Recurrent Graph Neural Networks and Their Connections to Bisimulation and Logic

Maximilian Pflueger[1], **David Tena Cucala**[1], **Egor V. Kostylev**[2]
[1]University of Oxford, United Kingdom
[2]University of Oslo, Norway
maximilian.pflueger@cs.ox.ac.uk, david.tena.cucala@cs.ox.ac.uk, egork@ifi.uio.no

Abstract

The success of Graph Neural Networks (GNNs) in practice has motivated extensive research on their theoretical properties. This includes recent results that characterise node classifiers expressible by GNNs in terms of first order logic. Most of the analysis, however, has been focused on GNNs with fixed number of message-passing iterations (i.e., layers), which cannot realise many simple classifiers such as reachability of a node with a given label. In this paper, we start to fill this gap and study the foundations of GNNs that can perform more than a fixed number of message-passing iterations. We first formalise two generalisations of the basic GNNs: recurrent GNNs (RecGNNs), which repeatedly apply message-passing iterations until the node classifications become stable, and graph-size GNNs (GSGNNs), which exploit a built-in function of the input graph size to decide the number of message-passings. We then formally prove that GNN classifiers are strictly less expressive than RecGNN ones, and RecGNN classifiers are strictly less expressive than GSGNN ones. To get this result, we identify novel semantic characterisations of the three formalisms in terms of suitable variants of bisimulation, which we believe have their own value for our understanding of GNNs. Finally, we prove syntactic logical characterisations of RecGNNs and GSGNNs analogous to the logical characterisation of plain GNNs, where we connect the two formalisms to monadic monotone fixpoint logic—a generalisation of first-order logic that supports recursion.

Introduction

Graph Neural Networks (*GNNs*) (Scarselli et al. 2008; Liu and Zhou 2020; Hamilton 2021) are a popular family of machine learning formalisms, which operate directly on graph-structured data. Systems based on GNNs have achieved remarkable success in applications to areas as diverse as biology (Fout et al. 2017), chemistry (Reiser et al. 2022), recommender systems (Ying et al. 2018), and data management (Schlichtkrull et al. 2018; Pflueger, Tena Cucala, and Kostylev 2022). Despite these successes, it is now well-understood that GNNs have limitations, and seemingly insignificant differences between GNN variants may lead to dramatic changes in performance on various tasks.

These successes and limitations have motivated an increasing interest in the theoretical properties of GNNs, which has been recently addressed by the AI community. One of the central topics being investigated is the *discriminative* (or *non-uniform expressive*) *power* of GNNs, which determines what kinds of graph patterns a GNN can distinguish. A seminal result shows that GNN variants that fall into a standard aggregate-combine scheme with fixed number of *layers*—that is, message-passing iterations—including popular in practice GCNs (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), and GINs (Xu et al. 2019), can not discriminate between two nodes if the Weisfeiler-Lehman graph-isomorphism test assigns them the same colour (Weisfeiler and Leman 1968; Morris et al. 2019; Xu et al. 2019). This result has motivated new variants of GNNs with greater discriminative power, such as GNNs with message passing between sets of nodes (Morris et al. 2019), and GNNs with unique or random initialisation of the node labels (Loukas 2020; Abboud et al. 2021; Sato, Yamada, and Kashima 2021).

A related line of research focuses on the (*uniform*) *expressive power* of GNNs, which studies the classes of functions that GNN variants can realise. It was initiated by drawing connections between several GNN variants and logic (Barceló et al. 2019). This connection is based on the observation that GNNs and logical formulas, though being very different formalisms, operate on essentially the same structures. Indeed, labelled graphs, which (trained) GNNs take as input, can be seen as relational structures, on which logical formulas are evaluated; furthermore both formalisms can realise Boolean *node classifiers*—that is, select subsets of the nodes in graphs: for a (trained) GNN, the selected subset is determined by the assignment of Boolean labels (obtained by, for example, thresholding final numeric labels), while for a (unary) formula, by logical entailment. The seminal result by Barceló et al. (2019) shows that the node classifiers realisable by both aggregate-combine GNNs and *first-order logic* (*FOL*) are precisely those that can be expressed by formulas of *graded modal logic* (de Rijke 1996) (or, equivalently, the description logic $\mathcal{ALCQ}$ (Baader 2003)). They also obtained similar results for a more expressive GNN variant, *aggregate-combine-readout GNNs*, and connected it to the logic C^2 (Cai, Fürer, and Immerman 1992). Further (uniform) expressibility results for GNN variants have followed, including their relations to the query language Datalog (Tena Cucala et al. 2022; Abiteboul, Hull, and Vianu

1995), the impact of the choice of the aggregation function (Rosenbluth, Toenshoff, and Grohe 2023), and a general characterisation of aggregate-combine GNNs in the spirit of descriptive complexity (Grohe 2023; Immerman 2012).

This analysis, however, concentrates almost ultimately on GNNs with fixed (i.e., independent on the size of the input graph) number of layers. The majority of GNNs used in practice indeed have this property, but their relaxations have also been often discussed. In fact, the original GNN paper (Scarselli et al. 2008) and its early follow up (Gallicchio and Micheli 2010) consider a model which recurrently applies a contracting message-passing layer until the updates of the feature vectors are smaller than a fixed threshold, thus making the number of message-passing iterations a function of the input graph, rather than taking it fixed. Another example of a GNN variant with more than a fixed number of layers is the recent system RunCSP for approximate solutions for constraint satisfaction problems (Toenshoff et al. 2021), which limits the number of iterations by a function of the number of nodes in the input graph. It is straightforward to see that these GNN variants can realise node classifiers not realisable by GNNs with a fixed number of layers; for example, they can express *reachability*—the function that assigns *true* to all nodes reachable from a node with a specific label via an arbitrary number of edges.

To the best of our knowledge, there is no systematic study of GNNs that apply more than a fixed number of message-passing iterations, and this paper is a first step towards closing this gap. Our first contribution is a formalisation of two GNN variants, both of which recurrently perform message-passing until a stopping condition is satisfied. First, we consider *recurrent GNNs* (*RecGNNs*), which iterate message-passing until the classification labels become stable; for those nodes where stability cannot be achieved, a fixed label is returned. Second, we consider *graph-size GNNs* (*GS-GNNs*), which use a function of the input graph size to decide the number of message-passing iterations.

