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

How Aggregation Functions Affect the Uniform Expressiveness of Graph Neural Networks

Stan P Hauke[1] , **Przemysław Andrzej Wałęga**[2]
[1]Department of Informatics, King's College London, UK
[2]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
stanislaw.hauke@kcl.ac.uk, p.walega@qmul.ac.uk

Abstract

We analyse how the choice of the aggregation function in graph neural networks (GNNs) affects their uniform expressiveness. We show that arbitrary aggregation yields expressiveness of infinitary graded modal logic. Expressiveness strictly decreases when moving from arbitrary aggregation to sum, from sum to mean, and from mean to max. When GNNs are equipped with global readout and arbitrary aggregation, they have the expressiveness of infinitary C^2 logic and restricting aggregation to sum does not decrease their expressiveness. However, still, the expressiveness strictly decreases when moving from sum aggregation to mean, and from mean to max. In the case of simple GNNs, where combination functions are linear transformations followed by a non-linearity and the classification function is a threshold function, the landscape differs. In particular GNNs with sum aggregation and readout no longer have the expressiveness of full infinitary C^2. These results provide us with new insights on the expressiveness of GNNs, showing that even subtle architectural modifications can significantly influence their expressive power.

1 Introduction

Graph neural networks (GNNs) are specialised machine learning models for processing graph-structured data (Gilmer et al. 2017). They achieve state-of-the-art performance in a number of graph learning tasks, including graph and node classification or link prediction (Besharatifard and Vafaee 2024; Derrow-Pinion et al. 2021; Chen et al. 2024; Zhang and Chen 2018; Huang et al. 2025). Given their practical success, there is growing interest in obtaining a deeper theoretical understanding of the computational mechanisms underpinning this effectiveness.

Recent research has provided a number of results on the expressive power of GNNs, focusing mostly on the standard message-passing architecture, also called *aggregate-combine* GNNs (AC-GNNs) (Barceló et al. 2020). Since computations of AC-GNNs are based on passing messages between neighbour nodes in a graph, they are closely related to well-known graph isomorphism heuristics of colour refinement and Weisfeiler-Leman test, as well as to two variable and modal logics. Indeed, the celebrated result obtained independently by Morris et al. (2019) and Xu et al. (2019) shows that no AC-GNN can distinguish a pair of graphs which is indistinguishable by colour refinement or, equivalently, by the one-dimensional Weisfeiler-Leman test. If colour refinement distinguishes two graphs, there exists an AC-GNN distinguishing them (Xu et al. 2019). Furthermore, for any graph G, there is an AC-GNN that accepts precisely those graphs that colour refinement cannot distinguish from G (Morris et al. 2019).

Later, AC-GNNs have been analysed in terms of their *uniform expressiveness*, which asks about a characterisation of all functions which can be implemented by GNNs. The first result on the logical expressiveness, opening this research line, shows that node classifiers which are both expressible by AC-GNNs and by the first-order logic (FO) are exactly the classifiers expressible in graded modal logic (GML) (Barceló et al. 2020). Moreover ACR-GNNs, obtained by extending AC-GNNs with global readout, can express all properties from C^2—the two-variable fragment of FO with counting quantifiers. However, ACR-GNNs can also express FO properties outside C^2 (Hauke and Wałęga 2026), but the proof of this fact requires using a non-simple architecture.

Further results have characterised the uniform expressiveness of various GNN architectures, using first-order and modal logics (Schönherr and Lutz 2026; Cuenca Grau, Feng, and Wałęga 2026; Ahvonen et al. 2026), counting terms (Grohe 2024; Huang et al. 2023), Presburger quantifiers (Benedikt et al. 2024), linear programming (Nunn et al. 2024), fixpoint operators and infinitary logics (Ahvonen et al. 2024; Pflueger, Cucala, and Kostylev 2024), Datalog rules (Tena Cucala et al. 2023; Tena Cucala and Cuenca Grau 2024; Morris and Horrocks 2025), two-dimensional logics (Sälzer, Wałęga, and Lange 2025), MPLang declarative language (Barceló et al. 2025), and message-passing algorithms (Rosenbluth and Grohe 2026). Such characterisations often rely on specific architectural restrictions, for example imposed on the aggregation, combination, classification, readout, and precision.

In this paper, we take a different direction. Instead of searching for specific architectures that can be matched with logical languages, we consider the basic message-passing GNN architectures and ask the following research question:

- How does the uniform expressiveness of GNNs depend on the form of the aggregation function? In particular, what are the expressive power relations between the architectures in which the aggregation function is an arbitrary

function, summation, mean, and max?

To obtain a more detailed understanding of the impact of aggregation on the relative expressive power, we also consider the following questions:

- How does the landscape of relative expressive power look in the case of simple GNNs—where the combination function is a linear transformation followed by a non-linear activation, and the final classification is a threshold function?
- How does the landscape look in the case of ACR-GNNs, which have a global readout allowing to aggregate over all nodes in a graph?

It is worth observing that the above questions are in line with current research interests in the area. Indeed, the recent work has analysed how different types of aggregation functions in GNNs can approximate one another (Rosenbluth, Toenshoff, and Grohe 2023), but not the relations between their uniform expressiveness. The uniform expressiveness of GNNs with mean aggregation has been studied from the logical perspective (Schönherr and Lutz 2026), but only lower bounds have been provided; tight characterisation has not been achieved so far. This is in contrast to max aggregation, for which tight characterisations are not hard to obtain (Cuenca Grau, Feng, and Wałęga 2026).

Following the research questions presented above, we will consider four main types of GNNs in this paper, namely AC-GNNs, simple AC-GNNs, ACR-GNNs, and simple ACR-GNN, as depicted in Figure 1. In each of these settings, we will analyse relations between the uniform expressiveness of architectures with aggregation being arbitrary, summation, mean, and max functions. Moreover, we will relate the expressiveness of these architectures to infinitary versions of logics C^2 and GML, namely $\mathcal{L}\mathrm{C}^2_\omega$ and $\mathcal{L}\mathrm{GML}_\omega$.

Our results are summarised in Venn diagrams from Figure 1, where "all", "sum", "mean", and "max" stand for the type of the aggregation function allowed in GNNs. In the case of ACR-GNNs, we always assume that the same restriction is imposed on readout functions as on the aggregation. All the expressiveness relations presented pictorially in Figure 1 are strict, in the sense that each region of the induced partitions is non-empty. The only exception is in the cases of simple AC-GNNs and simple ACR-GNNs, where it is not known whether there are properties expressible with mean, but not expressible with sum aggregation. We have marked it with dashed region borders of the diagrams. Our conjecture is that these regions are non-empty, but it remains an open problem to verify this conjecture.

The paper is structured as follows. In Section 2 we present basic notions used in the paper. Then, in Section 3, we discuss the similarities and differences between the colour refinement (cr) and the folklore one-dimensional Weisfeiler-Leman (wl) algorithms, which we link later with the expressiveness of, respectively, AC-GNNs and ACR-GNNs. In Section 4 we show that a class of pointed graphs is closed under ℓ-round cr (resp. wl) if and only if it is definable by a formula of infinitary extension $\mathcal{L}\mathrm{GML}$ of GML (resp. infinitary extension $\mathcal{L}\mathrm{C}^2$ of C^2), with quantifier depth ℓ. We use

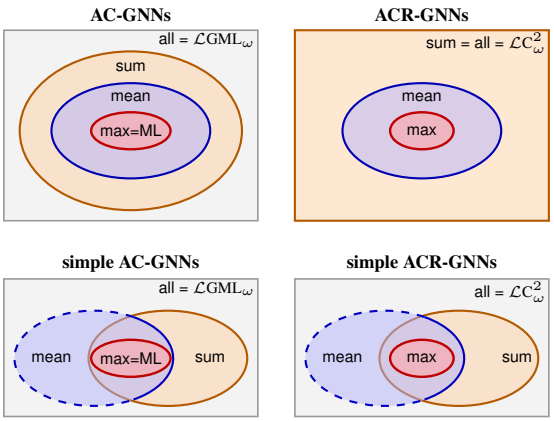

Figure 1: Impact of the choice of the aggregation function on the expressiveness of AC-GNNs and ACR-GNNs

this result to show that when aggregation function is arbitrary, both AC-GNNs and simple AC-GNNs have the same expressiveness as $\mathcal{L}\mathrm{GML}_\omega$, whereas ACR-GNNs and simple ACR-GNNs have the same expressiveness as $\mathcal{L}\mathrm{C}^2_\omega$. If aggregation is summation, the expressiveness strictly drops in three out of four cases. The exception is for ACR-GNNs, where summation aggregation yields the expressiveness of the full $\mathcal{L}\mathrm{C}^2_\omega$. In Section 5 we provide the remaining results comparing the expressiveness of GNNs with sum, mean, and max aggregation. Max leads to strictly lower expressiveness than sum, in all cases, whereas mean leads to strictly lower expressiveness than sum for non-simple GNNs. We also show that max yields strictly lower expressiveness than mean in all cases. In Section 6 we briefly conclude the paper.

2 Preliminaries

Graphs All the *graphs* we consider in the paper are undirected, node-labelled, finite, and simple. A graph of dimension $d \in \mathbb{N}$ is $G = (V, E, \lambda)$, where V is a finite set of nodes, $E \subseteq \{\{v, w\} \mid v, w \in V \text{ and } v \neq w\}$ is a set of undirected edges with no loops, and $\lambda : V \to \{0, 1\}^d$ assigns to each node a binary vector[1] of dimension d. We assume that all vectors are represented as row vectors. We will assume that d is arbitrary but fixed, namely all the graphs we consider are of the same dimension. A *pointed graph* is a pair (G, v) consisting of a graph and one of its nodes. The *neighbourhood*, $N_G(v)$, of a node v in a graph $G = (V, E, \lambda)$, is the set of all nodes w such that $\{v, w\} \in E$.

