2, v_3 red, that is, 

.
- Colouring c' colours v_2 blue, and v_0, v_1, v_3, v_4 red, that is, 

.

We define a numerical query f which counts the number of blue neighbours of a node which have at least two red neighbours. We have $f(P_5, c)(v_1) = 0$ and $f(P_5, c')(v_1) = 1$. Yet, (P_5, v_1, c) and (P_5, v_1, c') are walk-equivalent. Indeed, for $i \geq 0$ and $t \in \{0, 1, 2, 3, 4\}$ let $\mathsf{w}_i(t)$ be the number of walks of length i from v_1 to v_t in P_5. We note that $\mathsf{w}_{i+1}(t) = \sum_{\{v_u, v_t\} \in E} \mathsf{w}_i(u)$. For all $i \geq 0$, we have

$$\mathsf{w}_i(2) = \mathsf{w}_i(0) + \mathsf{w}_i(4).$$

For $i = 0$ this is immediate and for $i > 0$, using the adjacency of P_5 we have $\mathsf{w}_{i+1}(2) = \mathsf{w}_i(1) + \mathsf{w}_i(3)$, and since v_0 (resp. v_4) has the unique neighbor v_1 (resp. v_3), $\mathsf{w}_{i+1}(0) = \mathsf{w}_i(1)$ and $\mathsf{w}_{i+1}(4) = \mathsf{w}_i(3)$. Thus $\mathsf{w}_{i+1}(2) = \mathsf{w}_{i+1}(0) + \mathsf{w}_{i+1}(4)$, as desired. Now, for all i, $(\Diamond^i 1)(P_5, c)(v_1) = (\Diamond^i 1)(P_5, c')(v_1)$ since the underlying pointed graph is the same. Moreover,

$$(\Diamond^i \mathsf{blue})(P_5, c)(v_1) = \mathsf{w}_i(0) + \mathsf{w}_i(4), \qquad (\Diamond^i \mathsf{blue})(P_5, c')(v_1) = \mathsf{w}_i(2),$$

which agree, as we have just shown. Finally $\mathsf{red} = 1 - \mathsf{blue}$ holds pointwise in both colourings, so $\Diamond^i \mathsf{red}$ also agrees. Hence the two pointed structures are walk-equivalent. Hence, f is not expressible in A-MPLang (over 2-coloured graphs). □

This example leverages that A-MPLang can only inspect the colours of terminal nodes along walks, and computing the query relies on a property of the middle node in walks of length 2.

4.2 Boolean expressivity

We know from Example 4.1 that A-MPLang can express Boolean queries beyond FO. We now ask whether it forms a reasonable Boolean query language, which would at the very least require closure under the Boolean operations. In the following, we refer to the class of Boolean queries expressible in (rational) A-MPLang as (rational) A-MPLang$_{\mathbb{B}}$. Our first result is positive.

Proposition 4.5. *Rational A-MPLang$_{\mathbb{B}}$ on k-coloured graphs is closed under negation.*

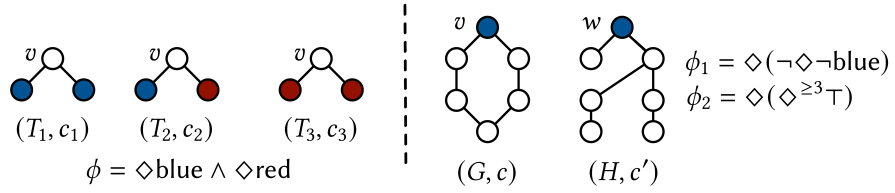


Fig. 1. Graphs and Boolean queries used to show non-closure of A-MPLang$_{\mathbb{B}}$ in the proof of Proposition 4.6.

Proof. Let e be a rational A-MPLang expression of $\oplus$-depth n on k-coloured graphs. By the proof of Theorem 4.2, e is of the form $\sum_{i=0}^{n}\left(c_i \oplus^i 1 + \sum_{j=1}^{k} c_{i,j} \oplus^i P_j\right)$, for suitable rational coefficients $c_i, c_{i,j}$. Let d_e be a common denominator of all these coefficients. Since each P_j is Boolean, every value $e(G,c)(v)$ is an integer multiple of $1/d_e$, so any positive value is at least $1/d_e$. Set $k_e := 1/2d_e$ and define $e' := k_e 1 - e$. Then, for a pointed k-coloured graph (G, c, v):

- if $e(G,c)(v) > 0$, then $e'(G,c)(v) < 0$;
- if $e(G,c)(v) \leq 0$, then $e'(G,c)(v) \geq k_e > 0$.

Thus e' expresses the negation of the Boolean query defined by e. □

But this is as far as rational A-MPLang$_{\mathbb{B}}$ goes in behaving like a Boolean query language.

Proposition 4.6. *Rational A-MPLang$_{\mathbb{B}}$ on k-coloured graphs is neither closed under conjunction nor under the Boolean $\Diamond$-modality.*

In other words, even if ϕ_1 and ϕ_2 are in rational A-MPLang$_{\mathbb{B}}$, the Boolean queries $\phi_1 \wedge \phi_2$ and $\Diamond\phi_1$ need not be expressible in this way.

- The proof of non-closure under conjunction uses the 2-coloured trees (T, c_1), (T, c_2), and (T, c_3) (shown on the left in Figure 1) together with a Boolean conjunction ϕ. We show that any A-MPLang expression e must satisfy either $e(T, c_1)(v) \geq e(T, c_2)(v)$ or $e(T, c_3)(v) \geq e(T, c_2)(v)$, contradicting that ϕ holds on (T, c_2, v) but fails on both (T, c_1, v) and (T, c_3, v).
- We show non-closure under the Boolean $\Diamond$-modality by exhibiting pointed 2-coloured graphs (G, c, v) and (H, c', w) (depicted on the right in Figure 1) that are walk-equivalent but can be distinguished by Boolean queries ϕ_1 (in ML) and ϕ_2 (in GML). Each of these queries applies $\Diamond$ to a Boolean query that is itself in A-MPLang$_{\mathbb{B}}$.