Our second contribution is the (uniform) expressivity relations between the classes of binary node classifiers realised by plain (i.e., with a fixed number of layers) GNNs, RecGNNs, and GSGNNs. We show that plain GNN classifiers are strictly less expressive than RecGNNs—that is, RecGNNs can realise all plain GNN classifiers, but not the other way round—and that RecGNNs are strictly less expressive than GSGNNs. To obtain these results, we establish a novel *semantic* characterisation of the node classifiers realisable by GNNs, RecGNNs, and GSGNNs in terms of suitable variants of *bisimulation*—a well-known concept in various branches of computer science and other fields (Baader et al. 2017; Libkin 2004; Milner 1989). In particular, we show that these GNN variants can realise precisely all node classifiers that are *bisimulation-invariant*, for the respective variant of bisimulation in each case. We believe that these characterisations have their own value for our understanding of GNNs and hence are our third contribution.

Our fourth and most technically difficult contribution are *syntactic* characterisations of the expressivity of RecGNNs and GSGNNs in terms of logics, in the same spirit as the characterisation of plain GNNs by Barceló et al. (2019). Note, however, that it is not very meaningful to do this for FOL as the base logic, as it is done for plain GNNs, because FOL cannot express any kind of recursion. Thus, we do this for a fragment of *monadic monotone fixpoint logic* (Libkin 2004), a generalisation of FOL with a least fixpoint operator, which can express recursion. Then, the characterising logics are appropriate generalisations of the graded modal logic. To get these characterisations, we prove van Benthem-Rosen type theorems characterising the fragments of the base logic that are invariant under the relevant variants of bisimulation, thus continuing a long line of such fundamental characterisations (van Benthem 1976; Rosen 1997; Janin and Walukiewicz 1996; Otto 2004, 2019).

GNNs and Logics

The purpose of this section is twofold. First, we outline the main result of the seminal paper (Barceló et al. 2019), which gives a logical characterisation of node classifiers realisable by GNNs on relational structures and which motivates our paper. Second, we use this section to introduce the basic concepts and notation used throughout the paper. We start with a formalisation of GNNs, then proceed to relevant logics, and conclude with the connections between them.

Graph Neural Networks. It is common that Graph Neural Networks (GNNs) are applied to undirected graphs with edges of one type. We consider a more general setting with directed edges of several types (or *colours*), which is also often considered in the literature (Schlichtkrull et al. 2018; Pflueger, Tena Cucala, and Kostylev 2022). As usual when comparing GNNs and logics, we concentrate on Boolean GNN classifiers.

For Col a finite set of *colours* and $\delta \in \mathbb{N}$, a (Col, δ)-*graph* is a triple $(\mathcal{V}, \mathcal{E}, \lambda)$ where $\mathcal{V}$ is a non-empty set of *nodes*, $\mathcal{E}$ is a set of directed *edges* of the form (v, c, u) with $c \in \mathsf{Col}$ and $v, u \in \mathcal{V}$, and λ is a *labelling* function that assigns a *feature* vector $\lambda(v) \in \mathbb{R}^\delta$ to each $v \in \mathcal{V}$. A (Col, δ)-graph is *Boolean* if $\lambda(v) \in \{0,1\}^\delta$ for each $v \in \mathcal{V}$. Given a (Col, δ)-graph $G = (\mathcal{V}, \mathcal{E}, \lambda)$, we sometimes write the vector $\lambda(v)$ for a node v as $\mathbf{v}$ and its i-th element as $(\mathbf{v})_i$. Furthermore, for each $v \in \mathcal{V}$ and $c \in \mathsf{Col}$, we define the *c-neighbourhood* $N^c_G(v)$ of v in G as the set $\{u \mid (v, c, u) \in \mathcal{E}\}$.

We next define a general form of (Boolean) (Col, δ)-GNN classifiers, a family of neural networks that partition nodes in (Col, δ)-graphs into two classes by assigning to each node the label 0 or 1. These networks work by repeatedly applying a (usually, trainable) (Col, δ)-*layer*.

Definition 1. *For colours* Col *and* $\delta \in \mathbb{N}$*, a* (Col, δ)-layer *is a pair* $(\mathsf{Aggr}, \mathsf{Comb})$ *of functions such that*

- Aggr *maps a multiset of* messages*—that is, pairs* $(c, \mathbf{u})$ *with* $c \in \mathsf{Col}$ *and* $\mathbf{u} \in \mathbb{R}^\delta$*—to a vector in* $\mathbb{R}^\delta$*;*
- Comb *maps two vectors in* $\mathbb{R}^\delta$ *to a vector in* $\mathbb{R}^\delta$*.*

For a (Col, δ)*-graph* $G = (\mathcal{V}, \mathcal{E}, \lambda)$*, a* (Col, δ)*-layer induces the infinite sequence* $\lambda^0, \lambda^1, \lambda^2, \ldots$ *of node labellings where* $\lambda^0 = \lambda$ *and, for each* $\ell \in \mathbb{N}$ *and* $v \in \mathcal{V}$*,*

$$\mathbf{v}^{\ell+1} = \mathsf{Comb}\Big(\mathbf{v}^\ell, \mathsf{Aggr}\big(\{\{(c, \mathbf{u}^\ell) \mid u \in N^c_G(v), c \in \mathsf{Col}\}\}\big)\Big).$$

For a number $L \in \mathbb{N}$*, an* L-layer (Col, δ) multi-colour *(or* relational*)* graph neural network classifier *(*(Col, δ)*-GNN)* $\Re$

is a pair $(\mathsf{Layer}, \mathsf{Cls})$ *where* Layer *is a* (Col, δ)*-layer and* $\mathsf{Cls} : \mathbb{R}^\delta \to \{0,1\}$ *is a* classifying *function. The result* $\Re(G)$ *of applying* $\Re$ *to a* (Col, δ)*-graph* G *is the Boolean* $(\mathsf{Col}, 1)$*-graph with the same nodes and edges as* G*, but where each node* v *is labelled by* $\mathsf{Cls}(\lambda^L(v))$*, with* λ^L *from the sequence induced by* Layer *on* G*.*

Abusing our notation, we will use $\Re(G)$ to denote also the set of all nodes labelled by 1 in $\Re(G)$.

Our definition of GNNs is restricted to so-called *uniform* GNNs, which repeatedly apply the same layer to the input graph; these stand in contrast to *non-uniform* GNNs, which may apply several distinct layers in sequence. Our definition is, however, without loss of generality, since uniform GNNs are known to be able to simulate arbitrary non-uniform GNNs (Barceló et al. 2019) (they showed this for undirected and uncoloured graphs, but their proof applies to our setting with minor modifications). Specific layer and classification functions (possibly parametrised) define the architecture of a concrete GNN variant. A great diversity of such variants are considered in the literature, and they usually involve trainable parameters as well as fixed linear and non-linear transformations. However, concrete instantiations of the general scheme and training details are not essential for us, and we abstract away from them in our presentation.

