Logical Expressiveness of Graph Neural Networks

Pablo Barceló
IMC, PUC & IMFD Chile

Egor V. Kostylev
University of Oxford

Mikaël Monet
IMFD Chile

Jorge Pérez
DCC, UChile & IMFD Chile

Juan Reutter
DCC, PUC & IMFD Chile

Juan-Pablo Silva
DCC, UChile

1 Introduction

Graph Neural Networks (*GNNs*) [11, 14] are a family of machine learning architectures that has recently become popular for applications dealing with structured data, such as molecule classification and knowledge graph completion [3, 6, 9, 15]. Recent work on the expressive power of GNNs has established a close connection between their ability to classify nodes in a graph and the *Weisfeiler-Lehman* (*WL*) test for checking graph isomorphism [12, 17]. Specifically, the authors of these two papers independently observe that the classifications of nodes produced by the WL test always refines the classification produced by any GNN, and moreover that there are GNNs that can reproduce the WL test. These results establish that GNNs can be as powerful as the WL test for node classification. However, this does not imply that GNNs can express any classifier that is refined by the WL test.

Our work aims to answer the following question: what are the node classifiers that can be captured with GNNs? In this paper we look at this question from a logical perspective, restricting ourselves to the properties expressible in FOC_2, the two-variable fragment of first-order logic extended with counting capabilities [4]. This choice is justified by a seminal result by Cai et al. [4] 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. Since the expressive power of FOC_2 as a declarative logical language is well-understood [4, 8], the precise relationship between FOC_2 and GNNs in terms of expressivity can shed a light on the expressive power of GNNs.

To establish the relationship, we start by considering GNNs that update the feature vector of a node by *combining* it with the *aggregation* of the vectors of its neighbors; we call these *aggregate-combine GNNs* (*AC-GNNs*). Note that recent studies on the expressive power of GNNs concentrate mostly on this architecture [12, 17]. On the negative side, we present some very simple FOC_2 node properties that cannot be captured by AC-GNN classifiers. On the positive side, we identify a natural fragment of FOC_2 whose expressiveness is subsumed by that of AC-GNNs. This fragment corresponds to *graded modal logic* [5], or, equivalently, to the *description logic* $\mathcal{ALCQ}$, which has received considerable attention in the knowledge representation community [1, 2]. Next we extend the AC-GNN architecture 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* (ACR-GNNs). These networks are a special case of the ones proposed by Battaglia et al. [3] for relational reasoning over graph representations. In this setting, we prove that each FOC_2 formula is captured by an ACR-GNN classifier. Note that, besides their own value, our two results put together indicate that readouts strictly increase the discriminative power of GNNs.

We experimentally validate our theoretical findings by showing that, on synthetic graph data conforming to a specific FOC_2 formula that is not in $\mathcal{ALCQ}$, AC-GNNs struggle to fit the training data while ACR-GNNs can generalize even to graphs of sizes not seen during training.

2 Graph Neural Networks

Next we formalize the relevant GNN architecture and introduce other related notions. We concentrate on *boolean node classifiers*—that is, networks that clasiify each graph node as `true` or `false`.

To simplify the presentation, we concentrate on undirected simple node colored graphs; however, all our results generalize to directed multigraphs with self-loops and colored edges in a straightforward way. Formally, a graph $G = (V, E, c)$ consists of a set V of *nodes*, a set E of (undirected) *edges* of the form $\{v, u\}$ for $v, u \in V$ with $v \neq u$, and a *coloring* $c : V \to \mathrm{Col}$ assigning a unique color from a finite set Col to each node. The *neighborhood* $\mathcal{N}_G(v)$ of a node $v \in V$ is the set $\{u \mid \{v, u\} \in E\}$.

The most basic architecture for GNNs, which is also considered in the recent studies on GNN expressibility [12, 17], consists of a sequence of *layers* that combine the features of every node with the multiset of feature vectors of its neighbors. Formally, an *aggregate-combine GNN* (*AC-GNN*) boolean node classifier is specified by a (positive integer) number L of *layers*, numbers $\{d_i\}_{i=0}^{L}$ (which are also positive integers) representing the *features' dimensionlities* such that d_0 is the number of colors in Col, *aggregation* functions $\{\mathrm{AGG}^{(i)}\}_{i=1}^{L}$ with each $\mathrm{AGG}^{(i)}$ mapping a multiset of rational feature vectors in $\mathbb{R}^{d_{i-1}}$ to one such vector, *combining* functions $\{\mathrm{COM}^{(i)}\}_{i=1}^{L}$ mapping each vector in $\mathbb{R}^{2d_{i-1}}$ to a vector in $\mathbb{R}^{d_i}$, and a *classification* function CLS mapping each vector in $\mathbb{R}^{d_L}$ to true or false. Given a graph $G = (V, E, c)$, each layer $i = 1, \ldots, L$ of such a classifier $\mathcal{A}$ computes feature vectors $\boldsymbol{x}_v^{(i)}$ of size d_i for every node $v \in V$ with the recursive formula

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

assuming that the initial feature vector $\boldsymbol{x}_v^{(0)}$ for each node v is the *one-hot* encoding of its color $c(v)$—that is, the vector whose k-th dimention is 1 if $c(v)$ is the color number k and 0 otherwise. Then, to solve a node-classification problem for a node $v \in V$, the vector $\boldsymbol{x}_v^{(L)}$ is used as the input to CLS to produce the output for v—that is, the (boolean) classification $\mathcal{A}(G, v)$ of v by $\mathcal{A}$ is $\mathrm{CLS}(\boldsymbol{x}_v^{(L)})$.

