Aggregate-Combine-Readout GNNs Can Express Logical Classifiers Beyond the Logic C2

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

In recent years, there has been growing interest in understanding the expressive power of graph neural networks (GNNs) by relating them to logical languages. This research has been initialised by an influential result of Barceló et al. (2020), who showed that the graded modal logic (or a guarded fragment of the logic C^2), characterises the logical expressiveness of aggregate-combine GNNs. As a "challenging open problem" they left the question whether C^2 characterises the logical expressiveness of aggregate-combine-readout GNNs. This question has remained unresolved despite several attempts. In this paper, we solve the above open problem by proving that aggregate-combine-readout GNNs can express logical classifiers beyond C^2. This result holds over both undirected and directed graphs. Beyond its implications for GNNs, our work also leads to purely logical insights on the expressive power of infinitary logics.

Extended version — https://arxiv.org/abs/2508.06091

1 Introduction

Graph Neural Networks (GNNs) (Gilmer et al. 2017) are state-of-the-art machine learning models tailored for processing graph-structured data. They have been successfully applied across numerous domains, including molecular property prediction (Besharatifard and Vafaee 2024), traffic forecasting and navigation (Derrow-Pinion et al. 2021), visual scene interpretation (Chen et al. 2024), personalised recommendations (Ying et al. 2018), as well as knowledge graph completion and reasoning under partial information (Tena Cucala et al. 2022; Zhang and Chen 2018; Huang et al. 2025a).

In recent years, there has been growing interest in understanding the expressive power of GNNs, particularly focusing on the message-passing architectures. A key result (Morris et al. 2019; Xu et al. 2019) shows that GNNs have the same *distinguishing power* as the Weisfeiler–Leman (WL) algorithm (Weisfeiler and Leman 1968)—a widely used heuristic for testing graph isomorphism (Babai and Kucera 1979). This means that a pair of graphs can be distinguished by WL if and only if there exists a GNN that can distinguish them. The result by Cai, Fürer, and Immerman (1992), in turn, shows that WL has the same distinguishing power as the fragment C^2 of first-order logic (FO), in which formulas are restricted to two variables but may use counting quantifiers $\exists_k$, interpreted as "there exist at least k distinct elements such that … ." Hence, we obtain a tight correspondence between GNNs, the Weisfeiler–Leman algorithm, and the logic C^2.

The results on the distinguishable power, however, do not allow us to establish a one-to-one mapping between GNNs and logical formulas expressing the same properties. This finer correspondence is known as *logical expressiveness*, or *uniform expressive power*, and has attracted growing interest in recent years (Benedikt et al. 2024; Ahvonen et al. 2025; Schönherr and Lutz 2025; Nunn et al. 2024; Tena Cucala and Cuenca Grau 2024; Huang et al. 2025b,a; Grohe 2021). The main, and historically first, results in this direction have been established by Barceló et al. (2020). They have studied two architectures of message-passing GNNs: the standard aggregate-combine GNNs (AC-GNNs) and their extension with readout function called aggregate-combine-readout GNNs (ACR-GNNs). The two main results of Barceló et al. (2020) are as follows:

(i) the FO node properties expressible by AC-GNNs are exactly those definable in graded modal logic,
(ii) the FO node properties expressible by ACR-GNNs contain all properties definable in C^2.

Note that, in contrast to Result (i), Result (ii) does not provide an exact logical characterisation. This was left by the authors' as an open problem.

The Open Problem The precise formulation of the open problem of Barceló et al. (2020) is whether the FO node properties expressible by ACR-GNNs are exactly those definable in C^2. Their paper explicitly states that this is "a challenging open problem." This question was subsequently highlighted in later papers, for example by Grohe (2021) as Question 4 on his list of "interesting theoretical questions that remain open."

Several research groups have attempted to solve this problem (Pflueger, Tena Cucala, and Kostylev 2024;

*These authors contributed equally.

Benedikt et al. 2024), but without success. As a result the question has remained unresolved for the past five years.

Contributions In this paper we will solve the above open problem, by showing that ACR-GNNs can express FO node classifiers beyond C^2.

We will show that this result holds not only in the setting of undirected graphs—as originally considered by Barceló et al. (2020)—but also in the setting of directed graphs. In both cases, our proofs follow a common structure: (1) we define a node property, (2) we show that it is expressible both in FO and by an ACR-GNN, and (3) we show that the property is not expressible in C^2. In the directed case (Section 4), the property we consider is that of "being a node of a graph whose edge relation forms a strict linear order." In the undirected case (Section 5), we simulate directed edges using paths of three undirected edges, where direction of an edge is encoded by colours of the two middle nodes in the path. We then consider the property of being a node of an undirected graph that simulates a strict linear order. To show Results (1) and (2) we provide constructions of FO formulas and ACR-GNNs, respectively. To show the inexpressibility Results (3), we introduce in Section 3 a bounded version of WL algorithm, which characterises the expressive power of C^2 formulas whose counting quantifiers $\exists_k$ have bounded k.

Note that, in particular, we show that linear orders cannot be expressed in C^2. Inexpressibility of linear orders in logics is a well studied topic; it is known that FO^2 (i.e. FO with two variables) cannot express linear orders (Immerman and Kozen 1989; Szwast and Tendera 2013) and a similar result for C^2 can be inferred from results of Charatonik and Witkowski (2016). We will prove the latter result directly, which will be useful afterwards for the inexpressibility Result (3) for undirected graphs. Finally, in Section 6 we exploit our results to the study the expressive power of infinitary logics. As we show, the infinitary version of C^2 can express strictly more FO properties than the standard, finitary C^2.

2 Preliminaries

We introduce notions and notation for graphs, GNNs, and logics. Our setting extends that ofBarceló et al. (2020) by considering not only undirected, but also directed graphs.