Logics. A *(graph) relational signature* is a finite set of *unary* and *binary predicates*. For the rest of this paper, we fix an arbitrary relational signature σ, as well as let σ_1 and σ_2 denote the sets of unary and binary predicates in σ, respectively. A *σ-structure* is a triple $\mathfrak{A} = (A, \{C^\mathfrak{A}\}_{C\in\sigma_1}, \{R^\mathfrak{A}\}_{R\in\sigma_2})$, where A is a finite non-empty set of *elements*, called the *universe* of $\mathfrak{A}$, each $C^\mathfrak{A}$ is a unary relation over A, and each $R^\mathfrak{A}$ is a binary relation over A. In what follows, we will write σ-structures as $\mathfrak{A}, \mathfrak{B}, \ldots$ and their universes as $A, B, \ldots$. We also write $N^R_\mathfrak{A}(a)$ and $N^{\bar{R}}_\mathfrak{A}(a)$ to denote the sets $\{a' \mid (a,a') \in R^\mathfrak{A}\}$ and $\{a' \mid (a',a) \in R^\mathfrak{A}\}$, respectively.

We assume standard *first-order logic* (*FOL*) with equality but without function symbols (and therefore, without constants) over σ, where the syntax of formulas is defined by the following standard grammar:

$$\phi ::= C(x) \mid R(x,y) \mid x = y \mid (\phi \wedge \phi) \mid \neg\phi \mid \exists x.\, \phi; \quad (1)$$

here C and R range over σ_1 and σ_2, respectively, while x and y are variables. As usual, we may use abbreviations such as $\vee$, $\forall$, $\neq$, and $\exists^{\geq n}$. We may write a FOL formula φ as $\varphi(\bar{x})$ to emphasise that the free (i.e., non-quantified) variables of φ are $\bar{x}$. We omit the standard semantics for brevity, referring instead to literature (Rautenberg 2009). Given a FOL formula $\varphi(\bar{x})$, a σ-structure $\mathfrak{A}$, and a tuple $\bar{a}$ of elements in A of the same length as $\bar{x}$, we let $\mathfrak{A} \models \varphi(\bar{a})$ denote that $\mathfrak{A}$ and the assignment of $\bar{a}$ to $\bar{x}$ satisfy $\varphi(\bar{x})$.

We next introduce a syntactic fragment FOL(GML$^-$) of FOL, which is the embedding of *graded two-way (multi-)modal logic* (or, equivalently, the *description logic* $\mathcal{ALCIQ}$) into FOL (Blackburn, van Benthem, and Wolter 2006). Formulas in FOL(GML$^-$), which are all unary, are defined by the following grammar:

$$\phi(x) ::= C(x) \mid (\phi(x) \wedge \phi(x)) \mid \neg\phi(x) \mid \exists^{\geq n} y.\, (R(x,y) \wedge \phi(y)) \mid \exists^{\geq n} y.\, (R(y,x) \wedge \phi(y)),$$

where C and R are as above, while x and y are different variables. The semantics of FOL(GML$^-$) is inherited from FOL. This well-known and widely-used formalism has significant expressive power and yet decidable reasoning (Blackburn, van Benthem, and Wolter 2006).

Correspondence between GNNs and Logics. As mentioned in the introduction, the key observation of (Barceló et al. 2019) is that GNNs and logical formulas, while looking very different, have a lot in common: they apply to very similar objects, (Col, δ)-graphs and σ-structures, and can select subsets of graph nodes and universe elements, respectively. We next formalise this correspondence.

Definition 2. *Let* $\sigma_1 = \{U_1, \ldots, U_m\}$ *and let* $\mathsf{Col}_\sigma = \sigma_2 \cup \{\bar{R} \mid R \in \sigma_2\}$*, where each* $\bar{R}$ *is a fresh binary predicate unique for* R*. The* encoding $G_\mathfrak{A}$ *of a* σ*-structure* $\mathfrak{A}$ *is the Boolean* $(\mathsf{Col}_\sigma, |\sigma_1|+1)$*-graph* $(\mathcal{V}, \mathcal{E}, \lambda)$ *where* $\mathcal{V} = A$*,* $\mathcal{E}$ *includes* (a,R,b) *and* $(b,\bar{R},a)$ *for every* $R \in \sigma_2$ *and* $(a,b) \in R^\mathfrak{A}$*, and* λ *labels each* $a \in \mathcal{V}$ *with* $\mathbf{a} \in \mathbb{R}^{|\sigma_1|+1}$ *such that* $(\mathbf{a})_i = 1$ *for each* $a \in U_i^\mathfrak{A}$ *or* $i = |\sigma_1|+1$*, and* $(\mathbf{a})_i = 0$ *otherwise.*

Note that the encoding is defined so that the last element of each label does not correspond to any unary predicate and is always set to 1; this ensures that encodings always have non-zero feature vectors of dimension at least one even when σ contains no unary predicates. Note also that the encoding of structures as Boolean graphs is injective, and hence we may switch between $\mathfrak{A}$ and $G_\mathfrak{A}$ silently. Since σ is fixed and we are interested in $(\mathsf{Col}_\sigma, |\sigma_1|+1)$-GNNs, we will often refer to them as just GNNs.

The final ingredient for formalising correspondence between GNNs and logics is the notion of a node classifier.

Definition 3. *A* (node) classifier *is a function that maps a* σ*-structure to a subset of its universe.*

By definition, each unary FOL formula $\phi(x)$ *realises* the node classifier that maps each σ-structure $\mathfrak{A}$ to the set of all $a \in A$ with $\mathfrak{A} \models \phi(a)$. Similarly, each $(\mathsf{Col}_\sigma, |\sigma_1|+1)$-GNN $\Re$ *realises* the node classifier mapping each $\mathfrak{A}$ to $\Re(G_\mathfrak{A})$.

We conclude this section with the main theorem of Barceló et al. (2019), which establishes a connection between the classifiers realisable by these two formalisms (again, they showed this for undirected uncoloured graphs–that is, for $|\sigma_2| = 1$ and with the only binary relation being symmetric—but their proof applies to our setting with minor modifications).

