Logical Characterizations of Recurrent Graph Neural Networks with Reals and Floats

Veeti Ahvonen^1, **Damian Heiman**^{1}, **Antti Kuusisto**^{1}, **Carsten Lutz**^{2,3}
^{1}Tampere University, ^{2}Leipzig University, ^{3}ScaDS.AI, Dresden/Leipzig
^{1}`firstname.lastname@tuni.fi`, ^{2,3}`clu@informatik.uni-leipzig.de`

Abstract

In pioneering work from 2019, Barceló and coauthors identified logics that precisely match the expressive power of constant iteration-depth graph neural networks (GNNs) relative to properties definable in first-order logic. In this article, we give exact logical characterizations of recurrent GNNs in two scenarios: (1) in the setting with floating-point numbers and (2) with reals. For floats, the formalism matching recurrent GNNs is a rule-based modal logic with counting, while for reals we use a suitable infinitary modal logic, also with counting. These results give exact matches between logics and GNNs in the recurrent setting without relativising to a background logic in either case, but using some natural assumptions about floating-point arithmetic. Applying our characterizations, we also prove that, relative to graph properties definable in monadic second-order logic (MSO), our infinitary and rule-based logics are equally expressive. This implies that recurrent GNNs with reals and floats have the same expressive power over MSO-definable properties and shows that, for such properties, also recurrent GNNs with reals are characterized by a (finitary!) rule-based modal logic. In the general case, in contrast, the expressive power with floats is weaker than with reals. In addition to logic-oriented results, we also characterize recurrent GNNs, with both reals and floats, via distributed automata, drawing links to distributed computing models.

1 Introduction

Graph Neural Networks (GNNs) [11, 30, 34] have proven to be highly useful for processing graph data in numerous applications that span a remarkable range of disciplines including bioinformatics [35], recommender systems [33], traffic forecasting [21], and a multitude of others. The success of GNNs in applications has stimulated lively research into their theoretical properties such as expressive power. A landmark result is due to Barceló et al. [5] which was among the first to characterize the expressive power of GNNs in terms of *logic*, see [12, 8, 6, 26] and references therein for related results. More precisely, Barceló et al. show that a basic GNN model with a constant number of iterations has exactly the same expressive power as graded modal logic GML in restriction to properties definable in first-order logic FO.

In this article, we advance the analysis of the expressive power of GNNs in two directions. First, we study the relation between GNN models based on real numbers, as mostly studied in theory, and GNN models based on floating-point numbers, as mostly used in practice. And second, we focus on a family of basic recurrent GNNs while previous research has mainly considered GNNs with a constant number of iterations, with the notable exception of [26]. The GNNs studied in the current paper have a simple and natural termination (or acceptance) condition: termination is signaled via designated feature vectors and thus the GNN "decides" by itself about when to terminate. We remark that some of our results also apply to constant iteration GNNs and to recurrent GNNs with a termination condition based on fixed points, as used in the inaugural work on GNNs [11]; see the conclusion section for further details.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

We provide three main results. The first one is that recurrent GNNs with floats, or $\mathrm{GNN[F]}$s, have the same expressive power as the *graded modal substitution calculus* GMSC [24, 1, 3]. This is a rule-based modal logic that extends the *modal substitution calculus* MSC [24] with counting modalities. MSC has been shown to precisely correspond to distributed computation models based on automata [24] and Boolean circuits [1]. GMSC is related to the graded modal μ-calculus, but orthogonal in expressive power. The correspondence between $\mathrm{GNN[F]}$s and GMSC is as follows.