Graphs A directed (node-labelled, finite, and simple) *graph* of dimension $d \in \mathbb{N}$ is a tuple $G = (V, E, \lambda)$, where V is a finite set of nodes, $E \subseteq V \times V$ is a set of directed edges with no loops $E(v, v)$, and $\lambda : V \to \{0,1\}^d$ assigns to each node a binary vector of a dimension d. We will identify *undirected* graphs with directed graphs that have a symmetric edge relation, and write $\{v, w\}$ for a pair of edges (v, w), (w, v).

The *neighbourhoud*, $N_G(v)$, of a node v in a graph G, is the set of all nodes w such that G has an edge (in any direction) between w and v. The *in-neighbourhoud*, $\overleftarrow{N}_G(v)$, are nodes w such that G has an edge from w to v, whereas the *out-neighbourhoud*, $\overrightarrow{N}_G(v)$, are nodes w such that G has an edge from v to w. Hence, in undirected graphs we have $N_G(v) = \overleftarrow{N}_G(v) = \overrightarrow{N}_G(v)$.

GNN Node Classifiers We focus on *aggregate-combine-readout* GNNs (ACR-GNNs) introduced by Barceló et al. (2020), which extend the standard message-passing mechanism with readout functions. First, we introduce ACR-GNN architecture for processing undirected graphs. In such GNNs, each layer is a triple $(\mathsf{agg}, \mathsf{comb}, \mathsf{read})$ consisting of an *aggregation function*, agg, mapping a multiset (a generalisation of a set so that elements can have multiple occurrences) of vectors into a single vector, a *combination function* comb, mapping three vectors to one vector, and a *readout function*, read, mapping a multiset of vectors into a single vector. Such layers applied to a graph $G = (V, E, \lambda)$ computes a graph $G' = (V, E, \lambda')$ with a new labelling function λ' such that for each v, the labelling $\lambda'(v)$ is given by

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

where $\{\!\{\cdot\}\!\}$ stands for a multiset. In the spirit of Rossi et al. (2023), we also consider a straightforward generalisation of ACR-GNN architecture for processing directed graphs. In this case a GNN is a tuple $(\overleftarrow{\mathsf{agg}}, \overrightarrow{\mathsf{agg}}, \mathsf{comb}, \mathsf{read})$, which has two types of aggregation: $\overleftarrow{\mathsf{agg}}$ for incoming edges and $\overrightarrow{\mathsf{agg}}$ for outgoing edges. In such ACR-GNNs, a new labelling $\lambda'(v)$ is computes as

$$\begin{aligned}\mathsf{comb}\big(\lambda(v), &\overleftarrow{\mathsf{agg}}(\{\!\{\lambda(w)\}\!\}_{w \in \overleftarrow{N}_G(v)}),\\ &\overrightarrow{\mathsf{agg}}(\{\!\{\lambda(w)\}\!\}_{w \in \overrightarrow{N}_G(v)}), \mathsf{read}(\{\!\{\lambda(w)\}\!\}_{w \in V})\big).\end{aligned}$$

An *ACR-GNN classifier* $\mathcal{N}$ of dimension d consists of a fixed number L of layers[1] and a classification function cls from vectors to truth values; once applied to a graph of dimension d, the classifier $\mathcal{N}$ computes for each node v a truth value denoted as $\mathcal{N}(G, v)$.

Logical Node Classifiers By FO we mean the standard first-order logic with identity $=$, one binary predicate E for edges, and unary predicates $P_1, \dots, P_d$ for node labels. Let C^2 be the fragment of FO, which allows for using only two variables in formulas, but allows for additional counting quantifiers $\exists_k$, for any $k \in \mathbb{N}$, where $\exists_k x \varphi(x)$ means that φ holds in at least k different nodes. We will write $\exists_{=k} \varphi(x)$ as an abbreviation for $\exists_k \varphi(x) \wedge \neg \exists_{k+1} \varphi(x)$. Note that we write $\varphi(x)$ for a formula with exactly one free variable x, and similarly we will use $\varphi(x, y)$ for a formula with exactly two free variables. We let the quantifier *depth* of a formula φ be its maximum nesting of quantifiers. Moreover, for C^2 formulas we define the *counting rank*, $\mathsf{rk}_\#(\varphi)$, as the maximal among numbers k occurring in its counting quantifiers. For a logic $\mathcal{L}$, we let $\mathcal{L}_{\ell,c}$ be the fragment with formulas of depth at most ℓ and counting rank at most c. In the paper we pay special attention to $C^2_{\ell,c}$.

[1]We assume that functions in the layers are of matching dimensions, so that they can be applied.

A *logical node classifier* is a formula $\varphi(x)$ in FO (or its fragment) with one free variable. To evaluate logical classifiers, we identify a graph $G = (V, E, \lambda)$ of dimension d with the FO structures $\mathfrak{M}_G = (V, P_1, \dots, P_d, E)$, with domain V, sets $P_i = \{v \in V \mid \lambda(v)_i = 1\}$ containing all nodes v with 1 on the ith position of $\lambda(v)$, and the binary relation E being the graph edges. We assume the standard FO semantics over such models and write $G \models \varphi(v)$ if classifier $\varphi(x)$ holds in $\mathfrak{M}_G$ at the node v. If this is the case, we say that the application of the logical classifier $\varphi(x)$ to G at node v is true, and otherwise it is false. We write $G, u \equiv_{\mathcal{L}} H, v$, if $G \models \varphi(u)$ is equivalent to $H \models \varphi(v)$, for each logical classifier $\varphi(x)$ in a logic $\mathcal{L}$.

3 WL Algorithm with Bounded Counting

In this section, we will introduce a bounded version of the one dimensional WL algorithm (Weisfeiler and Leman 1968). Our version WL_c is parametrised by $c \in \mathbb{N}$, which bounds the "counting abilities" of the algorithm. As we will show, ℓ rounds of application of WL_c allows us to characterise expressiveness of the fragment $\mathrm{C}^2_{\ell,c}$ of C^2, where formulas have depth bounded by ℓ and counting rank by c. This result will play a crucial role to establish non-expressivity results in the latter sections of the paper.