Theorem 1 (implicitly in (Barceló et al. 2019)). *A node classifier is realised by both a GNN and a unary FOL formula if and only if it is realised by a FOL (GML$^-$) formula.*

Recurrent and Graph-Size GNNs

It is well known (and confirmed by Theorem 1) that there are many classifiers not realisable by (plain) GNNs. Thus, the community often identifies such limitations and suggests

appropriate GNN extensions. One notable classifier not realisable by GNNs (already mentioned in the introduction) is Reachability, which maps every σ-structure $\mathfrak{A}$ to the set of elements in A that are reachable from elements in $C^{\mathfrak{A}}$ through edges of any colour, where $C \in \sigma_1$ is a dedicated unary predicate. The missing component is *recursion*, which allows to iterate a computation step (e.g., a layer) beyond a fixed number of times. There have been several proposals to realise Reachability and similar functions by means of various types of GNNs with recursion (Scarselli et al. 2008; Gallicchio and Micheli 2010; Toenshoff et al. 2021); however, as far as we are aware, no systematic study of such generalisations exists. We start filling this gap and consider various aspects of such GNNs, including possible definitions and expressivity relations. More specifically, we introduce two alternative definitions of GNNs with more than a fixed number of iterations: one that iterates a GNN layer until a fixpoint and one where the number of iterations is determined by a function of the input graph size. We also make some initial observations about the expressivity relations between these GNN variants. We begin with the first definition.

Definition 4. *For a finite set* Col *of colours and* $\delta \in \mathbb{N}$*, a* (Col, δ)-recurrent graph neural network *(RecGNN)* $\Re$ *is a pair* $(\mathsf{Layer}, \mathsf{Cls})$ *where the components are as in a* (Col, δ)*-GNN. The result* $\Re(G)$ *of applying* $\Re$ *to a* (Col, δ)*-graph* G *is the Boolean* $(\mathsf{Col}, 1)$*-graph with the same nodes and edges as* G*, but where each node* v *is labelled by*

- $\mathsf{Cls}(\lambda^{\ell_v}(v))$*, if there exists a number* $\ell_v \in \mathbb{N}$ *such that* $\mathsf{Cls}(\lambda^{\ell}(v)) = \mathsf{Cls}(\lambda^{\ell_v}(v))$ *for every* $\ell \geq \ell_v$*, where* λ^{ℓ} *is from the sequence induced by* Layer *on* G*;*
- 0*, otherwise.*

Our definition of RecGNNs captures a natural class of GNN models that may iterate beyond any fixed number of iterations and only stop when the feature vectors (or, more generally, the classifications) stabilise. In particular, RecGNNs capture the original GNNs with recursion by Scarselli et al. (2008) and their follow-up GraphESN (Gallicchio and Micheli 2010), assuming that they both iterate a layer (trainable or hand-engineered) that implements a *contracting* mapping until a predefined precision is reached, and hence the feature vector stabilises.

Similarly to GNNs, each RecGNN *realises* a node classifier that maps each $\mathfrak{A}$ to $\Re(G_{\mathfrak{A}})$. RecGNNs, however, can realise many more classifiers than GNNs, including Reachability.

Example 1. Reachability is realised by the RecGNN that, in each iteration, sets the feature's element corresponding to C to 1 if and only if it is either already 1 or has a neighbour with such element. So, when the classification stabilises, only nodes reachable from an instance of C will be assigned 1.

The following is another, less straightforward example.

Example 2. Let BetweenCycles be the node classifier such that $a \in \mathsf{BetweenCycles}(\mathfrak{A})$ for a σ-structure $\mathfrak{A}$ if there are elements $a_1^l, \ldots, a_n^l$ and $a_1^r, \ldots, a_m^r$ in A such that

$$\{(a_1^l, a_2^l), \ldots, (a_n^l, a), (a, a_1^r), \ldots, (a_{m-1}^r, a_m^r)\} \subseteq \bigcup_{R \in \sigma_2} R^{\mathfrak{A}},$$

and either $a_1^l = a_m^r$, or $a_1^l = a_i^l$ for some $i > 1$ and $a_m^r = a_j^r$ for some $j < m$. By reasons similar to the ones for Reachability, plain GNNs cannot realise BetweenCycles. In contrast, one can construct a simple RecGNN that identifies nodes between cycles by pruning nodes of all tree-like substructures (e.g., by iteratively assigning the all-zero feature vector to each node with either no ingoing or no outgoing edge leading to an element with a non-zero element in the feature).

An obvious limitation of RecGNNs is that they can only classify an element as 1 (i.e., true) if the classification stabilises for the respective node. There is no guarantee, however, that the classification will stabilise for any node, since our definition allows arbitrary Aggr and Comb functions.

We address this shortcoming in our second generalisation of GNNs, *graph-size graph neural networks*. Similarly to RecGNNs, they apply a single GNN layer more than a fixed number of times; however, they do not require the classification to stabilise but rather specify the number of iterations as a function of the number of nodes in the input graph.

Definition 5. *For colours* Col *and* $\delta \in \mathbb{N}$*, a* (Col, δ)-graph-size graph neural network *(GSGNN)* $\Re$ *is a triple* $(\mathsf{Layer}, \mathsf{Cls}, \mathsf{Iter})$ *where* Layer *and* Cls *are as in a* (Col, δ)*-GNN, while* $\mathsf{Iter} : \mathbb{N} \to \mathbb{N}$ *is a* stopping *function. The* result $\Re(G)$ *of applying* $\Re$ *to a* (Col, δ)*-graph* G *is the Boolean* (Col, δ)*-graph with the same nodes and edges as* G*, but where each node* v *is labelled by* $\mathsf{Cls}(\lambda^{\mathsf{Iter}(|\mathcal{V}|)}(v))$*, for* $\lambda^{\mathsf{Iter}(|\mathcal{V}|)}$ *from the sequence induced by* Layer *on* G *and* $|\mathcal{V}|$ *the number of nodes in* G*.*

Note that GSGNNs capture existing GNN models with recursion such as RunCSP (Toenshoff et al. 2021), which combines a recursive LSTM-based layer, a simple linear stopping function, and a FNN-based classification function. GSGNNs *realise* node classifiers in the same way as GNNs and RecGNNs. Moreover, it follows from the definition that GSGNNs generalise GNNs: each L-layer GNN can be seen as a GSGNN with Iter mapping each input to the constant L. So, every classifier realised by a GNN can also be realised by a GSGNN. The converse result does not hold; for example, Reachability can be realised by a GSGNN with identity as Iter that operates as the RecGNN in Example 1; such Iter is enough since the shortest path between any two nodes is never longer than the total number of graph nodes. Moreover, BetweenCycles can be realised by a GSGNN in an analogous way.

The following example shows that there are node classifiers realisable by GSGNNs but not RecGNNs.

Example 3. For each $n \in \mathbb{N}$, let $\mathsf{GraphSize}_n$ be the classifier mapping each σ-structure $\mathfrak{A}$ to (the whole of) A if $|A| \leq n$ and to $\emptyset$ otherwise. There is neither a GNN nor a RecGNN realising $\mathsf{GraphSize}_n$, because none of them can distinguish nodes in an n-cycle from nodes in an $(n+1)$-cycle. We can, however, define a (Col, δ)-GSGNN $(\mathsf{Layer}, \mathsf{Cls}, \mathsf{Iter})$ realising $\mathsf{GraphSize}_n$: we take Layer counting the number of iterations, identity as Iter, and Cls giving 1 if the count does not exceed n.