These negative results continue to hold for A-MPLang when we drop the restriction to rational coefficients. However, the proof of closure under negation relies in an essential way on having only rational coefficients. Once we lift this assumption, even that remaining Boolean closure property is lost.

Proposition 4.7. *A-MPLang$_{\mathbb{B}}$ on k-coloured graphs is not closed under negation.*

In particular, our proof demonstrates that the query $e := r \oplus \text{red} - \oplus\text{blue}$ has no negation when r is irrational. In fact, our proof requires r to be a Liouville number, a special type of non-algebraic real that can be approximated particularly well in terms of rational numbers. On a technical level this leads to e becoming arbitrarily close to 0 at a very fast rate, making it analytically impossible for a hypothetical MPLang expression that negates e to separate between cases where e evaluates to 0 or positive values. It remains an interesting open question whether non-closure under negation can also be observed with algebraic irrational coefficients.

Remark 4.8. The previous results, together with Example 4.1, show that A-MPLang$_{\mathbb{B}}$ and ML are incomparable in expressive power on coloured graphs. On uncoloured graphs, however, every ML formula is equivalent to one of $0, 1, \Diamond 1$, or $\neg\Diamond 1$ and is then subsumed by A-MPLang$_{\mathbb{B}}$. In contrast,

the GML formula ϕ_2 in Figure 1 is not expressible in A-MPLang$_{\mathbb{B}}$ even on uncoloured graphs. In conclusion, Theorem 3.3 holds only in the presence of non-linearities in GNNs. □

As we now show, activation functions restore closure under Boolean operations. In fact, MPLang with any eventually constant activation is closed under Boolean operators and subsumes earlier logical frameworks for characterizing ReLU-GNNs.

5 Bounded Activation Functions

Having established a baseline for the numerical and Boolean expressive power of MPLang, we now investigate what happens when we equip the language with bounded non-linear activation functions. We will start with the activation function bool, show that it is canonical, and then connect it to related work on the logical expressivity of GNNs with eventually constant activation functions.

5.1 MPLang with Booleanisation

Our guiding theme here is *Booleanisation*: using activations to turn numerical MPLang-expressions into robust Boolean query languages. For this, we introduce the activation function bool:

$$\mathrm{bool}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x \leq 0. \end{cases}$$

This is the indicator of the positive reals, also known as the *unit step function*. It bridges numerical and Boolean MPLang queries: the map $e \mapsto \mathrm{bool}(e)$ turns any numerical expression into a Boolean one.

Following the notation Σ-MPLang introduced in Section 3.1, the expressions that can use bool as activation function is denoted by bool-MPLang. Note that this is still a numerical query language; we write bool-MPLang$_{\mathbb{B}}$ for the class of resulting Boolean queries. As we see next, this activation function already yields strictly more expressive power than A-MPLang.

Example 5.1. Consider the Boolean expressions ϕ, ϕ_1 and ϕ_2 used in the proof of Proposition 4.6 and depicted in Figure 1. These Boolean queries are not expressible in A-MPLang. However, they become definable once we allow the bool activation. Indeed, it is readily verified that the ML formula $\phi = \Diamond\mathsf{red} \wedge \Diamond\mathsf{blue}$ can be expressed in bool-MPLang as $e = \mathrm{bool}(\oplus\mathsf{red}) + \mathrm{bool}(\oplus\mathsf{blue}) - 1$, that $\phi_1 = \Diamond(\neg\Diamond\neg\mathsf{blue})$ can be expressed as $e_1 = \oplus\big(1 - \mathrm{bool}(\oplus(1 - \mathsf{blue}))\big)$, and that $\phi_2 = \Diamond(\Diamond^{\geq 3}\top)$ can be expressed as $e_2 = \oplus\big(\mathrm{bool}((\oplus 1) - 2)\big)$. As a final example, we observe that

$$\oplus\mathrm{bool}\big(\mathsf{blue} + (\mathrm{bool}(\oplus\mathsf{red} - 1)) - 1\big)$$

returns the number of blue neighbours of a node which have at least two red neighbours. This computes the function from Example 4.4 that was not computable using A-MPLang. Hence, bool-MPLang is more expressive than A-MPLang on both numerical and Boolean queries. □

More generally, we prove that bool-MPLang$_{\mathbb{B}}$ is closed under Boolean operations.

Proposition 5.2. *bool-MPLang$_{\mathbb{B}}$ is closed under negation and disjunction.*

Proof. Recall that a Boolean query ϕ is expressed by an expression e if ϕ holds at v on (G, γ) iff $e(G, \gamma)(v) > 0$, for every pointed embedded graph (G, γ, v). Let ϕ_1 and ϕ_2 be Boolean queries expressed by e_1 and e_2 in bool-MPLang. For $\phi = \neg\phi_1$ we set $e \coloneqq 1 - \mathrm{bool}(e_1)$, and for $\phi = \phi_1 \vee \phi_2$ we set $e \coloneqq \mathrm{bool}(e_1) + \mathrm{bool}(e_2)$. □

We can therefore treat bool-MPLang$_{\mathbb{B}}$ as a genuine Boolean query language: whenever an expression e defines a Boolean query via the threshold condition $e(G, \gamma)(v) > 0$, we may freely

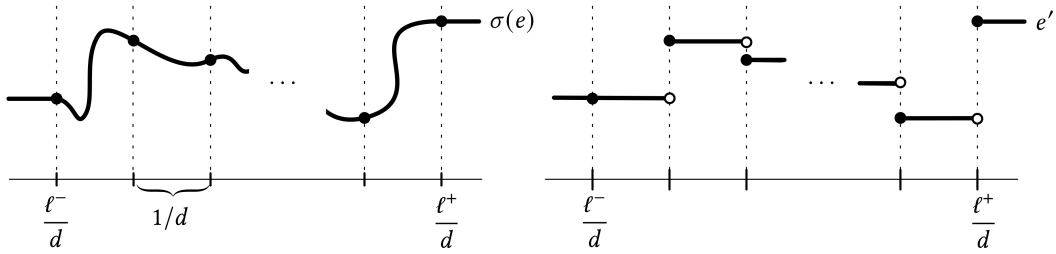


Fig. 2. Simulation of $\sigma(e)$ by a bool-MPLang expression e' as described in the proof of Theorem 5.5.