Isomorphism Heuristics Graphs $G = (V, E, \lambda)$ and $H = (V', E', \lambda')$ are *isomorphic* if there is a bijection $f : V \to V'$, satisfying the following conditions for any $v, w \in V$: (i) $\lambda(v) = \lambda'(f(v))$ and (ii) $\{v, w\} \in E$ if and only if $\{f(v), f(w)\} \in E'$. *Colour refinement*, cr, is a heuristic for

[1]The restriction to binary vectors in graph labels is a standard assumption allowing to compare expressiveness of GNNs and logics (Barceló et al. 2020; Cuenca Grau, Feng, and Wałęga 2026; Tena Cucala and Cuenca Grau 2024)

checking graph isomorphism. Given a graph $G = (V, E, \lambda)$, it iteratively computes values $\mathsf{cr}^{(\ell)}(v)$, for all $v \in V$ and $\ell \in \mathbb{N}$, as follows:

$$\mathsf{cr}^{(0)}(v) \coloneqq \lambda(v),$$
$$\mathsf{cr}^{(\ell+1)}(v) \coloneqq \Big(\mathsf{cr}^{(\ell)}(v), \{\!\{\mathsf{cr}^{(\ell)}(w)\}\!\}_{w \in N_G(v)}\Big),$$

where $\{\!\{\cdot\}\!\}$ stands for a multiset. We let $\mathsf{cr}^{(\ell)}(G) \coloneqq \{\!\{\mathsf{cr}^{(\ell)}(v)\}\!\}_{v \in V}$ be the multiset of colours computed by $\mathsf{cr}^{(\ell)}$ for all nodes in G. If two graphs G and H are isomorphic, then $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$, for any ℓ. However, even if for all ℓ it holds that $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$, the graphs do not need to be isomorphic.

The (folklore) *one-dimensional Weisfeiler-Leman algorithm* (1-WL) (Weisfeiler and Leman 1968; Babai and Kucera 1979; Huang and Villar 2021) is an extension of the colour refinement algorithm. Given a graph $G = (V, E, \lambda)$, the algorithm iteratively computes values $\mathsf{wl}^{(\ell)}(v)$, for all $v \in V$ and $\ell \in \mathbb{N}$, as follows:

$$\mathsf{wl}^{(0)}(v) \coloneqq \lambda(v),$$
$$\mathsf{wl}^{(\ell+1)}(v) \coloneqq \Big(\mathsf{wl}^{(\ell)}(v), \{\!\{(\mathsf{atp}(v, w), \mathsf{wl}^{(\ell)}(w))\}\!\}_{w \in V}\Big),$$

where $\mathsf{atp}(v, w)$ is the atomic type (Grohe 2021), which in the setting of simple undirected graphs is determined by the existence of an edge between v and w, so we let:

$$\mathsf{atp}(v, w) \coloneqq \begin{cases} 1 & \text{if } \{v, w\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Hence, in contrast to $\mathsf{cr}^{(\ell+1)}(v)$, the multiset occurring in the definition of $\mathsf{wl}^{(\ell+1)}(v)$ does not only contain colours of v neighbours, but also colours of v non-neighbours. We let $\mathsf{wl}^{(\ell)}(G) \coloneqq \{\!\{\mathsf{wl}^{(\ell)}(v)\}\!\}_{v \in V}$.

Graph Neural Networks We consider two types of *message-passing graph neural networks* (GNNs): one with *aggregate-combine* (AC) layers and the other with *aggregate-combine-readout* (ACR) layers. An AC layer is a pair $(\mathsf{agg}, \mathsf{comb})$ of *aggregation* and *combination* functions, whereas an ACR layer is a triple $(\mathsf{agg}, \mathsf{comb}, \mathsf{read})$ containing also a *readout* function. Functions agg and read map multisets of vectors into single vectors, whereas comb maps vectors to vectors. An application of a layer to a graph $G = (V, E, \lambda)$ yields a graph $G' = (V, E, \lambda')$ with updated labelling function λ'. For an AC layer, the updated vector $\lambda'(v)$ is computed as

$$\mathsf{comb}\Big(\lambda(v), \mathsf{agg}(\{\!\{\lambda(w)\}\!\}_{w \in N_G(v)})\Big),$$

whereas for an ACR layer, $\lambda'(v)$ is computed as

$$\mathsf{comb}\Big(\lambda(v), \mathsf{agg}(\{\!\{\lambda(w)\}\!\}_{w \in N_G(v)}), \mathsf{read}(\{\!\{\lambda(w)\}\!\}_{w \in V})\Big).$$

Each of the functions agg, comb, and read has a domain and a range of a fixed dimension (but each can have a different dimension), which we refer to as input and output dimensions. These dimensions need to match: if in an AC layer agg has input dimension d and output dimension d', then the input dimension of comb is $d + d'$; in an ACR layer, if additionally read has output dimension d'', then the input dimension of comb is $d + d' + d''$.

A *GNN classifier* $\mathcal{N}$ of dimension d consists of a fixed number L of layers and a single classification function cls from vectors to truth values. The input dimension of the first layer is d and consecutive layers have matching dimensions: the output dimension of layer ℓ matches the input dimension of layer $\ell+1$. An AC-GNN is a GNN with AC layers and an ACR-GNN is a GNN with ACR layers. We write $\lambda^{(\ell)}(v)$ for the vector of node v upon application of layer ℓ, so $\lambda^{(0)}(v)$ is the initial label of v, and $\lambda^{(L)}(v)$ is its final label. The application of a classifier $\mathcal{N}$ to a pointed graph (G, v) is the truth value $\mathcal{N}(G, v) = \mathsf{cls}(\lambda^{(L)}(v))$.

In the most general setting, we do not make any assumptions on the form of functions agg, comb, read, and cls in AC-GNNs and ACR-GNNs. However, we also consider more practical architectures, where specific restrictions are imposed. In particular, we let a *simple* AC-GNN have $\mathsf{comb}(\mathbf{x}, \mathbf{y})$ of the form $\sigma(\mathbf{x}\mathbf{A} + \mathbf{y}\mathbf{C} + \mathbf{b})$, where $\mathbf{A}$,$\mathbf{C}$ are matrices and $\mathbf{b}$ is a vector, all with rational number entries, whereas σ is a position-wise activation function. In the paper we will always use ReLU as the activation function σ, except in Theorem 21 where we allow σ to be a threshold function. Recall that ReLU is the Rectified Linear Unit with $\mathsf{ReLU}(x) = \max(0, x)$ applied coordinate-wise to a vector. Hence, in simple AC-GNNs, $\lambda'(v)$ is computed as

$$\sigma\Big(\lambda(v)\mathbf{A} + \mathsf{agg}(\{\!\{\lambda(w)\}\!\}_{w \in N_G(v)})\mathbf{C} + \mathbf{b}\Big).$$

Moreover, in simple GNNs we will assume that the classification function cls is a *threshold function*, namely $\mathsf{cls}(\mathbf{x}) = 1$ if and only if $x_i \geq t$ (or $x_i > t$) for all positions x_i of the vector $\mathbf{x}$, where $t \in \mathbb{Q}$ is some threshold value. *Simple* ACR-GNNs are defined similarly, but the readout function is multiplied by additional matrix $\mathbf{D}$ with rational entries, so $\lambda'(v)$ is computed as

$$\sigma\Big(\lambda(v)\mathbf{A} + \mathsf{agg}(\{\!\{\lambda(w)\}\!\}_{w \in N_G(v)})\mathbf{C} + \mathsf{read}(\{\!\{\lambda(w)\}\!\}_{w \in V})\mathbf{D} + \mathbf{b}\Big).$$

Furthermore, we consider specific choices for agg and read function, namely the summation `sum`, average `mean`, or maximum `max` functions. We specify the choice of aggregation and readout functions in parentheses after a symbol of the architecture; if it is not specified, then no restriction is imposed. For example AC-GNN(`sum`) are AC-GNN with aggregation being summation. In total, it gives rise to sixteen architectures, namely AC-GNN, AC-GNN(`sum`), AC-GNN(`mean`), AC-GNN(`max`); their simple versions; and another eight architectures obtained by adding the readout function, that is, replacing AC layers with ACR layers.

Logics We consider a language of function-free *first-order logic* (FO) in which formulae are built in the standard way from variables, the identity operator $=$, unary predicates $P_1, \ldots, P_d$ for node colours, the binary predicate E for

edges, the logical connectives $\neg$, $\wedge$, and the quantifier $\exists$. The language of *two-variable logic* (FO^2) allows for formulae of FO with at most two variables. The *two-variable counting logic* (C^2) allows for FO^2 formulae using additional counting quantifiers $\exists_k$ for every $k \in \mathbb{N}$. A formula $\exists_k x\, \varphi(x)$ states that φ holds for at least k distinct elements. Such counting quantifiers are expressible in FO, so we can treat C^2 as a fragment of FO. The language of *guarded fragment* of FO^2 restricts the allowed usage of quantifiers, so that they are always guarded by the edge predicate E. In particular, formulae with quantifiers must be of the form $\exists y(E(x,y) \wedge \varphi(y))$, where $\varphi(y)$ is a guarded FO^2 formula with exactly one free variable y. Similarly, the guarded fragment of C^2 requires that formulae with counting quantifiers are of the form $\exists_k y(E(x,y) \wedge \varphi(y))$, where $\varphi(y)$ is a guarded C^2 formula with exactly one free variable y. Every guarded FO^2 formula is equivalent to a formula of modal logic (ML) and vice versa. Similar equivalence holds between guarded C^2 and graded modal logic (GML). To simplify presentation we do not introduce the syntax and semantics of modal logics, but we will use the symbols ML and GML when referring to the guarded fragments of FO^2 and C^2, respectively.

