The Expressive Power of Graph Neural Networks as a Query Language

Pablo Barceló[*]
IMC, PUC & IMFD Chile

Egor V. Kostylev
University of Oxford

Mikaël Monet
IMFD Chile

Jorge Pérez[†]
DCC, UChile & IMFD Chile

Juan L. Reutter[‡]
DCC, PUC & IMFD Chile

Juan-Pablo Silva
DCC, UChile

ABSTRACT

In this paper we survey our recent results characterizing various graph neural network (GNN) architectures in terms of their ability to *classify* nodes over graphs, for classifiers based on unary logical formulas– or queries. We focus on the language FOC_2, a well-studied fragment of FO. This choice is motivated by the fact that FOC_2 is related to the Weisfeiler-Lehman (WL) test for checking graph isomorphism, which has the same ability as GNNs for distinguishing nodes on graphs. We unveil the exact relationship between FOC_2 and GNNs in terms of node classification. To tackle this problem, we start by studying a popular basic class of GNNs, which we call AC-GNNs, in which the features of each node in a graph are updated, in successive layers, according only to the features of its neighbors. We prove that the unary FOC_2 formulas that can be captured by an AC-GNN are exactly those that can be expressed in its guarded fragment, which in turn corresponds to graded modal logic. This result implies in particular that AC-GNNs are too weak to capture all FOC_2 formulas. We then seek for what needs to be added to AC-GNNs for capturing all FOC_2. We show that it suffices to add readouts layers, which allow updating the node features not only in terms of its neighbors, but also in terms of a global attribute vector. We call GNNs with readouts ACR-GNNs. We also describe experiments that validate our findings by showing that, on synthetic data conforming to FOC_2 but not to graded modal logic, AC-GNNs struggle to fit in while ACR-GNNs can generalise even to graphs of sizes not seen during training.

[*]Institute for Mathematical and Computational Engineering, School of Engineering, Faculty of Mathematics, Pontificia Universidad Catolica de Chile.
[†]Department of Computer Science, University of Chile.
[‡]Department of Computer Science, School of Engineering, Pontificia Universidad Catolica de Chile.

1. INTRODUCTION

Graph neural networks (*GNNs*), which were introduced about a decade ago [21, 29], are a class of artificial neural network architectures that has recently become popular for a wide range of applications dealing with structured data, such as molecule classification, knowledge graph completion, and Web page ranking [6, 13, 17, 30]. The main idea behind GNNs is that the connections between neurons are not arbitrary but reflect the structure of the input data, which is given as a graph. Specifically, each node in the graph is associated a neuron, and the forward propagation of the neuron's data depends on the connections–or neighbors–of this neuron in the graph. This approach is motivated by convolutional and recurrent neural networks, and actually generalizes both of them [6].

Despite the fact that GNNs have recently been proven very efficient in many applications, their theoretical properties are not yet well-understood. We focus on the *expressive power* of GNNs, and concentrate on the ability of GNNs to express *node classifiers*, that is, functions assigning 1 (true) or 0 (false) to every node in a graph. More precisely, let us assume we have a GNN whose last layer behaves like a classifier: for every node v of the graph the last layer simply outputs a number 0 or 1. We say that this GNN can express a particular node classifier f, if for every graph G we have that the computation of the GNN assigns to every node v in G the value $f(v)$. This leads us to the following question:

> What type of node classifiers can be expressed as GNNs?

In the context of databases, one can see a graph as a *graph database* [27, 5], and a classifier f as a query language: On input graph (database) G, the

query would return all the nodes in G that are classified as true by f. Thus, answering the question above implies understanding what type of queries can be expressed by GNNs.

Our first observation draws from an interesting result published independently by Morris et al. [23] and Xu et al. [34] that establishes a connection between GNNs and the *Weisfeiler-Lehman* (*WL*) test for checking graph isomorphism. The WL test works by constructing a labeling of the nodes of the graph, in an incremental fashion, and then decides whether two graphs are isomorphic by comparing the labeling of each graph. To state the connection between GNNs and this test, consider the popular GNN architecture that updates the feature vector of each graph node by combining it with the (aggregation of) the feature vectors of its neighbors. We call such GNNs *aggregate-combine GNNs*, or *AC-GNNs*. The authors of these papers independently observe that the node labeling produced by the WL test always refines the labeling produced by any GNN. More precisely, if two nodes are labeled the same by the algorithm underlying the WL test, then the feature vectors of these nodes produced by any AC-GNN will always be the same. Moreover, there are AC-GNNs that can reproduce the WL labeling, and hence AC-GNNs can be as powerful as the WL test for distinguishing nodes.

In terms of queries, these connections give us a sort of upper bound: we see that AC-GNNs can only express queries that agree with the WL test, in the sense that all nodes assigned the same WL label are either all part of the answer, or none of them is. However, this gives us little in terms of understanding the actual queries that can be expressed by GNNs.

To pursue further in this topic, we concentrate on queries expressible in first-order logic. For AC-GNNs, a meaningful starting point to measure their expressive power is the logic FOC_2, the two-variable fragment of FO extended with counting quantifiers of the form $\exists^{\geq N} x\, \varphi(x)$, which state that there are at least N nodes satisfying formula φ [7].[1] This choice of FOC_2 is justified by a classical result establishing a tight connection between FOC_2 and WL: two nodes in a graph are classified the same by the WL test if and only if they satisfy exactly the same unary FOC_2 formulas [7].

Given the connection between AC-GNNs and WL on the one hand, and that between WL and FOC_2 on the other hand, it is natural to think that the expressivity of AC-GNNs coincides with that of FOC_2, at least in terms of classifiers or unary queries. Surprisingly, this is not the case; indeed, we will see that there are many FOC_2 unary formulas that cannot be expressed by AC-GNNs. This leaves us with the following natural questions. First, what is the largest fragment of FOC_2 that can be captured by AC-GNNs? Second, is there an extension of AC-GNNs that allows to express all FOC_2 (unary) formulas? In this paper we provide answers to these two questions. The following are the main results outlined in this paper.

First, we characterize exactly the fragment of FOC_2 that can be expressed as AC-GNNs. This fragment corresponds to *graded modal logic* [9], or, equivalently, to the *description logic* $\mathcal{ALCQ}$, which has received considerable attention in the knowledge representation community [2, 3]. What is more, we show that formulas of this kind can be expressed in terms of a particularly simple class of GNNs, which we call *homogeneous* AC-GNNs. We present these results in Section 4.