build new Boolean queries from it using the usual logical connectives and comparisons, and stay within bool-MPLang$_{\mathbb{B}}$. More precisely, for bool-MPLang-expressions e_1 and e_2, we abbreviate:

$$e_1 \vee e_2 \coloneqq \mathsf{bool}(e_1) + \mathsf{bool}(e_2), \qquad e_1 \wedge e_2 \coloneqq \mathsf{bool}(e_1) + \mathsf{bool}(e_2) - 1, \qquad \neg e_1 \coloneqq 1 - \mathsf{bool}(e_1),$$

$$e_1 > e_2 \coloneqq \mathsf{bool}(e_1 - e_2), \qquad e_1 \leq e_2 \coloneqq \neg(e_1 > e_2), \qquad e_1 = e_2 \coloneqq \neg\big(e_1 > e_2 \vee e_2 > e_1\big),$$

and similarly for the remaining comparison operators.

Example 5.3. These shorthand notations make it easy to specify more involved Boolean queries directly at the level of bool-MPLang. For instance, consider

$$(\oplus\mathsf{blue} > \oplus\mathsf{red} \wedge \oplus\mathsf{white} = 0) \vee (\oplus\mathsf{white} + \oplus\mathsf{blue} \leq \oplus\mathsf{red}),$$

which says that either a node has strictly more blue than red neighbours and no white neighbours, or the total number of white and blue neighbours does not exceed the number of red neighbours. □

The syntactic sugar will also make it very clear to see the connections to other logical frameworks for GNNs with eventually constant activation functions. To that end, we first show that the choice of bool as our non-linear bounded activation function is not arbitrary.

5.2 Eventually constant activation functions

We show that bool-MPLang acts as a canonical language, capturing the expressive power of various MPLang variants with different activation functions over coloured graphs. Our focus is on eventually constant activations, a standard choice in the study of neural networks [9, 43].

Definition 5.4 (Eventually constant function)**.** An *eventually constant function* is a function $\sigma : \mathbb{R} \to \mathbb{R}$ for which there exist $t^- < t^+$ such that $\sigma(t) = \sigma(t^-)$ for all $t \leq t^-$ and $\sigma(t) = \sigma(t^+)$ for all $t \geq t^+$. If $\sigma(t^-) \neq \sigma(t^+)$, we call σ *uneven eventually constant*.

The Booleanisation function bool is an uneven eventually constant activation; others include TrReLU, sign, and hard versions of sigmoid and tanh. We are not aware of any eventually constant activation used in machine learning that is not uneven. We show that, with rational coefficients, the specific choice of uneven eventually constant activation does not affect expressive power.

Theorem 5.5. *Let Σ be a finite set of uneven eventually constant functions. Then the rational fragment of Σ-MPLang is numerically equivalent to bool-MPLang on k-coloured graphs.*

Proof sketch. The proof relies on the fact that any rational σ-MPLang-expression on k-coloured graphs can take only finitely many values on a fixed interval, which can be encoded in bool-MPLang using suitable affine transformations. To see this, note that if d is the lowest common denominator in a rational MPLang expression e, then on any pointed coloured graph (G, c, v), the evaluation

$e(G, c)(v)$ only takes values of the form ℓ/d for $\ell \in \mathbb{Z}$. It thus suffices to simulate $\sigma(e)$ on that finite set only, which can be done by appropriately rescaling and combining bool-functions applied on e, as shown in Figure 2. □

This strictly extends the result of Benedikt et al. [9], who established a similar correspondence for Boolean queries computed by GNNs with eventually constant activations *but without linear layers*. In contrast, our setting allows linear layers and numerical (not only Boolean) queries, and therefore goes beyond the expressive scope considered in their work.

Remark 5.6. If σ has limits at $\pm\infty$, then for any $\varepsilon > 0$ any σ-MPLang-expression can be ε-approximated on all k-coloured graphs by a bool-MPLang-expression. This paves the way for quantitative approximation results for MPLang with bounded activations. □

5.3 Connections with logics for Boolean GNN expressivity

Using bool-MPLang as a stand-in for MPLang equipped with any eventually constant function, we can now see how the Boolean fragment bool-MPLang$_{\mathbb{B}}$ connects directly to modal logics with counting and Presburger constraints that have recently been used to characterise GNNs [9, 38].[1]

Benedikt et al. [9], introduces *local modal logics with Presburger quantifiers*, such as $\mathcal{L}$-MP^2. This logic extends ML with numerical atoms of the form

$$\sum_i \lambda_i \cdot \#y \left[E(x, y) \wedge \varphi_i(y)\right] \text{ op } \delta,$$

where the φ_i are themselves formulas in $\mathcal{L}$-MP^2, $\lambda_i, \delta \in \mathbb{Q}$, and op $\in \{=, \neq, \leq, \geq, <, >\}$. Such atoms evaluate to true or false depending on whether the indicated linear inequality is satisfied by the counts of one-hop neighbours whose endpoints satisfy the respective formulas.

$\mathcal{L}$-MP^2 is strictly contained in bool-MPLang$_{\mathbb{B}}$. Using the shorthands we developed for the language bool-MPLang, it is easy to see the connection:

Proposition 5.7. *$\mathcal{L}$-MP^2 is Boolean strictly contained in bool-MPLang over k-coloured graphs (strict containment holds even when $k = 2$).*

Proof. We first show containment. The Presburger quantifiers can be translated to

$$\sum_i c_i \diamond e_i \text{ op } \delta,$$

where $c_i, \delta \in \mathbb{Q}$ and op $\in \{=, \neq, \leq, \geq, <, >\}$, which can be translated using our shorthand for inequalities. Also recall the closure of bool-MPLang$_{\mathbb{B}}$ under Boolean combinations. The P_i represent unary predicates and 1 is the formula that is always true.