There are many possible aggregation and combining functions, which produce different classes of GNNs [7, 9, 12, 17]. A simple choice is to consider the sum of the feature vectors as aggregation and

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

as the combining function, where $\boldsymbol{C}^{(i)}$ and $\boldsymbol{A}^{(i)}$ are matrices of rational parameters and $\boldsymbol{b}^{(i)}$ is a *bias* vector of rationals, all with appropriate dimensions, while f is a *non-linearity* function, such as ReLU or sigmoid. We say that an AC-GNN classifier using these functions is *simple*. Furthermore, we say that an AC-GNN classifier is *homogeneous* if all $\mathrm{AGG}^{(i)}$ are the same and all $\mathrm{COM}^{(i)}$ are the same (implying that all d_i are also the same). In our positive results we always construct simple homogeneous GNNs, while our negative results hold in general (i.e., for arbitrary aggregation and combining functions, and not necessarily homogeneous).

The *Weisfeiler-Lehman* (*WL*) test is a powerful heuristic used to solve the graph isomorphism problem [16], or, for our purposes, to determine if the neighborhoods of two nodes in a given graph are structurally close or not. Due to space limitations, we refer to [4] for a formal definition of the underlying algorithm, giving only its informal description: starting from a colored graph G, the algorithm iteratively assigns, for a certain number of *rounds*, a new color to every node in the graph; the color of a node in round i depends only its own color in round $i - 1$ and on the multiset of colors of its neighbors in round $i - 1$. An important observation is that the rounds of the WL algorithm can be seen as the layers of an AC-GNN classifier whose aggregation and combining functions are all injective [12, 17]. Furthermore, as the following proposition says, an AC-GNN classification can never contradict the WL test.

Proposition 2.1 ([12, 17])**.** *Let $\mathcal{A}$ be an AC-GNN classifier with L layers, G be a graph, and v, u be nodes of G. If the WL test assigns the same color to v and u after L rounds then $\mathcal{A}(G, v) = \mathcal{A}(G, u)$.*

3 AC-GNNs and Logics

In this section we present our results on the relationship of AC-GNNs and fragments of *first order* (FO) logic. To match the inputs and outputs of these formalisms, we concentrate on unary FO formulas over graphs. As illustrated below, each such formula α takes a node v in a graph G and classifies it as true or false, which is written as $(G, v) \models \alpha$ or $(G, v) \not\models \alpha$, respectively. We then say that an AC-GNN boolean classifier $\mathcal{A}$ *captures* a unary formula α if the classifications of $\mathcal{A}$ and α are the same for every node v in every graph G—that is, $\mathcal{A}(G, v) = \mathsf{true}$ if and only if $(G, v) \models \alpha$.

Next we briefly introduce first-order logic, as well as its fragment FOC_2, assuming the node colors Col as unary predicates, and the (undirected) edge relation E as a binary predicate. Due to space

limitations, we concentrate on examples, referring to Cai et al. [4] and any textbook on logic for a formal presentation. First, for the (unary) FO formula $\alpha(v) \;:=\; R(v) \wedge \exists x\big(E(v,x) \wedge B(x)\big)$ we have $(G, v) \models \alpha$ for all nodes v with color R (i.e., red) in a graph G that have at least one neighbor with color B (i.e., blue). The unary formulas we consider have only one *free* (i.e., output) variable, which is v in the example. However, FO formulas may arbitrarily use any number of quantified variables, and restricting this number has an impact on the expressive power of the logic. As an illustration, consider the following FO formula expressing that v is a red node, and there is another node, x_1, that is not connected to v and that has at least two blue neighbors, x_2 and x_3:

$$\beta(v) \coloneqq R(v) \wedge \exists x_1\big(\neg E(v,x_1) \wedge \exists x_2 \exists x_3\big[E(x_1,x_2) \wedge E(x_1,x_3) \wedge x_2 \neq x_3 \wedge B(x_2) \wedge B(x_3)\big]\big).$$

The formula $\beta(v)$ uses four variables, but it is possible to find an equivalent one with just three: the trick is to *reuse* variable v and replace every occurrence of x_3 in $\beta(v)$ by v. However, this is as far as we can go with this trick: $\beta(v)$ does not have an equivalent formula with less than three variables.

That being said, it is possible to extend the logic so that some properties can be expressed with even less variables. Consider the *counting quantifiers* $\exists^{\geq N}$ for every positive integer N. Similar to how the quantifier $\exists$ expresses the existence of a node satisfying a property, $\exists^{\geq N}$ expresses the existence of *at least N different nodes* satisfying a property. With $\exists^{\geq N}$ we can express $\beta(v)$ using only two variables by the following formula:

$$\gamma(v) \coloneqq R(v) \wedge \exists x\big(\neg E(v,x) \wedge \exists^{\geq 2} v\big[\,E(x,v) \wedge B(v)\,\big]\big).$$

Based on this idea, the logic FOC_2 allows for formulas using all FO constructs and counting quantifiers, but restricted to only two variables. Note that every FOC_2 formula has an equivalent FO formula (i.e., FOC_2 is a *semantic fragment* of FO), because counting quantifiers can be expressed via usual quantifiers and inequalities (but at the cost of using more variables). The following result establishes a classical connection between FOC_2 and the WL test.

Proposition 3.1 (Cai et al. [4])**.** *For any graph G and nodes u, v in G, the WL test colors v and u the same after any number of rounds iff u and v are classified the same by all FOC_2 unary formulas.*

Having Propositions 2.1 and 3.1, one may be tempted to combine them and claim that every FOC_2 formula can be captured by an AC-GNN. Yet, the following result shows that this is not the case.

Proposition 3.2. *There is an FOC_2 unary formula that cannot be captured by any AC-GNN classifier.*

One such formula is $\gamma(v)$, but there are infinitely many and even simpler FOC_2 formulas that cannot be captured by AC-GNNs. Intuitively, the main problem is that each AC-GNN has only a fixed number of layers L and hence the information of local aggregations cannot travel further than at distance L of every node along edges in the graph. Moreover, the information cannot travel across different connected components, and hence Proposition 3.2 may even be shown for AC-GNNs that dispose of an arbitrary number of layers (for instance, on input graph G, one might want to run a homogeneous AC-GNN for $|G|$ layers).