The main idea behind WL_c is that the algorithm is insensitive to multiplicities (occurring in processed multisets) greater than c. Standard WL computes new node labels based on multisets $\{\!\{\cdot\}\!\}$ of neighbours labels. In WL_c the computations are based on the *c-bounded* multisets $\{\!\{\cdot\}\!\}^c$, obtained by reducing all multiplicities to at most c. For example $\{\!\{7,7,7,3\}\!\}^2 = \{\!\{7,7,3\}\!\}$. In particular, over undirected graphs, labelling $W_c^{\ell+1}(v)$ of a node v in iteration $\ell+1$ will depend on the the previous label $W_c^{\ell}(v)$, the c-bounded multiset of labels of v neighbours, and the c-bounded multiset of non-neighbours, so $W_c^{\ell+1}(v)$ equals

$$\big(W_c^{\ell}(v), \{\!\{W_c^{\ell}(w)\}\!\}^c_{w \in N_G(v)}, \{\!\{W_c^{\ell}(w)\}\!\}^c_{w \in V \setminus \{N_G(v) \cup \{v\}\}}\big).$$

Characterising $\mathrm{C}^2_{\ell,c}$ over directed graphs is more challenging. In this case, instead of considering in WL_c one multiset of neighbours' labels, we consider separately nodes which belong to $\overleftarrow{N}_G$ and $\overrightarrow{N}_G$, those which belong to $\overleftarrow{N}_G$ only, and those which belong to $\overrightarrow{N}_G$ only. Below we define WL_c for directed graphs, but for undirected graphs it reduces to the computations described above.

Definition 1. *Let $c \in \mathbb{N}$. The* c-bounded WL algorithm, WL_c*, takes as an input a graph $G = (V, E, \lambda)$, and computes labels $W_c^{\ell}(v)$ for all $v \in V$ as follows:*

$$\begin{aligned} W_c^0(v) &= \lambda(v) \\ W_c^{\ell+1}(v) &= \Big(W_c^{\ell}(v), \{\!\{W_c^{\ell}(w)\}\!\}^c_{w \in \overleftarrow{N}_G(v) \cap \overrightarrow{N}_G(v)}, \qquad (1) \\ &\quad \{\!\{W_c^{\ell}(w)\}\!\}^c_{w \in \overleftarrow{N}_G(v) \setminus \overrightarrow{N}_G(v)}, \{\!\{W_c^{\ell}(w)\}\!\}^c_{w \in \overrightarrow{N}_G(v) \setminus \overleftarrow{N}_G(v)}, \\ &\quad \{\!\{W_c^{\ell}(w)\}\!\}^c_{w \in V \setminus \{N_G(v) \cup \{v\}\}}\Big). \end{aligned}$$

We note that, over undirected graphs, WL_c with $c < \infty$ is strictly less expressive than the standard WL, whereas WL_c with $c = \infty$ coincides with WL.

We will show that WL_c characterises the expressiveness of C^2 with counting rank c. To this end, we will exploit a modal logic characterising C^2 with counting rank c, which can be easily obtained based on the results of Lutz, Sattler, and Wolter (2001) and Barceló et al. (2020). Note that our proof below is over directed graphs, but since we treat undirected graphs as a special case of directed graphs, we can also use this result in the undirected setting.

Theorem 2. *Let $\ell, c \in \mathbb{N}$. For any directed graphs G and H with nodes u and v, the following holds:*

$$G, u \equiv_{\mathrm{C}^2_{\ell,c}} H, v \quad \textit{if and only if} \quad W_c^{\ell}(u) = W_c^{\ell}(v).$$

Proof sketch. Barceló et al. (2020) showed a modal logic $\mathcal{EMLC}$ that over undirected simple graphs has the same expressive power as C^2. Formulas of $\mathcal{EMLC}$ are given by $\varphi := p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle S \rangle^{\geq k}\varphi$, where p are propositional variables, $k \in \mathbb{N}$, and S are modal parameters given by $S := id \mid e \mid S \cup S \mid S \cap S \mid \neg S$. The logic is similar to (graded) modal logic, but allows for complex modal operators introduced by Lutz, Sattler, and Wolter (2001), which are constructed from the identity modality id (self-access) and standard modality e corresponding to edges in the graph, combined using Boolean operations. For example $G, v \models \langle e \cup \neg e \rangle^{\geq 3} p$ means that there are at least 3 nodes w such that $E(v, w)$ or $\neg E(v, w)$, and p holds at w.

To characterise C^2 over simple directed graphs we extend the grammar of modal parameters in $\mathcal{EMLC}$ with S^- expressing the inverse of S. Let $\mathcal{EMLC}^-_{\ell,c}$ be formulas in this extension with modal depth at most ℓ and with $k \leq c$ in graded modalities. We can show that over simple directed graphs $\mathcal{EMLC}^-_{\ell,c}$ has the same expressiveness as $\mathrm{C}^2_{\ell,c}$. Hence, it remains to show that $G, u \equiv_{\mathcal{EMLC}^-_{\ell,c}} H, v$ if and only if $W_c^{\ell}(u) = W_c^{\ell}(v)$.