Second, we extend the AC-GNN architecture in a simple way by allowing global *readouts*, where in each layer we also compute a feature vector for the whole graph and combine it with local aggregations; we call these *aggregate-combine-readout GNNs*, or ACR-GNNs. These networks are a special case of the networks proposed by Battaglia et al. [6] for relational reasoning over graph representations. In this setting, we prove that each FOC_2 formula can be captured by an ACR-GNN. In this setting, we also prove that each FOC_2 formula can be captured by an ACR-GNN using a single readout. These results are presented in Section 5.

Finally, we experimentally validate our findings in Section 6, where we show that the theoretical expressiveness of ACR-GNNs, as well as the differences between AC-GNNs and ACR-GNNs, can be observed when we learn from examples. In particular, we show that on synthetic graph data conforming to FOC_2 formulas, AC-GNNs struggle to fit the training data while ACR-GNNs can generalize even to graphs of sizes not seen during training.

Remark. This paper summarizes recent results published by the same authors in a machine learning conference paper [4]; however, the presentation is adapted to a reader in the database community.

2. PRELIMINARIES

In this section we describe the architecture of basic GNN classifiers, AC-GNNs, and introduce other related notions. We consider the problem of Boolean

[1] Note that every formula in FOC_2 can also be expressed in FO, albeit with more than just two variables.

node classification in graphs, where we wish to classify each graph node as true or false on the base of the structure of its neighborhood. We concentrate on undirected graphs without self-loops and multiedges and where each node is assigned with a unique color from a finite set; however, our results can be generalised to directed edge-colored multigraphs with loops in a straightforward way.

Graph neural networks. The basic architecture for GNNs, and the one studied in recent articles on GNN expressibility [23, 34], consists of a sequence of *layers* that combine the feature vectors of every node with the multiset of feature vectors of its neighbors, as formalized in the following definition.

DEFINITION 2.1. *An* aggregate-combine GNN *(*AC-GNN*)* $\mathcal{A}$ *with* $L \geq 1$ *layers is specified by two sets of functions,* $\{\mathrm{AGG}^{(i)}\}_{i=1}^{L}$ *and* $\{\mathrm{COM}^{(i)}\}_{i=1}^{L}$*, called* aggregation *and* combination *functions, respectively, and a* classification *function* CLS*. Each aggregation function* $\mathrm{AGG}^{(i)}$ *takes a multiset of (rational) vectors and returns one such vector, each combination function* $\mathrm{COM}^{(i)}$ *takes a pair or vectors and returns one vector, and the classification function* CLS *takes a vector and returns a Boolean value,* true *or* false *(the vector dimensions of these functions are assumed to match the semantics of the GNN as defined next).*

An AC-GNN $\mathcal{A}$ takes a graph G as input and computes feature vectors $\boldsymbol{x}_v^{(i)}$, for each node v of G and each layer $i = 1, \ldots, L$, via the recursive formula

$$\boldsymbol{x}_v^{(i)} = \mathrm{COM}^{(i)}\Big(\boldsymbol{x}_v^{(i-1)}, \mathrm{AGG}^{(i)}\big(\{\!\!\{\boldsymbol{x}_u^{(i-1)} \mid u \in \mathcal{N}_G(v)\}\!\!\}\big)\Big),$$

where $\mathcal{N}_G(v)$ is the neighborhood of v in G and the initial vector $\boldsymbol{x}_v^{(0)}$ is the one-hot encoding of the color of v in G (i.e., the dimension of $\boldsymbol{x}_v^{(0)}$ is the number of possible colors and $\boldsymbol{x}_v^{(0)}$ has the k-th component 1 if its color has number k and 0 otherwise). Finally, each node v of G is classified as true or false according to CLS applied to $\boldsymbol{x}_v^{(L)}$. We define $\mathcal{A}(G, v) := \mathrm{CLS}(\boldsymbol{x}_v^{(L)})$, for each node v in G.

Aggregation, combination, classification functions. Many possible aggregation, combination, and classification functions exist, which produce different classes of GNNs [14, 17, 23, 34]. A simple, yet common choice is to consider the sum of feature vectors as the aggregation function, the sign of one of the elements in $\boldsymbol{x}_v^{(L)}$ as the classification function, and the combination function

$$\mathrm{COM}^{(i)}(\boldsymbol{x}_1, \boldsymbol{x}_2) = f\big(\boldsymbol{x}_1 \boldsymbol{C}^{(i)} + \boldsymbol{x}_2 \boldsymbol{A}^{(i)} + \boldsymbol{b}^{(i)}\big), \quad (1)$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are row vectors, $\boldsymbol{C}^{(i)}$ and $\boldsymbol{A}^{(i)}$ are matrices of parameters (of appropriate dimensions), $\boldsymbol{b}^{(i)}$ is a *bias* row vector of parameters, and f is a *non-linear* function, such as (truncated) ReLU or sigmoid [22]. We call *simple* an AC-GNN using these functions. Note that the parameters in $\boldsymbol{C}^{(i)}$, $\boldsymbol{A}^{(i)}$, and $\boldsymbol{b}^{(i)}$ are usually found during the training of the GNN (e.g., using standard ML techniques [22]). We say that an AC-GNN is *homogeneous* if all $\mathrm{AGG}^{(i)}$ are the same and all $\mathrm{COM}^{(i)}$ are the same (i.e., share the same parameters across layers). In most of our positive results we construct simple and homogeneous GNNs, while our negative results hold in general, i.e., for GNNs with arbitrary aggregation, combination, and classification functions and that are not necessarily homogeneous.

We note that besides node classification, which we consider in this paper, one can use GNNs to classify whole graphs. This can be done, for example, by considering that the classification function CLS inputs the multiset $\{\!\!\{\boldsymbol{x}_v^{(L)}\}\!\!\}$ of feature vectors over all nodes v in the graph and outputs a classification of the whole graph. In this case the classification function is often called *readout* [23, 34]. In this paper, however, we use the term "readout" to refer to functions applied globally on intermediate layers of ACR-GNNs (i.e., GNNs that are more expressive than AC-GNNs, see Section 5).