The result of Proposition 3.2 opens up the following questions: (1) What kind of FOC_2 formulas can be captured by AC-GNN boolean classifiers?, and (2) How can we extend the AC-GNN architecture to capture all FOC_2 unary formulas? We provide answers to these questions in the next section.

4 ALCQ Description Logic and AC-GNN Classifiers

We start with the answer to the first question at the end of the previous section, and show that AC-GNNs capture a well-known fragment of FOC_2, namely, the one defined by graded modal logic [5], or, equivalently, the description logic $\mathcal{ALCQ}$ which is fundamental for knowledge representation: for example, the Semantic Web ontology language OWL 2 is largely based on $\mathcal{ALCQ}$ [13].

Again, due to space limitations we do not present $\mathcal{ALCQ}$ formally, concentrating on informal description and examples (but we give the formal definition in the appendix). Essentially, in the undirected node-colored graph settings, $\mathcal{ALCQ}$ restricts unary FOC_2 by forbidding several combinations of constructs, most importantly negations of binary atoms (i.e., $\neg E(x,y)$), closed sub-formulas (i.e., sub-formulas without free variables), and disconnected conjunctions (such as in $\exists y(R(y) \wedge B(v))$). Such a regular shape of formulas allows for a variable free syntax, which is in fact the standard in description logics. For instance, the FO formula $\alpha(v) \coloneqq R(v) \wedge \exists x\big(E(v,x) \wedge B(x)\big)$ considered

above is written simply as $R \sqcap \exists E.B$, and counting quantifiers can be used similarly. Note, however, that FOC_2 formula $\beta(v)$ above cannot be equivalently written in $\mathcal{ALCQ}$, because it essentially uses binary negation. This last observation is crucial for us, because the negative result of Proposition 3.2 relies on the fact that AC-GNNs are local (in the sense described above), and $\mathcal{ALCQ}$ restricts FOC_2 by imposing similar locality by forbidding negation, closed sub-formulas, and disconnected conjunctions. This intuition is formalised in the following theorem, which answers our first question even for simple and homogeneous AC-GNNs.

Theorem 4.1. *Each $\mathcal{ALCQ}$ formula can be captured by a simple homogeneous AC-GNN classifier.*

In the construction we use the piecewise linear sigmoid $\sigma(x) = \max(0, \min(1, x))$ for each dimension in the combining functions. A key idea of the AC-GNN is that the dimensions of the feature vectors represent the sub-formulas of the captured formula. Thus, if a feature in a node is 1 then the node satisfies the corresponding sub-formula, and the opposite holds after evaluating L layers, where L is the depth of the formula (which importantly does not depend on the graph). We do not know whether $\mathcal{ALCQ}$ is the *largest* fragment of FOC_2 for which Theorem 4.1 holds, and leave this question for future work.

Next, we answer our second question. To this end, we extend the AC-GNN architecture by a global feature vector for the whole graph, which is computed on each layer by aggregating the multiset of feature vectors of all the nodes and then used in the combining functions as a separate parameter. This architecture is essentially the one proposed by Battaglia et al. [3]. Formally, an *aggregate-combine-readout GNN* (*ACR-GNN*) is the same as AC-GNN except that the input vectors to $\{\text{COM}^{(i)}\}_{i=1}^{L}$ are of size $3d_{i-1}$ (rather than $2d_{i-1}$) and it is additionally specified by *readout* functions $\{\text{READ}^{(i)}\}_{i=1}^{L}$ from multisets of vectors in $\mathbb{R}^{d_{i-1}}$ to one such vector. Each layer of an ACR-GNN classifier computes feature vectors $\boldsymbol{x}_v^{(i)}$ for every node v in a graph G with the formula

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

Intuitively, every layer an ACR-GNN first computes an aggregation over all the nodes in G; then, for every node v, it computes an aggregation over the neighbors of v; and finally it combines the features of v with the aggregation over its neighbors and the aggregation over the whole graph. All the notions about AC-GNNs extend to ACR-GNNs in a straightforward way; for example, combination functions of *simple* ACR-GNN classifiers have an additional parameter vector $\boldsymbol{x}_3$ as well as a term $\boldsymbol{x}_3\boldsymbol{R}^{(i)}$ in the summation, for rational matrices $\boldsymbol{R}^{(i)}$. The following result answers our second question, and complements Proposition 3.2 and Theorem 4.1.

Theorem 4.2. *Each unary FOC_2 formula is captured by a simple homogeneous ACR-GNN classifier.*

The construction is similar to that of Theorem 4.1; however, in contrast to $\mathcal{ALCQ}$ formulas, with FOC_2 we have to deal with formulas asserting the existence of a node that is not connected to the current node in the graph. 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 in [10].

5 Preliminary Experimental Results

For the sake of the space we just briefly describe our experimental results and refer to the appendix for the details. We consider a simple FOC_2 formula $R(v) \wedge \exists x\ B(x)$, which is satisfied by every red node in a graph provided that the graph contains at least one blue node (note that this is the formula that we use in the proof of Proposition 3.2). We use synthetic data with nodes having five different initial colors, with at least 18% of nodes in the positive class. We tested with path graphs and Erdös-Renyi graphs with different connectivities. For the case of paths and sparse graphs, AC-GNNs are not able to fit the training data. For dense graphs and with enough layers AC-GNNs can fit the training and generalize. We also consider the GIN architecture [17] and it exhibits a performance similar to AC-GNNs. In contrast, in all our experiments ACR-GNNs totally fit the training data and generalize (100% train and test accuracy) even for graphs larger than those seen during training which may be an indication that they are actually learning the formula. Our experiments show that ACR-GNNs perform better than AC-GNNs specially for sparsely connected graphs and even for a very simple FOC_2 property. We leave for future work the testing of more complex FOC_2 properties and the comparison of ACR-GNNs and AC-GNNs with non-synthetic data.