We prove this equivalence by induction on $i \leq \ell$. In the basis, we have $G, u \equiv_{\mathcal{EMLC}^-_{0,c}} H, v$ if and only if u and v satisfy the same unary predicates, which is equivalent to $W_c^0(u) = W_c^0(v)$. For the inductive step we observe that each $\mathcal{EMLC}^-_{\ell,c}$ formula can be equivalently written in the normal form, where $S := id \mid e^- \cap e \mid e^- \cap \neg e \mid e \cap \neg(e^-) \mid \neg e \cap \neg(e^-) \cap \neg id \mid S \cup S$. In the forward implication assume that $W_c^{i+1}(u) \neq W_c^{i+1}(v)$, so $W_c^{i+1}(u)$ and $W_c^{i+1}(v)$ differ one of the five components from Equation (1). Since these components correspond to components of S grammar in our normal form, we can show that $G, u \not\equiv_{\mathcal{EMLC}^-_{i+1,c}} H, v$. For the backwards implication assume that $W_c^{i+1}(u) = W_c^{i+1}(v)$. We show by induction on the structure of $\mathcal{EMLC}_{i+1,c}$ formulas φ that $G, u \models \varphi$ if and only if $H, v \models \varphi$. The interesting case is for $\varphi = \langle S \rangle^{\geq k}\psi$. Suppose towards a contradiction that $G, u \models \langle S \rangle^{\geq k}\psi$, but $H, v \not\models \langle S \rangle^{\geq k}\psi$. Since atomic parameters in the normal form of S have disjoint interpretations, there are $k' \leq k$

and an atomic parameter A such that $G, u \models \langle A \rangle^{\geq k'} \psi$, but $H, v \not\models \langle A \rangle^{\geq k'} \psi$. By the inductive assumption and correspondence of atomic parameters to the sets occurring in Equation (1), we can show that $W_c^{i+1}(u) = W_c^{i+1}(v)$, raising a contradiction. $\square$

We will use Theorem 2 in two following sections: in Section 4 for directed graphs (Theorem 6) and in Section 5 for undirected graphs (Theorem 12).

4 Logical Expressiveness Over Directed Graphs

In this section, we will study the expressiveness of ACR-GNNs over directed graphs. In this setting, we will consider an analogous question to the open problem of Barceló et al. (2020), namely: are C^2 node classifiers exactly FO classifiers expressible by ACR-GNNs? As we will show, and which may be surprising, the answer is negative. To this end, we will prove that checking if edges of a graph form a strict linear order is expressible in FO and by ACR-GNNs, but cannot be expressed in C^2. Although this is a property of graphs, we can formulate it also as a node classifier as follows.

Definition 3. *We let $\varphi_{Lin}(x)$ be a node classifier accepting a node of a graph G if and only if G is a strict linear order.*

Clearly, strict linear orders can be defined in FO with a formula ψ being a conjunction of the following three:

$$\forall x \, \neg E(x, x) \qquad \text{irreflexivity}$$

$$\forall x \, \forall y \left((x = y) \vee E(x, y) \vee E(y, x) \right) \qquad \text{totality}$$

$$\forall x \, \forall y \, \forall z \left(E(x, y) \wedge E(y, z) \rightarrow E(x, z) \right) \qquad \text{transitivity}$$

Since we are considering simple graphs, irreflexivity can be omitted from ψ. Notice that ψ has no free variables, but we can always turn it into a node classifier by writing it as $(x = x) \wedge \psi$. Thus, $\varphi_{Lin}(x)$ is expressible in FO.

Next, we will show that $\varphi_{Lin}(x)$ can be expressed as an ACR-GNN. This is more challenging, since ACR-GNNs cannot detect transitivity. To address this challenge, we will exploit the following equivalent definition of linear orders.

Proposition 4. *A finite binary relation E is a strict linear order if and only if E is irreflexive, total, and each element has a different number of E-successors.*

Proof sketch. Strict linear orders clearly satisfy the three properties. For the opposite direction we show that E enjoying these properties is transitive. Assume that there are n elements. As each element has a different number of E-successors and E is irreflexive, we can call the elements $v_0, \dots, v_{n-1}$, where v_i is the unique element whose number of E-successors is i. By a strong induction on $i \leq n-1$, we can show that, for all v_j, we have $(v_i, v_j) \in E$ if and only if $i > j$. It implies that E must be transitive. $\square$

We will use Proposition 4 to construct an ACR-GNN which detects strict linear orders.

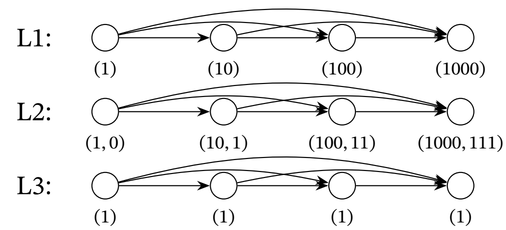

Figure 1: Application of layers 1–3 of the ACR-GNN from Theorem 5 to the strict linear order with four nodes

Theorem 5. *Over directed graphs, $\varphi_{Lin}(x)$ is expressible by an ACR-GNN. It can be achieved using only 3 layers and no aggregation over the out-neighbourhood.*

Proof. We will construct the required ACR-GNN $\mathcal{N}$, whose application to a linear order of length four is presented in Figure 1. The first layer maps the initial vector of a node v into the number 10^n, where n is the in-degree of v. This is obtained by setting $\overleftarrow{\mathsf{agg}}(M) = 10^{|M|}$ and $\mathsf{comb}(x, y, z) = y$. The second layer maps a vector of v into a vector in $\mathbb{R}^2$ of the form $(10^n, 10^{k_1} + \dots + 10^{k_n})$ where 10^n is as in the first layer, whereas each k_i is the in-degree of the ith among the n in-neighbours of v. This is obtained by setting $\overleftarrow{\mathsf{agg}}(M) = sum(M)$ and $\mathsf{comb}(x, y, z) = (x, y)$. The third layers maps each vector into 1 or 0 by setting $\mathsf{read}(M) = 1$ if both of the following conditions hold:

(i) $x[1] \neq y[1]$, for every pair $x, y \in M$.
(ii) if $x[1] = 10^n$, then $x[2] = \underbrace{1 \dots 1}_{n \text{ times}}$, for each $x \in M$.

If any of the conditions does not hold, we set $\mathsf{read}(M) = 0$. Finally, we let $\mathsf{comb}(x, y, z) = y$.

Condition (i) guarantees that each node has a different in-degree. If this is the case, then Condition (ii)—which can be equivalently written as $\frac{x[1]-1}{9} = x[2]$—checks if the graph is total. Hence, for any graph $G = (V, E, \lambda)$, if E is a strict linear order, then $\mathcal{N}(G, v) = 1$ and otherwise $\mathcal{N}(G, v) = 0$, for any node v in G. $\square$

To finish this section, we need to show that $\varphi_{Lin}(x)$ cannot be expressed in C^2. For this, we will exploit our bounded WL algorithm and corresponding Theorem 2.