Theorem 3.2. *The following have the same expressive power:* $\mathrm{GNN[F]}$*s,* GMSC*, and R-simple aggregate-combine* $\mathrm{GNN[F]}$*s.*

Here *R-simple aggregate-combine* $\mathrm{GNN[F]}$s mean $\mathrm{GNN[F]}$s that use basic aggregate-combine functions as specified by Barceló et al. [5] and the truncated ReLU as the non-linearity function, see Section 2.1 for the formalities. The theorem shows that an R-simple model of $\mathrm{GNN[F]}$s suffices, and in fact $\mathrm{GNN[F]}$s with a more complex architecture can be turned into equivalent R-simple ones. We emphasize that the characterization provided by Theorem 3.2 is *absolute*, that is, not relative to a background logic. In contrast, the characterization by Barceló et al. [5] is relative to first-order logic. Our characterization does rely, however, on an assumption about floating-point arithmetics. We believe that this assumption is entirely natural as it reflects practical implementations of floats.

Our second result shows that recurrent GNNs with reals, or $\mathrm{GNN}[\mathbb{R}]$s, have the same expressive power as the infinitary modal logic ω-GML that consists of infinite disjunctions of GML-formulas.

Theorem 3.4. $\mathrm{GNN}[\mathbb{R}]$s have the same expressive power as ω-GML.

Again, this result is absolute. As we assume no restrictions on the arithmetics used in $\mathrm{GNN}[\mathbb{R}]$s, they are very powerful: with the infinitary logic ω-GML it is easy to define even undecidable graph properties. We regard the theorem as an interesting theoretical upper bound on the expressivity of GNNs operating in an unrestricted, recurrent message passing scenario with messages flowing to neighbouring nodes. We note that $\mathrm{GNN}[\mathbb{R}]$s can easily be shown more expressive than $\mathrm{GNN[F]}$s.

Our third result considers $\mathrm{GNN}[\mathbb{R}]$s and $\mathrm{GNN[F]}$s relative to a very expressive background logic, probably the most natural choice in the recurrent GNN context: *monadic second-order logic* MSO.

Theorem 4.3. *Let $\mathcal{P}$ be a property expressible in* MSO*. Then $\mathcal{P}$ is expressible as a* $\mathrm{GNN}[\mathbb{R}]$ *if and only if it is expressible as a* $\mathrm{GNN[F]}$.

This result says that, remarkably, for the very significant and large class of MSO-expressible properties, using actual reals with unrestricted arithmetic gives no more expressive power than using floats, by Theorem 3.2 even in the R-simple aggregate-combine setting. Thus, for this class of properties, the theoretical analyses from the literature do not diverge from the practical implementation! Taken together, the above results also imply that in restriction to MSO-expressible properties, $\mathrm{GNN}[\mathbb{R}]$s are equivalent to the (finitary!) graded modal substitution calculus GMSC.

We also develop characterizations of GNNs in terms of distributed automata. These are in fact crucial tools in our proofs, but the characterizations are also interesting in their own right as they build links between GNNs and distributed computing. We study a class of distributed automata called counting message passing automata (CMPAs) that may have a countably infinite number of states. Informally, these distributed automata update the state of each node according to the node's own state and the *multiset* of states received from its out-neighbours. We also study their restriction that admits only a finite number of states (FCMPAs) and, furthermore, bounded FCMPAs. In the bounded case the multiplicities of states in the received multisets are bounded by some constant $k \in \mathbb{N}$. A summary of our main results, both absolute and relative to MSO, is given in Table 1.

Table 1: A summary of our main results. The first row contains the results obtained without relativising to a background logic and the second row contains results relative to MSO. Here $x \equiv y$ means that x and y have the same expressive power while $x < y$ means that y is strictly more expressive than x. Further, "bnd." stands for "bounded" and "AC" for "aggregate-combine".