The following proposition summarises our initial observations about the expressivity of the three GNN variants.

Proposition 1. *The following hold:*

- *RecGNNs realise some classifiers that GNNs do not;*
- *GSGNNs realise some classifiers that neither GNNs nor RecGNNs realise;*
- *GSGNNs realise all classifiers that GNNs realise.*

Thus, to obtain the full expressivity picture for these GNN variants, we need to understand whether RecGNNs realise all GNN classifiers, and whether GSGNNs realise all RecGNN classifiers. These less straightforward relationships will be obtained in the next section.

GNNs, RecGNNs & GSGNNs, and Bisimulations

In this section, we provide novel semantic characterisations of the classes of node classifiers realised by (plain) GNNs, RecGNNs, and GSGNNs in terms of appropriate variants of *bisimulation*—a well-known concept widely used in various fields of computer science, including logic and automata, game theory, and geometric topology (Baader et al. 2017; Libkin 2004; Milner 1989). In particular, the three variants of GNN classifiers are characterised precisely by invariance under (graded two-way) ℓ-*bisimulation*, *bisimulation*, and *gs-bisimulation* over σ-structures, respectively. Note, the presented characterisation of GNNs in terms of ℓ-*bisimulation* can be viewed as a uniform formulation of the results by Morris et al. (2019) and Xu et al. (2019) that connect GNNs to the Weisfeiler-Lehman graph-isomorphism test when assuming the fixed signature σ contains one binary relation and considering only structures where this relation is symmetric. As corollaries of these results, we will get the two missing expressivity relationships.

We first introduce the notion of ℓ-bisimulation (Otto 2019), which, as we will see, corresponds to GNNs.

Definition 6. *A relation $\rho \subseteq A \times B$ is a (*graded two-way*)* 0-bisimulation *between σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ whenever, for every $C \in \sigma_1$ and $(a,b) \in \rho$, we have $a \in C^{\mathfrak{A}}$ if and only if $b \in C^{\mathfrak{B}}$. For each $\ell \geq 1$, ρ is an* ℓ-bisimulation *between $\mathfrak{A}$ and $\mathfrak{B}$ if there is an $(\ell-1)$-bisimulation ρ' between $\mathfrak{A}$ and $\mathfrak{B}$ such that, for every $(a,b) \in \rho$, we have that $(a,b) \in \rho'$ and, for each $R \in \sigma_2$, there exist bijections*

$$\begin{aligned} h_R &: N^R_{\mathfrak{A}}(a) \to N^R_{\mathfrak{B}}(b), \\ h_{\bar{R}} &: N^{\bar{R}}_{\mathfrak{A}}(a) \to N^{\bar{R}}_{\mathfrak{B}}(b) \end{aligned} \tag{2}$$

satisfying $(a', h_R(a')) \in \rho'$ for each a' in the domain of h_R and $(a', h_{\bar{R}}(a')) \in \rho'$ for each a' in the domain of $h_{\bar{R}}$.

Note that the union of two ℓ-bisimulations between every two σ-structures is also an ℓ-bisimulation between them. So, there exists a unique maximal (possibly empty) ℓ-bisimulation between every two σ-structures. We next exploit this notion to define a semantic property of classifiers.

Definition 7. *Given a number $\ell \in \mathbb{N}$, a node classifier f is* ℓ-bisimulation-invariant *whenever, for every two σ-structures $\mathfrak{A}$ and $\mathfrak{B}$, and every ℓ-bisimulation ρ between them, $a \in f(\mathfrak{A})$ if and only if $b \in f(\mathfrak{B})$ for every $(a,b) \in \rho$*

Observe that we could equivalently formulate the definition in terms of the maximal ℓ-bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$, instead of all ℓ-bisimulations between them.

The following lemma will be useful for establishing all three of our semantic characterisations.

Lemma 1. *Let* Layer *be a $(\mathsf{Col}_\sigma, |\sigma_1|+1)$-layer. Then, for every ℓ-bisimulation ρ between σ-structures $\mathfrak{A}$ and $\mathfrak{B}$, and every $(a,b) \in A \times B$, $(a,b) \in \rho$ implies $\lambda^\ell_{\mathfrak{A}}(a) = \lambda^\ell_{\mathfrak{B}}(b)$, where $\lambda^\ell_{\mathfrak{A}}$ and $\lambda^\ell_{\mathfrak{B}}$ are from the sequences induced by* Layer *on $G_{\mathfrak{A}}$ and $G_{\mathfrak{B}}$, respectively. If both components of* Layer *are injective and ρ is maximal, then the reverse implication also holds: $\lambda^\ell_{\mathfrak{A}}(a) = \lambda^\ell_{\mathfrak{B}}(b)$ implies $(a,b) \in \rho$.*

The proof of the lemma, which is by induction on ℓ, is based on the observation that an injective aggregation-combination layer and the inductive step in the definition of maximal bisimulation can essentially mimic each other. Indeed, two nodes receive the same labelling in iteration ℓ of a layer whenever they have the same labelling in the previous iteration and they have the same numbers of (colour-respecting) neighbours with same labels in that iteration; at the same time, two elements are ℓ-bisimilar whenever they are $(\ell-1)$-bisimilar and have the same numbers of $(\ell-1)$-bisimilar neighbours for each colour.

In the following, we will use Lemma 1 to prove our semantic characterisation theorems, and we start with the connection between plain GNNs and ℓ-bisimulation.

Theorem 2. *A node classifier is realised by an L-layer GNN if and only if it is L-bisimulation-invariant.*

We can prove both directions of this theorem using Lemma 1. In particular, for the forward direction we observe that every given L-layer GNN must assign the same feature after L layers to all L-bisimilar elements for every L-bisimulation; hence, the GNN classifies these elements the same, whatever the classification function is. For the backward direction, we construct, given an L-bisimulation-invariant classifier f, a GNN with an injective layer and the classification function based on the classification of f; such a function is well-defined because every two nodes with the same label after L layers must be in the maximal L-bisimulation and hence must have the same f classification.

Having the correspondence between plain GNNs and ℓ-bisimulation, we move towards a similar characterisation of RecGNNs. Using the intuition that RecGNNs are essentially GNNs with an infinite number of layers, we aim for an ℓ-bisimulation with infinite ℓ, which turns out to be the classic (graded two-way) bisimulation (Otto 2004, 2019).