Theorem 6. *Over directed graphs, $\varphi_{Lin}(x)$ is not expressible in C^2.*

Proof sketch. Suppose towards a contradiction that $\varphi_{Lin}(x)$ is expressible in C^2, so it is definable by a formula in $C^2_{\ell,c}$, for some $\ell, c \in \mathbb{N}$. To obtain a contradiction, we will construct a graph G with nodes v_i and a graph G' with corresponding nodes v_i', such that $G \models \varphi_{Lin}(v_i)$ and $G' \not\models \varphi_{Lin}(v_i')$, but $G, v_i \equiv_{C^2_{l,c}} G', v_i$ for all nodes v_i.

Let $n = \ell \cdot c + 1$. We define $G = (V, E, \lambda)$ as a strict linear order over $2n + 1$ nodes $V = \{v_{-n}, \dots, v_n\}$, with $E = \{(v_i, v_j) : i < j\}$, and $\lambda(v_i) = 0$ for each v_i.

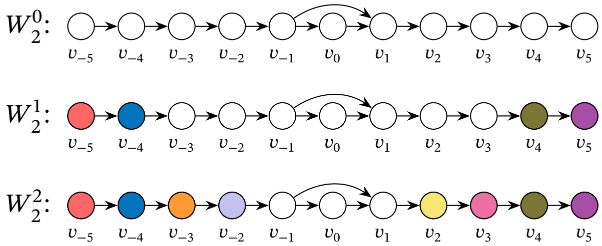

Figure 2: Application of WL_c to G from Theorem 6; for readability we draw only arrows (v_i, v_{i+1}) between consecutive nodes and (v_{-1}, v_1) distinguishing G from G'

We let $G' = (V', E', \lambda')$ be such that $V' = \{v'_{-n}, \dots, v'_n\}$, $E' = \{(v'_i, v'_j) : i < j\} \setminus \{(v'_{-1}, v'_1)\} \cup \{(v'_1, v'_{-1})\}$, and $\lambda'(v'_i) = 0$ for each v'_i. For example, if $c = 2$ and $\ell = 2$, the graphs G is depicted on top of Figure 2; graph G' is similar, but instead of (v'_{-1}, v'_1) it has the opposite edge (v'_1, v'_{-1}). Notice that both graphs are irreflexive, asymmetric, and total, but only G is transitive. Hence, for all nodes v_i, we have $G \models \varphi_{Lin}(v_i)$ and $G' \not\models \varphi_{Lin}(v'_i)$.

It remains to show that $G, v_i \equiv_{\mathrm{C}^2_{l,c}} G', v'_i$. To this end, by Theorem 2, it suffices to show that $W_c^\ell(v_i) = W_c^\ell(v'_i)$. We can prove it by showing, with a simultaneous induction on $k \leq \ell$, the following two statements:

(i) $W_c^k(v_i) = W_c^k(v'_i)$, for $i \in \{-n, \dots, n\}$,

(ii) $W_c^k(v_i) = W_c^k(v_j)$, for $i, j \in \{-(n - ck), \dots, n - ck\}$.

Statement (ii) ensures that all 'middle nodes' have the same colour; for instance in Figure 2 nodes $v_{-3}, \dots, v_3$ have the same colour in W_2^1. We use it to show Statement (i), which implies required $G, v_i \equiv_{\mathrm{C}^2_{l,c}} G', v'_i$. □

Hence, we can conclude this sections as follows.

Corollary 7. *Over directed graphs, there are FO node classifiers expressible by ACR-GNNs which are not expressible in C^2. In particular, $\varphi_{Lin}(x)$ is such a classifier.*

5 Logical Expressiveness Over Undirected Graphs

Now we consider the setting of undirected graphs. We will solve the open problem of Barceló et al. (2020), asking whether over undirected graphs the FO node properties expressible by ACR-GNNs are exactly those definable in C^2. We will show that, the answer is negative. In particular, we will show that, similarly to the case of directed graphs in Section 4, there is a property expressible by both FO and ACR-GNNs, but which cannot be expressed in C^2. Our proofs will build on some ideas from Section 4, but no access to directed edges will require more complex argumentation.

In place of $\varphi_{Lin}(x)$ from Section 4, we will use now classifier $\varphi_{GadLin}(x)$. It checks if a node belongs to a *gadgetised linear order*, which is an undirected graph $\mathsf{gad}(G)$ obtained by encoding (gadgetising) some strict linear order G. Intuitively, $\mathsf{gad}(G)$ is obtained by replacing each directed edge (u, w) in G with a path of three undirected edges—called *gadgetised edges*—as depicted in Figure 3. Next, we present a formal definition of gadgetisation.

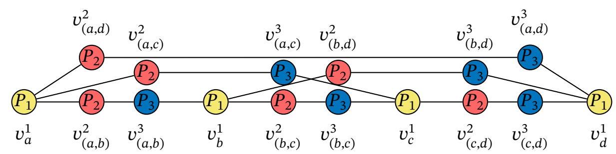

Figure 3: Gadgetisation of the linear order from Figure 1 assuming its nodes are called a, b, c, and d; labels $(1, 0, 0)$, $(0, 1, 0)$, and $(0, 0, 1)$ are represented as P_1, P_2, and P_3, respectively (and also with colours)

Definition 8. *The* gadgetisation, $\mathsf{gad}(G)$, *of a directed graph $G = (V, E, \lambda)$ is an undirected graph $G' = (V', E', \lambda')$ of dimension 3 such that for each edge $(u, w) \in E$, the graph G' has:*