Absolute:	$\mathrm{GNN[F]} \equiv$ R-simple AC-$\mathrm{GNN[F]} \equiv \mathrm{GMSC} \equiv$ bnd. FCMPA $<$ $\mathrm{GNN}[\mathbb{R}] \equiv \omega$-GML $\equiv$ CMPA
MSO:	$\mathrm{GNN[F]} \equiv$ R-simple AC-$\mathrm{GNN[F]} \equiv \mathrm{GMSC} \equiv$ bnd. FCMPA $\equiv$ $\mathrm{GNN}[\mathbb{R}] \equiv \omega$-GML $\equiv$ CMPA

Related Work. Barceló et al. [5] study aggregate-combine GNNs with a constant number of iterations. They characterize these GNNs—in restriction to properties expressible in first-order logic—in terms of graded modal logic GML and show that a *global readout mechanism* leads to a model equivalent to the two-variable fragment of first-order logic with counting quantifiers FOC^2. Our work extends the former result to include recurrence in a natural way while we leave studying global readouts as future work; see the conclusion section for further details. Grohe [13] connects the *guarded fragment of first-order logic with counting* $\mathrm{GFO} + \mathrm{C}$ and polynomial-size bounded-depth circuits, linking (non-recurrent) GNNs to the circuit complexity class TC^0. Grohe's characterization utilises dyadic rationals rather than floating-point numbers. Benedikt et al. [6] use logics with a generalized form of counting via *Presburger quantifiers* to obtain characterizations for (non-recurrent) GNNs with a bounded activation function. The article also investigates questions of decidability concerning GNNs—a topic we will not study here. As a general remark on related work, it is worth mentioning that the expressive power of (basic) recurrent GNN-models is invariant under the Weisfeiler-Leman test. This link has been recently studied in numerous articles [25, 36, 12, 5].

Pfluger et al. [26] investigate recurrent GNNs with two kinds of termination conditions: one based on reaching a fixed point when iteratively generating feature vectors, and one where termination occurs after a number of rounds determined by the size of the input graph. They concentrate on the case of unrestricted aggregation and combination functions, even including all uncomputable ones. Their main result is relative to a logic LocMMFP introduced specifically for this purpose, extending first-order logic with a least fixed-point operator over unary monotone formulas. The characterization itself is given in terms of the graded two-way μ-calculus. We remark that MSO significantly generalizes LocMMFP and that the graded two-way μ-calculus is incomparable in expressive power to our GMSC. In contrast to our work and to Barcelo et al. [5], Pfluger et al. do not discuss the case where the aggregation and combination functions of the GNNs are R-simple or restricted in any other way. We view our work as complementing yet being in the spirit of both Barceló et al. [5] and Pfluger et al. [26].

GNNs are essentially distributed systems, and logical characterizations for distributed systems have been studied widely. A related research direction begins with Hella et al. [16], Kuusisto [24] and Hella et al. [17] by results linking distributed computation models to modal logics. The articles [16] and [17] give characterizations of constant-iteration scenarios with modal logics, and [24] lifts the approach to recurrent message-passing algorithms via showing that the modal substitution calculus MSC captures the expressivity of finite distributed message passing automata. This generalizes the result from [16] that characterized the closely related class $\mathsf{SB}(1)$ of local distributed algorithms with modal logic. Later Ahvonen et al. [1] showed a match between MSC and circuit-based distributed systems. Building on the work on MSC, Reiter showed in [27] that the μ-fragment of the modal μ-calculus captures the expressivity of finite message passing automata in the asynchronous scenario.

2 Preliminaries

We let $\mathbb{N}$, $\mathbb{Z}_+$ and $\mathbb{R}$ denote the sets of non-negative integers, positive integers, and real numbers respectively. For all $n \in \mathbb{Z}_+$, we let $[n] := \{1, \ldots, n\}$ and for all $n \in \mathbb{N}$, we let $[0; n] := \{0, \ldots, n\}$. With $|X|$ we denote the cardinality of the set X, with $\mathcal{P}(X)$ the power set of X and with $\mathcal{M}(X)$ the set of multisets over X, i.e., the set of functions $X \to \mathbb{N}$. With $\mathcal{M}_k(X)$ we denote the set of k-multisets over X, i.e., the set of functions $X \to [0; k]$. Given a k-multiset $M \in \mathcal{M}_k(X)$ and $x \in X$, intuitively $M(x) = n < k$ means that there are exactly n copies of x and $M(x) = k$ means that there are k *or more* copies of x.