We now show that the containment is strict. Note that bool-MPLang$_{\mathbb{B}}$ extends $\mathcal{L}$-MP^2 by allowing arbitrarily many hops, precisely the extra expressive power enabled by the identity function. This is illustrated by the numerical query:

$$\diamond\diamond \,\text{green} = \diamond\diamond \,\text{blue},$$

which asks whether a node has the same number of green and blue grandchildren. Since this is an equivalence of two A-MPLang expressions, it is directly recognisable as expressible in bool-MPLang. But it is known that this query is inexpressible in $\mathcal{L}$-MP^2 [9]. □

This suggests that the Boolean queries definable by TrReLU-GNNs—equivalent, over coloured graphs, to those of $\mathcal{L}$-MP^2 [9]—match a specific syntactic subfragment of bool-MPLang.

[1]The logic K# was independently introduced by Nunn et al. in [38], and also captures the Boolean expressivity of TrReLU-GNNs without linear layers. However, the setting in this work is restricted to integer coefficients.

Linear layers add expressive power. By Theorem 5.5 and Proposition 3.1, bool-MPLang is expressively equivalent to any $\{\sigma, \mathsf{id}\}$-GNN with σ eventually constant. Likewise, $\mathcal{L}$-MP^2 is known to capture σ-GNNs for such σ [9]. Our comparison therefore shows that adding the identity activation strictly increases expressivity for uneven eventually constant activations (even for Boolean queries).

Proposition 5.8. *Let Σ, Σ' be sets of uneven eventually constant activation functions. Then Σ'-GNNs are Boolean strictly contained in $(\Sigma \cup \{\mathsf{id}\})$-GNNs over k-coloured graphs (even for $k = 2$).*

Theorem 5.5 and Proposition 5.8 are the exact type of results that motivate the study of GNNs from the perspective of query languages. We see how the formal study of these query languages provides real insight for the design of GNNs. Concretely, these two results show that combining multiple activation functions (of the form in Theorem 5.5) cannot increase expressivity. That is, a $\{\mathsf{TrReLU}, \mathsf{sign}, \mathsf{id}\}$-GNN is just as powerful as a $\{\mathsf{TrReLU}, \mathsf{id}\}$-GNN. Yet, the presence of linear layers in GNNs does increase expressivity.

6 Beyond Bounded Activation Functions

A natural next question is how the expressive power of MPLang changes once we allow *unbounded* activation functions. In what follows, we fix the activation to be ReLU. Recall from Proposition 3.1 that ReLU-MPLang captures exactly ReLU-GNNs, and more generally σ-MPLang captures $\{\mathsf{id}, \sigma\}$-GNNs. Thus, the question we are facing is:

How do ReLU-GNNs and $\{\mathsf{id}, \sigma\}$-GNNs compare in expressive power, when σ ranges over eventually constant activation functions?

This turns out to be a delicate problem. For Boolean queries, Benedikt et al. [9] showed that, on k-coloured graphs, rational ReLU-GNNs are strictly more expressive than rational $\{\sigma\}$-GNNs. However, by Proposition 5.8, σ-MPLang (or, equivalently, $\{\sigma, \mathsf{id}\}$-GNNs) is strictly more expressive than $\{\sigma\}$-GNNs, so the comparison between ReLU-GNNs and $\{\sigma, \mathsf{id}\}$-GNNs requires a fresh analysis.

Our main result shows that adding ReLU increases the *numerical* expressive power of MPLang.

Theorem 6.1. *For any set of uneven eventually constant activation functions Σ, rational ReLU-MPLang numerically strictly contains rational Σ-MPLang on k-coloured graphs.*

The proof of this theorem hinges on the following stronger separation, which does not rely on rational coefficients.

Proposition 6.2. *ReLU-MPLang numerically strictly contains bool-MPLang on k-coloured graphs.*

Combined with Theorem 5.5, which shows that all eventually constant activation functions are equivalent to bool, the proposition yields Theorem 6.1 immediately.

The following important corollary of Proposition 6.2 constitutes an important, non-trivial extension of the aforementioned result by Benedikt et al. stating that TrReLU-GNNs are numerically strictly contained in ReLU-GNNs [9].

Corollary 6.3. *ReLU-GNNs numerically strictly contain $\{$TrReLU, $\mathsf{id}\}$-GNNs on k-coloured graphs.*

The following subsection is dedicated to the proof of Proposition 6.2.

6.1 Proof of Proposition 6.2

We start from a very simple numerical query, definable in ReLU-MPLang:

$$Q \coloneqq \mathsf{ReLU}\big(\oplus\mathsf{red} - \oplus\mathsf{blue}\big),$$

which counts how many more red than blue neighbours a node has, truncated at 0. Our aim is to show that Q is *not* definable in bool-MPLang. This will show the desired strictness in Proposition 6.2.

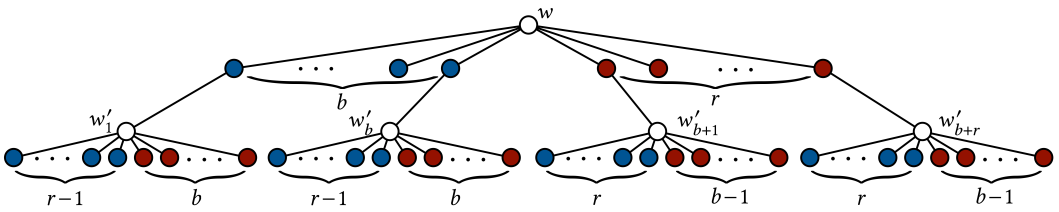


Fig. 3. The red–blue symmetric tree $T[r, b, 3]$.

Red–blue symmetric trees. Our argument uses a family of highly symmetric trees whose root can "see" the imbalance between red and blue children through the numerical power of ReLU. For readability, we introduce white nodes as a third colour, but note that the following proof is exactly analogous if all of the white nodes in the construction get coloured red. In total, we only require two colours, which can be realized by a 1-colouring (which assigns 1 to red nodes, 0 to blue nodes).