References

[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[2] Franz Baader and Carsten Lutz. Description logic. In *Handbook of Modal Logic*, pages 757–819. North-Holland, 2007.

[3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[4] Jin-Yi Cai, Martin Fürer, and Neil Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[5] Maarten de Rijke. A Note on Graded Modal Logic. *Studia Logica*, 64(2):271–283, 2000.

[6] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George 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.

[7] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[8] Neil Immerman. *Descriptive complexity*. Graduate texts in computer science. Springer, 1999.

[9] Thomas N. Kipf and Max 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.

[10] Carsten Lutz, Ulrike Sattler, and Frank Wolter. Modal logic and the two-variable fragment. In *Proceedings of the International Workshop on Computer Science Logic*, pages 247–261. Springer, 2001.

[11] Christian Merkwirth and Thomas Lengauer. Automatic Generation of Complementary Descriptors with Molecular Graph Networks. *J. of Chemical Information and Modeling*, 45(5):1159–1168, 2005.

[12] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin 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*, pages 4602–4609, 2019.

[13] Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. OWL 2 Web Ontology Language Profiles (Second Edition). W3C recommendation, W3C, 2012.

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

[15] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings*, pages 593–607, 2018.

[16] Boris Yu. Weisfeiler and Andrei 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.

[17] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie 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.

A Proof of Proposition 3.2

Consider the following FOC_2 node property $\alpha(v) \coloneqq \text{red}(v) \wedge \exists x\, \text{green}(x)$. We will show by contradiction that there is no AC-GNN that captures α (no matter which aggregation, combining, and final classifier functions are considered). 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 $(G, v_0) \not\models \alpha$, we have that $\mathcal{A}$ labels v_0 with false, i.e., $\mathcal{A}(G, v_0) = \mathsf{false}$. 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.

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 u cannot travel further than at distance L from u. 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 an homogeneous AC-GNN for $f(|G|)$ layers, for a fixed function f). Assume again by way of contradiction that $\mathcal{A}$ is an AC-GNN capturing α (not necessarily with a fixed number of layers). Consider the graph G consisting of two disconnected nodes u, v, with u being red and v being green. Then, since $(G, u) \models \alpha$, we have $\mathcal{A}(G, u) = 1$. Now consider the graph G' obtained from G by changing the color of v from green to red. Observe that, since the two nodes are not connected, we will again have $\mathcal{A}(G', u) = 1$, contradicting the fact that $(G', u) \not\models \alpha$ and that $\mathcal{A}$ is supposed to capture α.

By contrast, it is easy to see that this formula can be done with only one intermediate readout, using the technique in the proof of Theorem 4.2.

B Proof of Theorem 4.1

We first define formally the logic $\mathcal{ALCQ}$ (or, equivalently, graded modal logic [5]) over simple undirected node-colored graphs.

Definition B.1 (See [2])**.** An $\mathcal{ALCQ}$ formula is constructed as follows:

1. if r is one of the base colors, then r is a formula in $\mathcal{ALCQ}$, and
2. if C and D are formulas, E is the neighbor relation in a graph, and n is a natural number, then $C \sqcap D$, $C \sqcup D$, $\neg C$ and $\exists^{\geq n} E.C$ are ALCQ formulas.

We define when a node v in a graph G *satisfies* an $\mathcal{ALCQ}$ formula α, denoted by $v \models \alpha$, recursively as follows:

- if $\alpha = r$, then $v \models \alpha$ if and only if r is the base color of v in G,
- if $\alpha = C \sqcap D$, then $v \models \alpha$ if and only if $v \models C$ and $v \models D$ (and similarly with $C \sqcup D$ and $\neg C$), and
- if $\alpha = \exists^{\geq n} E.C$, then $v \models \alpha$ if and only if the set of nodes $\{u \mid u \in \mathcal{N}_G(v) \text{ and } v \models C\}$ has at least size n.

Note that every $\mathcal{ALCQ}$ formula expresses a node property.

We can now proceed to prove Theorem 4.1. Let α be an $\mathcal{ALCQ}$ formula. We will construct an AC-GNN $\mathcal{A}_\alpha$ that is further simple and homogeneous. Let $\text{sub}(\alpha) = (\alpha_1, \alpha_2, \ldots, \alpha_K)$ be an enumeration of the sub-formulas of α. The idea of the construction of $\mathcal{A}_\alpha$ is to have (row) feature vectors in $\mathbb{R}^{1\times K}$ such that every component of those vectors represents a different formula in $\text{sub}(\alpha)$. Then $\mathcal{A}_\alpha$ will update the feature vector $\boldsymbol{x}_v$ of node v ensuring that component i of $\boldsymbol{x}_v$ gets a value 1 if and only if the formula α_i is satisfied in node v. We note that $\alpha \in \text{sub}(\alpha)$ and thus, there is one component of each feature vector in every node that gets a value 1 if and only if the node satisfies α. We will then be able to use a final classification function CLS that simply extracts that particular component. The simple homogeneous GNN $\mathcal{A}_\alpha$ uses the aggregation and combine operators defined by

$$\begin{aligned} \text{AGG}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_\ell) &= \sum_{i=1}^{\ell} \boldsymbol{x}_i \\ \text{COM}(\boldsymbol{x}, \boldsymbol{y}) &= \sigma\big(\boldsymbol{x}\boldsymbol{C} + \boldsymbol{y}\boldsymbol{A} + \boldsymbol{b}\big) \end{aligned}$$

where $\boldsymbol{A}, \boldsymbol{C} \in \mathbb{R}^{K\times K}$, and $\boldsymbol{b} \in \mathbb{R}^{1\times K}$ are defined next, and σ is the *hard-sigmoid* activation defined by $\sigma(u) = \min(\max(0, u), 1)$, applied on each component of the feature vectors. The entries of the i-th columns of $\boldsymbol{A}$, $\boldsymbol{C}$, and $\boldsymbol{b}$ depend on the sub formulas of α as follows:

Case 1: if $\alpha_i = r$ with r one of the base colors, then $C_{ii} = 1$,

Case 2: if $\alpha_i = \alpha_j \sqcap \alpha_\ell$ then $C_{ji} = C_{\ell i} = 1$ and $b_i = -1$,

Case 3: if $\alpha_i = \alpha_j \sqcup \alpha_\ell$ then $C_{ji} = C_{\ell i} = 1$,

Case 4: if $\alpha_i = \neg\alpha_j$ then $C_{ji} = -1$ and $b_i = 1$,

Case 5: if $\alpha_i = \exists^{\geq n} E.\alpha_j$ then $A_{ji} = 1$ and $b_i = -n + 1$,

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

We now prove that $\mathcal{A}_\alpha$ indeed captures α. Let G be a colored graph. For every $v \in G$ we consider the initial feature vector $\boldsymbol{x}_v^0 = (x_1, \ldots, x_K)$ such $x_i = 1$ if and only if sub-formula α_i is the initial color assigned to v, and $x_i = 0$ otherwise. $\mathcal{A}_\alpha$ will iterate the aggregation and combine operators defined above for K-rounds (K layers) to produce feature vectors $\boldsymbol{x}_v^t$ for every node $v \in G$ and $t = 1, \ldots, K$. That is

$$\boldsymbol{x}_v^t = \mathrm{COM}(\boldsymbol{x}_v^{t-1}, \mathrm{AGG}(\{\boldsymbol{x}_u^{t-1} \mid E(u,v) \in G\})) \tag{1}$$

$$= \sigma\Big(\boldsymbol{x}_v^{t-1}\boldsymbol{C} + \boldsymbol{b} + \sum_{E(u,v)\in G} \boldsymbol{x}_u^{t-1}\boldsymbol{A}\Big) \tag{2}$$

We next prove that for every $\alpha_i \in \mathrm{sub}(\alpha)$, if α_i has t sub-formulas, then for every $T \geq t$ and every $v \in G$ it holds that

$$(\boldsymbol{x}_v^T)_i = 1 \text{ if } (G,v) \models \alpha_i, \text{ and } (\boldsymbol{x}_v^T)_i = 0 \text{ otherwise} \tag{3}$$

That is, the i-th component of $\boldsymbol{x}_v^T$ has a 1 if and only if v satisfies α_i in G. In the rest of the proof we will be continuously using the value of $(\boldsymbol{x}_v^t)_i$ whose general expression is

$$(\boldsymbol{x}_v^t)_i = \sigma\Big(\sum_{j=1}^{K}(\boldsymbol{x}_v^{t-1})_j C_{ji} + b_i + \sum_{E(u,v)\in G}\sum_{j=1}^{K}(\boldsymbol{x}_u^{t-1})_j A_{ji}\Big). \tag{4}$$

We proceed to prove (3) by induction on the number of sub-formulas of every α_i. If α_i has one sub-formula, then $\alpha_i = r$ with r a base color. We next prove that $(\boldsymbol{x}_v^1)_i = 1$ if and only if v hast r as its initial color. Since $\alpha_i = r$ we know that $C_{ii} = 1$ and $C_{ji} = 0$ for every $j \neq i$ (see Case 1 above). Moreover, we know that $b_i = 0$ and $A_{ji} = 0$ for every j. Then, from Equation (4) we obtain that

$$(\boldsymbol{x}_v^1)_i \;=\; \sigma\Big(\sum_{j=1}^{K}(\boldsymbol{x}_v^0)_j C_{ji} + b_i + \sum_{E(u,v)\in G}\sum_{j=1}^{K}(\boldsymbol{x}_u^0)_j A_{ji}\Big) \;=\; \sigma\big((\boldsymbol{x}_v^0)_i\big)$$

Then, given that $(\boldsymbol{x}_v^0)_i = 1$ if the initial color of v is $\alpha_i = r$ and $(\boldsymbol{x}_v^0)_i = 0$ otherwise, we have that $(\boldsymbol{x}_v^1)_i = 1$ if $(G,v) \models \alpha_i G$ and $(\boldsymbol{x}_v^1)_i = 0$ otherwise. From this it is easy to prove that for every $T \geq 1$ $(\boldsymbol{x}_v^T)_i$ satisfies the same property. Now assume that α_i has $t > 1$ sub-formulas, and assume that for every α_j with less than t sub-formulas the property (3) holds. Let $T \geq t$. We proceed by cases.