Weisfeiler-Lehman. The *Weisfeiler-Lehman* (*WL*) test (also called *node coloring*) is a powerful heuristic used to solve the graph isomorphism problem [7, 32], or, for our purposes, to determine whether the neighborhoods of two nodes in a graph are structurally close. Formally, the L-round WL algorithm takes as input a (node-colored) graph G and iteratively assigns, for L rounds, a new color to every node in the graph in such a way that the color of a node assigned in round i is uniquely and unambiguously defined by (i.e., has a one-to-one correspondence with) its own color in round $i-1$ and with the multiset of colors of its neighbors in G in round $i-1$. The result of the algorithm is the coloring of the nodes after round L; then, the multisets of the resulting colors in two graphs can be compared for testing their (non-)isomorphism [7, 32]. An important observation is that the rounds of the WL algorithm can be seen as the layers of an AC-GNN whose aggregation and combination functions are all injective [23, 34]. Furthermore, as independently

shown by Morris et al. [23] and Xu et al. [34], an AC-GNN classification can never contradict the WL test, in the following sense.

PROPOSITION 2.2. *If the L-round WL algorithm assigns the same color to two nodes in a graph, then every AC-GNN with L-layers classifies both nodes the same (i.e., either both as* true *or both as* false*).*

3. GNNS AND LOGIC

Our study relates the expressive power of GNNs to that of classifiers formalized as unary formulas in first order logic with equality (FO) and some of its fragments. It is well-known that FO logic underlies many standard database query languages, such as SQL, and thus our work bridges the gap between structure-aware machine learning architectures on the one side and classic declarative query formalisms on the other side.

Since we concentrate on undirected node-colored graphs, we consider the signature consisting of a single binary predicate Edge and unary predicates corresponding to the possible node colors, as well as assume that all the logical structures encode such graphs (in particular, the interpretation of Edge is always symmetric). As formalized in the following definition, we say that a GNN classifier (i.e., an AC-GNN or a GNN of more expressive architecture as described later) captures a logical classifier when both classifiers agree on every node in every graph.

DEFINITION 3.1. *A GNN classifier $\mathcal{A}$* captures *a logical formula $\varphi(x)$ if for every graph G and node v in G, it holds that $\mathcal{A}(G,v) = \mathsf{true}$ if and only if $(G,v) \models \varphi$.*

3.1 Logic FOC_2 and the WL test

As we have outlined in the introduction, we focus on formulas in FOC_2, the fragment of FO logic that only allows formulas with two variables, but in turn permits the use of *counting quantifiers* [7]. Such quantifiers have the form $\exists^{\geq N}$ for a positive integer N, and a formula $\exists^{\geq N} x\, \varphi(x)$ holds if there are *at least N different nodes* for which φ holds. For example, in FOC_2 we can express a formula that checks whether x is a red node, and there is another node that is not connected to x and that has at least two blue neighbors:

$$\gamma(x) \coloneqq \mathsf{Red}(x) \wedge \exists y\big(\neg\mathsf{Edge}(x,y) \wedge \exists^{\geq 2} x\big[\,\mathsf{Edge}(y,x) \wedge \mathsf{Blue}(x)\,\big]\big).$$

Despite that FOC_2 is not a syntactic fragment of FO logic due to the counting quantifiers, it is a semantic fragment, because these quantifiers can be expressed via usual existential quantifiers and disequalities. For example, the formula $\gamma(x)$ above can be written in FO as

$$\beta(x) \coloneqq \mathsf{Red}(x) \wedge \exists y\big(\neg\mathsf{Edge}(x,y) \wedge \exists z_1 \exists z_2\big[\,\mathsf{Edge}(y,z_1) \wedge \mathsf{Edge}(y,z_2) \wedge z_1 \neq z_2 \wedge \mathsf{Blue}(z_1) \wedge \mathsf{Blue}(z_2)\,\big]\big).$$

Note, however, that this rewriting is possible only by means of increasing the number of used variables, and it is easy to see that this formula cannot be expressed in FO_2, the fragment of FO that allows only two variables (and no counting quantifiers). On the other hand, FO is strictly more expressive than FOC_2; this is witnessed, for example, by a formula checking whether a graph has a triangle as a subgraph.

The following result, which is due to Cai et al. [7], establishes a classical connection between FOC_2 and the WL test. Together with Proposition 2.2, it provides a justification for our choice of the logic FOC_2 for measuring the expressiveness of AC-GNNs.

PROPOSITION 3.2. *For every graph G and nodes u, v in G, we have that u and v agree on every* FOC_2 *unary formula if and only if the WL algorithm colors v and u the same after arbitrary many rounds.*

3.2 FOC_2 and AC-GNN classifiers

Having Propositions 2.2 and 3.2 at hand, one may be tempted to combine them and claim that every FOC_2 formula can be captured by an AC-GNN. Yet this is not the case, as we show in Proposition 3.3 below. In fact, while it is true that two nodes are indistinguishable by the WL test if and only if they are indistinguishable by FOC_2 (Proposition 3.2), and if the former holds then such nodes cannot be distinguished by AC-GNNs (Proposition 2.2), this by no means tells us that every FOC_2 formula can be captured by an AC-GNN.

PROPOSITION 3.3. *There are* FOC_2 *formulas that are not captured by any AC-GNN. In fact, this holds even for FO formulas using only two variables and no counting quantifiers.*

PROOF. Consider the formula $\alpha(v) \coloneqq \mathsf{Red}(v) \wedge \exists x\ \mathsf{Green}(x)$. We will show by contradiction that there is no AC-GNN that captures α, no matter which aggregation, combination, and final classification functions are allowed. Indeed, assume that $\mathcal{A}$ is an AC-GNN capturing α, and let L be its number of layers. Consider the graph G that is a chain of $L+2$ nodes colored Red, and consider the first node v_0 in that chain. Since $\mathcal{A}$ captures α, and since $(G, v_0) \not\models$

α, we have that $\mathcal{A}$ labels v_0 with false, that is, $\mathcal{A}(G, v_0) = \mathsf{false}$. Now, consider the graph G' obtained from G by coloring the last node in the chain with Green (instead of Red). Then one can easily show that $\mathcal{A}$ again labels v_0 by false in G'. But we have $(G', v_0) \models \alpha$, a contradiction. $\square$