We consider infinitary languages $\mathcal{L}\mathrm{C}^2$ and $\mathcal{L}$GML obtained from, respectively, C^2 and GML, by allowing for infinitary conjunctions and disjunctions. Furthermore, we consider restrictions of the above languages based on the *quantifier rank* of formulae—the maximal nesting depth of quantifiers in a formula. Given a language L and $\ell \in \mathbb{N}$, we let L_ℓ be the fragment of L consisting of formulae with quantifier rank at most ℓ. If L is an infinitary language, we consider also the fragment L_ω, in which every formula has arbitrarily large, but finite quantifier rank. The following languages will be particularly important for the paper: C^2, C^2_ℓ, $\mathcal{L}\mathrm{C}^2_\ell$, $\mathcal{L}\mathrm{C}^2_\omega$, FO^2, GML, GML_ℓ, $\mathcal{L}\mathrm{GML}_\ell$, $\mathcal{L}\mathrm{GML}_\omega$, and ML. A *logical classifier*, in any of these languages, is a formula $\varphi(x)$ with one free variable.

We interpret logical classifiers $\varphi(x)$ over graphs $G = (V, E, \lambda)$. To this end, we identify a graph of dimension d (i.e. whose vectors have dimension d) with an FO structure $\mathfrak{M} = (V^{\mathfrak{M}}, P_1^{\mathfrak{M}}, \ldots, P_d^{\mathfrak{M}}, E^{\mathfrak{M}})$, where the domain $V^{\mathfrak{M}} = V$ is the set of nodes, each unary predicate P_i is interpreted as the set $P_i^{\mathfrak{M}} = \{v \in V \mid \lambda(v)_i = 1\}$ of nodes with 1 on the ith position of the labelling vector, and the binary relation $E^{\mathfrak{M}}$ corresponds to the edge relation $E^{\mathfrak{M}} = E$. We use the standard FO semantics and write $G \models \varphi(v)$ for a node v, if $\mathfrak{M} \models \varphi(v)$. In this case, we say that classifier $\varphi(x)$ accepts v, or that applied to v it returns true; otherwise, it rejects and returns false. For a language L, we write $G, v \equiv_L H, w$ if $G \models \varphi(v)$ is equivalent to $H \models \varphi(w)$, for every logical classifier $\varphi(x)$ in L. We say that a class $\mathcal{K}$ of pointed graphs is *definable* by a classifier $\varphi(x)$ if $\mathcal{K} = \{(G, v) \mid G \models \varphi(v)\}$.

Expressive Power We consider the *uniform expressiveness* as a measure of expressive power. Recall that both GNNs and logical classifiers are functions mapping pointed graphs into one of the truth values: true or false.

We say that two classifiers are *equivalent* if they compute the same function. A family $\mathcal{F}$ of classifiers is *at most as expressive* as $\mathcal{F}'$, written $\mathcal{F} \leq \mathcal{F}'$, if each classifier in $\mathcal{F}$ has an equivalent one in $\mathcal{F}'$. If $\mathcal{F} \leq \mathcal{F}'$ and $\mathcal{F}' \not\leq \mathcal{F}$, then $\mathcal{F}'$ is *strictly more expressive* than $\mathcal{F}$. If $\mathcal{F} \leq \mathcal{F}'$ and $\mathcal{F}' \leq \mathcal{F}$, we write $\mathcal{F} = \mathcal{F}'$, and say that $\mathcal{F}$ and $\mathcal{F}'$ have the *same expressiveness*.

It is worth emphasising that uniform expressiveness compares classification functions expressible by different types of devices (e.g. GNNs and logics), and not which pairs of graphs can be distinguished. The latter is called the distinguishing or the separating power.

3 Colour Refinement, Weisfeiler-Leman Test, and Logics

In this section, we discuss the relationship between the colour refinement and the (folklore) one-dimensional Weisfeiler-Leman algorithms, as well as their correspondence to logical languages. It will be important for establishing connections between GNNs and infinitary logics in the following section.

First, we recall that colour refinement distinguishes exactly the same pairs of graphs as one-dimensional Weisfeiler-Leman test:

Proposition 1. *(Grohe 2021, Proposition V.4) Let G and H be graphs, and let $\ell \in \mathbb{N}$, then*

$$\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H) \quad \textit{if and only if} \quad \mathsf{wl}^{(\ell)}(G) = \mathsf{wl}^{(\ell)}(H).$$

The above, however, does not mean that cr and wl distinguish the same nodes in graphs. For example in the graphs from Figure 2 we have $\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w)$ for any ℓ, but $\mathsf{wl}^{(\ell)}(v) \neq \mathsf{wl}^{(\ell)}(w)$ for any $\ell \geq 2$.

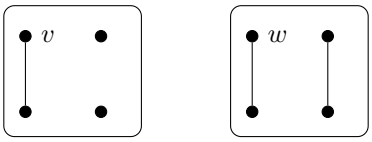

Figure 2: Nodes v and w distinguishable by wl, but not by cr

As we show next, a sufficient condition for wl and cr to distinguish the same nodes is that $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$.

Proposition 2. *Let (G, v) and (H, w) be pointed graphs, and let $\ell \in \mathbb{N}$. If $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$, then*

$$\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w) \quad \textit{if and only if} \quad \mathsf{wl}^{(\ell)}(v) = \mathsf{wl}^{(\ell)}(w).$$

Proof. We prove the equivalence by induction on $i \leq \ell$. The base case follows from the fact that $\mathsf{cr}^{(0)}(z) = \mathsf{wl}^{(0)}(z)$ for any z. In the inductive step, the backwards implication is trivial since $\mathsf{wl}^{(\ell)}$ refines $\mathsf{cr}^{(\ell)}$. For the forward implication assume that $\mathsf{cr}^{(i+1)}(v) = \mathsf{cr}^{(i+1)}(w)$. Hence, $\mathsf{cr}^{(i)}(v) = \mathsf{cr}^{(i)}(w)$ and $\{\!|\mathsf{cr}^{(i)}(z)|\!\}_{z \in N_G(v)} = \{\!|\mathsf{cr}^{(i)}(z)|\!\}_{z \in N_H(w)}$. Since $\mathsf{cr}^{(\ell)}$ refines $\mathsf{cr}^{(i)}$, the assumption $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$ implies that $\mathsf{cr}^{(i)}(G) = \mathsf{cr}^{(i)}(H)$. Together with $\{\!|\mathsf{cr}^{(i)}(z)|\!\}_{z \in N_G(v)} = \{\!|\mathsf{cr}^{(i)}(z)|\!\}_{z \in N_H(w)}$ it implies that $\{\!|\mathsf{cr}^{(i)}(z)|\!\}_{z \notin N_G(v)} = \{\!|\mathsf{cr}^{(i)}(z)|\!\}_{z \notin N_H(w)}$. By the

inductive assumption, we have that if $\mathsf{cr}^{(i)}(v) = \mathsf{cr}^{(i)}(w)$ then $\mathsf{wl}^{(i)}(v) = \mathsf{wl}^{(i)}(w)$ for all v, w. Hence, $\mathsf{wl}^{(i)}(v) = \mathsf{wl}^{(i)}(w)$ and $\{\!|\mathsf{wl}^{(i)}(z)|\!\}_{z \in N_G(v)} = \{\!|\mathsf{wl}^{(i)}(z)|\!\}_{z \in N_H(w)}$, and $\{\!|\mathsf{wl}^{(i)}(z)|\!\}_{z \notin N_G(v)} = \{\!|\mathsf{wl}^{(i)}(z)|\!\}_{z \notin N_H(w)}$. Therefore $\mathsf{wl}^{(i+1)}(v) = \mathsf{wl}^{(i+1)}(w)$. □

Next, we consider the relation between the algorithms cr and wl, and distinguishability in logical languages. The seminal result of Cai, Fürer, and Immerman (1992) shows that $\mathsf{wl}^{(\ell)}$ distinguishes the same nodes as the formulae of C^2_ℓ:

Theorem 3. *(Cai, Fürer, and Immerman 1992, Theorem 5.2) Let (G, v) and (H, w) be pointed graphs, and $\ell \in \mathbb{N}$:*

$$\mathsf{wl}^{(\ell)}(v) = \mathsf{wl}^{(\ell)}(w) \quad \text{if and only if} \quad G, v \equiv_{\mathrm{C}^2_\ell} H, w$$

From this and Proposition 2 we immediately obtain the following corollary.

Corollary 4. *Let G and H be graphs, and let $\ell \in \mathbb{N}$. If $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$, then*

$$\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w) \quad \text{if and only if} \quad G, v \equiv_{\mathrm{C}^2_\ell} H, w$$

for any nodes v and w of G and H, respectively.

Hence, $\mathsf{cr}^{(\ell)}$ algorithm and C^2_ℓ formulae distinguish the same pairs of nodes, whenever $\mathsf{cr}^{(\ell)}(G) = \mathsf{cr}^{(\ell)}(H)$.

Next, we observe that removing this assumption leads to a correspondence between $\mathsf{cr}^{(\ell)}$ and GML_ℓ.

Proposition 5. *Let (G, v) and (H, w) be pointed graphs, and let $\ell \in \mathbb{N}$. Then*

$$\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w) \quad \text{if and only if} \quad G, v \equiv_{\mathit{GML}_\ell} H, w.$$

Proof. The equivalence is implicit in the results of Barceló et al. (2020). By their Observation C.3, $\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w)$ if and only if the tree unravellings $Unr^\ell_G(v)$ and $Unr^\ell_H(w)$ are isomorphic. This in turn, by their Theorem C.5, is equivalent to $G, v \equiv_{\mathrm{GML}_\ell} H, w$. □

In the next section we will exploit the above observations to show the expressive power correspondence between GNNs and infinitary logics.

4 GNNs and Infinitary Logics

Some connections of GNNs and infinitary logics have been already observed in the literature. In particular, Ahvonen et al. (2024), in their Theorem 3.6, have shown that constant-iteration recursive GNNs with arbitrary real functions agg and comb have the same expressive power as an infinitary version of GML. We will provide a similar result for AC-GNNs in our setting, as well as extend this connection to the setting of ACR-GNNs, and to simple GNNs.

We start by observing a connection between infinitary logics and algorithms cr and wl. The following definition will play a crucial role.

Definition 6. *We say that a class $\mathcal{K}$ of pointed graphs is* closed under $\mathsf{cr}^{(\ell)}$*, for $\ell \in \mathbb{N}$, if for any (G, v) and (H, w) the fact that $(G, v) \in \mathcal{K}$ and $\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w)$ implies $(H, w) \in \mathcal{K}$. We define closure under $\mathsf{wl}^{(\ell)}$ in a similar way.*

Models definable by AC-GNNs are closed under cr, and those definable by ACR-GNNs are closed under wl, as stated below.