- Assume that $\alpha_i = \alpha_j \sqcap \alpha_\ell$. Then $C_{ji} = C_{\ell i} = 1$ and $b_i = -1$. Moreover for every $k \neq j, \ell$ we have that $C_{ki} = 0$ and $A_{mi} = 0$ for every m (see Case 2 above). Then, from Equation (4) we obtain that

  $$(\boldsymbol{x}_v^T)_i \;=\; \sigma\Big((\boldsymbol{x}_v^{T-1})_j + (\boldsymbol{x}_v^{T-1})_\ell - 1\Big).$$

  Let t_1 be the number of sub-formulas of α_j. Since $t_1 < t$ and $T \geq t$ we know that $T - 1 \geq t_1$. Then by induction hypothesis we know that $(\boldsymbol{x}_v^{T-1})_j = 1$ if and only if $(G,v) \models \alpha_j$ and $(\boldsymbol{x}_v^{T-1})_j = 0$ otherwise. Similarly, $(\boldsymbol{x}_v^{T-1})_\ell = 1$ if and only if $(G,v) \models \alpha_\ell$ and $(\boldsymbol{x}_v^{T-1})_\ell = 0$ otherwise. Now, since $(\boldsymbol{x}_v^T)_i = \sigma((\boldsymbol{x}_v^{T-1})_j + (\boldsymbol{x}_v^{T-1})_\ell - 1)$ we have that $(\boldsymbol{x}_v^T)_i = 1$ if and only if $(\boldsymbol{x}_v^{T-1})_j + (\boldsymbol{x}_v^{T-1})_\ell - 1 \geq 1$ that can only happen if $(\boldsymbol{x}_v^{T-1})_j = (\boldsymbol{x}_v^{T-1})_\ell = 1$. Then $(\boldsymbol{x}_v^T)_i = 1$ if and only if $(G,v) \models \alpha_j$ and $(G,v) \models \alpha_\ell$, that is, if and only if $(G,v) \models (\alpha_j \sqcap \alpha_\ell) = \alpha_i$, and $(\boldsymbol{x}_v^T)_i = 0$ otherwise. This is exactly what we wanted to prove.
- The case $\alpha_i = \alpha_j \sqcup \alpha_\ell$ is similar to $\alpha_i = \alpha_j \sqcap \alpha_\ell$.
- Assume that $\alpha_i = \neg\alpha_j$. Then $C_{ji} = -1$ and $b_i = 1$. Moreover for every $\ell \neq j$ we have that $C_{\ell i} = 0$ and $A_{ki} = 0$ for every k (see Case 4 above). Then, from Equation (4) we obtain that

  $$(\boldsymbol{x}_v^T)_i \;=\; \sigma\Big(-(\boldsymbol{x}_v^{T-1})_j + 1\Big).$$

  By induction hypothesis we know that $(\boldsymbol{x}_v^{T-1})_j = 1$ if and only if $(G,v) \models \alpha_j$ and $(\boldsymbol{x}_v^{T-1})_j = 0$ otherwise. Since $(\boldsymbol{x}_v^T)_i = \sigma(-(\boldsymbol{x}_v^{T-1})_j + 1)$ we have that $(\boldsymbol{x}_v^T)_i = 1$ if and only if $1 - (\boldsymbol{x}_v^{T-1})_j \geq 1$ that can only happen if $(\boldsymbol{x}_v^{T-1})_j = 0$. Then $(\boldsymbol{x}_v^T)_i = 1$ if and only if $(G,v) \not\models \alpha_j$, that is, if and only if $(G,v) \models \neg\alpha_j$, i.e., if and only if $(G,v) \models \alpha_i$, and $(\boldsymbol{x}_v^T)_i = 0$ otherwise. This is exactly what we wanted to prove.

- Assume that $\alpha_i = \exists^{\geq n} E.\alpha_j$. Then $A_{ji} = 1$ and $b_i = -n + 1$. Moreover for every k we have that $C_{ki} = 0$ (see Case 5 above). Then, from Equation (4) we obtain that

$$(\boldsymbol{x}_v^T)_i \;=\; \sigma\Big(-n + 1 + \sum_{E(u,v)\in G} (\boldsymbol{x}_u^{T-1})_j \Big).$$

  By induction hypothesis we know that $(\boldsymbol{x}_u^{T-1})_j = 1$ if and only if $(G, v) \models \alpha_j$ and $(\boldsymbol{x}_u^{T-1})_j = 0$ otherwise. Then we can write $(\boldsymbol{x}_v^T)_i = \sigma(-n + 1 + m)$ where

$$m = |\{u \mid E(u, v) \text{ and } (G, u) \models \alpha_j\}|.$$

  Thus, we have that $(\boldsymbol{x}_v^T)_i = 1$ if and only if $m \geq n$, that is if and only if there exists at least n nodes connected with v that satisfies α_j, and $(\boldsymbol{x}_v^T)_i = 0$ otherwise. From that we obtain that $(\boldsymbol{x}_v^T)_i = 1$ if and only if $(G, v) \models (\exists^{\geq n} E.\alpha_j) = \alpha_i$ which is what we wanted to prove.

To complete the proof we only need to add a final classification layer after the K iterations of the aggregate and combine layers that simply classifies a node v as 1 if the component of $\boldsymbol{x}_v^K$ corresponding to α holds a 1.

C Proof of Theorem 4.2

To prove Theorem 4.2, we will use a characterization of the unary FOC_2 formulas provided by [10] that uses a specific modal logic. That logic is defined via what are called *modal parameters*. We adapt the definitions of [10] to deal with simple undirected node-colored graphs.

Definition C.1. A *modal parameter* is an expression built from the following grammar:

$$S \coloneqq \text{id} \mid E \mid S_1 \cup S_1 \mid S_1 \cap S_2 \mid \neg S$$

Given an undirected colored graph $G = (V, E', c)$ and a node u of G, the interpretation of S on u is the set $\varepsilon_S(u) \subseteq V$ defined inductively as follows:

- if $S = \text{id}$ then $\varepsilon_S(u) \coloneqq \{u\}$;
- if $S = E$ then $\varepsilon_S(u) \coloneqq \{v \mid \{u, v\} \in E'\}$;
- if $S = S_1 \cup S_2$ then $\varepsilon_S(u) \coloneqq \varepsilon_{S_1}(u) \cup \varepsilon_{S_2}(u)$;
- if $S = S_1 \cap S_2$ then $\varepsilon_S(u) \coloneqq \varepsilon_{S_1}(u) \cap \varepsilon_{S_2}(u)$;
- if $S = \neg S'$ then $\varepsilon_S(u) \coloneqq V \setminus \varepsilon_S(u)$;

Definition C.2. The modal logic $\mathcal{EMLC}$ consists of all the unary formulas that are built with the following grammar:

$$\varphi \coloneqq C|\varphi_1 \wedge \varphi_2|\varphi_1 \vee \varphi_2|\neg\varphi|\langle S\rangle^{\geq N}\varphi$$

Where C is a color, S is a modal parameter and $N \in \mathbb{N}$. The semantics of the first 4 constructs is defined as expected, and for an undirected colored graph $G = (V, E, c)$ and node $u \in V$, we have $(G, u) \models \langle S\rangle^{\geq N}\varphi$ iff there exist at least N nodes v in $\varepsilon_S(u)$ such that $(G, v) \models \varphi$.