The above proof relies on the following weakness of AC-GNNs: if the number of layers is fixed (i.e., does not depend on the input graph), then the information of the color of a node v cannot travel further than at distance L from v. Nevertheless, we can show that the same holds even when we consider AC-GNNs that dispose of an arbitrary number of layers (for instance, one may want to run a homogeneous AC-GNN for $f(|E|)$ layers for each graph $G = (V, E)$, for a fixed function f). Assume again by way of contradiction that $\mathcal{A}$ is such an extended AC-GNN capturing α. Consider the graph G consisting of two disconnected nodes v, u, with v colored Red and y colored Green. Then, since $(G, v) \models \alpha$, we have $\mathcal{A}(G, v) = \mathsf{true}$. Now consider the graph G' obtained from G by changing the color of u from Green to Red. Observe that, since the two nodes are not connected, we will again have $\mathcal{A}(G', v) = \mathsf{true}$, contradicting the fact that $(G', v) \not\models \alpha$ and that $\mathcal{A}$ is supposed to capture α.

From these proofs we get two pieces of intuition. One problem is that an AC-GNN has only a fixed number L of layers and hence the information of local aggregations cannot travel further than at distance L of every node along edges in the graph. But there are times when no number of layer suffices, simply because two nodes may be disconnected in the graph. This negative result opens up the following questions.

1. What kind of FOC_2 formulas can be captured by AC-GNNs?
2. Can we capture FOC_2 classifiers with GNNs using a simple extension of AC-GNNs?

We answer these questions in the next two sections.

4. EXPRESSIVE POWER OF AC-GNNS

Towards answering our first question, we recall that the problem with AC-GNN classifiers is that they are local, in the sense that they cannot see across a distance beyond their number of layers. Thus, if we want to understand which queries this architecture is capable of expressing, we must consider logics built with similar limitations in mind. And indeed, in this section we show that AC-GNNs capture any FOC_2 formula as long as they satisfy such a locality property. This happens to be a well-known restriction of FOC_2 that corresponds to *graded modal logic* [9] or, equivalently, to the description logic $\mathcal{ALCQ}$ [2], which is fundamental for knowledge representation: for instance, the OWL 2 Web Ontology Language [24, 31] relies on $\mathcal{ALCQ}$.

The idea of graded modal logic is to force all subformulas to be *guarded* by the edge predicate Edge. This means that one cannot express in graded modal logic arbitrary formulas of the form $\psi(x) = \exists y\, \varphi(y)$ (that is, whether there is some node that satisfies property φ). Instead, one is allowed to check whether some *neighbor* y of the node x where the formula is being evaluated satisfies φ. For instance, we are allowed to express the formula $\psi(x) = \exists y\, (\mathsf{Edge}(x, y) \wedge \varphi(y))$ in the logic as in this case $\varphi(y)$ is guarded by $\mathsf{Edge}(x, y)$.

We can formally define this logic using FOC_2 syntax as follows (note that both graded modal logic and $\mathcal{ALCQ}$ have their own syntaxes, but we stick to the general FO syntax for uniformity).

Definition 4.1. *A* graded modal logic formula *is either* $\mathrm{Col}(x)$*, for* Col *a node color, or one of the following, where* φ *and* ψ *are graded modal logic formulas and* N *is a positive integer:*

$$\neg\varphi(x), \quad \varphi(x) \wedge \psi(x), \quad \exists^{\geq N} y\, (\mathsf{Edge}(x, y) \wedge \varphi(y)).$$

For example, the formula

$$\delta(x) \;\; := \;\; \mathsf{Red}(x) \;\wedge\; \exists y \left(\mathsf{Edge}(x, y) \;\wedge\; \mathsf{Blue}(y)\right)$$

is in graded modal logic, but the formula

$$\gamma(x) := \mathsf{Red}(x) \wedge \exists y\big(\neg\mathsf{Edge}(x, y) \wedge \exists^{\geq 2} x\big[\,\mathsf{Edge}(y, x) \wedge \mathsf{Blue}(x)\,\big]\big).$$

of Section 3 is not, because the use of $\neg\mathsf{Edge}(x, y)$ as a guard is disallowed. Observe that all graded modal logic formulas are unary by definition, so all of them define unary queries. As promised, we now show that AC-GNNs can indeed capture all graded modal logic classifiers.

Proposition 4.2. *Each graded modal logic classifier is captured by a simple homogeneous AC-GNN.*

Proof sketch. The key idea of the construction is that the components of a node's feature vector can represent the subformulas of the captured logical classifier that hold in the node. An AC-GNN then can implement a standard dynamic programming algorithm over the graph G such that, after k layers, it declares a feature in a node v to be 1 iff v satisfies the corresponding subformula φ over G.

Let $\varphi(x)$ be a formula in graded modal logic. Let $\text{sub}(\varphi) = (\varphi_1, \varphi_2, \ldots, \varphi_L)$ be an enumeration of the sub-formulas of φ such that if φ_k is a sub-formula of φ_ℓ then $k \leq \ell$. We show how to construct a simple and homogeneous AC-GNN $\mathcal{A}_\varphi$ capturing $\varphi(x)$. As mentioned, the idea is that $\mathcal{A}_\varphi$ uses feature vectors in $\mathbb{R}^L$ such that every component of those vectors represents a different formula in $\text{sub}(\varphi)$. Then $\mathcal{A}_\varphi$ will update the feature vector $\boldsymbol{x}_v^{(i)}$ of node v ensuring that component ℓ of $\boldsymbol{x}_v^{(i)}$ gets a value 1 if and only if the formula φ_ℓ is satisfied in node v, for every $i \geq l$. We note that $\varphi = \varphi_L$ and thus, the last component of each feature vector after evaluating L layers in every node gets a value 1 if and only if the node satisfies φ. We will then be able to use a final classification function CLS that simply extracts that particular component.