Fix $r, b, k \in \mathbb{N}$. We write $T[r, b, k]$ for the *red–blue symmetric tree* with parameters (r, b, k). To define $T[r, b, k]$, it is convenient to first introduce a basic gadget. Let $H[r, b]$ be the star graph consisting of a single white node with r red children and b blue children. Intuitively, $T[r, b, k]$ is obtained by gluing together copies of $H[r, b]$ and $H[b, r]$ in such a way that red and blue nodes always "sit between" two white nodes of different types. Formally, $T[r, b, k]$ is the unique finite rooted tree of height k satisfying the following properties. There are four kinds of nodes:

- *w-type* white nodes: they have exactly r red and b blue neighbours;
- *w′-type* white nodes: they have exactly b red and r blue neighbours;
- red nodes: they have exactly one w-type and one w'-type white neighbour;
- blue nodes: they have exactly one w-type and one w'-type white neighbour.

Moreover, the root is a w-type white node. We remark that all nodes lie at distance at most k from the root. See Figure 3 for the case $k = 3$.

The parameter k will always be chosen to bound the $\oplus$-depth of the expressions we evaluate: if e has $\oplus$-depth at most k, then at each node v the value $e(T[r, b, k])(v)$ only depends on the radius-k neighbourhood of v, which is completely contained in $T[r, b, k]$ and isomorphic for all nodes of the same type. Whenever k is fixed and understood from context, we write

$$e(v) \;\coloneqq\; e\big(T[r, b, k]\big)(v)$$

for the value of e at a node v of $T[r, b, k]$.

Nice functions on (r, b). We now describe the shape of the numerical functions that arise when evaluating bool-MPLang-expressions on $T[r, b, k]$. The key point is that, because bool is eventually constant and the tree is highly symmetric, the dependence on (r, b) is very tame.

Definition 6.4 (Simple and symmetric functions)**.** A function $\beta : \mathbb{R}^n \to \mathbb{R}$ is called *simple* if there exist real constants $a_1, \ldots, a_m$ and a partition $\mathbb{R}^n = I_1 \dot\cup \ldots \dot\cup I_m$ such that

$$\beta(\mathbf{x}) = a_j \qquad \text{for all } \mathbf{x} \in I_j.$$

A function $s : \mathbb{R}^2 \to \mathbb{R}$ is *symmetric* if $s(x, y) = s(y, x)$ for all $x, y \in \mathbb{R}$. □

Thus simple functions take only finitely many values, and symmetric functions are invariant under swapping their arguments.

Definition 6.5 (Nice functions)**.** A function $F : \mathbb{R}^2 \to \mathbb{R}$ is called *nice* if it can be written as

$$F(x, y) = \sum_{i=1}^{m} p_i(x, y)\, \beta_i(x, y),$$

where each p_i is a polynomial and each β_i is simple and symmetric. □

We observe that nice functions are closed under addition and multiplication by polynomials.

What bool-MPLang can see on $T[r, b, k]$. We now show that, on the red–blue symmetric trees $T[r, b, k]$, every bool-MPLang-expression has an extremely rigid form.

Lemma 6.6. *Let e be a bool-MPLang-expression of $\oplus$-depth at most k. Then there exist*

- *nice functions $F, S_r, S_b : \mathbb{R}^2 \to \mathbb{R}$, where S_r and S_b are symmetric,*
- *an affine function $\alpha : \mathbb{R}^2 \to \mathbb{R}$, and*
- *a simple function $\beta : \mathbb{R}^2 \to \mathbb{R}$,*

such that, for all $r, b \in \mathbb{N}$ and all nodes v of $T[r, b, k]$:

- *if v is red, then $e(v) = S_r(r, b)$;*
- *if v is blue, then $e(v) = S_b(r, b)$;*
- *if v is a w-type white node, then $e(v) = F(r, b) + \alpha(r, b) + \beta(r, b)$;*
- *if v is a w'-type white node, then $e(v) = F(b, r) + \alpha(b, r) + \beta(b, r)$.*

Proof sketch. The proof is by structural induction on e. The base cases are easy: constants and colour predicates P_i clearly give rise to constant functions, which are both nice and symmetric. The inductive steps for scalar multiplication and addition use the closure of nice, affine and simple functions under these operations. The only interesting cases are the applications of bool and of $\oplus$. For $e = \mathrm{bool}(e')$, the induction hypothesis provides functions of the desired shape for e'. Applying bool then simply replaces these functions by simple ones: on each region where e' is constant, $\mathrm{bool}(e')$ is also constant. Symmetry is preserved because bool is applied pointwise. For $e = \oplus e'$, we distinguish the four node types. Red and blue nodes each have one w-type and one w'-type neighbour. The value of $\oplus e'$ at such a node is therefore the sum of the values at these two neighbours, which yields a symmetric nice function in (r, b) by the induction hypothesis and closure properties. On a w-type node, the value of $\oplus e'$ is a linear combination of $S_r(r, b)$ and $S_b(r, b)$ with coefficients r and b. This again yields a nice function of (r, b). The case of w'-type nodes is obtained from this by swapping r and b, which explains the occurrence of $F(b, r)$, $\alpha(b, r)$ and $\beta(b, r)$. A straightforward bookkeeping of the affine and simple parts gives the desired shape. □

In particular, evaluating e at the root of $T[r, b, k]$ always yields a value of the form

$$e(\mathrm{root}) = F(r, b) + \alpha(r, b) + \beta(r, b),$$

for some nice F, affine α, and simple β depending only on e. We recall that the parameter k bounds the $\oplus$-depth of e, ensuring the root "sees" precisely the portion of the tree we control.

Nice functions cannot encode $|r - b|$. The next lemma is the technical heart of the argument: functions of the above form cannot coincide with the absolute difference $|r - b|$ over the integers.