Proposition 7. *Let $\mathcal{N}$ be an AC-GNN with ℓ layers. The class of pointed graphs definable by $\mathcal{N}$ is closed under $\mathsf{cr}^{(\ell)}$. If $\mathcal{N}$ is an ACR-GNN, then the class is closed under $\mathsf{wl}^{(\ell)}$.*

Proof sketch. For $\mathcal{N}$ being an AC-GNN the result follows from Theorem 1 of Morris et al. (2019), stated more explicitly in Proposition 2.1 of Barceló et al. (2020). If $\mathcal{N}$ is an ACR-GNN we can show by induction $i \leq \ell$ that $\mathsf{wl}^{(i)}(v) = \mathsf{wl}^{(i)}(w)$ implies $\lambda^{(i)}(v) = \lambda^{(i)}(w)$ for any pointed graphs (G, v) and (H, w). Hence $(G, v) \in \mathcal{K}$ and $\mathsf{wl}^{(\ell)}(v) = \mathsf{wl}^{(\ell)}(w)$ implies $(H, w) \in \mathcal{K}$ as required. □

Next, we show that classes of models closed under cr and wl are exactly the classes definable by infinitary versions of GML and C^2, respectively.

Theorem 8. *Let $\mathcal{K}$ be a class of pointed graphs and let $\ell \in \mathbb{N}$. Then $\mathcal{K}$ is:*

- *definable in $\mathcal{L}\mathit{GML}_\ell$ if and only if it is closed under $\mathsf{cr}^{(\ell)}$,*
- *definable in $\mathcal{L}\mathrm{C}^2_\ell$ if and only if it is closed under $\mathsf{wl}^{(\ell)}$.*

Proof sketch. Consider the first Item in the theorem, and assume that $\mathcal{K}$ is definable by some $\varphi_\mathcal{K} \in \mathcal{L}\mathrm{GML}_\ell$. Let us fix pointed graphs (G, v) and (H, w) such that $(G, v) \in \mathcal{K}$ and $\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w)$. We need to show that $(H, w) \in \mathcal{K}$. By $\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w)$ and Proposition 5 we obtain that $G, v \equiv_{\mathrm{GML}_\ell} H, w$. Now, let the *counting rank* of a formula be the greatest integer k appearing in its counting quantifiers $\exists_k$, and let L_c be the restriction of language L to formulae of counting rank at most c. Let us set $c = \max(|G|, |H|) + 1$, where $|G|$ is the number of nodes in G (and similarly for H).

Clearly $G, v \equiv_{\mathrm{GML}_\ell} H, w$ implies $G, v \equiv_{\mathrm{GML}_{\ell,c}} H, w$. Language $\mathrm{GML}_{\ell,c}$ contains only finitely many non-equivalent formulae, so every infinite conjunction of $\mathrm{GML}_{\ell,c}$ formulae is equivalent to a finite one (and the same holds for infinite disjunctions). Hence, $G, v \equiv_{\mathcal{L}\mathrm{GML}_{\ell,c}} H, w$. Moreover, since both G and H have less than c nodes, for any $\mathcal{L}\mathrm{GML}_\ell$ classifier φ we have $G \models \varphi(v)$ if and only if $G \models \varphi'(v)$, where φ' is obtained from φ, by replacing each $\exists_k$ with $k > c$ by $\exists_c$. The same observation is valid for (H, w), and so, $G, v \equiv_{\mathcal{L}\mathrm{GML}_{\ell,c}} H, w$ implies $G, v \equiv_{\mathcal{L}\mathrm{GML}_\ell} H, w$. Since $(G, v) \in \mathcal{K}$, we have $G \models \varphi_\mathcal{K}(v)$. Moreover, as $\varphi_\mathcal{K} \in \mathcal{L}\mathrm{GML}_\ell$, we obtain that $H \models \varphi_\mathcal{K}(w)$, and so, $(H, w) \in \mathcal{K}$.

For the opposite direction assume that $\mathcal{K}$ is closed under $\mathsf{cr}^{(\ell)}$. For a pointed graph (G, v), let its type $t_{G,v}$ be the conjunction of all GML_ℓ classifiers φ such that $G \models \varphi(v)$. We will show that $\mathcal{K}$ is definable by the $\mathcal{L}\mathrm{GML}_\ell$ classifier $\varphi_\mathcal{K} \coloneqq \bigvee_{(G,v) \in \mathcal{K}} t_{G,v}$. Indeed, if $(G, v) \in \mathcal{K}$, then $G \models t_{G,v}(v)$, and so, $G \models \varphi_\mathcal{K}(v)$. Conversely, if $G \models \varphi_\mathcal{K}(v)$, then $G \models t_{H,w}$ for some $(H, w) \in \mathcal{K}$. Hence, $G, v \equiv_{\mathrm{GML}_\ell} H, w$ and so, by Proposition 5, $\mathsf{cr}^{(\ell)}(v) = \mathsf{cr}^{(\ell)}(w)$. Since $\mathcal{K}$ is closed under $\mathsf{cr}^{(\ell)}$ and $(H, w) \in \mathcal{K}$, we obtain that $(G, v) \in \mathcal{K}$.

The second Item of the theorem has a similar proof, but it uses Theorem 3 instead of Proposition 5. □

By Theorem 8 and Proposition 7 we obtain that AC-GNN $\leq \mathcal{L}\text{GML}_\omega$ and ACR-GNN $\leq \mathcal{L}\text{C}^2_\omega$. Next, we show that the opposite relations also hold, so these GNNs can express all classifiers from the corresponding infinitary logics. Moreover, this result holds already for simple GNNs.

Theorem 9. *The following expressive power results hold:*
- *simple AC-GNN = AC-GNN = $\mathcal{L}GML_\omega$,*
- *simple ACR-GNN = ACR-GNN = $\mathcal{L}C^2_\omega$.*

Proof sketch. We focus on the first Item, for which it suffices to show that $\mathcal{L}\text{GML}_\omega \leq$ simple AC-GNN. Assume that input graphs G are of dimension d and let φ be an $\mathcal{L}\text{GML}_\omega$ classifier, so φ is in $\mathcal{L}\text{GML}_\ell$, for some ℓ. We will construct an equivalent simple AC-GNN $\mathcal{N}$. The first layer of $\mathcal{N}$ computes $\lambda^{(1)}(v) = \lambda^{(0)}(v)(2^0, 2^1, \ldots, 2^{d-1})^\top$, so $\lambda^{(1)}(v)$ is the natural number whose binary encoding is $\lambda^{(0)}(v)$. The next ℓ layers compute, for each $2 \leq i \leq \ell + 1$

$$\lambda^{(i)}(v) = \lambda^{(i-1)}(v) \parallel \mathsf{Hash}(\{\!|\lambda^{(i-1)}(w)|\!\}_{w \in N_G(v)}),$$

where $\parallel$ is concatenation and Hash is an injective function from multisets to natural numbers. This can be achieved by a simple AC-GNN because it can use any aggregation, and because it can compute $\mathbf{x} \parallel \mathbf{y}$ if lengths of $\mathbf{x}$ and $\mathbf{y}$ are known. Indeed, if $\mathbf{x} \in \mathbb{Q}^m$ and $\mathbf{y} \in \mathbb{Q}^n$, then $\mathbf{x} \parallel \mathbf{y} = \mathbf{x}\,(\mathbf{I}_n \quad \mathbf{0}_{m\times n}) + \mathbf{y}\,(\mathbf{0}_{n\times m} \quad \mathbf{I}_n)$, where $\mathbf{I}_n \in \mathbb{Q}^{n\times n}$ and $\mathbf{0}_{m\times n} \in \mathbb{Q}^{m\times n}$ are, respectively, identity and zero matrices. Hence, $\lambda^{(\ell+1)}(v)$ is a vector of $\ell+1$ natural numbers. On the first position it has $\lambda^{(1)}(v)$. We can show inductively that on a position $i > 1$ it has an encoding of $\{\!|\mathsf{cr}^{(i-2)}(w)|\!\}_{w\in N_G(v)}$, namely $\mathsf{Hash}'(\{\!|\mathsf{cr}^{(i-2)}(w)|\!\}_{w\in N_G(v)})$, for some injective function Hash'. The argument is based on the observation that each $\mathsf{cr}^{(i)}(v)$ is determined by $\mathsf{cr}^{(0)}(v)$ and $\{\!|\mathsf{cr}^{(i-1)}(w)|\!\}_{w\in N_G(v)}$. Thus $\lambda^{(i)}(v)$ determines $\mathsf{cr}^{(i-2)}(v)$.

It remains to show how a simple AC-GNN can use $\lambda^{(\ell+1)}$ to determine whether $G \models \varphi(v)$, for an input (G, v). By Theorem 8, $G \models \varphi(v)$ is determined by $\mathsf{cr}^{(\ell)}(v)$ and, as we have observed, $\mathsf{cr}^{(\ell)}(v)$ is determined by $\mathsf{cr}^{(0)}(v)$ and $\{\!|\mathsf{cr}^{(\ell-1)}(w)|\!\}_{w\in N(v)}$. Thus, there is a function $f : \mathbb{N} \to \{0,1\}^{2^d}$ that maps encoding $\mathsf{Hash}(M)$ of a multiset M into a vector, such that for every (G, v) with $\{\!|\mathsf{cr}^{(\ell-1)}(w)|\!\}_{w\in N_G(v)} = M$ we have $G \models \varphi(v)$ if and only if the vector $f(M)$ has 1 on the position $\lambda^{(1)}(v)$. Based on this observation, layer $\ell + 2$ will compute $f(\mathsf{Hash}(\{\!|\lambda^{(\ell+1)}(w)|\!\}_{w\in N_G(v)}))$. Then layers $\ell + 3$ and $\ell + 4$ will compute $\mathsf{onehot}(\lambda^{(1)}(v))$ for all v, where $\mathsf{onehot}(n)$ maps n into its one-hot encoding (a vector in $\{0,1\}^{2^d}$). Finally, layer $\ell + 5$ will compute $f(\mathsf{Hash}(\{\!|\lambda^{(\ell+1)}(w)|\!\}_{w\in N_G(v)})) - \mathsf{onehot}(\lambda^{(1)}(v))$. Hence we obtain that $G \models \varphi(v)$ if and only if $\lambda^{(\ell+5)}(v)$ has only non-negative entries. The latter is verified by a final threshold classification function of $\mathcal{N}$. Hence $G \models \varphi(v)$ if and only if $\mathcal{N}(G, v) = \mathsf{true}$.