We work with node-labeled **directed graphs** (possibly with self-loops), and *simply refer to them as **graphs***. Let LAB denote a countably infinite set of **node label symbols**, representing features. We denote finite sets of node label symbols by $\Pi \subseteq \mathrm{LAB}$. Given any $\Pi \subseteq \mathrm{LAB}$, a **$\Pi$-labeled graph** is a triple $G = (V, E, \lambda)$ where V is a set of **nodes**, $E \subseteq V \times V$ is a set of **edges** and $\lambda \colon V \to \mathcal{P}(\Pi)$ is a **node labeling function**. Note that a node can carry multiple label symbols. A **pointed graph** is a pair (G, v) with $v \in V$. Given a graph (V, E, λ), the set of **out-neighbours** of $v \in V$ is $\{ w \mid (v, w) \in E \}$. Unless stated otherwise, we only consider *finite* graphs, i.e., graphs where the set of nodes is finite. A **node property over** Π is a class of pointed Π-labeled graphs. A **graph property over** Π is a class of Π-labeled graphs. A graph property $\mathcal{G}$ over Π corresponds to a node property $\mathcal{N}$ over Π if the following holds for all Π-labeled graphs G: $G \in \mathcal{G}$ iff $(G, v) \in \mathcal{N}$ for every node v of G. Henceforth a property means a node property. We note that many of our results hold even with infinite graphs and infinite sets of node label symbols. Our results easily extend to graphs that admit labels on both nodes and edges.

2.1 Graph neural networks

A graph neural network (GNN) is a neural network architecture for graph-structured data. It may be viewed as a distributed system where the nodes of the (directed, node-labeled) input graph calculate with real numbers and communicate synchronously in discrete rounds. More formally, a **recurrent graph neural network** $\mathrm{GNN}[\mathbb{R}]$ over (Π, d), with $\Pi \subseteq \mathrm{LAB}$ and $d \in \mathbb{Z}_+$, is a tuple $\mathcal{G} = (\mathbb{R}^d, \pi, \delta, F)$. A recurrent graph neural network computes in a (node-labeled) directed graph as follows. In any Π-labeled graph (V, E, λ), the **initialization function** $\pi \colon \mathcal{P}(\Pi) \to \mathbb{R}^d$ assigns to each node v an initial **feature vector** or **state** $x_v^0 = \pi(\lambda(v))$.[1] In each subsequent round $t = 1, 2, \ldots$, every node computes a new feature vector x_v^t using a **transition function** $\delta \colon \mathbb{R}^d \times \mathcal{M}(\mathbb{R}^d) \to \mathbb{R}^d$, $\delta(x, y) = \mathrm{COM}(x, \mathrm{AGG}(y))$, which is a composition of an **aggregation function** $\mathrm{AGG} \colon \mathcal{M}(\mathbb{R}^d) \to \mathbb{R}^d$ (typically sum, min, max or average) and a **combination function** $\mathrm{COM} \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ such that $x_v^t = \mathrm{COM}\left(x_v^{t-1}, \mathrm{AGG}\left(\{\{\, x_u^{t-1} \mid (v,u) \in E \,\}\}\right)\right)$ where double curly brackets $\{\{...\}\}$ denote multisets.[2] The recurrent GNN $\mathcal{G}$ **accepts** a pointed Π-labeled graph (G, v) if v visits (at least once) a state in the set $F \subseteq \mathbb{R}^d$ of **accepting feature vectors**, i.e., $x_v^t \in F$ for some $t \in \mathbb{N}$. When we do not need to specify d, we may refer to a $\mathrm{GNN}[\mathbb{R}]$ over (Π, d) as a $\mathrm{GNN}[\mathbb{R}]$ over Π. A **constant-iteration** $\mathrm{GNN}[\mathbb{R}]$ is a pair $(\mathcal{G}, N)$ where $\mathcal{G}$ is a $\mathrm{GNN}[\mathbb{R}]$ and $N \in \mathbb{N}$. It **accepts** a pointed graph (G, v) if $x_v^N \in F$. Informally, we simply run a $\mathrm{GNN}[\mathbb{R}]$ for N iterations and accept (or do not accept) based on the last iteration. We say that $\mathcal{G}$ (resp., $(\mathcal{G}, N)$) **expresses** a node property $\mathcal{P}$ over Π, if for each pointed Π-labeled graph (G, w): $(G, w) \in \mathcal{P}$ iff $\mathcal{G}$ (resp., $(\mathcal{G}, N)$) accepts (G, w). A node property $\mathcal{P}$ over Π is **expressible** as a $\mathrm{GNN}[\mathbb{R}]$ (resp. as a constant-iteration $\mathrm{GNN}[\mathbb{R}]$) if there is a $\mathrm{GNN}[\mathbb{R}]$ (resp. constant-iteration $\mathrm{GNN}[\mathbb{R}]$) expressing $\mathcal{P}$.