Definition 8. *A relation $\rho \subseteq A \times B$ is a (*graded two-way*)* bisimulation *between σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ whenever*

- *for each $C \in \sigma_1$ and $(a,b) \in \rho$, we have $a \in C^{\mathfrak{A}}$ if and only if $b \in C^{\mathfrak{B}}$; and*
- *for each $R \in \sigma_2$ and $(a,b) \in \rho$, there are bijections* (2) *satisfying $(a', h_R(a')) \in \rho$ for each a' in the domain of h_R and $(a', h_{\bar{R}}(a')) \in \rho$ for each a' in the domain of $h_{\bar{R}}$.*

Similarly to ℓ-bisimulations, there is a unique maximal bisimulation between every two σ-structures. We also note that every bisimulation is an ℓ-bisimulation for every $\ell \in \mathbb{N}$. The notion of invariance transfers to bisimulations.

Definition 9. *A node classifier f is* bisimulation-invariant *whenever, for every two σ-structures $\mathfrak{A}$ and $\mathfrak{B}$, and ev-*

ery bisimulation ρ between them, $a \in f(\mathfrak{A})$ if and only if $b \in f(\mathfrak{B})$ for every $(a, b) \in \rho$.

We are ready to formulate our semantic characterisation of RecGNNs in terms of bisimulation.

Theorem 3. *A node classifier is realised by a RecGNN if and only if it is bisimulation-invariant.*

To prove this theorem, we again exploit Lemma 1. The forward direction relies on the fact that each bisimulation is an ℓ-bisimulation for every ℓ and the same argument as in Theorem 2. For the backward direction, as in Theorem 2, we construct a RecGNN with an injective layer, which, additionally, ensures that the same label cannot occur in two distinct iterations; then, the classification function is defined as before, and we can show, by exploiting the maximal bisimulation, that this is again a well-defined function.

Since each bisimulation is an ℓ-bisimulation for all $\ell \in \mathbb{N}$, we have that ℓ-bisimulation-invariance implies bisimulation-invariance, for all $\ell \in \mathbb{N}$. Thus, Theorem 3 implies our first missing expressivity result.

Corollary 1. *RecGNNs realise all classifiers that GNNs do.*

Finally, we obtain a similar semantic characterisation for GSGNNs. To arrive at the appropriate notion of bisimulation, we observe that GSGNNs are essentially RecGNNs that are aware of the size of the input graph; thus, so should be their bisimulation counterpart.

Definition 10. *A relation $\rho \subseteq A \times B$ is a (*graded two-way*)* gs-bisimulation *between σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ whenever either $|A| = |B|$ and ρ is a bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$, or ρ is empty.*

This version of bisimulation induces the following notion of invariance for node classifiers.

Definition 11. *A node classifier f is* gs-bisimulation-invariant *whenever, for every two σ-structures $\mathfrak{A}$ and $\mathfrak{B}$, and every gs-bisimulation ρ between them, $a \in f(\mathfrak{A})$ if and only if $b \in f(\mathfrak{B})$ for every $(a, b) \in \rho$.*

This notion allows us to formulate our semantic characterisation of GSGNNs.

Theorem 4. *A node classifier is realised by a GSGNN if and only if it is gs-bisimulation-invariant.*

The proof idea is similar to the one for Theorems 2 and 3.

Finally, we observe that every gs-bisimulation is a bisimulation, and so bisimulation-invariance implies gs-bisimulation-invariance. Together with Theorem 4, this leads to our second missing expressivity result.

Corollary 2. *GSGNNs realise all classifiers that RecGNNs realise.*

RecGNNs & GSGNNs, and Logic

In the previous section, we have given precise semantic characterisations of the classifiers realised by the three GNN variants in terms of appropriate variants of bisimulation. In this section, we aim to complement these results by showing how these semantic characterisations are utilised to obtain syntactic characterisations of RecGNNs and GSGNNs in terms of logics, in the same spirit as the characterisation for plain GNNs in Theorem 1 by Barceló et al. (2019).

To this end, we will first define a base logic $\mathcal{L}$—that is, a set of logical formulas—and then prove theorems that identify the sub-logic of $\mathcal{L}$ realising all bisimulation-invariant or gs-bisimulation-invariant node classifiers over finite structures—that is, the corresponding versions of the van Benthem-Rosen theorem, which originally covers FOL as the base logic and non-graded one-way bisimulation (Rosen 1997; Otto 2019)). Then, Theorems 3 and 4, respectively, immediately imply syntactic characterisations for node classifiers expressible by RecGNNs and GSGNNs.

Corollary 3. *Let $\mathcal{L}$ be a logic and $\mathcal{S}$ be the sublogic of $\mathcal{L}$ that realises all bisimulation-invariant or gs-bisimulation-invariant classifiers over finite structures. Then, a node classifier is realised by both a RecGNN or a GSGNN, respectively, and a formula in $\mathcal{L}$ if and only if it is realised by a formula in $\mathcal{S}$.*

Our first step, therefore, is to specify an appropriate base logic. In the case of plain GNNs, FOL is a natural choice because it can express many interesting classifiers that GNNs can express. Indeed, the proof of Theorem 1 is based on a recent van Benthem-Rosen type theorem for (graded two-way) bisimulation, which states that a FOL formula is bisimulation-invariant if and only if it is equivalent, over finite structures, to a FOL(GML^-) formula (Otto 2019) (see also relevant papers (Rosen 1997; Otto 2004)). FOL, however, is not expressive enough for interesting results about RecGNNs and GSGNNs, as it lacks recursion and, hence, cannot realise many classifiers that separate RecGNNs and GSGNNs from GNNs such as Reachability and BetweenCycles. Better candidates are monadic monotone fixpoint logic (MMFP), monadic least fixpoint logic (MLFP), and monadic second-order logic (MSO)—three well-known general-purpose logics with recursion (Libkin 2004). Obtaining van Benthem-Rosen type theorems on finite structures for these base logics, however, are long-standing open problems, which are famously extremely challenging (Blumensath and Wolf 2020). Therefore, we leave such general characterisations open and concentrate on local MMFP, a fragment of MMFP where the free variable of a sub-formula defining a fixpoint predicate must always be witnessed by a path of bounded length and for which we can obtain a van Benthem-Rosen type theorem for bisimulations and gs-bisimulations. Next to the desired syntactic characterisations of RecGNNs and GSGNNs, our proof technique is, to the best of our knowledge, new and valuable on its own, because it may lead to van Benthem-Rosen type theorems for full MMFP and MLFP.

We next formally define MMFP, which is FOL extended with the least fixpoint operators of all unary formulas inducing monotone operators, and its local fragment.