- *nodes v^1_u, $v^2_{(u,w)}$, $v^3_{(u,w)}$, v^1_w in V',*
- *edges $\{v^1_u, v^2_{(u,w)}\}$, $\{v^2_{(u,w)}, v^3_{(u,w)}\}$, $\{v^3_{(u,w)}, v^1_w\}$ in E',*
- *labelling of nodes with $\lambda'(v^1_u) = \lambda'(v^1_w) = (1, 0, 0)$, $\lambda'(v^2_{(u,w)}) = (0, 1, 0)$, and $\lambda'(v^3_{(u,w)}) = (0, 0, 1)$.*

Recall that we identify undirected graphs with symmetric directed graphs, so an undirected edge, like $\{v^1_u, v^2_{(u,w)}\}$ in the definition above, can be seen as a pair of directed edges $(v^1_u, v^2_{(u,w)})$, $(v^2_{(u,w)}, v^1_u)$. Note also that our construction of $\mathsf{gad}(G)$ does not depend on the labelling λ in G. Now, the formal definition of $\varphi_{GadLin}(x)$ is as follows:

Definition 9. *We let $\varphi_{GadLin}(x)$ be a node classifier accepting a node of a graph G if and only if G is isomorphic to $\mathsf{gad}(G')$, for some strict linear order G'.*

It the remaining part of this section, we will show that $\varphi_{GadLin}(x)$ is expressible in FO and by ACR-GNNs, but it is not expressible in C^2.

Theorem 10. *Over undirected graphs, $\varphi_{GadLin}(x)$ is expressible in FO.*

Proof sketch. We will express $\varphi_{GadLin}(x)$ as a conjunction of four FO formulas φ_1, φ_2, φ_3, and φ_4. Recall that we identify graphs with FO structures interpreting unary predicates $P_1, \dots, P_d$, where d is the dimension of the graph, and one binary predicate E. Since gadgetisations are always of dimension $d = 3$, our formulas will mention three unary predicated P_1, P_2, and P_3.

Formula φ_1 states that P_1, P_2, and P_3 partition the set of nodes. Formula φ_2 states that every node satisfying P_2 has exactly two E-neighbours: one satisfying P_1 and the other satisfying P_3. It states also that every node satisfying P_3 has exactly two E-neighbours: one satisfying P_1 and the other satisfying P_2. Finally, it states that if u and v are nodes satisfying P_1, then $E(u, v)$ cannot be true. Formulas φ_3 and φ_4 are about *gadgetised edges*, which are paths in gad(G) that correspond to directed edges in G. In particular, we let a gadgetised edge from u to z be a path of the form $E(u,w)$, $E(w,v)$, $E(v,z)$ with $P_1(u)$, $P_2(w)$, $P_3(v)$, and $P_1(z)$.

Formula φ_3 states that between any two distinct nodes satisfying P_1 there is exactly one gadgetised edge. Formula φ_4, in turn, states that there are no nodes u, w, v with gadgetised edges from v to w, from w to u, and from u to v.

All formulas φ_1–φ_4 can be written in FO, and we can show that a graph satisfies all of them if and only if the graph is a gadgetised linear order. □

In Theorem 10 we have showed how to express $\varphi_{GadLin}(x)$ with FO formulas φ_1–φ_4. We observe that φ_1 and φ_2 are in C^2, so by the result of Barceló et al. (2020), we can express them with ACR-GNNs. However φ_3 and φ_4 cannot be expressed by ACR-GNNs. However, as will show, φ_3 and φ_4 can be replaced with a property that is expressible by ACR-GNNs. This will show that $\varphi_{GadLin}(x)$ is expressible by ACR-GNNs.

Theorem 11. *Over undirected graphs, $\varphi_{GadLin}(x)$ is expressible by an ACR-GNN.*

Proof sketch. We can show that a graph G is a gadgetised linear order if and only if G satisfies φ_1, φ_2 (see the proof of Theorem 10) and a property ψ explained next. Property ψ states that for all $i < j < |P_1|$, the graph has nodes v_i and v_j such that (1) both v_i and v_j satisfy P_1, (2) v_i has i neighbours satisfying P_2 and v_j has j such neighbours, and (3) there is a gadgetised edge (see the proof of Theorem 10) from v_j to v_i. Since φ_1 and φ_2 are C^2 formulas, they can be expressed by ACR-GNNs (Barceló et al. 2020, Theorem 5.1). It remains to construct an ACR-GNN $\mathcal{N}$ which expresses ψ, since it is straightforward to combine the three ACR-GNNs into a single GNN.

Recall that gadgetised linear orders are graphs of dimension three, so we will consider application of $\mathcal{N}$ to such graphs G. In each layer, $\mathcal{N}$ will assign to nodes vectors of dimesion five, where the first three positions are always as in the input graph G, so information about P_1, P_2, and P_3 in the input graph is preserved across all layers. The fourth and fifth positions will always keep binary numbers. The details of $\mathcal{N}$ are provided next and and example of its application is visualised in Figure 4.

The first layer assigns to the fourth position of nodes v satisfying P_1 the number 10^n, where n is the number of neighbours of v satisfying P_2. Fourth and fifth positions of other nodes are set to 0. The next three layers will compute bitwise OR applied to binary numbers, for example $OR(100, 10, 10) = 110$. The second layer assigns to the fourth position of nodes v satisfying P_3 the value of OR over the fourth positions of v neighbours satisfying satisfy P_1.

The third layer assigns to the fourth position of nodes v satisfying P_2 the value of OR over the fourth positions of v neighbours satisfying P_3. The fourth layer assigns to the fifth position of nodes v satisfying P_1 the value of OR over the fourth positions of v neighbours satisfying P_2. Finally, the fifth layer uses a global readout to assign 1 to each node if for all $i < j < |P_1|$ there exists a node whose fourth position of the vector is 10^j and the fifth position of the vector has 1 as the ith bit from the right (when counting from 0).