One common, useful and simple possibility for the aggregation and combination functions, which is also used by Barceló et al. (see [5], and also the papers [25, 15]) is defined by $\mathrm{COM}\left(x_v^{t-1}, \mathrm{AGG}\left(\{\{\, x_u^{t-1} \mid (v,u) \in E \,\}\}\right)\right) = f(x_v^{t-1} \cdot C + \sum_{(v,u) \in E} x_u^{t-1} \cdot A + \mathbf{b})$, where $f \colon \mathbb{R}^d \to \mathbb{R}^d$ is a non-linearity function (such as the truncated ReLU also used by Barceló et al. in [5], defined by $\mathrm{ReLU}^*(x) = \min(\max(0, x), 1)$ and applied separately to each vector element), $C, A \in \mathbb{R}^{d \times d}$ are matrices and $\mathbf{b} \in \mathbb{R}^d$ is a bias vector. We refer to GNNs that use aggregation and combination functions of this form and ReLU^* as the non-linearity function as **R-simple aggregate-combine** GNNs (here 'R' stands for ReLU^*).

Example 2.1. Given Π and $p \in \Pi$, *reachability of node label symbol* p is the property $\mathcal{P}$ over Π that contains those pointed Π-labeled graphs (G, v) where a path exists from v to some u with $p \in \lambda(u)$. An R-simple aggregate-combine $\mathrm{GNN}[\mathbb{R}]$ over $(\Pi, 1)$ (where $C = A = 1$, $\mathbf{b} = 0$ and 1 is the only accepting feature vector) can express $\mathcal{P}$: In round 0, a node w's state is 1 if $p \in \lambda(w)$ and else 0. In later rounds, w's state is 1 if w's state was 1 or it gets 1 from its out-neighbours; else the state is 0.

Remark 2.2. Notice that unrestricted $\mathrm{GNN}[\mathbb{R}]$s can express, even in a single iteration, node properties such as that the number of immediate out-neighbours of a node is a prime number. In fact, for any $U \subseteq \mathbb{N}$, including any undecidable set U, a $\mathrm{GNN}[\mathbb{R}]$ can express the property that the number l of immediate out-neighbours is in the set U. See [6] for related undecidability results.

Informally, a floating-point system contains a finite set of rational numbers and arithmetic operations $\cdot$ and $+$. Formally, if $p \in \mathbb{Z}_+$, $n \in \mathbb{N}$ and $\beta \in \mathbb{Z}_+ \setminus \{1\}$, then a **floating-point system** is a tuple $S = ((p, n, \beta), +, \cdot)$ that consists of the set D_S of all rationals accurately representable in the form $0.d_1 \cdots d_p \times \beta^e$ or $-0.d_1 \cdots d_p \times \beta^e$ where $0 \leq d_i \leq \beta - 1$ and $e \in \{-n, \ldots, n\}$. It also consists of arithmetic operations $+$ and $\cdot$ of type $D_S \times D_S \to D_S$. We adopt the common convention where $+$ and $\cdot$ are defined by taking a precise sum/product w.r.t. reals and then rounding to the closest float in D_S, with ties rounding to the float with an even least significant digit, e.g., $0.312 + 0.743$ evaluates to 1.06 if the real sum 1.055 is not in the float system. Thus, our float systems handle overflow by capping at the maximum value instead of wrapping around. We typically just write S for D_S.