Example C.3. On an undirected graph $G = (V, E, c)$), the $\mathcal{EMLC}$ formula $\langle\neg E\rangle^{\geq 2}(\langle E\rangle^{\geq 3}\text{green})$ holds on a node $u \in V$ if u has at least two nonadjacent nodes v (and since our graphs have no self-loops, v could be u) such that v has at least three green neighbors.

The following can be obtained from [10]:

Theorem C.4 ([10, Theorem 1])**.** *For every $\varphi \in \mathcal{EMLC}$, there exists an equivalent FOC_2 unary formula. Conversely, for every unary FOC_2 formula, there exists an equivalent $\mathcal{EMLC}$ formula.*[1]

In order to simplify the proof, we will use the following observation:

Observation C.5. Let φ be an $\mathcal{EMLC}$ formula. Then there exists an $\mathcal{EMLC}$ formula φ' equivalent to φ such that each modal parameter appearing in φ' is one of the following:

a) id; thus representing the current node;

b) E; thus representing the neighbours of the current node;

c) $\neg E \cap \neg\text{id}$; thus representing the nodes distinct from the current node and that are not neighbours of the current node;

[1] In fact, [10] shows this for FO_2 without counting quantifiers and for $\mathcal{EMLC}$ without counting, but an inspection of the proofs reveals that the result extends to counting quantifiers.

d) $\mathrm{id} \cup E$; thus representing the current node and its neighbors;

e) $\neg\mathrm{id}$; thus representing all the nodes distinct from the current node:

f) $\neg E$; thus representing the nodes that are not neighbours of the current node (note that this *includes* the current node);

g) $E \cup \neg E$; thus representing all the nodes;

h) $E \cap \neg E$; thus representing the emptyset.

Proof. Direct by case by a tedious case analysis. □

We are now ready to proceed with the proof of Theorem 4.2. The proof is similar to that of Theorem 4.1. Let φ be an $\mathcal{EMLC}$ formula equivalent to the targeted FOC_2 unary formula that is of the form givem by Observation C.5, and let $\mathrm{sub}(\varphi) = (\varphi_1, \varphi_2, \ldots, \varphi_K)$ be an enumeration of the sub-formulas of φ. We will build a simple homogeneous ACR-GNN $\mathcal{A}_\varphi$ computing feature vectors $\boldsymbol{x}_v^t$ in $\mathbb{R}^{1\times K}$ such that every component of those vectors represents a different formula in $\mathrm{sub}(\varphi)$. In addition, we will also have feature vectors $\boldsymbol{x}_G^t$ in $\mathbb{R}^{1\times K}$. The GNN $\mathcal{A}_\varphi$ will update the feature vector $\boldsymbol{x}_v$ of node v ensuring that component i of $\boldsymbol{x}_v$ gets a value 1 if and only if the formula φ_i is satisfied in node v. Similarly, $\boldsymbol{x}_G$ will be updated to make sure that every component represents the number of nodes in G that satisfy the corresponding subformula. The readouts and aggregate simply sum the input feature vectors. When φ_i is of the form described by the cases (1-4) in the proof of Theorem 4.1, we define the i-th columns of the matrices $\boldsymbol{A}, \boldsymbol{C}$ and bias $\boldsymbol{b}$ as in that proof, and the i-th column of $\boldsymbol{R}$ (the matrix that multiplies the global readout feature vector) is full of zeros. We now explain how we define their i-th columns when φ_i is of the form $\langle S\rangle^{\geq N}\varphi_j$, according to the 8 cases given by Observation C.5:

Case a: if $\varphi_i = \langle \mathrm{id}\rangle^{\geq N}\varphi_j$, then $\boldsymbol{C}_{ji} = 1$ if $N = 1$ and 0 otherwise;

Case b: if $\varphi_i = \langle E\rangle^{\geq N}\varphi_j$, then $\boldsymbol{A}_{ji} = 1$ and $\boldsymbol{b}_i = -N + 1$;

Case c: if $\varphi_i = \langle \neg E \cap \neg\mathrm{id}\rangle^{\geq N}\varphi_j$, then $\boldsymbol{R}_{ji} = 1$ and $\boldsymbol{C}_{ji} = \boldsymbol{A}_{ji} = -1$ and $\boldsymbol{b} = -N + 1$;

Case d: if $\varphi_i = \langle \mathrm{id} \cup E\rangle^{\geq N}\varphi_j$, then $\boldsymbol{C}_{ji} = 1$ and $\boldsymbol{A}_{ji} = 1$ and $\boldsymbol{b} = -N + 1$;

Case e: if $\varphi_i = \langle \neg\mathrm{id}\rangle^{\geq N}\varphi_j$, then $\boldsymbol{R}_{ji} = 1$ and $\boldsymbol{C}_{ji} - 1$ and $\boldsymbol{b} = -N + 1$;

Case f: if $\varphi_i = \langle \neg E\rangle^{\geq N}\varphi_j$, then $\boldsymbol{R}_{ji} = 1$ and $\boldsymbol{A}_{ji} = -1$ and $\boldsymbol{b} = -N + 1$;

Case g: if $\varphi_i = \langle E \cup \neg E\rangle^{\geq N}\varphi_j$, then $\boldsymbol{R}_{ji} = 1$ and $\boldsymbol{b} = -N + 1$;

Case h: if $\varphi_i = \langle E \cap \neg E\rangle^{\geq N}\varphi_j$, then nothing.

and all other values in the i-th columns of $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{R}$, and $\boldsymbol{b}$ are 0. The proof then goes on exactly as in that of Theorem 4.1.