The simple homogeneous AC-GNN $\mathcal{A}_\varphi$ has L layers and uses aggregation and combine functions

$$\begin{aligned} \text{AGG}(X) &= \sum_{\boldsymbol{x} \in X} \boldsymbol{x}, \\ \text{COM}(\boldsymbol{x}, \boldsymbol{y}) &= \sigma\big(\boldsymbol{x}\boldsymbol{C} + \boldsymbol{y}\boldsymbol{A} + \boldsymbol{b}\big), \end{aligned}$$

where $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{R}^{L \times L}$, and $\boldsymbol{b} \in \mathbb{R}^L$ are defined next, and σ is the truncated ReLU activation defined by $\sigma(x) := \min(\max(0, x), 1)$. The entries of the ℓ-th columns of $\boldsymbol{A}, \boldsymbol{C}$, and $\boldsymbol{b}$ depend on the sub-formulas of φ as follows:

- if $\varphi_\ell(x) = \text{Col}(x)$ with Col one of the (base) colors, then $C_{\ell\ell} = 1$,
- if $\varphi_\ell(x) = \varphi_j(x) \wedge \varphi_k(x)$ then $C_{j\ell} = C_{k\ell} = 1$ and $b_\ell = -1$,
- if $\varphi_\ell(x) = \neg\varphi_k(x)$ then $C_{k\ell} = -1$ and $b_\ell = 1$,
- if $\varphi_\ell(x) = \exists^{\geq N}(E(x, y) \wedge \varphi_k(y))$ then $A_{k\ell} = 1$ and $b_\ell = -N + 1$,

and all other values in the ℓ-th columns of $\boldsymbol{A}, \boldsymbol{C}$, and $\boldsymbol{b}$ are 0.

To show correctness, let $G = (V, E)$ be a colored graph. For every node v in G we consider the initial feature vector $\boldsymbol{x}_v^{(0)} = (x_1, \ldots, x_L)$ such that $x_\ell = 1$ if sub-formula φ_ℓ is the initial color assigned to v, and $x_\ell = 0$ otherwise. By definition, AC-GNN $\mathcal{A}_\varphi$ will iterate the aggregation and combine functions defined above for L rounds (L layers) to produce feature vectors $\boldsymbol{x}_v^{(i)}$ for every node $v \in G$ and $i = 1, \ldots, L$. All that remains is to show that for every $\varphi_\ell \in \text{sub}(\varphi)$, every $i \in \{\ell, \ldots, L\}$, and every node v in G it holds that:

$(\boldsymbol{x}_v^{(i)})_\ell = 1$ if $(G, v) \models \varphi_\ell$ and $(\boldsymbol{x}_v^{(i)})_\ell = 0$ otherwise,

where $(\boldsymbol{x}_v^{(i)})_\ell$ is the ℓ-th component of $\boldsymbol{x}_v^{(i)}$. But this can easily be proved by induction on the number of sub-formulas of every φ_ℓ. □

An interesting open question is whether the same kind of construction can be done with AC-GNNs using different aggregate and combine operators from the ones we consider here; for instance, using max instead of sum to aggregate the feature vectors of the neighbors, or using other non-linearities such as sigmoid.

Interestingly, the relationship between AC-GNNs and graded modal logic goes further: we can show that graded modal logic is the *largest* class of FO logical classifiers captured by AC-GNNs—that is, the only FO formulas that AC-GNNs are able to learn accurately are those in graded modal logic.

THEOREM 4.3. *A logical classifier is captured by AC-GNNs if and only if it can be expressed in graded modal logic.*

The backward direction of this theorem is Proposition 4.2. On the other hand, the proof of the forward direction is based on a van Benthem & Rosen characterization obtained by Otto [26, Theorem 2.2] for finite graphs, stating that an FO formula can be expressed in graded modal logic if and only if the formula only depends on the *unraveling* of the nodes, which in turn correspond to the colors assigned by the WL test. While the setting considered by Otto is slightly different from ours (in particular, we consider directed graphs, as opposed to undirected), these differences can be shown to be inessential, and the proof carries over to this setting. We point out that the forward direction holds no matter which aggregate and combine operators are considered—that is, this is a limitation of the AC-GNN architecture, not of the specific functions that one chooses to update the features.

5. GNNS FOR CAPTURING FOC_2

In this section we tackle our second question: Which GNN architectures do we need to capture all FOC_2 classifiers? Recall that the main shortcoming of AC-GNNs for expressing such classifiers is their local behavior. A natural way to avoid this behavior is to allow for a global feature computation on each layer of the GNN. This is called a *global attribute* computation in the framework of [6]. Following the recent GNN literature [13, 23, 34], we refer to this global operation as a *readout*. We begin with formalizing the GNN architecture with readouts. We then show how readouts serve in capturing all of FOC_2, and finish with an observation on the number of readouts needed in these neural networks.

5.1 GNNs with global readouts

Our definition of GNNs with readouts is a generalization of the Definition 2.1 for AC-GNNs.

DEFINITION 5.1. *An* aggregate-combine-readout GNN *(*ACR-GNN*) with L layers extends AC-GNNs by* readout *functions* $\{\text{READ}^{(i)}\}_{i=1}^{L}$*, which aggregate the (multiset of the) current feature vectors of all the nodes in a graph to a single vector; additionally, the combination functions* $\text{COM}^{(i)}$ *take three arguments rather than two. Then, the feature vector* $\boldsymbol{x}_v^{(i)}$ *of each node v in a graph G on each layer i is computed by the following recursive formula, where V is the set of all nodes in G:*

$$\boldsymbol{x}_v^{(i)} = \text{COM}^{(i)}\Big(\boldsymbol{x}_v^{(i-1)}, \text{AGG}^{(i)}\left(\{\!\{\boldsymbol{x}_u^{(i-1)} \mid u \in \mathcal{N}_G(v)\}\!\}\right), \text{READ}^{(i)}\left(\{\!\{\boldsymbol{x}_u^{(i-1)} \mid u \in V\}\!\}\right)\Big).$$

Intuitively, every layer in an ACR-GNN first computes (i.e., "reads out") the aggregation over all the nodes in G; then, for every node v, it computes the aggregation over the neighbors of v; and finally it combines the features of v with the two aggregation vectors.