Consider GNNs using floats in the place of reals. In GNNs, sum is a common aggregation function (also used in R-simple GNNs), and the sum of floats can depend on the ordering of floats, since it

[1] In [5, 26] an initialization function is not explicitly included in GNNs; instead each node is labeled with an initial feature vector in place of node label symbols. However, these two approaches are essentially the same.

[2] In GNNs here, messages flow in the direction opposite to the edges of graphs, i.e., an edge $(v, u) \in E$ means that v receives messages sent by u. This is only a convention that could be reversed via using a modal logic that scans the *inverse relation* of E instead of E.

is not associative. In real-life implementations, the set V of nodes of the graph studied can typically be associated with an implicit linear order relation $<^V$ (which is not part of the actual graph). It is then natural to count features of out-neighbours in the order $<^V$. However, this allows float GNNs to distinguish isomorphic nodes, which violates the desire that GNNs should be invariant under isomorphism. For example, summing 1, -1 and 0.01 in two orders in a system where the numbers must be representable in the form $0.d_1d_2 \times 10^e$ or $-0.d_1d_2 \times 10^e$: first $(1 + (-1)) + 0.01 = 0 + 0.01 = 0.01$ while $(1 + 0.01) + (-1) = 1 + (-1) = 0$, since 1.01 is not representable in the system. A float GNN could distinguish two isomorphic nodes with such ordering of out-neighbours.

To ensure isomorphism invariance for GNNs with floats that use sum, it is natural to order the floats instead of the nodes. For example, adding floats in increasing order (of the floats) is a natural and simple choice. Summing in this increasing order is also used widely in applications, being a reasonable choice w.r.t. accuracy (see, e.g., [32],[29], [18]). Generally, floating-point sums in applications have been studied widely, see for example [19]. Summing multisets of floats in increasing order leads to a bound in the multiplicities of the elements of the sum; see Proposition 2.3 for the formal statement. Before discussing its proof, we define the **k-projection** of a multiset M as $M_{|k}$ where $M_{|k}(x) = \min\{M(x), k\}$. Given a multiset N of floats in float system S, we let $\mathrm{SUM}_\mathrm{S}(N)$ denote the output of the sum $f_1 + \cdots + f_\ell$ where **(1)** f_i appears $N(f_i)$ times (i.e., its multiplicity) in the sum, **(2)** the floats appear and are summed in increasing order and **(3)** $+$ is according to S.

Proposition 2.3. *For all floating-point systems S, there exists a $k \in \mathbb{N}$ such that for all multisets M over floats in S, we have* $\mathrm{SUM}_S(M) = \mathrm{SUM}_S(M_{|k})$.

Proof. (Sketch) See also in Appendix A.2. Let $u = 0.0 \cdots 01 \times \beta^e$ and $v = 0.10 \cdots 0 \times \beta^{e+1}$. Now notice that for a large enough ℓ, summing u to itself $m > \ell$ times will always give v. □

Due to Proposition 2.3, GNNs with floats using sum in increasing order are bounded in their ability to fully count out-neighbours. Thus, it is natural to assume that floating-point GNNs are **bounded** GNNs, i.e., the aggregation function can be written as $\mathcal{M}_k(U^d) \to U^d$ for some **bound** $k \in \mathbb{N}$, i.e., for every multiset $M \in \mathcal{M}(U^d)$, we have $\mathrm{AGG}(M) = \mathrm{AGG}(M_{|k})$, where $M_{|k}$ is the k-projection of M (and U^d is the set of states of the GNN). We finally give a formal definition for floating-point GNNs: a **floating-point graph neural network** (GNN[F]) is simply a bounded GNN where the set of states and the domains and co-domains of the functions are restricted to some floating-point system S instead of $\mathbb{R}$ (note that S can be any floating-point system). In R-simple GNN[F]s, SUM_S replaces the sum of reals as the aggregation function, and their bound is thus determined by the choice of S. A GNN[F] obtained by removing the condition on boundedness is called an **unrestricted** GNN[F]. Note that by default and unless otherwise stated, a GNN[$\mathbb{R}$] is unbounded, whereas a GNN[F] is bounded. Now, it is immediately clear that unrestricted GNN[F]s (with an unrestricted aggregation function) are more expressive than GNN[F]s: expressing the property that a node has an even number of out-neighbours is easy with unrestricted GNN[F]s, but no bounded GNN[F] with bound k can distinguish the centers of two star graphs with k and $k+1$ out-neighbours.