It remains to show that layers $\ell + 2$ to $\ell + 5$ can be implemented by a simple AC-GNN. To compute $f(\mathsf{Hash}(\{\!|\lambda^{(\ell+1)}(w)|\!\}_{w\in N_G(v)}))$ in layer $\ell + 2$, it suffices to use $f(\mathsf{Hash}(\mathbf{y}))$ as the aggregation function. For layers $\ell + 3$ and $\ell + 4$ it suffices to show how two layers can compute $\mathsf{onehot}(x)$, for $0 \leq x < 2^d$. The first layer computes:

$$(\mathbf{y}, \mathbf{z}) = \mathsf{ReLU}((x \cdot \mathbf{1}_{2^d} - \mathbf{b}) \parallel (\mathbf{b} - x \cdot \mathbf{1}_{2^d})),$$

where $\mathbf{b} = (0, 1, \ldots, 2^d - 1)$ and $\mathbf{1}_{2^d}$ is a vector of 2^d ones. The second layer computes $\mathsf{ReLU}(\mathbf{1}_{2^d} - \mathbf{y} - \mathbf{z})$. We can observe that only on position $x - 1$ both $\mathbf{y}$ and $\mathbf{z}$ have value 0. Thus, the second layer computes $\mathsf{onehot}(x)$. Layer $\ell + 5$ computes subtraction, which is clearly implementable by a simple AC-GNN.

The proof for the second Item is similar, but now the hashing function needs to be applied also to non-neighbours. □

It is worth observing that the proof of the above result relies on the access to arbitrary aggregation functions. As we show next, in the case of AC-GNNs, restricting the form of aggregation functions to summation strictly lowers the expressiveness.

Proposition 10. *AC-GNN($\mathtt{sum}$) < $\mathcal{L}GML_\omega$.*

Proof. By Theorem 9, it suffices to show an $\mathcal{L}\text{GML}_\omega$ classifier which is not expressible in AC-GNN($\mathtt{sum}$). To this end, consider a classifier accepting pointed graphs (G, v) such that v has a neighbour w with $|N_G(v)| < |N_G(w)|$. It is known to be not expressible by AC-GNN($\mathtt{sum}$) (Grohe and Rosenbluth 2024, Theorem 5.1), whereas we can express it with an $\mathcal{L}\text{GML}_\omega$ classifier $\varphi(x) = \bigvee_{k\in\mathbb{N}}(\neg\exists_k y E(x, y) \wedge \exists y\,(E(x, y) \wedge \exists_k x E(y, x)))$. □

Therefore, we obtain the following expressive power results for AC-GNNs.

Corollary 11. *AC-GNN($\mathtt{sum}$) < AC-GNN and simple AC-GNN($\mathtt{sum}$) < simple AC-GNN.*

In the case of ACR-GNNs, the expressiveness relations are different. Interestingly, restricting the form of aggregation to summation does not decrease expressiveness at all.

Theorem 12. *ACR-GNN($\mathtt{sum}$) = $\mathcal{L}C^2_\omega$.*

Proof sketch. It suffices to show $\mathcal{L}\text{C}^2_\omega \leq$ ACR-GNN($\mathtt{sum}$). Let φ be an $\mathcal{L}\text{C}^2_\omega$ classifier, so it is in $\mathcal{L}\text{C}^2_\ell$ for some ℓ. We will construct an equivalent ACR-GNN($\mathtt{sum}$) $\mathcal{N}$ with $\ell + 2$ layers. The first layer computes $\lambda^{(1)}(v) = (1, \lambda^{(0)}(v))$. The next $\ell + 1$ layers will compute $\lambda^{(i)}(v) = (|G|, (|G| + 1)^{\rho_{i-2}(\mathsf{wl}^{(i-2)}(v))})$, where $|G|$ is the size of the input graph and ρ_{i-2} injectively maps values of $\mathsf{wl}^{(i-2)}$ into natural numbers. As we show, this can be done with agg and read being summation $\sum$, together with non-simple comb. Layer 2 uses only comb and read to compute $\lambda^{(2)}(v) = \mathsf{comb}(\lambda^{(1)}(v)_2,\ \sum_{w\in V}\lambda^{(1)}(w)_1) = \mathsf{comb}(\lambda^{(0)}(v), |G|) = (|G|,\ (|G| + 1)^{\rho_0(\lambda^{(0)}(v))}) = (|G|,\ (|G| + 1)^{\rho_0(\mathsf{wl}^{(0)}(v))})$. The remaining ℓ layers compute $\lambda^{(i)}(v)$ as

$$\mathsf{comb}(\lambda^{(i-1)}(v), \sum_{w\in N_G(v)} \lambda^{(i-1)}(w), \sum_{w\in V} \lambda^{(i-1)}(w)),$$

for $\mathsf{comb}(x_1,\dots,x_6) = (x_1,(x_1+1)^{2^{x_2}3^{x_4}5^{x_6}}) = (|G|,(|G|+1)^{2^{x_2}3^{x_4}5^{x_6}})$, where x_2, x_4, and x_6 represent $\mathsf{wl}^{(i-3)}(v)$, $\{\!|\mathsf{wl}^{(i-3)}(w)|\!\}_{w\in N_G(v)}$, and $\{\!|\mathsf{wl}^{(i-3)}(w)|\!\}_{w\in V}$, respectively. The trick that we use here is based on the fact that aggregation and readout are sums of multisets of at most $|G|$ numbers. The form of these sums (using exponentiation of $|G|+1$) allows us to uniquely determine the multisets of the summands. For example, if $|G|+1=10$ and the multiset is $M=\{\!|1,2,2,2,4,4|\!\}$, then $\sum_{a\in M}10^a = 20310$. Since we are adding at most 9 numbers 10^a, the sum 20310 uniquely determines the multiset M. Consequently, x_4 and x_6 injectively encode $\{\!|\mathsf{wl}^{(i-3)}(w)|\!\}_{w\in N_G(v)}$ and $\{\!|\mathsf{wl}^{(i-3)}(w)|\!\}_{w\in V}$, so $2^{x_2}3^{x_4}5^{x_6}$ encodes $\mathsf{wl}^{(i-2)}(v)$.

We obtain that $\lambda^{(\ell+2)}(v)=\lambda^{(\ell+2)}(w)$ if and only if $\mathsf{wl}^{(\ell)}(v)=\mathsf{wl}^{(\ell)}(w)$. By Theorem 8 two pointed graphs are distinguished by $\mathsf{wl}^{(\ell)}$ if and only if they are distinguished by an $\mathcal{LC}^2_\ell$ classifier. Since φ is in $\mathcal{LC}^2_\ell$, there is a classification function cls which applied to $\mathcal{N}$ yields an ACR-GNN($\mathtt{sum}$) classifier equivalent to φ. □

In contrast to ACR-GNNs, restricting aggregation to summation strictly decreases the expressiveness of simple ACR-GNNs, as we observe next.

Proposition 13. *Simple ACR-GNN($\mathtt{sum}$) $< \mathcal{LC}^2_\omega$.*

Proof. By Theorem 9, ACR-GNN($\mathtt{sum}$) $= \mathcal{LC}^2_\omega$, so simple ACR-GNN($\mathtt{sum}$) $\leq \mathcal{LC}^2_\omega$. Observe that simple ACR-GNN($\mathtt{sum}$) can express only computable classifiers, as their aggregation, combination, and classification are all computable. In contrast, $\mathcal{LC}^2_\omega$ can express uncomputable classifiers, e.g. $\varphi(x)=\bigvee_{k\in S}\exists_k y E(x,y)$ for some (arbitrary) uncomputable set S of natural numbers. □

We finish this section by relating expressiveness of GNNs to non-counting logics FO^2 and ML. It is worth observing that due to the lack of counting operators in these logics, we have $\mathcal{L}\mathrm{FO}^2_\ell = \mathrm{FO}^2_\ell$ and $\mathcal{L}\mathrm{ML}_\ell = \mathrm{ML}_\ell$.

Proposition 14. *The following results hold:*
- *simple AC-GNN($\mathtt{max}$) = AC-GNN($\mathtt{max}$) = ML,*
- *ACR-GNN($\mathtt{max}$) $< FO^2$.*

Proof sketch. The first Item follows from Corollary 10 of Cuenca Grau, Feng, and Wałęga (2026) and an observation that their construction uses simple AC-GNN($\mathtt{max}$). The result ACR-GNN($\mathtt{max}$) $\leq FO^2$ follows from their Corollary 15, so it remains to show strictness. To this end consider an FO^2 classifier $\varphi(x)=\exists y\neg E(x,y)$. To show that it is not expressible by any ACR-GNN($\mathtt{max}$) let G be a graph with nodes v,w,u, all labelled by the same vector, and with edges $\{v,w\}$, $\{v,u\}$. For every ACR-GNN($\mathtt{max}$) $\mathcal{N}$ we have $\mathcal{N}(G,v)=\mathcal{N}(G,w)$, but $G\not\models\varphi(v)$ and $G\models\varphi(w)$. □

5 Comparison of Max, Mean, and Sum

In this section, we will compare the uniform expressiveness of GNNs with max, mean, and sum aggregations. We start our analysis by showing that max aggregation is less expressive than sum in all GNN architectures we consider.