All the notions about AC-GNNs extend to ACR-GNNs in a straightforward way; for example, a *simple* ACR-GNN uses the sum as the function $\text{READ}^{(i)}$ in each layer, and the following combination function, generalizing Equation (1):

$$\text{COM}^{(i)}(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = f\big(\boldsymbol{x}_1\boldsymbol{C}^{(i)} + \boldsymbol{x}_2\boldsymbol{A}^{(i)} + \boldsymbol{x}_3\boldsymbol{R}^{(i)} + \boldsymbol{b}^{(i)}\big),$$

where $\boldsymbol{R}^{(i)}$ is one more matrix of parameters.

5.2 ACR-GNNs and FOC_2

To see how readout functions could help in capturing non-local properties, consider again the formula $\gamma(x)$ from above, that assigns true to every red node v unless there is another node not connected with v having at least two blue neighbors. It is easy to show, by adapting the proof of Proposition 3.3, that no AC-GNN can capture this classifier. However, using a single readout and local aggregations, one can implement this classifier as follows. Let B be the property "having at least 2 blue neighbors". Then an ACR-GNN that implements $\gamma(x)$ can first use a local aggregation to store in the feature of every node if the node satisfies B, then use a readout function to count the nodes satisfying B in the whole graph, and finally use another local aggregation to count neighbors of every node satisfying B. Then γ is obtained by classifying as true every red node having less neighbors satisfying B than the total number of nodes satisfying B in the whole graph. It turns out that the usage of readout functions is enough to capture all non-local properties of FOC_2 classifiers.

THEOREM 5.2. *Each* FOC_2 *classifier can be captured by a simple homogeneous ACR-GNN.*

PROOF SKETCH. As an intermediate step in the proof, we use a characterization of FOC_2 using an extended version of graded modal logic, which was obtained by Lutz et al. [19], and relates FO_2 with a modal logic that can use parameters to navigate to all nodes not connected, or different to, the current node. This connection can be extended to FOC_2 and the counting version of this modal logic, which is denoted as $\mathcal{EMLC}$.

Next, we show how to capture any $\mathcal{EMLC}$ formula with an ACR-GNN. Since $\mathcal{EMLC}$ formulas are essentially the extension of graded modal logic with these negated modalities, we can reuse most of the proof of Proposition 4.2. The novelty is that we use readouts to take care of subformulas with negated modalities. Thus, readout functions are only used to deal with subformulas asserting the existence of a node that is not connected to the current node in the graph, just as we did for classifier $\gamma(x)$. □

Note that Proposition 4.2 has two directions while Theorem 5.2 just one; we leave as a challenging open problem the other direction of Theorem 5.2—that is, whether the FOC_2 classifiers are exactly the logical classifiers (i.e., FO logic unary formulas) captured by ACR-GNNs.

5.3 The number of layers with readouts

The proof of Theorem 5.2 constructs ACR-GNNs whose number of layers depends on the size of the formula being captured; moreover, readouts are used on unboundedly many (in some cases all) layers of these GNNs. Given that a global computation can be costly, one might wonder whether this is really needed, or if it is possible to cope with all the complexity of such classifiers by performing only a few readouts. We next show the surprising fact that just one readout is actually always enough. However, this reduction in the number of readouts comes at the cost of severely complicating the resulting GNN.

DEFINITION 5.3. *An* aggregate-combine GNN with final readout *(*AC-FR-GNN*) is the same as an ACR-GNN except that only the final layer uses a readout function.*

The following theorem formalizes the result of this section.

THEOREM 5.4. *Each* FOC_2 *classifier is captured by an AC-FR-GNN.*

The AC-FR-GNN construction in the proof of this theorem is not based on the idea of evaluating the formula incrementally along layers, as in the proofs of Proposition 4.2 and Theorem 5.2, and it is not simple (note that AC-FR-GNNs are never homogeneous). Instead, it is based on a refinement of the GIN architecture proposed by Xu et al. [34] (which is also used in the proof of the second claim of Proposition 2.2) to obtain as much information as possible about the local neighborhood in graphs, followed by a readout and combination functions that use this information to deal with non-local constructs in formulas. The first component we build is an AC-GNN that computes an invertible function mapping each node to a number representing its neighborhood (how big is this neighborhood depends on the classifier to be captured). This information is aggregated so that we know for each different type of a neighborhood how many times it appears in the graph. We then use the combine function to evaluate FOC_2 formulas by decoding back the neighborhoods.

6. EXPERIMENTAL RESULTS

In this section we report on our experiments, which are aimed to empirically validate our theoretical findings.

6.1 Overview and Set Up

Our main motivation was to show that the theoretical expressiveness of ACR-GNNs, as well as the difference between AC- and ACR-GNNs, can actually be observed when we learn from examples. To this end, we performed two sets of experiments on synthetic data: experiments to show that ACR-GNNs can learn a very simple FOC_2 node classifier that AC-GNNs cannot learn, and experiments involving complex FOC_2 classifiers that need more intermediate readouts to be learned. Besides testing simple AC-GNNs, we also tested the GIN network proposed by Xu et al. [34] (we used the implementation by Fey and Lenssen [11] and adapted it to classify nodes).

We performed these two experiments using synthetic graphs with five initial colors; these graphs are divided in three sets: train set with 5000 graphs with 50 to 100 nodes, test set with 500 graphs with a similar number of nodes, and another test set with 500 graphs about twice larger than the train set.

We tried several configurations for the aggregation, combination and readout functions, but observed a consistent pattern in which the setting of simple AC(R)-GNNs with ReLU activation as described above produced the most accurate results. Besides this, we did not do any hyperparameter search and did not use any regularisation. Accuracy in our experiments is computed as the total number of nodes correctly classified among all nodes in all the graphs in the dataset. In addition, we report on our preliminary experiments on a real-life *Protein-Protein Interaction* (*PPI*) benchmark [36], where we did not observe an improvement of ACR-GNNs over AC-GNNs.

We implemented our experiments using the PyTorch Geometric library [11]. In all cases we trained with a batch-size of 128, and run up to 50 epochs with the Adam optimizer and default PyTorch parameters.[2]

6.2 Separating AC-GNNs and ACR-GNNs

In our first set of experiments we considered a very simple FOC_2 classifier defined by

$$\alpha(x) \coloneqq \mathsf{Red}(x) \wedge \exists y\ \mathsf{Blue}(y),$$

which is satisfied by every red node in a graph provided that the graph contains at least one blue node. This classifier is not expressible in graded modal logic, so we expected very good performance from ACR-GNNs but difficulties for AC-GNNs.