Definition 12. *The syntax of (*parameter-free*)* MMFP *extends the FOL syntax with a constructor $[\mathbf{lfp}_{U,y}\phi](x)$, where ϕ is the unary* iterating *formula over the signature extended with a fresh unary predicate U such that $F^U_{\mathfrak{A},\phi}$ is monotone for every structure $\mathfrak{A}$ over the extended signature; here, $F^U_{\mathfrak{A},\phi} : 2^A \to 2^A$ is the function that maps each $X \subseteq A$*

to the set $\{a \in A \mid (\mathfrak{A}, X/U) \models \phi(a)\}$*, where* $(\mathfrak{A}, X/U)$ *is the structure extending* $\mathfrak{A}$ *by interpreting* U *as* X*; moreover,* $F^U_{\mathfrak{A},\phi}$ *is* monotone *for a structure* $\mathfrak{A}$ *if* $F^U_{\mathfrak{A},\phi}(X) \subseteq F^U_{\mathfrak{A},\phi}(Y)$ *for every* $X, Y \subseteq A$ *such that* $X \subseteq Y$*. The semantics of* MMFP *extends the one of FOL so that*

$$\mathfrak{A} \models [\mathbf{lfp}_{U,y}\phi(y)](a) \quad \text{if} \quad a \in \mathsf{lfp}(F^U_{\mathfrak{A},\phi}),$$

where $\mathsf{lfp}(F^U_{\mathfrak{A},\phi})$ *is the least fixpoint of* $F^U_{\mathfrak{A},\phi}$*—that is, the smallest set* $X \in 2^A$ *such that* $X = F^U_{\mathfrak{A},\phi}(X)$*.*

Note that the monotonicity requirement ensures, by the Knaster-Tarski theorem, that $F^U_{\mathfrak{A},\phi}$ has a unique least fixpoint (Libkin 2004); therefore, MMFP semantics are well-defined. Note also that the syntax of MMFP is undecidable (Libkin 2004); however, this shortcoming is not essential for our results.

Finally, we define the local fragment of MMFP, which requires that each existentially quantified variable must be witnessed by an element reachable from the witnesses of other variables via a path of a certain length.

Definition 13. *An MMFP formula* ϕ *is* local *if its each* $\exists$*-subformula* ϕ' *is of the form* $\exists y.\, (d_k(x, y) \wedge \psi)$ *for some* $k \in \mathbb{N}$*, where* x *is a free variable of the most-specific* **lfp***-iterating subformula of* ϕ *containing* ϕ'*, if it exists, or of* ϕ*, otherwise; and*

$$d_k(x, y) = \exists z_1. \ldots \exists z_{k-1}.\, d_1(x, z_1) \wedge d_1(z_1, z_2) \wedge \cdots \wedge d_1(z_{k-2}, z_{k-1}) \wedge d_1(z_{k-1}, y),$$

for $d_1(x, y) = \bigvee_{R \in \sigma_2} R(x, y) \vee R(y, x)$*.* LocMMFP *is the language of unary local MMFP formulas.*

To have a feeling of LocMMFP, one can verify that it can express both Reachability and BetweenCycles, but is incomparable with unary FOL. Having our base logic defined, we next specify our target logic. This is MMFP(GML$^-$), which is the extension of FOL (GML$^-$) with the same constructor that extends FOL to MMFP.

Definition 14. *The syntax of* MMFP(GML$^-$) *extends the one of FOL (GML*$^-$*) with the constructor that extends FOL to MMFP (with the same restrictions). The semantics of MMFP(GML*$^-$*) is inherited from MMFP.*

Note, each MMFP(GML$^-$) formula is equivalent to a LocMMFP formula, and so we do not need a separate language of local MMFP(GML$^-$) formulas.

We are now ready to state our van Benthem-Rosen type characterisation theorems and their corollaries.

Theorem 5. *A LocMMFP formula is bisimulation-invariant if and only if it is equivalent, over finite structures, to an MMFP(GML*$^-$*) formula.*

The proof of this theorem relies on a lemma that shows that every LocMMFP formula is equivalent to an MMFP(GML$^-$) formula on n-acyclic structures for a sufficiently large $n \in \mathbb{N}$, as well as a construction of bisimilar n-acyclic structures for any given structure (i.e., bisimilar structures that do not contain any cycles of length smaller than n); our construction builds upon ideas of Bednarczyk et al. (2021).[1] Then, the theorem is proved by showing equivalence between every bisimulation-invariant LocMMFP formula ϕ and the MMFP(GML$^-$) formula ψ obtained by applying our lemma to ϕ. Specifically, for the forward direction, we first fix an arbitrary structure $\mathfrak{A}$ and $a \in A$ with $\mathfrak{A} \models \phi(a)$, and then apply our construction to obtain an n-acyclic structure $\mathfrak{A}'$ for some sufficiently large n. By bisimulation invariance of ϕ, we have $\mathfrak{A}' \models \phi(a')$ for some a' bisimilar to a, and $\mathfrak{A}' \models \psi(a')$ by our lemma. So, we obtain $\mathfrak{A} \models \psi(a)$ by bisimulation-invariance of ψ. The backward direction is analogous.

Theorems 3 and 5 imply our first logical characterisation.

Corollary 4. *A node classifier is realised by both a RecGNN and a LocMMFP formula if and only if it is realised by a MMFP(GML*$^-$*) formula.*

Having presented the syntactic characterisations for RecGNNs, we move on to the one for GSGNNs. We again begin with an appropriate van Benthem-Rosen type theorem, which complements Theorem 5 by saying that bisimulation-invariance and gs-bisimulation-invariance are equivalent for LocMMFP formulas.

Theorem 6. *A LocMMFP formula is gs-bisimulation-invariant if and only if it is equivalent, over finite structures, to an MMFP(GML*$^-$*) formula.*

This theorem follows from Theorem 5 and two lemmas: one saying that every LocMMFP formula is invariant under disjoint unions, and another saying that each classifier is invariant under disjoint unions and gs-bisimulation-invariant if and only if it is bisimulation-invariant; here, a classifier f is *invariant under disjoint unions* whenever, for every two structures $\mathfrak{A}$ and $\mathfrak{B}$ with disjoint sets of elements and every $a \in A$, we have $a \in f(\mathfrak{A})$ if and only if $a \in f(\mathfrak{A} \uplus \mathfrak{B})$, where the disjoint union $\mathfrak{A} \uplus \mathfrak{B}$ is defined as expected. The first lemma is rather straightforward; the second lemma holds because the invariance under disjoint unions allows us to construct, for σ-structures $\mathfrak{A}$ and $\mathfrak{B}$ with bisimilar elements a and b, σ-structures of the form $\mathfrak{A} \uplus \mathfrak{A}'$ and $\mathfrak{B} \uplus \mathfrak{B}'$ whose universes are of the same size, and a and b are still bisimilar.