Theorem 15. *AC-GNN($\mathtt{max}$) $<$ AC-GNN($\mathtt{sum}$). The same holds if we replace AC-GNN with ACR-GNN, or with simple versions of AC-GNN and ACR-GNN.*

Proof sketch. Recall that by Theorem 14 we have that simple AC-GNN($\mathtt{max}$) = AC-GNN($\mathtt{max}$) = ML. From the proof of Proposition 4.1 by Barceló et al. (2020) we conclude that ML $\leq$ simple AC-GNN($\mathtt{sum}$); this is because in their construction the classifying function can be simulated with a projection onto the last coordinate and a threshold function at $\frac{1}{2}$. Moreover, simple AC-GNN($\mathtt{sum}$) $\not\leq$ ML because $\varphi(x)=\exists_k y E(x,y)$ cannot be expressed in ML, but simple AC-GNN($\mathtt{sum}$) can express it. Thus AC-GNN($\mathtt{max}$) $<$ AC-GNN($\mathtt{sum}$) and simple AC-GNN($\mathtt{max}$) $<$ simple AC-GNN($\mathtt{sum}$).

Next, note that Theorem 14 also shows that simple ACR-GNN($\mathtt{max}$) $\leq$ ACR-GNN($\mathtt{max}$) $< FO^2$. From the proof of Theorem 5.1 by Barceló et al. (2020) we conclude that $FO^2 \leq$ simple ACR-GNN($\mathtt{sum}$); again, this is because their classifying function can be simulated with a projection onto the last coordinate and a threshold function at $\frac{1}{2}$. Thus ACR-GNN($\mathtt{max}$) $<$ ACR-GNN($\mathtt{sum}$) and simple ACR-GNN($\mathtt{max}$) $<$ simple ACR-GNN($\mathtt{sum}$). □

Next we show that mean aggregation, similarly as max, is strictly less expressive than sum aggregation. Note however, that the result below is stated for non-simple GNNs.

Theorem 16. *AC-GNN($\mathtt{mean}$) $<$ AC-GNN($\mathtt{sum}$) and ACR-GNN($\mathtt{mean}$) $<$ ACR-GNN($\mathtt{sum}$).*

Proof sketch. First, we observe that mean aggregation can be simulated using sum by exploiting additional layers and non-simple combination functions. Indeed, to compute $\mathtt{mean}(\{\!|\lambda(w)|\!\}_{w\in N_G(v)})$, a GNN can replace each $\lambda(w)$ with $(\lambda(w),1)$ and then compute $\mathsf{comb}(\mathtt{sum}\{\!|(\lambda(w),1)|\!\}_{w\in N_G(v)})$ for $\mathsf{comb}(x_1,\dots,x_n) = (x_1/x_n,\dots,x_{n-1}/x_n)$. Hence AC-GNN($\mathtt{mean}$) $\leq$ AC-GNN($\mathtt{sum}$). The same idea applies for readout, so ACR-GNN($\mathtt{mean}$) $\leq$ ACR-GNN($\mathtt{sum}$).

To show strictness of these relations, fix $k>1$ and consider a classifier accepting nodes with at least k neighbours. This is expressible by GML classifier $\varphi(x)=\exists_k y E(x,y)$, and so also by an AC-GNN($\mathtt{sum}$) due to results of Barceló et al. (2020). However, $\varphi(x)$ cannot be expressed by an ACR-GNN($\mathtt{mean}$). Indeed, in any ACR-GNN($\mathtt{mean}$) $\mathcal{N}$ we have $\mathcal{N}(K_{k+1},v)=\mathcal{N}(K_k,v)$, for K_i a complete graph of size i. However, $K_{k+1}\models\varphi(v)$ and $K_k\not\models\varphi(v)$. □

Since the separating example from Theorem 16 applies also for the case of simple GNNs, we obtain the following.

Corollary 17. *It holds that simple AC-GNN($\mathtt{sum}$) $\not\leq$ simple AC-GNN($\mathtt{mean}$) and simple ACR-GNN($\mathtt{sum}$) $\not\leq$ simple ACR-GNN($\mathtt{mean}$).*

Next, we mention a hypothesis regarding the relation between simple AC-GNN($\mathtt{mean}$) and simple ACR-GNN($\mathtt{sum}$). We will introduce a property that is expressible in simple AC-GNN($\mathtt{mean}$) but, as we conjecture, cannot be expressed in simple ACR-GNN($\mathtt{sum}$).

Let the P-ratio of a node be the fraction of its neighbours that satisfy a unary predicate P. We define the property $\text{AVG}_{\text{ratio}}(v)$ to be true if the average P-ratio of the neighbours of v is greater than $\frac{1}{2}$.

We now explain how to construct a two-layer simple AC-GNN$(\texttt{mean})$ equivalent to $\text{AVG}_{\text{ratio}}$. The first layer uses neighbourhood aggregation to compute the P-ratio for each node. The second layer uses aggregation to compute the average of these ratios over the neighbours. Finally, a threshold function accepts a node if the second layer computes a value greater than $\frac{1}{2}$ for it.

Our conjecture, as stated below, is that $\text{AVG}_{\text{ratio}}$ cannot be expressed by simple ACR-GNN$(\texttt{sum})$, which would imply that simple AC-GNN$(\texttt{mean}) \not\leq$ simple ACR-GNN$(\texttt{sum})$

Hypothesis 18. *Classifier AVG_{ratio} is not expressible by simple ACR-GNN($\texttt{sum}$).*

In the rest of this section we show that max aggregation is always weaker than mean aggregation (Theorem 21). It requires two technical lemmas which we present below.

Lemma 19. *Let* comb *be a combination function mapping pairs of d dimensional vectors to vectors, and let $S \subseteq \mathbb{Q}^d$ be a finite set. Then there exist functions $(\mathsf{comb}_i)_{i=1}^3$ such that for any graph $G = (V, E, \lambda)$ with labels $\lambda(v) \in S$, and for all $v \in V$: $\mathsf{comb}(\lambda(v), max\{\!|\lambda(w)|\!\}_{w \in N_G(v)}) = \lambda^{(3)}(v)$, where $\lambda^{(0)} = \lambda$, for $k = 1, 3$ $\lambda^{(k)}(v) := \mathsf{comb}_k(\lambda^{(k-1)}(v))$ and $\lambda^{(2)}(v) := \mathsf{comb}_2(\lambda^{(1)}(v), mean\{\!|\lambda^{(1)}(w)|\!\}_{w \in N_G(v)})$. The above holds also in the setting with readout functions.*

Proof sketch. We will simulate max with mean using the thermometer encoding. As an example, assume we have four numbers $-7 < 0 < 3.39 < 5.5$. We map each value to its thermometer encoding, $T(x)$, as follows: $T(-7) = [1,0,0,0]$, $T(0) = [1,1,0,0]$, $T(3.39) = [1,1,1,0]$, and $T(5.5) = [1,1,1,1]$. To compute the max of the multiset $\{\!|-7, 3.39, 3.39|\!\}$, we average their encodings: $\frac{[1,0,0,0]+[1,1,1,0]+[1,1,1,0]}{3} = [1, 0.66, 0.66, 0]$. The $\mathbb{1}_{>0}$ threshold gives $[1,1,1,0]$, which reverses to the correct maximum of 3.39.

For the general case, let $U = \{u_1 < u_2 < \cdots < u_q\}$ be the sorted set of rational coordinates in S. Define the thermometer encoding $T : U \to \{0,1\}^u$ such that the j-th component of $T(r_k)$ is 1 if $j \leq k$ and 0 otherwise. Let $\bar{T} : S \to \{0,1\}^{ud}$ be the coordinate-wise application of T.

We will now show that the coordinate-wise application of the thermometer encoding allows for simulating max with mean, precisely. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq S$ be a finite multiset of vectors. Then

$$\bar{T}^{-1}\left(\mathbb{1}_{>0}\left[\texttt{mean}_{i=1}^n \bar{T}(\mathbf{x}_i)\right]\right) = \texttt{max}_{i=1}^n \mathbf{x}_i. \qquad (1)$$

Indeed, first denote $\mathbf{x}_i = (u_{q_{i,1}}, \ldots, u_{q_{i,d}})$, then we can visualize the vectors as $\bar{T}(\mathbf{x}_i) = (\underbrace{1, \ldots, 1}_{q_{i,1}}, \underbrace{0, \ldots, 0}_{u - q_{i,1}}, \ldots)$. The summation and thresholding operations act coordinate-wise. For any position where at least one $\mathbf{x}_i$ has a 1, the sum is positive, and $\mathbb{1}_{>0}$ returns 1. This effectively computes the union of the non-zero indices: $\mathbb{1}_{>0}\left[\texttt{mean}_{i=1}^n \bar{T}(\mathbf{x}_i)\right] = \left(\underbrace{1, \ldots, 1}_{\max_i q_{i,1}}, \underbrace{0, \ldots, 0}_{u - \max_i q_{i,1}}, \ldots\right)$. Since the encoding $\bar{T}$ is monotonic with respect to the values in U, we have $u_{\max_i q_{i,j}} = \max_i u_{q_{i,j}}$. Thus: $\bar{T}^{-1}\left(\mathbb{1}_{>0}\left[\texttt{mean}_{i=1}^n \bar{T}(\mathbf{x}_i)\right]\right) = (\texttt{max}_i u_{q_{i,1}}, \ldots, \texttt{max}_i u_{q_{i,d}}) = \texttt{max}_{i=1}^n \mathbf{x}_i$.

We now proceed to defining the three layers as follows: $\mathsf{comb}_1(\mathbf{x}, \mathbf{y}) = \bar{T}(\mathbf{x})$, $\mathsf{comb}_2(\mathbf{x}, \mathbf{y}) = \mathbb{1}_{>0}(\mathbf{x} \parallel \mathbf{y})$, and $\mathsf{comb}_3(\mathbf{x}, \mathbf{y}) = \mathsf{comb}\big(\bar{T}^{-1}(\mathbf{x}_{\text{left}}), \bar{T}^{-1}(\mathbf{x}_{\text{right}})\big)$. In comb_3, the input $\mathbf{x}$ comes from the previous layer (which output a concatenation), so we treat it as $\mathbf{x}_{\text{left}} \parallel \mathbf{x}_{\text{right}}$. Let λ' be the output of the initial layer.