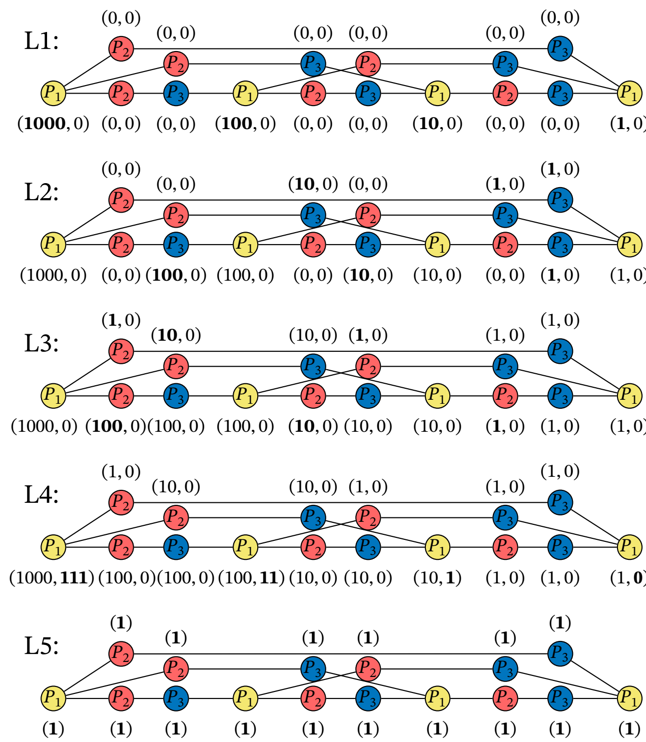

Figure 4: Application of the ACR-GNN from Theorem 11 to the graph from Figure 3; we present only the fourth and fifth components of vectors, and write in bold values updated in a given layer

The first four layers can be implemented without readout functions. The fifth layer, in contrast, requires using readout, but no aggregation. To show that the construction is correct, we can show that in layer 4, each node v satisfying P_1 has on the fourth position of its vector 10^j, where j is the number of v neighbours satisfying P_2. On the fifth position v has a binary number, whose ith bit is 1 if there is a gadgetised edge from v to some node with i neighbours satisfying P_2. Therefore, the fifth layer assigns 1 to all nodes if the graphs satisfies ψ, and otherwise it assigns 0 to all nodes. □

To finish this section, it remains to show that gadgetised linear orders are not expressible in C^2. To this end, we will again use bounded WL from Section 3, as it is applicable to both directed and undirected graphs.

Theorem 12. *Over undirected graphs, the classifier $\varphi_{GadLin}(x)$ is not expressible in C^2.*

Proof sketch. The proof is similar to the one for Theorem 6, namely we suppose towards a contradiction that $\varphi_{GadLin}(x)$ is expressible by a $C^2_{\ell,c}$ formula, for some $\ell, c \in$

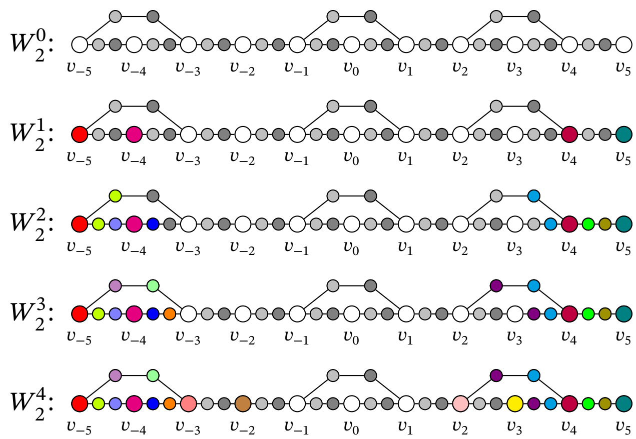

Figure 5: Application of WL_2 to $H = \mathsf{gad}(G)$, for G from Theorem 6; for readability we draw only gadgetised edges corresponding to v_i, v_{i+1} in G, as well as to edges (v_{-5}, v_{-3}), (v_{-1}, v_{-1}), and (v_2, v_4), which helps to understand better the colourings

$\mathbb{N}$. In the proof of Theorem 6 we have obtained contradiction by applying WL_c to directed graphs G and G'. Now, we will apply WL_c to their gadgetisations $H = \mathsf{gad}(G)$ and $H' = \mathsf{gad}(G')$. Since G is a strict linear order, but G' is not, we obtain that H is a gadgetised linear order, but H' is not. Hence, by Theorem 2, it remains to show that W_c^ℓ outputs the same colourings on H and H'. The proof is similar as in Theorem 6. Colourings obtained by applying W_c^ℓ to H are presented in Figure 5; application of W_c^ℓ to H' results in the exactly same colourings. □

By combining Theorems 10 and 11, we obtain a solution to the open problem of Barceló et al. (2020).

Corollary 13. *Over undirected graphs, there are FO node classifiers expressible by ACR-GNNs which are not expressible in* C^2*. In particular,* $\varphi_{GadLin}(x)$ *is such a classifier.*

The above result, shows that ACR-GNNs can express FO node classifiers beyond C^2. Consequently, we establish that the converse of the result of Barceló et al. (2020, Theorem 5.1) does not hold. As we show in the following short section, our results have interesting implications beyond the expressive power of GNNs, contributing to a better understanding of the expressiveness of logics.

6 Impact on the Expressiveness of Logics

It turns out that our results can be used to show an interesting relation between the expressive power of finitary and infinitary logics. To formulate this result, let us use $\mathsf{inf}\text{-}\mathrm{C}^2$ for an extension of C^2 which allows for infinitary conjunctions and disjunctions. Notice that the expressive power of $\mathsf{inf}\text{-}\mathrm{C}^2$ is not only beyond C^2, but also beyond the whole FO. For example $\mathsf{inf}\text{-}\mathrm{C}^2$ allows us to express parity of a graph size using the infinite formula:

$$\exists_{=2} x (x = x) \lor \exists_{=4} x (x = x) \lor \exists_{=6} x (x = x) \lor \ldots$$

which is well-known to be inexpressible in FO—it can be shown by a standard application of Ehrenfeucht–Fraïssé games (Libkin 2004).