Lemma 6.7. *Let $F : \mathbb{R}^2 \to \mathbb{R}$ be nice, let $\alpha : \mathbb{R}^2 \to \mathbb{R}$ be affine, and let $\beta : \mathbb{R}^2 \to \mathbb{R}$ be simple. Then there exist $x, y \in \mathbb{N}$ such that*

$$F(x, y) + \alpha(x, y) + \beta(x, y) \;\neq\; |x - y|.$$

Proof sketch. Write $F(x,y) = \sum_{j=1}^{k} p_j(x,y)\,\beta_j(x,y)$ with p_j polynomials and β_j simple symmetric. Assume for contradiction that

$$G(x,y) \coloneqq F(x,y) + \alpha(x,y) + \beta(x,y) = |x-y| \quad \text{for all } (x,y) \in \mathbb{N}^2.$$

Fix $\delta \in \mathbb{N}$ and consider the diagonal

$$L_\delta \coloneqq \{(m+\delta, m) \mid m \in \mathbb{N}\}.$$

Along L_δ we have $G(m+\delta, m) = \delta$, while the tuple

$$\big(\beta_1(m+\delta,m), \ldots, \beta_k(m+\delta,m), \beta(m+\delta,m)\big)$$

takes only finitely many values. Hence, on an infinite subset of L_δ, all these simple functions are constant: there exist real constants $c_1(\delta), \ldots, c_k(\delta)$ and b_δ such that $\beta_j(m+\delta,m) = c_j(\delta)$ for all j and $\beta(m+\delta,m) = b_\delta$ on that subset. On this subset, G reduces to a single polynomial

$$Q_\delta(r,b) = p_0(r,b) + \sum_{j=1}^{k} c_j(\delta)\,p_j(r,b)$$

satisfying $Q_\delta(m+\delta,m) + b_\delta = \delta$, hence

$$Q_\delta(x+\delta,x) \equiv \delta - b_\delta,$$

where $\equiv$ denotes equality of polynomials on the entire domain. Using the symmetry of the β_j and again the finiteness of the range of β, we similarly obtain

$$Q_\delta(x,x+\delta) \equiv \delta - b'_\delta$$

for some constant b'_δ.

Also when δ varies, the finite tuple $(c_1(\delta), \ldots, c_k(\delta), b_\delta, b'_\delta)$ can take only finitely many values. Thus there exists an infinite set $D \subseteq \mathbb{N}$ and fixed constants $c_1, \ldots, c_k, b, b'$ such that for all $d \in D$ the corresponding polynomial Q_d equals a single polynomial

$$Q(r,b) \coloneqq p_0(r,b) + \sum_{j=1}^{k} c_j\,p_j(r,b),$$

and satisfies

$$Q(x+d,x) \equiv d - b, \qquad Q(x,x+d) \equiv d - b'.$$

Set $u = x - y$ and $v = y$ and define $\widetilde{Q}(u,v) \coloneqq Q(u+v,v)$. For each $d \in D$ we have $\widetilde{Q}(d,v) = Q(v+d,v) \equiv d - b$, which is constant in v. Hence $\widetilde{Q}(u,v)$ does not depend on v, so there is a univariate polynomial C such that $Q(x,y) = C(x-y)$ depends only on $x-y$. For $d \in D$ this gives $C(d) = d - b$, and using $Q(x,x+d) \equiv d - b'$ also $C(-d) = d - b'$. Thus

$$C(d) + C(-d) = 2d - (b + b').$$

The left-hand side is an even polynomial in d, whereas the right-hand side is not. The equality holds for infinitely many d values, so they should be also equal as polynomials, which is impossible. This is our contradiction. □

Table 1. Graphical summary of results by *query type* (vertical) and *input setting* (horizontal). Results highlighted in gray represent ideal placement (most restrictive for negative results, least restrictive for positive results).

	d-embedded graphs	coloured graphs
numerical	**A-MPLang** is equivalent to affine combinations of walk-summed features (Thm. 4.2)	Σ-**MPLang** $\equiv$ **bool-MPLang** for Σ eventually constant and rational coefficients (Thm. 5.5) **ReLU-MPLang** $\supsetneq$ Σ-**MPLang** for Σ eventually constant and rational coefficients (Thm. 6.1) **ReLU-MPLang** $\supsetneq$ **bool-MPLang** (Prop. 6.2)
Boolean	**bool-MPLang**$_{\mathbb{B}}$ is closed under Boolean operations; ML connectives and thresholds (Sec. 5)	**A-MPLang**$_{\mathbb{B}}$ with rational coefficients is closed under $\neg$, but not under $\wedge$ nor Boolean $\Diamond$ (Prop. 4.5, Prop. 4.6) **A-MPLang**$_{\mathbb{B}}$ with real coefficients is not closed under $\neg$ (Prop. 4.7) **A-MPLang**$_{\mathbb{B}}$ **and ML** incomparable on coloured graphs (Remark 4.8) $\mathcal{L}$-**MP**^{2} $\subsetneq$ **bool-MPLang** (Prop. 5.7)

ReLU beats eventually constant activations. We can now put everything together.

Theorem 6.8. *The following numerical query is not expressible in bool-MPLang:*

$$Q \coloneqq \mathsf{ReLU}\big(\oplus\mathsf{red} - \oplus\mathsf{blue}\big).$$

Proof. Suppose, towards a contradiction, that there is a bool-MPLang-expression e defining the numerical query Q. By exchanging the roles of red and blue in e we obtain another bool-MPLang-expression e' that defines the query $\mathsf{ReLU}\big(\oplus\mathsf{blue} - \oplus\mathsf{red}\big)$. Consider now the bool-MPLang-expression $e^* \coloneqq e - e'$. By construction, e^* defines the absolute difference $\big|\oplus\mathsf{red} - \oplus\mathsf{blue}\big|$ at every node. Let k be the $\oplus$-depth of e^*. Evaluating e^* at the root of $T[r, b, k]$ only depends on the radius-k neighbourhood of the root, so Lemma 6.6 applies: there exist a nice function F, an affine function α and a simple function β such that $e^*(\text{root of } T[r, b, k]) = F(r, b) + \alpha(r, b) + \beta(r, b)$ for all $r, b \in \mathbb{N}$. On the other hand, by the definition of $T[r, b, k]$, the root has exactly r red and b blue neighbours, so $e^*(\text{root of } T[r, b, k]) = |r - b|$. This contradicts Lemma 6.7. □

6.2 Comparison with the separation of ReLU-GNNs from TrReLU-GNNs.

Let us once more clarify the relationship between our results in this section and the work by Benedikt et al. [9]. They separate ReLU-GNNs from TrReLU-GNNs *without* linear layers by showing that the query $\oplus\oplus$ green $= \oplus\oplus$ blue is expressible by ReLU-GNNs but not by TrReLU-GNNs, where they could restrict attention to graphs of depth two. In contrast, our separation must handle identity (linear) layers, and as proven in Proposition 5.7, we *can* express said query in bool-MPLang, thus with $\{\mathsf{TrReLU}, \mathrm{id}\}$-GNNs. Moreover, on graphs of bounded depth, the query of Theorem 6.8 is in fact expressible (proof omitted). This explains why our proof must consider graphs of arbitrary depth.