We now verify that $\lambda^{(3)}(v) = \lambda'(v)$ for any vertex v: $\lambda^{(3)}(v) = \mathsf{comb}_3(\lambda^{(2)}(v), \texttt{mean}_{w \in N(v)} \lambda^{(2)}(w)) = \mathsf{comb}(\bar{T}^{-1}(\lambda^{(2)}(v)_{\text{left}}), \bar{T}^{-1}(\lambda^{(2)}(v)_{\text{right}})) = \mathsf{comb}(\bar{T}^{-1}(\bar{T}(\lambda(v))), \bar{T}^{-1}(\mathbb{1}_{>0}[\texttt{mean}_{w \in N(v)} \bar{T}(\lambda(w))]))$ By applying Equation (1) to the multiset $\{\lambda(w) : w \in N_G(v)\}$ we get $\mathsf{comb}(\lambda(v), \texttt{max}_{w \in N_G(v)} \lambda(w)) = \lambda'(v)$. The case of ACR-GNN follows in the same way. □

Lemma 20. *If in Lemma 19* comb *is of the form of* ReLU *or threshold applied to a linear combination, then each* comb_i *can be chosen to be of that form too.*

Proof sketch. The proof is similar to that of Lemma 19, except that we build the thermometer encoding using linear steps, which requires additional argumentation.

Let $\mathsf{comb}(\mathbf{x}, \mathbf{y}) = \sigma(\mathbf{x}\mathbf{A} + \mathbf{y}\mathbf{C} + \mathbf{b})$. Let p be the smallest positive integer that makes $p\mathbf{x}\mathbf{A}$ and $p\mathbf{x}\mathbf{C}$ integer for all $\mathbf{x}$ in S. Let m be the largest absolute value of any single coordinate inside $p\mathbf{x}\mathbf{A}$ or $p\mathbf{x}\mathbf{C}$, for $\mathbf{x} \in S$. Let t be the output size of $\mathbf{A}$ and $\mathbf{C}$.

We define a matrix $\mathbf{T} \in \{0,1\}^{(2m+1)t \times t}$ and a bias vector $\mathbf{k} \in \mathbb{Z}^{(2m+1)t}$ as follows:

- Set the entry $\mathbf{T}_{i,j} = p$ if $j = \lceil \frac{i}{2m+1} \rceil$, and 0 otherwise.
- Set the entry $\mathbf{k}_i = ((i-1) \pmod{2m+1}) - m$.

To see that this construction yields a thermometer encoding, consider the scalar case ($d = t = p = 1$) with range $m = 2$. We define thresholds $\mathbf{k} = [-2, -1, 0, 1, 2]^\top$ and let $\mathbf{T} = [1,1,1,1,1]^\top$. The encoding is $\mathbf{v} = \mathbb{1}_{>0}(x\mathbf{T} - \mathbf{k})$ and its inverse is $\mathbf{T}^T\mathbf{v} - m = x$, for example

- If $x = -1$, then $\mathbf{v} = \mathbb{1}_{>0}(-1\mathbf{T} - \mathbf{k}) =$
$$= \mathbb{1}_{>0}([1, 0, -1, -2, -3]^\top) = [1,0,0,0,0]^\top,$$
and the inverse: $\mathbf{T}^T\mathbf{v} - m = 1 - 2 = -1 = x$.
- If $x = 0$, then $\mathbf{v} = \mathbb{1}_{>0}(0\mathbf{T} - \mathbf{k}) =$
$$= \mathbb{1}_{>0}([2, 1, 0, -1, -2]^\top) = [1,1,0,0,0]^\top,$$
and the inverse: $\mathbf{T}^T\mathbf{v} - m = 2 - 2 = 0 = x$.
- If $x = 2$, then $\mathbf{v} = \mathbb{1}_{>0}(2\mathbf{T} - \mathbf{k}) =$
$$= \mathbb{1}_{>0}([4, 3, 2, 1, 0]^\top) = [1,1,1,1,0]^\top,$$
and the inverse: $\mathbf{T}^T\mathbf{v} - m = 4 - 2 = 2 = x$.

We will now show that this thermometer encoding allows for simulating max with mean, precisely. Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq S$ be a finite multiset of vectors. Then

$$\frac{1}{p^2}\mathbf{T}^T \mathbb{1}_{>0}\left(\texttt{mean}_i \mathbb{1}_{>0}\left(\mathbf{x}_i\mathbf{T} - \mathbf{k}\right)\right) - \frac{m}{p}\mathbf{1} = \texttt{max}_i \mathbf{x}_i. \quad (2)$$

Indeed, first denote $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$. Then:

$$\mathbf{x}_i\mathbf{T} = (\underbrace{px_{i,1}, \ldots, px_{i,1}}_{2m+1 \text{ times}}, \ldots), \quad \text{so}$$

$$\mathbb{1}_{>0}(\mathbf{x}_i\mathbf{T} - \mathbf{k}) = (\underbrace{\mathbb{1}_{>0}(px_{i,1} + m), \ldots, \mathbb{1}_{>0}(px_{i,1} - m)}_{2m+1 \text{ times}}, \ldots).$$

For any pair (d', m') with $1 \leq d' \leq d$, $-m \leq m' \leq m$ where at least one $\mathbf{x}_i$ satisfies $px_{i,d'} - m' > 0$, the mean is positive, and the outer $\mathbb{1}_{>0}$ returns 1, so:

$$\mathbb{1}_{>0}(\texttt{mean}_i \mathbb{1}_{>0}(\mathbf{x}_i\mathbf{T} - \mathbf{k})) =$$
$$= (\underbrace{\texttt{max}_i \mathbb{1}_{>0}(px_{i,1} + m), \ldots, \texttt{max}_i \mathbb{1}_{>0}(px_{i,1} - m)}_{2m+1 \text{ times}}, \ldots).$$

Multiplying by $\mathbf{T}^T$ aggregates these indicator bits. Since $\sum_{m'=-m}^{m} \mathbb{1}_{>0}(k - m') = k + m$ for integer k, we obtain:

$$\mathbf{T}^T \mathbb{1}_{>0}\left(\texttt{mean}_i \mathbb{1}_{>0}(\mathbf{x}_i\mathbf{T} - \mathbf{k})\right) =$$
$$= (p \sum_{s=-m}^{m} \mathbb{1}_{>0}(p\texttt{max}_i x_{i,1} - s), \ldots) =$$
$$= (p^2\texttt{max}_i x_{i,1} + pm, \ldots, p^2\texttt{max}_i x_{i,d} + pm).$$

Finally, normalizing the result recovers the maximums:

$$\frac{1}{p^2}\mathbf{T}^T \mathbb{1}_{>0}\left(\texttt{mean}_i \mathbb{1}_{>0}(\mathbf{x}_i\mathbf{T} - \mathbf{k})\right) - \frac{m}{p} \cdot \mathbf{1} =$$
$$= (\max_i x_{i,1}, \ldots, \max_i x_{i,d}) = \texttt{max}_i\, \mathbf{x}_i.$$

We now proceed to defining the three layers as follows:

$$\mathsf{comb}_1(\mathbf{x}, \mathbf{y}) = \mathbb{1}_{>0}(\mathbf{x}\mathbf{T} - \mathbf{k}), \mathsf{comb}_2(\mathbf{x}, \mathbf{y}) = \mathbb{1}_{>0}(\mathbf{x} \parallel \mathbf{y}),$$

$$\mathsf{comb}_3(\mathbf{x}, \mathbf{y}) = \sigma\Big(\frac{1}{p^2}\left(\mathbf{A}\mathbf{T}^T\mathbf{x}_{\text{left}} + \mathbf{C}\mathbf{T}^T\mathbf{x}_{\text{right}}\right) - \frac{m}{p}(\mathbf{A} + \mathbf{C})\mathbf{1} + \mathbf{b}\Big).$$

We now verify that $\lambda^{(3)}(v) = \lambda'(v)$ for any vertex v:

$$\lambda^{(3)}(v) = \mathsf{comb}_3\Big(\lambda^{(2)}(v), \texttt{mean}_{w \in N(v)}\lambda^{(2)}(w)\Big)$$
$$= \sigma\Bigg(\frac{1}{p^2}\Big(\mathbf{A}\mathbf{T}^T[\lambda^{(2)}(v)]_{\text{left}} + \mathbf{C}\mathbf{T}^T[\lambda^{(2)}(v)]_{\text{right}}\Big) - \frac{m}{p}(\mathbf{A} + \mathbf{C})\mathbf{1} + \mathbf{b}\Bigg)$$
$$= \sigma\Bigg(\mathbf{A}\left(\frac{1}{p^2}\mathbf{T}^T\mathbb{1}_{>0}(\lambda(v)\mathbf{T} - \mathbf{k}) - \frac{m}{p}\mathbf{1}\right) + \mathbf{C}(\frac{1}{p^2}\mathbf{T}^T \mathbb{1}_{>0}\Big[\texttt{mean}_{w \in N(v)}\mathbb{1}_{>0}(\lambda(w)\mathbf{T} - \mathbf{k})\Big] - \frac{m}{p}\mathbf{1}) + \mathbf{b}\Bigg)$$
$$= [\text{Apply (2) to both terms}]$$
$$= \sigma\Big(\mathbf{A}\lambda(v) + \mathbf{C}\left(\texttt{max}_{w \in N(v)}\lambda(w)\right) + \mathbf{b}\Big) = \lambda'(v).$$

The construction for ACR-GNN follows in a similar way. □

Equipped with Lemmas 19 and 20 we are ready to show the last result of this section.

Theorem 21. *AC-GNN(max) < AC-GNN(mean). The same holds if we replace AC-GNN with ACR-GNN, or with simple versions of AC-GNN and ACR-GNN.*

Proof sketch. First, we show that AC-GNN(max) $\leq$ AC-GNN(mean). There are only finitely many choices for the initial label, bounded by 2^d, where d is the initial dimension. Note that a max layer keeps this property. So, at each layer, the max networks we look at have only a finite number of possible label vectors. Therefore, Lemma 19 applies. This lets us replace every max with mean, working from top to bottom.