Finally, Theorems 4 and 6 imply our logical characterisation for GSGNNs.

Corollary 5. *A node classifier is realised by both a GSGNN and a LocMMFP formula if and only if it is realised by a MMFP(GML*$^-$*) formula.*

Conclusion

In this paper, we formalised two variants of GNNs with recursion, RecGNNs and GSGNNs, and began to study their properties. We established a strict expressiveness hierarchy between them and plain GNNs, identified novel semantic characterisations for them in terms of appropriate bisimulations, and complemented them with syntactic characterisations in term of a simple fixpoint logic. To the best of our knowledge, this is the first systematic study of recurrent GNNs. Many questions about GNNs with recursion remain open, including the characterisation in terms of more expressive logics, the identification of more fine-grained and more general recurrent GNN classes, and the study of their practical applicability for various tasks and settings.

[1] A different construction yielding bisimilar structures with the same property was proposed by Otto (2004).

Acknowledgments

This work was supported by Siemens AG, Samsung Research UK, the EPSRC projects ConCur (EP/V050869/1), OASIS (EP/S032347/1) and UK FIRES (EP/S019111/1), and the Research Council of Norway through its Centres of Excellence scheme, Integreat—Norwegian Centre for knowledge-driven machine learning, project 332645.

For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

Abboud, R.; Ceylan, I. I.; Grohe, M.; and Lukasiewicz, T. 2021. The Surprising Power of Graph Neural Networks With Random Node Initialization. In *International Joint Conference on Artificial Intelligence 30*, 2112–2118.

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Baader, F. 2003. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *Introduction to Description Logic*. Cambridge University Press.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J.; and Silva, J. P. 2019. The Logical Expressiveness of Graph Neural Networks. In *International Conference on Learning Representations 7*.

Bednarczyk, B.; Orłowska, M.; Pacanowska, A.; and Tan, T. 2021. On Classical Decidable Logics Extended With Percentage Quantifiers and Arithmetics. In *Annual Conference on Foundations of Software Technology and Theoretical Computer Science 41*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.

Blackburn, P.; van Benthem, J.; and Wolter, F. 2006. *Handbook of Modal Logic*. Elsevier.

Blumensath, A.; and Wolf, F. 2020. Bisimulation Invariant Monadic-Second Order Logic in the Finite. *Theoretical Computer Science*, 26–43.

Cai, J.-Y.; Fürer, M.; and Immerman, N. 1992. An Optimal Lower Bound on the Number of Variables for Graph Identification. *Combinatorica*, 389–410.

de Rijke, M. 1996. A Note on Graded Modal Logic. *Stud Logica*, 271–283.

Fout, A.; Byrd, J.; Shariat, B.; and Ben-Hur, A. 2017. Protein Interface Prediction Using Graph Convolutional Networks. In *Advances in Neural Information Processing Systems 30*.

Gallicchio, C.; and Micheli, A. 2010. Graph Echo State Networks. In *International Joint Conference on Neural Networks*, 1–8. IEEE.

Grohe, M. 2023. The Descriptive Complexity of Graph Neural Networks. In *ACM/IEEE Symposium on Logic in Computer Science 38*, 1–14. IEEE.

Hamilton, W. L. 2021. Graph Representation Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 1–159.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30*.

Immerman, N. 2012. *Descriptive Complexity*. Springer.

Janin, D.; and Walukiewicz, I. 1996. On the Expressive Completeness of the Propositional Mu-Calculus With Respect to Monadic Second Order Logic. In *International Conference on Concurrency Theory 6*, 263–277. Springer.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations 5*.

Libkin, L. 2004. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer.

Liu, Z.; and Zhou, J. 2020. *Introduction to Graph Neural Networks*. Morgan and Claypool Publishers.

Loukas, A. 2020. What Graph Neural Networks Cannot Learn: Depth vs Width. In *International Conference on Learning Representations 8*.

Milner, R. 1989. *Communication and Concurrency*. Prentice-Hall International Series in Computer Science. Prentice Hall.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *AAAI Conference on Artificial Intelligence 33*, 4602–4609.

Otto, M. 2004. Modal and Guarded Characterisation Theorems Over Finite Transition Systems. *Annals of Pure and Applied Logic*, 173–205.

Otto, M. 2019. Graded Modal Logic and Counting Bisimulation. *arXiv preprint arXiv:1910.00039*.

Pflueger, M.; Tena Cucala, D. J.; and Kostylev, E. V. 2022. GNNQ: A Neuro-Symbolic Approach to Query Answering Over Incomplete Knowledge Graphs. In *International Semantic Web Conference 21*, 481–497. Springer.

Rautenberg, W. 2009. *A Concise Introduction to Mathematical Logic*. Springer.

Reiser, P.; Neubert, M.; Eberhard, A.; Torresi, L.; Zhou, C.; Shao, C.; Metni, H.; van Hoesel, C.; Schopmans, H.; Sommer, T.; et al. 2022. Graph Neural Networks for Materials Science and Chemistry. *Communications Materials*, 93.

Rosen, E. 1997. Modal Logic Over Finite Structures. *Journal of Logic, Language and Information*, 427–439.

Rosenbluth, E.; Toenshoff, J.; and Grohe, M. 2023. Some Might Say All You Need Is Sum. In *International Joint Conference on Artificial Intelligence 32*, 4172–4179.

Sato, R.; Yamada, M.; and Kashima, H. 2021. Random Features Strengthen Graph Neural Networks. In *SIAM International Conference on Data Mining*, 333–341. SIAM.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 61–80.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling Relational Data With Graph Convolutional Networks. In *European Semantic Web Conference 15*, 593–607. Springer.

Tena Cucala, D. J.; Cuenca Grau, B.; Kostylev, E. V.; and Motik, B. 2022. Explainable GNN-Based Models Over Knowledge Graphs. In *International Conference on Learning Representations 10*.

Toenshoff, J.; Ritzert, M.; Wolf, H.; and Grohe, M. 2021. Graph Neural Networks for Maximum Constraint Satisfaction. *Frontiers in Artificial Intelligence*.

van Benthem, J. 1976. *Modal Correspondence Theory*. Ph.D. thesis, University of Amsterdam.

Weisfeiler, B. Y.; and Leman, A. A. 1968. A Reduction of a Graph to a Canonical Form and an Algebra Arising During This Reduction. *Nauchno-Technicheskaya Informatsia*, 12–16.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations 7*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *ACM/SIGKDD International Conference on Knowledge Discovery & Data Mining 24*, 974–983.