We tested the GNN architectures with two classes of graphs. First, we considered line-shaped graphs, each of which has $2n$ nodes $v_1, \ldots, v_{2n}$ such that each v_i is connected to v_{i+1}, and such that only nodes $v_1, \ldots, v_n$ can be colored blue and only others can be colored red. Second, we considered Erdös-Renyi random graphs of two flavors: the graphs with the same number of nodes and edges, and the graphs where the number of edges is twice the number of nodes. In every set we had 50% of graphs containing no blue node, and others containing a fixed small number of blue nodes (typically less than five). Also, to ensure that there is a significant number of nodes satisfying the formula, we forced graphs to have at least $1/4$ of its nodes colored red.

The results of these experiments are shown in Table 1. As we can see there, already ACR-GNNs with a single layer showed perfect performance for both types of graphs (ACR-1 in Table 1). This was what we expected given the simplicity of the property being checked. In contrast, AC-GNNs and GINs (shown in Table 1 as AC-L and GIN-L, representing AC-GNNs and GINs with L layers) struggle to fit the data. For the case of the line-shaped graph, they were not able to fit the train data even by al-

[2]All our code and data can be accessed online at https://github.com/juanpablos/GNN-logic.

	Line-Shaped Train	Line-Shaped Test		Erdös-Renyi Train	Erdös-Renyi Test	
		same-size	bigger		same-size	bigger
AC-5	0.887	0.886	0.892	0.951	0.949	0.929
AC-7	0.892	0.892	0.897	0.967	0.965	0.958
GIN-5	0.861	0.861	0.867	0.830	0.831	0.817
GIN-7	0.863	0.864	0.870	0.818	0.819	0.813
ACR-1	1.000	1.000	1.000	1.000	1.000	1.000

Table 1: Results on synthetic data for nodes labeled by classifier $\alpha(x) \coloneqq \mathsf{Red}(x) \wedge \exists y\ \mathsf{Blue}(y)$

lowing 7 layers. For the case of random graphs, the performance with 7 layers was considerably better but still did not fit the data perfectly. We allowed AC-GNNs with 7 layers to run for more epochs but the results did not improve.

In a closer look at the performance for different connectivities of E-R graphs, we found an improvement for AC-GNNs when we train them with more dense graphs (i.e., when the number of edges increases while the number of nodes stays the same). This is consistent with the fact that AC-GNNs are able to move information of local aggregations to distances up to their number of layers. This combined with the fact that random graphs that are more dense make the maximum distances between nodes shorter, may explain the boost in performance for AC-GNNs.

6.3 Complex FOC_2 Properties

In the second experiment we consider classifiers $\alpha_i(x)$ constructed as

$$\alpha_0(x) \coloneqq \mathsf{Blue}(x), \tag{2}$$

$$\alpha_i(x) \coloneqq \exists^{[N_i,M_i]} y \left(\alpha_{i-1}(y) \wedge \neg \mathsf{Edge}(x,y)\right), \tag{3}$$

where $\exists^{[N,M]}$ stands for "there are exactly between N and M nodes" satisfying a given property. Observe that each $\alpha_i(x)$ is in FOC_2, as $\exists^{[N,M]}$ can be expressed by combining $\exists^{\geq N}$ and $\neg\exists^{\geq M+1}$; however, the classifiers α_i, for $i \geq 1$, are not expressible in graded modal logic. In particular, we concentrated on $\alpha_1(x)$, $\alpha_2(x)$ and $\alpha_3(x)$ with $[N_1, M_1]$, $[N_2, M_2]$ and $[N_3, M_3]$ being $[8, 10]$, $[10, 20]$ and $[10, 20]$ (the choice of these intervals is technical, it results in the number of satisfying nodes in the graphs as described below).

We considered sets of Erdös-Renyi random graphs with the number of edges about 7 times greater than the number of nodes (i.e., more dense that in the first experiments), and colored to ensure that approximately one half of all nodes in the graphs in the set satisfy each of $\alpha_1(x)$, $\alpha_2(x)$ and $\alpha_3(x)$.

Our results, given in Table 2, show that when increasing i (i.e., the quantifier depth of the classifiers α_i) more layers are needed to increase train and test accuracy. We report ACR-GNNs performance up to 3 layers (ACR-L in Table 2) as beyond that we did not see any significant improvement. We also note that for the bigger test set, AC-GNNs and GINs are unable to substantially depart from a trivial baseline of 50%. We tested these networks with up to 10 layers but only report the best results on the bigger test set. We also test AC-FR-GNNs with two and three layers (AC-FR-L in Table 2). As we expected, although theoretically using a single readout gives the same expressive power as using several of them (Theorem 5.4), in practice more than a single readout can actually help the learning process of complex properties.

6.4 Experiments with PPI benchmark

Finally, we also tested AC- and ACR-GNNs on the PPI benchmark [36]. We chose PPI since it is a node classification benchmark with different graphs in the train set (as opposed to other popular benchmarks for node classification such as Core or Citeseer that have a single graph). Although the best results we obtained for both classes of GNNs on PPI were quite high (AC-GNNs: 97.5 F1, ACR-GNNs: 95.4 F1 in the test set), we did not observe an improvement of ACR-GNNs over AC-GNNs.

However, Chen et al. have recently observed that commonly used benchmarks are inadequate for testing advanced GNN variants, and ACR-GNNs might be suffering from this fact [8]. Thus, the fact that we do not observe any improvement may be an artefact of the simplicity of the benchmark. We left as future work a more thorough testing and tuning of ACR-GNNs for real data.

7. FINAL REMARKS

Our results show the theoretical advantages of mixing local and global information when classifying nodes in a graph. Recent works have also observed these advantages in practice; e.g., Deng et al. used global-context aware local descriptors to classify objects in 3D point clouds [10], You et

	α_1 Train	α_1 Test		α_2 Train	α_2 Test		α_3 Train	α_3 Test	
		same-size	bigger		same-size	bigger		same-size	bigger
AC	0.839	0.826	0.671	0.694	0.695	0.667	0.657	0.636	0.632
GIN	0.567	0.566	0.536	0.689	0.693	0.672	0.656	0.643	0.580
AC-FR-2	1.000	1.000	1.000	0.863	0.860	0.694	0.788	0.775	0.770
AC-FR-3	1.000	1.000	0.825	0.840	0.823	0.604	0.787	0.767	0.771
ACR-1	1.000	1.000	1.000	0.827	0.834	0.726	0.760	0.762	0.773
ACR-2	1.000	1.000	1.000	0.895	0.897	0.770	0.800	0.799	0.771
ACR-3	1.000	1.000	1.000	0.903	0.902	0.836	0.817	0.802	0.748

Table 2: Results on Erdös-Renyi graphs with nodes labeled according to classifiers α_i

al. construct node features by computing shortest-path distances to a set of distant anchor nodes [35], and Haonan et al. introduced the idea of a "star node" storing global information of the graph [15].