7 Conclusion and Open Problems

A summary of our results and how they align with our assumptions on the types of embeddings and queries is in Table 1. Ideal placements (most restrictive for negative results, least restrictive for positive results), are slightly gray-shaded.

Small design choices in message-passing—especially activation functions—can significantly affect expressive behaviour of GNNs. We have seen that MPLang provides an attractive and transparent logical framework to investigate this expressivity, not only for boolean queries, but also for numerical queries. Our work confirms that expressive power in GNN-style models is best viewed through the interaction of basic numerical operations and local graph structure, rather than architectural templates alone. A more systematic account of this perspective may help clarify which features truly matter for graph-based learning.

From a practical perspective, our results suggest that two architectural choices matter most: the activation class and whether identity layers are available. We show that for all uneven eventually constant activations, expressivity collapses to a common class, so combining several such activations does not by itself increase expressivity. By contrast, adding identity layers can increase Boolean expressivity, and unbounded activations (such as ReLU) are strictly stronger for numerical queries. In this sense, our results provide concrete and actionable information for a critical aspect of GNN architecture design.

Several natural questions remain open. Theorem 6.8 gives a numerical query on coloured graphs expressible in ReLU-MPLang but not in bool-MPLang. Does there exist a boolean such query? Also, while some of our positive results are valid over d-embeddings in general, Theorem 5.5 is stated over coloured graphs only. Does there similarly exist a "universal" activation function for the uneven eventually constant activation functions, on arbitrary d-embeddings?

There are also several directions for further research. Of course, the existing body of work on expressiveness of machine learning models is huge, so we focus on directions that directly link to our work presented here. For example, how does the expressive power of MPLang with ReLU compare to that with other unbounded activation functions? Also, even for bounded activation functions (cf. Remark 5.6), does expressiveness formally shift when considering "soft" activation functions such as sigmoid or tanh?

Finally, we believe numerical logic frameworks for graph embeddings, and methods for them such as those developed in this paper, may be a good starting point to understand expressiveness issues underlying the recent interest in algorithm learning (e.g., [37, 45, 46]).

Acknowledgments

Barceló is funded by ANID - Millennium Science Initiative Program - Code ICN17002, and also by the National Center for Artificial Intelligence CENIA FB210017, Basal ANID. Geerts is supported by FWO project G019222N. Pakhomenko and Van den Bussche are supported by the Flanders AI Program (FAIR). Lanzinger is supported by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT2201, 10.47379/VRG18013].

References

[1] Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. 2024. Logical characterizations of recurrent graph neural networks with reals and floats. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*. https://openreview.net/forum?id=atDcnWqG5n

[2] Renzo Angles, Marcelo Arenas, Pablo Barcelo, Peter Boncz, George Fletcher, Claudio Gutierrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan Sequeda, Oskar van Rest, and Hannes Voigt. 2018. G-CORE: A Core for Future Graph Query Languages. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. Association for Computing Machinery, 1421–1432. https://doi.org/10.1145/3183713.3190654

[3] Pablo Barceló. 2013. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. Association for Computing Machinery, 175–188. https://doi.org/10.1145/2463664.2465216

[4] Pablo Barceló, Diego Figueira, and Miguel Romero. 2019. Boundedness of Conjunctive Regular Path Queries. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP) (LIPIcs, Vol. 132)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 104:1–104:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.104

[5] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. 2020. The Logical Expressiveness of Graph Neural Networks. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. OpenReview.net. https://openreview.net/forum?id=r1lZ7AEKvB

[6] Pablo Barceló, Leonid Libkin, Anthony W. Lin, and Peter T. Wood. 2012. Expressive Languages for Path Queries over Graph-Structured Data. *ACM Trans. Database Syst.* 37, 4 (2012), 31:1–31:46. https://doi.org/10.1145/2389241.2389250

[7] Pablo Barceló, Floris Geerts, Matthias Lanzinger, Klara Pakhomenko, and Jan Van den Bussche. 2025. A Logical View of GNN-Style Computation and the Role of Activation Functions. arXiv:2512.19332 [cs.LG] https://arxiv.org/abs/2512.19332

[8] Timon Barlag, Vivian Holzapfel, Laura Strieker, Jonni Virtema, and Heribert Vollmer. 2024. Graph Neural Networks and Arithmetic Circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems (NeurIPS)*. https://openreview.net/forum?id=0ZeONp33f0

[9] Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan. 2024. Decidability of Graph Neural Networks via Logical Characterizations. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP) (LIPIcs, Vol. 297)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 127:1–127:20. https://doi.org/10.4230/LIPIcs.ICALP.2024.127

[10] Robert Brijder, Floris Geerts, Jan Van Den Bussche, and Timmy Weerwag. 2019. On the Expressive Power of Query Languages for Matrices. *ACM Trans. Database Syst.* 44, 4, Article 15 (2019), 31 pages. https://doi.org/10.1145/3331445

[11] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2000. Containment of Conjunctive Regular Path Queries with Inverse. In *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR)*. Morgan Kaufmann, 176–185.