This naturally leads us to the question: what are the FO properties expressible in $\mathsf{inf}\text{-}\mathrm{C}^2$? It maybe tempting to assume that those are exactly the properties expressible in C^2. In other words, that the (semantical) intersection of $\mathsf{inf}\text{-}\mathrm{C}^2$ and FO is exactly C^2. As we show next, it is not true.

Theorem 14. *There are strictly more FO properties expressible in* $\mathsf{inf}\text{-}\mathrm{C}^2$ *than the properties expressible in* C^2*. This result holds both over directed and undirected graphs.*

Proof sketch. Clearly each C^2 property can be expressed in both FO and in $\mathsf{inf}\text{-}\mathrm{C}^2$. Thus, it suffices to show properties which disprove the opposite implication. For this, we can show that both $\varphi_{Lin}(x)$ and $\varphi_{GadLin}(x)$ are expressible in $\mathsf{inf}\text{-}\mathrm{C}^2$. Indeed, by the results obtained in the paper it suffices to show that the third condition from Proposition 4 can be expressed in $\mathsf{inf}\text{-}\mathrm{C}^2$ over directed graphs as

$$\bigwedge_{i \in \mathbb{N}} \forall x \forall y \, (\exists_{=i} y E(x,y) \land \exists_{=i} x E(y,x) \rightarrow x = y)$$

whereas ψ from Theorem 11 is expressed in $\mathsf{inf}\text{-}\mathrm{C}^2$ over undirected graphs as

$$\bigwedge_{i \in \mathbb{N}} \bigwedge_{j \in \mathbb{N}: i<j} \Big[\exists_{j+1} x P_1(x) \rightarrow \exists x \Big(\exists_{=j} y (P_2(y) \land E(x,y)) \land P_1(x) \land \exists y \Big(P_2(y) \land E(x,y) \land \exists x \Big(P_3(x) \land E(y,x) \land \exists y \big(P_1(y) \land E(x,y) \land \exists_{=i} x (P_2(x) \land E(y,x)) \big) \Big) \Big) \Big) \Big].$$

Note that both formulas rely on infinite conjunctions. □

7 Conclusions

In this paper, we have solved the open problem asking whether FO classifiers expressible by aggregate-combine-readout GNNs are exactly the classifiers expressible in logic C^2 (Barceló et al. 2020). As we show, the answer is negative. In particular, over both directed and undirected graphs, FO classifiers expressible by ACR-GNNs have a strictly higher expressive power than C^2. Recall that the distinguishing power of AC-GNNs is the same as of the 1-dimensional Weisfeiler-Leman algorithm, and so, the same as of C^2. It turns out, however, that the logical (FO) expressive power of standard GNN architectures cannot be characterised by C^2. In particular, AC-GNNs can express strictly less FO properties than C^2, whereas ACR-GNNs can express strictly more FO properties than C^2. Interestingly our results transfer to results on the expressive power of infinitary logics. As we have shown, the infinitary version of C^2 can express strictly more FO properties than the standard, finitary, C^2.

Acknowledgements

We are grateful to Bernardo Cuenca Grau for insightful comments and for drawing our attention to relevant work on the (in)definability of linear orders in fragments of first-order logics.

References

Ahvonen, V.; Heiman, D.; Kuusisto, A.; and Lutz, C. 2025. Logical Characterizations of Recurrent Graph Neural Networks with Reals and Floats. arXiv:2405.14606.

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*.

Benedikt, M.; Lu, C.; Motik, B.; and Tan, T. 2024. Decidability of Graph Neural Networks via Logical Characterizations. In *Proc. of ICALP*.

Besharatifard, M.; and Vafaee, F. 2024. A Review on Graph Neural Networks For Predicting Synergistic Srug 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.

Charatonik, W.; and Witkowski, P. 2016. Two-variable Logic with Counting and a Linear Order. *Logical Methods in Computer Science*, Volume 12, Issue 2.

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.

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. 2021. The Logic of Graph Neural Networks. In *Proc. of LICS*, 1–17.

Huang, X.; Orth, M. A. R.; Barceló, P.; Bronstein, M. M.; and Ceylan, İ. İ. 2025a. Link Prediction with Relational Hypergraphs. *Trans. Mach. Learn. Res.*

Huang, X.; Romero, M.; Ceylan, İ. İ.; and Barceló, P. 2025b. Logical Expressiveness of Graph Neural Networks on Knowledge Graphs. In *Handbook on Neurosymbolic AI and Knowledge Graphs*, 68–95.

Immerman, N.; and Kozen, D. 1989. Definability with Bounded Number of Bound Variables. *Inf. Comput.*, 83(2): 121–139.

Libkin, L. 2004. *Elements of Finite Model Theory*. Springer.

Lutz, C.; Sattler, U.; and Wolter, F. 2001. Modal Logic and the Two-Variable Fragment. In *Proc. of CSL*, 247–261.

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.

Pflueger, M.; Tena Cucala, D.; and Kostylev, E. V. 2024. Recurrent Graph Neural Networks and Their Connections to Bisimulation and Logic. In *Proc. of LICS*, 14608–14616.

Rossi, E.; Charpentier, B.; Giovanni, F. D.; Frasca, F.; Günnemann, S.; and Bronstein, M. M. 2023. Edge Directionality Improves Learning on Heterophilic Graphs. In *Proc. of LoG*.

Schönherr, M.; and Lutz, C. 2025. Logical Characterizations of GNNs with Mean Aggregation. *arXiv preprint arXiv:2507.18145*.

Szwast, W.; and Tendera, L. 2013. FO^2 with one transitive relation is decidable. In *Proc. of STACS*, 317–328.

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. J.; Cuenca Grau, B.; Kostylev, E. V.; and Motik, B. 2022. Explainable GNN-Based Models over Knowledge Graphs. In *Proc. of ICLR*.

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*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proc. of KDD*, 974–983.

Zhang, M.; and Chen, Y. 2018. Link Prediction Based on Graph Neural Networks. In *Proc. of NeurIPS*, 5171–5181.