2.2 Logics

We then define the logics relevant for this paper. For $\Pi \subseteq \mathrm{LAB}$, the set of **$\Pi$-formulae of graded modal logic** (GML) is given by the grammar $\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond_{\geq k}\varphi$, where $p \in \Pi$ and $k \in \mathbb{N}$. The connectives $\vee, \rightarrow, \leftrightarrow$ are considered abbreviations in the usual way. Note that node label symbols serve as propositional symbols here. A Π-formula of GML is interpreted in pointed Π-labeled graphs. In the context of modal logic, these are often called (pointed) Kripke models. Let $G = (V, E, \lambda)$ be a Π-labeled graph and $w \in V$. The truth of a formula φ in a pointed graph (G, w) (denoted $G, w \models \varphi$) is defined as usual for the Boolean operators and $\top$, while for $p \in \Pi$ and $\Diamond_{\geq k}\varphi$, we define that $G, w \models p$ iff $p \in \lambda(w)$, and $G, w \models \Diamond_{\geq k}\varphi$ iff $G, v \models \varphi$ for at least k out-neighbours v of w. We use the abbreviations $\Diamond\varphi := \Diamond_{\geq 1}\varphi$, $\Box\varphi := \neg\Diamond\neg\varphi$ and $\Diamond_{=n}\varphi := \Diamond_{\geq n}\varphi \wedge \neg\Diamond_{\geq n+1}\varphi$. The set of **$\Pi$-formulae of** ω-GML is given by the grammar $\varphi ::= \psi \mid \bigvee_{\psi \in \Psi} \psi$, where ψ is a Π-formula of GML and Ψ is an at most countable set of Π-formulae of GML. The truth of infinite disjunctions is defined in the obvious way: $G, w \models \bigvee_{\psi \in \Psi} \psi \iff G, w \models \psi$ for some $\psi \in \Psi$.

We next introduce the graded modal substitution calculus (or GMSC), which extends the modal substitution calculus [24, 1, 3] with counting capabilities. Define the set $\mathrm{VAR} = \{ V_i \mid i \in \mathbb{N} \}$ of **schema variable symbols**. Let $\mathcal{T} = \{X_1, \ldots, X_n\} \subseteq \mathrm{VAR}$. The set of **$(\Pi, \mathcal{T})$-schemata** of

GMSC is defined by the grammar $\varphi ::= \top \mid p \mid X_i \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond_{\geq k}\varphi$ where $p \in \Pi$, $X_i \in \mathcal{T}$ and $k \in \mathbb{N}$. A $(\Pi, \mathcal{T})$-**program** Λ of GMSC consists of two lists of expressions

$$X_1(0) :- \varphi_1 \quad \cdots \quad X_n(0) :- \varphi_n \qquad\qquad X_1 :- \psi_1 \quad \cdots \quad X_n :- \psi_n$$