As mentioned before, our work is close in spirit to that of [34] and [23] establishing the correspondence between the WL test and GNNs. In contrast to our work, they focus on graph classification and do not consider the relationship with logical classifiers.

Regarding our results on the links between AC-GNNs and graded modal logic (Theorem 4.3), we point out that very recent work [1] establishes close relationships between GNNs and certain classes of *distributed local algorithms*. These in turn have been shown to have strong correspondences with modal logics [16]. Hence, it may be the case that variants of our Proposition 4.2 could be obtained by combining these two lines of work (but it is not clear if this combination would yield AC-GNNs that are *simple*), and we believe this is an interesting direction for future work. Moreover, we also don't know how to bridge our work with that of distributed algorithms when we add *non-local computations* to GNNs (such as the readouts that we consider).

Morris et al. [23] also studied k-GNNs, which are inspired by the k-dimensional WL test. In k-GNNs, graphs are considered as structures connecting k-tuples of nodes instead of just pairs of nodes. We plan to study how our results on logical classifiers relate to k-GNNs, in particular, with respect to the logic FOC_k that extends FOC_2 by allowing formulas with k variables, for each fixed $k > 1$. Recent work has also explored the extraction of finite state representations from recurrent neural networks as a way of explaining them [33, 18, 25]. We would like to study how our results can be applied for extracting logical formulas from GNNs as possible explanations for their computations.

We would like to remark that studying GNNs continues to be an important topic in the community, with new advances reported every year. The latest results involve the study of more complex GNN architectures, that take us beyond AC-GNNs and even k-GNNs. This extra power may come, e.g., as a result of allowing random information to be computed for each node [28], allowing for more complex aggregating functions [20], or different schemes of port assignments in a distributed setting [12].

Funding All authors but Kostylev are funded the Millennium Institute for Foundational Research on Data[3]. Barceló and Pérez are funded by Fondecyt grant 1200967.

8. REFERENCES

[1] Approximation ratios of graph neural networks for combinatorial problems.

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, 2003.

[3] F. Baader and C. Lutz. Description logic. In *Handbook of modal logic*, pages 757–819. North-Holland, 2007.

[4] P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. P. Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2019.

[5] P. Barceló Baeza. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI symposium on Principles of database systems*, pages 175–188, 2013.

[6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and

[3] https://imfd.cl/en/

R. Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[7] J.-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[8] T. Chen, S. Bian, and Y. Sun. Are powerful graph neural nets necessary? A dissection on graph classification. *CoRR*, abs/1905.04579, 2019.

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

[10] H. Deng, T. Birdal, and S. Ilic. PPFnet: Global context aware local features for robust 3d point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*, pages 195–205, 2018.

[11] M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. *CoRR*, abs/1903.02428, 2019.

[12] V. K. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020.

[13] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August, 2017*, pages 1263–1272, 2017.

[14] W. L. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, December 4–9, 2017*, pages 1024–1034, 2017.

[15] L. Haonan, S. H. Huang, T. Ye, and G. Xiuyan. Graph star net for generalized multi-task learning. *arXiv preprint arXiv:1906.12330*, 2019.

[16] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela, and J. Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Computing*, 28(1):31–53, 2015.

[17] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017*, 2017.

[18] A. Koul, S. Greydanus, and A. Fern. Learning finite state representations of recurrent policy networks. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*, 2019.

[19] C. Lutz, U. Sattler, and F. Wolter. Modal logic and the two-variable fragment. In *Proceedings of the International Workshop on Computer Science Logic, CSL 2001, Paris, France, September 10–13, 2001*, pages 247–261. Springer, 2001.

[20] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks, 2019.

[21] C. Merkwirth and T. Lengauer. Automatic generation of complementary descriptors with molecular graph networks. *J. of Chemical Information and Modeling*, 45(5):1159–1168, 2005.

[22] T. M. Mitchell et al. Machine learning, 1997.

[23] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: higher-order graph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 – February 1, 2019*, pages 4602–4609, 2019.

[24] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web ontology language profiles (second edition). W3C recommendation, W3C, 2012.

[25] C. Oliva and L. F. Lago-Fernández. On the interpretation of recurrent neural networks as finite state machines. In *Part I of the Proceedings of the 28th International Conference on Artificial Neural Networks, ICANN 2019, Munich, Germany, September 17–19, 2019*, pages 312–323. Springer, 2019.

[26] M. Otto. Graded modal logic and counting bisimulation. `https://www2.mathematik.tu-darmstadt.de/~otto/papers/cml19.pdf`, 2019.

[27] I. Robinson, J. Webber, and E. Eifrem. *Graph databases*. " O'Reilly Media, Inc.", 2013.

[28] R. Sato, M. Yamada, and H. Kashima. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

[29] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The

graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.
[30] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *Proceedings of The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018*, pages 593–607, 2018.
[31] W3C OWL Working Group. OWL 2 Web ontology language document overview (second edition). W3C recommendation, W3C, 2012.
[32] B. Y. Weisfeiler and A. A. Leman. A Reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968. Translated from Russian.
[33] G. Weiss, Y. Goldberg, and E. Yahav. Extracting automata from recurrent neural networks using queries and counterexamples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018*, pages 5244–5253, 2018.
[34] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How Powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*, 2019.
[35] J. You, R. Ying, and J. Leskovec. Position-aware graph neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, California, USA, June 9–15, 2019*, pages 7134–7143, 2019.
[36] M. Zitnik and J. Leskovec. Predicting multicellular function through multi-layer tissue networks. *CoRR*, abs/1707.04638, 2017.