[12] Mariano P. Consens and Alberto O. Mendelzon. 1990. GraphLog: a Visual Formalism for Real Life Recursion. In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. Association for Computing Machinery, 404–416. https://doi.org/10.1145/298514.298591

[13] Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. 1987. A graphical query language supporting recursion. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data (SIGMOD)*. Association for Computing Machinery, 323–330. https://doi.org/10.1145/38713.38749

[14] Katrin M. Dannert and Erich Grädel. 2021. Semiring Provenance for Guarded Logics. In *Hajnal Andréka and István Németi on Unity of Science: From Computing to Relativity Theory Through Algebraic Logic*. Springer, 53–79. https://doi.org/10.1007/978-3-030-64187-0_3

[15] Cristina Feier, Tomasz Gogacz, and Filip Murlak. 2024. Evaluating Graph Queries Using Semantic Treewidth. In *Proceedinfs of 27th International Conference on Database Theory (ICDT) (Leibniz International Proceedings in Informatics, Vol. 290)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 22:1–22:20. https://doi.org/10.4230/LIPIcs.ICDT.2024.22

[16] Diego Figueira. 2020. Containment of UC2RPQ: The Hard and Easy Cases. In *Proceedings of the 23rd International Conference on Database Theory (ICDT) (Leibniz International Proceedings in Informatics, Vol. 155)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 9:1–9:18. https://doi.org/10.4230/LIPIcs.ICDT.2020.9

[17] Diego Figueira, Adwait Godbole, S. Krishna, Wim Martens, Matthias Niewerth, and Tina Trautner. 2020. Containment of Simple Conjunctive Regular Path Queries. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. 371–380. https://doi.org/10.24963/kr.2020/38

[18] Diego Figueira and Rémi Morvan. 2025. Semantic Tree-Width and Path-Width of Conjunctive Regular Path Queries. *Logical Methods in Computer Science* Volume 21, Issue 1 (2025). https://doi.org/10.46298/lmcs-21(1:21)2025

[19] Diego Figueira, Rémi Morvan, and Miguel Romero. 2025. Minimizing Conjunctive Regular Path Queries. *Proc. ACM Manag. Data* 3, 2, Article 100 (2025), 25 pages. https://doi.org/10.1145/3725237

[20] Diego Figueira and Varun Ramanathan. 2022. When is the Evaluation of Extended CRPQ Tractable?. In *Proceedings of the 41st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS)*. Association for Computing Machinery, 203–212. https://doi.org/10.1145/3517804.3524167

[21] Nadime Francis, Amélie Gheerbrant, Paolo Guagliardo, Leonid Libkin, Victor Marsault, Wim Martens, Filip Murlak, Liat Peterfreund, Alexandra Rogova, and Domagoj Vrgoč. 2023. A Researcher's Digest of GQL. In *Proceedings of the 26th International Conference on Database Theory (ICDT) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 255)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 1:1–1:22. https://doi.org/10.4230/LIPIcs.ICDT.2023.1

[22] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An Evolving Query Language for Property Graphs.

In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. Association for Computing Machinery, 1433–1445. https://doi.org/10.1145/3183713.3190657

[23] Dominik D. Freydenberger and Nicole Schweikardt. 2013. Expressiveness and static analysis of extended conjunctive regular path queries. *J. Comput. System Sci.* 79, 6 (2013), 892–909. https://doi.org/10.1016/j.jcss.2013.01.008

[24] Floris Geerts and Juan L Reutter. 2022. Expressiveness and approximation properties of graph neural networks. In *ICLR*. OpenReview.net.

[25] Floris Geerts, Jasper Steegmans, and Jan Van den Bussche. 2022. On the Expressive Power of Message-Passing Neural Networks as Global Feature Map Transformers. In *FoIKS 2022*, Vol. 13388. Springer, 20–34.

[26] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1263–1272.

[27] Martin Grohe. 2021. The Logic of Graph Neural Networks. In *LICS*. IEEE, 1–17.

[28] Martin Grohe. 2024. The Descriptive Complexity of Graph Neural Networks. *TheoretiCS* 3 (2024).

[29] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1025–1035.

[30] Steve Harris and Andy Seaborne. 2013. SPARQL 1.1 Query Language. https://www.w3.org/TR/sparql11-query/. W3C Recommendation.

[31] Stan P. Hauke and Przemyslaw Andrzej Walega. 2025. Aggregate-Combine-Readout GNNs Are More Expressive Than Logic C2. *CoRR* abs/2508.06091 (2025).

[32] Sammy Khalife and Amitabh Basu. 2023. On the power of graph neural networks and the role of the activation function. *CoRR* abs/2307.04661 (2023).

[33] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[34] Leonid Libkin, Wim Martens, and Domagoj Vrgoc. 2016. Querying Graphs with Data. *J. ACM* 63, 2 (2016), 14:1–14:53.

[35] Alberto O. Mendelzon and Peter T. Wood. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Comput.* 24, 6 (1995), 1235–1258.

[36] Matthew Morris and Ian Horrocks. 2025. Sound Logical Explanations for Mean Aggregation Graph Neural Networks. In *NeurIPS*. OpenReview.net.

[37] R.R. Nerem et al. 2025. Graph neural networks extrapolate out-of-distribution for shortest paths. arXiv:2503.19173.

[38] Pierre Nunn, Marco Sälzer, François Schwarzentruber, and Nicolas Troquard. 2024. A logic for reasoning about aggregate-combine graph neural networks. *arXiv preprint arXiv:2405.00205* (2024).

[39] Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. 2017. Regular Queries on Graph Databases. *Theory Comput. Syst.* 61, 1 (2017), 31–83.

[40] Miguel Romero, Pablo Barceló, and Moshe Y. Vardi. 2017. The homomorphism problem for regular graph patterns. In *LICS*. IEEE Computer Society, 1–12.

[41] Marco Sälzer, Przemyslaw Andrzej Walega, and Martin Lange. 2025. The Logical Expressiveness of Temporal GNNs via Two-Dimensional Product Logics. In *NeurIPS*. OpenReview.net.

[42] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks* 20, 1 (2009), 61–80.

[43] H.T. Siegelmann and E.D. Sontag. 1995. On the computational power of neural nets. *J. Comput. System Sci.* 50, 1 (1995), 132–150.

[44] Arie Soeteman and Balder ten Cate. 2025. Logical Expressiveness of Graph Neural Networks with Hierarchical Node Individualization. In *NeurIPS*. OpenReview.net.

[45] P. Velickovic and Ch. Blundell. 2021. Neural algorithmic reasoning. *Patterns* 2 (2021). https://doi.org/10.1016/j.patter.2021.100273

[46] Solveig Wittig, Antonis Vasileiou, Robert R. Nerem, Timo Stoll, Floris Geerts, Yusu Wang, and Christopher Morris. 2026. Which Algorithms Can Graph Neural Networks Learn? arXiv:2602.13106. https://arxiv.org/abs/2602.13106

Received December 2025; accepted February 2026