To show that AC-GNN(mean) $\not\leq$ AC-GNN(max), consider a classifier $Q(x)$ that accepts nodes where at least half of their neighbours satisfy a unary predicate P. A simple AC-GNN(mean) can express this. One layer computes the average of the neighbour labels, and a threshold checks if the value is $\geq \frac{1}{2}$.

However, $Q(x)$ cannot be expressed by an ACR-GNN(max). Consider a graph G with nodes $V = \{v_1, v_2, w_1, w_2, w_3, w_4, w_5, w_6\}$ and the edge set E consisting of $\{v_1, w_1\}$, $\{v_1, w_2\}$, $\{v_1, w_3\}$, $\{v_2, w_4\}$, $\{v_2, w_5\}$, and $\{v_2, w_6\}$. Moreover, let P be true at nodes w_1, w_2, and w_4. So, $Q(v_1)$ holds and $Q(v_2)$ does not hold.

To show that any ACR-GNN(max) $\mathcal{N}$ computes the same output for v_1 and v_2, that is $\mathcal{N}(G, v_1) = \mathcal{N}(G, v_2)$, we use induction on the number of layers i. We will show that the node embeddings are always the same, namely $\lambda^{(i)}(v_1) = \lambda^{(i)}(v_2)$. At the same time, we will show that $\lambda^{(i)}(w_1) = \lambda^{(i)}(w_2) = \lambda^{(i)}(w_4)$, whereas $\lambda^{(i)}(w_3) = \lambda^{(i)}(w_5) = \lambda^{(i)}(w_6)$.

The base case (layer 0) holds by the definition of G. For the inductive step, we first consider the global readout. Readout combines information from all nodes in the graph, so it gives the exact same vector to every node in the graph at layer i. Next, we consider the w nodes. Each w node has exactly one neighbour: either v_1 or v_2. By our inductive assumption, $\lambda^{(i-1)}(v_1) = \lambda^{(i-1)}(v_2)$. As a result we can show that $\lambda^{(i)}(w_1) = \lambda^{(i)}(w_2) = \lambda^{(i)}(w_4)$ and $\lambda^{(i)}(w_3) = \lambda^{(i)}(w_5) = \lambda^{(i)}(w_6)$.

Finally, we need to show that $\lambda^{(i)}(v_1) = \lambda^{(i)}(v_2)$. Since the readout vector is the same for both nodes, we only need to show that the aggregation function, in this case max, computes the same vector for v_1 and v_2. There are two cases to consider. (Case 1) If $\lambda^{(i-1)}(w_1) > \lambda^{(i-1)}(w_3)$, the max function for v_1 outputs $\lambda^{(i-1)}(w_1)$. For v_2, the max function outputs $\lambda^{(i-1)}(w_4)$. Since we know $\lambda^{(i-1)}(w_4) = \lambda^{(i-1)}(w_1)$, the outputs match. (Case 2) If $\lambda^{(i-1)}(w_1) \leq \lambda^{(i-1)}(w_3)$, the max function for v_1 outputs $\lambda^{(i-1)}(w_3)$. For v_2, the max function outputs $\lambda^{(i-1)}(w_5)$ (or w_6). Since we know $\lambda^{(i-1)}(w_5) = \lambda^{(i-1)}(w_3)$, the outputs match again. Because both cases result in the same output, we obtain that $\lambda^{(i)}(v_1) = \lambda^{(i)}(v_2)$, as required. □

Since simple AC-GNN(max) can express all classifiers from ML, we obtain that also simple AC-GNN(mean) can express all classifiers from ML. It is worth comparing our result with Theorem 11 by Schönherr and Lutz (2026), which states that AC-GNN(mean) can express all classifiers from ML, but their construction requires GNNs to use complex functions and Cantor-style arguments. The authors conjecture that their construction can be simplified, by using only simple GNNs. Our proof validates this conjecture. Observe however, that our construction uses threshold function as an activation function of combination function. We leave it as an open question whether the result holds for GNNs which use only ReLU as activation functions.

6 Conclusions

We have established relative expressiveness results for GNN architectures with different aggregation functions, namely arbitrary aggregation, sum, mean, and max. In addition, we have connected these architectures to infinitary counting logics. The obtained expressiveness landscape is non-trivial and varies significantly depending on whether global readout is allowed and whether the GNNs are simple.

As future work, it is interesting to establish the missing relationship between simple GNNs with mean and sum aggregations, namely to determine whether Hypothesis 18 is true. Another open question is whether Theorem 21 holds if in simple GNNs the only activation function allowed is ReLU (currently the construction allows also for threshold).

AI Declaration

Generative AI tools were used only for language correction and for searching the literature. All results, proofs, and conclusions presented in this paper were obtained by the authors.

References

Ahvonen, V.; Heiman, D.; Kuusisto, A.; and Lutz, C. 2024. Logical characterizations of recurrent graph neural networks with reals and floats. In *Proc. of NeurIPS*.

Ahvonen, V.; Funk, M.; Heiman, D.; Kuusisto, A.; and Lutz, C. 2026. Expressive power of graph transformers via logic. In *Proc. of AAAI*.

Babai, L., and Kucera, L. 1979. Canonical labelling of graphs in linear average time. In *Proc. FOCS*, 39–46.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J. L.; and Silva, J. P. 2020. The logical expressiveness of graph neural networks. In *Proc. of ICLR*.

Barceló, P.; Geerts, F.; Lanzinger, M.; Pakhomenko, K.; and Bussche, J. V. d. 2025. A logical view of gnn-style computation and the role of activation functions. *arXiv preprint arXiv:2512.19332*.

Benedikt, M.; Lu, C.-H.; Motik, B.; and Tan, T. 2024. Decidability of Graph Neural Networks via Logical Characterizations. In *Proc. of ICALP*, volume 297, 127:1–127:20. Schloss Dagstuhl.

Besharatifard, M., and Vafaee, F. 2024. A review on graph neural networks for predicting synergistic drug combinations. *Artif. Intell. Rev.* 57(3):49.

Cai, J.; Fürer, M.; and Immerman, N. 1992. An optimal lower bound on the number of variables for graph identification. *Comb.* 12.

Chen, C.; Wu, Y.; Dai, Q.; Zhou, H.; Xu, M.; Yang, S.; Han, X.; and Yu, Y. 2024. A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective. *IEEE Trans. Pattern Anal. Mach. Intell.* 46(12):10297–10318.

Cuenca Grau, B.; Feng, E.; and Wałęga, P. A. 2026. The correspondence between bounded graph neural networks and fragments of first-order logic. In *Proc. of AAAI*.

Derrow-Pinion, A.; She, J.; Wong, D.; Lange, O.; Hester, T.; Perez, L.; Nunkesser, M.; Lee, S.; Guo, X.; Wiltshire, B.; Battaglia, P. W.; Gupta, V.; Li, A.; Xu, Z.; Sanchez-Gonzalez, A.; Li, Y.; and Velickovic, P. 2021. ETA prediction with graph neural networks in Google Maps. In *Proc. of CIKM*, 3767–3776.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *Proc. of ICML*, 1263–1272.

Grohe, M., and Rosenbluth, E. 2024. Are targeted messages more effective?

Grohe, M. 2021. The logic of graph neural networks. In *2021 36th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 1–17. IEEE.

Grohe, M. 2024. The descriptive complexity of graph neural networks. *TheoretiCS* 3.

Hauke, S. P., and Wałęga, P. A. 2026. Aggregate-combine-readout GNNs are more expressive than logic C2. In *Proc. of AAAI*.

Huang, N. T., and Villar, S. 2021. A short tutorial on the weisfeiler-lehman test and its variants. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8533–8537. IEEE.

Huang, X.; Romero, M.; Ceylan, İ. İ.; and Barceló, P. 2023. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Proc. of NeurIPS*.

Huang, X.; Orth, M. A. R.; Barceló, P.; Bronstein, M. M.; and Ceylan, İ. İ. 2025. Link prediction with relational hypergraphs. *Trans. Mach. Learn. Res.*

Morris, M., and Horrocks, I. 2025. Sound logical explanations for mean aggregation graph neural networks. In *Proc. of NeurIPS*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proc. of AAAI*, 4602–4609.

Nunn, P.; Sälzer, M.; Schwarzentruber, F.; and Troquard, N. 2024. A logic for reasoning about aggregate-combine graph neural networks. In *Proc. of IJCAI*, 3532–3540. ijcai.org.

Pflueger, M.; Cucala, D. T.; and Kostylev, E. V. 2024. Recurrent graph neural networks and their connections to bisimulation and logic. In *Proc. of LICS*, 14608–14616.

Rosenbluth, E., and Grohe, M. 2026. Repetition makes perfect: Recurrent sum-GNNs match message passing limit. In *Proc. of AAAI*.

Rosenbluth, E.; Toenshoff, J.; and Grohe, M. 2023. Some might say all you need is sum. In *Proc. of IJCAI*.

Sälzer, M.; Wałęga, P. A.; and Lange, M. 2025. The logical expressiveness of temporal gnns via two-dimensional product logics. In *Proc. of NeurIPS*.

Schönherr, M., and Lutz, C. 2026. Logical characterizations of GNNs with mean aggregation. In *Proc. of AAAI*.

Tena Cucala, D. J., and Cuenca Grau, B. 2024. Bridging max graph neural networks and datalog with negation. In *Proc. of KRR*.

Tena Cucala, D.; Cuenca Grau, B.; Motik, B.; and Kostylev, E. V. 2023. On the correspondence between monotonic max-sum gnns and datalog. In Marquis, P.; Son, T. C.; and Kern-Isberner, G., eds., *Proc. of KR*, 658–667.

Weisfeiler, B., and Leman, A. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *Nauchno-Technicheskaya Informatsia*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *Proc. of ICLR*.

Zhang, M., and Chen, Y. 2018. Link prediction based on graph neural networks. In *Proc. of NeurIPS*, 5171–5181.