D Details on the Experimental Setting and Results

We consider two classes of graphs: (1) **paths**: connected graphs in which every node in the graph has degree 2 except for two nodes (the extreme nodes) that have degree 1, and (2) **Erdös-Renyi graphs**: graphs generated randomly specifying the number of nodes and edges in the graph. For this last case, we consider as a extreme cases the case in which graphs contain the same number of nodes and edges and graphs in which the number of edges is twice the number of nodes.

For training and testing we constructed three sets of graphs for each case: (a) **Train set** containing 5k graphs with nodes between 50 and 100, (b) **Test set, same size**, containing 500 graphs with the same number of nodes as in the train set (between 50 and 100 nodes), and (c) **Test set, bigger size**, containing 500 graphs with nodes between 100 and 200.

All graphs contain up to 5 different colors, and every node is labeled according to the formula $R(v) \wedge \exists x B(x)$ (the node is *red* and there exists at least one *blue* node in the graph). To force the models to try to learn the formula, in every set (train and test) we consider 50% of graphs not containing any *blue* node, and 50% containing at least one *blue* node. The number of *blue* nodes in every graph is fixed to a small number (typically less than 5 nodes). Moreover, to ensure that there is a significant number of nodes satisfying the formula, we force graphs to contain at least 1/4 of its nodes colored with *red*. The colors of all the other nodes are distributed randomly. With all these restrictions, every dataset that we created had at least a 18% of nodes satisfying the property. For the case of the path graphs, and to mimic the impossibility proof in Proposition 3.2 we put the *blue* nodes in one of the "sides" of the path, and the *red* nodes in the other "side". More specifically, consider the path graph with 100 nodes $v_1, \ldots, v_{100}$ such that v_i is connected with v_{i+1} for every $i \in \{1, \ldots, 99\}$. Then, we ensure that every *blue* node appears in one of $v_1, \ldots, v_{50}$ and every *red* node appears in one of $v_{51}, \ldots, v_{100}$.

	Path			Erdös-Renyi		
	Train Acc.	Test Acc.		Train Acc.	Test Acc.	
		same-size	bigger		same-size	bigger
AC-2	0.872	0.871	0.877	0.823	0.826	0.790
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-2	0.865	0.864	0.870	0.803	0.805	0.780
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	1.000	1.000	1.000	1.000	1.000	1.000

Table 1: Results on synthetic data for nodes labeled according to the FOC_2 formula $R(v) \wedge \exists x B(x)$.

	Erdös-Renyi + 20%			Erdös-Renyi + 50%			Erdös-Renyi + 100%		
	Train Acc.	Test Acc.		Train Acc.	Test Acc.		Train Acc.	Test Acc.	
		same-size	bigger		same-size	bigger		same-size	bigger
AC-2	0.810	0.807	0.778	0.829	0.835	0.791	0.861	0.864	0.817
AC-5	0.940	0.937	0.901	0.975	0.971	0.958	0.994	0.994	0.993
AC-7	0.963	0.961	0.946	0.983	0.978	0.981	0.995	0.995	0.995
GIN-2	0.797	0.795	0.771	0.813	0.818	0.784	0.838	0.840	0.803
GIN-5	0.838	0.836	0.819	0.846	0.847	0.833	0.841	0.844	0.838
GIN-7	0.838	0.840	0.803	0.841	0.844	0.838	0.784	0.788	0.773
ACR	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

Table 2: Detailed results for Erdös-Renyi synthetic graphs with different connectivities.

We tested AC-GNNs, GINs [17], and ACR-GNNs for 2, 5, and 7 layers and with several configurations for the aggregate and combine (Table 1). We report the accuracy on the best configuration that we found for the aggregation and combination functions. Accuracy in our experiments is computed as the total number of nodes correctly classified among all nodes in all the graphs in the dataset. We run 10 epochs with the Adam optimizer and we did not use any regularization.[2] For both types of graphs, ACR-GNNs completely fit the training data and generalize even for graphs of size bigger compared to the ones seen during training (100% train and test accuracy). ACR-GNN with 2 layers already showed perfect performance (ACR in Table 1). This was what we expected given the simplicity of the property that we are checking. In contrast, AC-GNNs and GINs (shown in Table 1 as AC-k and GIN-k representing AC-GNNs and GINs with k layers) struggle to fit the data. For the case of the path graph, they were not able to completely fit the train data even if 7 layers are allowed. For the case of the random graphs, the performance with 7 layers is considerably better but still not perfectly fitting the data. We allowed AC-GNNs with 7 layers to run form more epochs but the results did not improve.

We also took a closer look at the performance for different connectivities of random graphs (Table 2). We define the set "Erdös-Renyi + k%" as a set of graphs in which the number of edges is k% larger than the number of nodes. For example, "Erdös-Renyi + 100%" contains random graphs in which the number of egdes doubles the number of nodes. We see a consistent improvement in the performance of AC-GNNs and GINs when we train and test them with more dense graphs and more layers (Table 2). This is also consistent with our theoretical observations that the main problem with AC-GNNs is that they are not able to move information of local aggregations to distances beyond the number of layers of each network. 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.

[2] Our code base can be accessed at
https://anonymous.4open.science/r/c61016f9-60da-493b-817e-766137bd921c/