where $\varphi_1, \dots, \varphi_n$ are Π-formulae of GML and $\psi_1, \dots, \psi_n$ are $(\Pi, \mathcal{T})$-schemata of GMSC. Moreover, each program is associated with a set $\mathcal{A} \subseteq \mathcal{T}$ of **appointed predicates**. A program of **modal substitution calculus** MSC is a program of GMSC that may only use diamonds $\Diamond$ of the standard modal logic. The expressions $X_i(0) :- \varphi_i$ are called **terminal clauses** and $X_i :- \psi_i$ are called **iteration clauses**. The schema variable X_i in front of the clause is called the **head predicate** and the formula φ_i (or schema ψ_i) is called the **body** of the clause. The terminal and iteration clauses are the rules of the program. When we do not need to specify $\mathcal{T}$, we may refer to a $(\Pi, \mathcal{T})$-program as a Π-**program** of GMSC. Now, the n**th iteration formula** X_i^n of a head predicate X_i (or the iteration formula of X_i in **round** $n \in \mathbb{N}$) (w.r.t. Λ) is defined as follows. The 0th iteration formula X_i^0 is φ_i and the $(n+1)$st iteration formula X_i^{n+1} is ψ_i where each head predicate Y in ψ_i is replaced by the formula Y^n. We write $G, w \models \Lambda$ and say that Λ **accepts** (G, w) iff $G, w \models X^n$ for some appointed predicate $X \in \mathcal{A}$ and some $n \in \mathbb{N}$. Moreover, for all $(\Pi, \mathcal{T})$-schemata φ that are not head predicates and for $n \in \mathbb{N}$, we let φ^n denote the formula (w.r.t. Λ) where each $Y \in \mathcal{T}$ in φ is replaced by Y^n.

Recall that **monadic second-order logic** MSO is obtained as an extension of **first-order logic** FO by allowing quantification of unary relation variables X, i.e., if φ is an MSO-formula, then so are $\forall X\varphi$ and $\exists X\varphi$, see e.g. [9] for more details. Given a set $\Pi \subseteq \mathrm{LAB}$ of node label symbols, an FO- or MSO-formula φ over Π is an FO- or MSO-formula over a vocabulary which contains exactly a unary predicate for each $p \in \Pi$ and the edge relation symbol E. Equality is admitted.

Let φ be an ω-GML-formula, GMSC-schema, GMSC-program, or a rule of a program. The **modal depth** (resp. the **width**) of φ is the maximum number of nested diamonds in φ (resp. the maximum number $k \in \mathbb{N}$ that appears in a diamond $\Diamond_{\geq k}$ in φ). If an ω-GML-formula has no maximum depth (resp., width), its modal depth (resp., width) is ∞. If an ω-GML-formula has finite modal depth (resp., width), it is **depth-bounded** (resp., **width-bounded**). The **formula depth** of a GML-formula or GMSC-schema is the maximum number of nested operators $\neg$, $\wedge$ and $\Diamond_{\geq k}$. Given a Π-program of GMSC or a Π-formula of ω-GML φ (respectively, a formula $\psi(x)$ of MSO or FO over Π, where the only free variable is the first-order variable x), we say that φ (resp., $\psi(x)$) **expresses** a node property $\mathcal{P}$ over Π, if for each pointed Π-labeled graph (G, w): $(G, w) \in \mathcal{P}$ iff $G, w \models \varphi$ (or resp. $G \models \psi(w)$). A node property $\mathcal{P}$ over Π is **expressible** in GMSC (resp., in ω-GML, FO or MSO) if there is a Π-program of GMSC (resp., a Π-formula of ω-GML, FO or MSO) expressing $\mathcal{P}$.

Example 2.4. Recall the property *reachability of node label symbol* p over Π defined in Example 2.1. It is expressed by the Π-program of GMSC $X(0) :- p$, $X :- \Diamond X$, where X is an appointed predicate. The ith iteration formula of X is $X^i = \Diamond \cdots \Diamond p$ where there are exactly i diamonds.

Example 2.5 ([24]). A pointed Π-labeled graph (G, w) has the *centre-point property* $\mathcal{P}$ over Π if there exists an $n \in \mathbb{N}$ such that each directed path starting from w leads to a node with no out-neighbours in exactly n steps. It is easy to see that the Π-program $X(0) :- \Box\bot$, $X :- \Diamond X \wedge \Box X$, where X is an appointed predicate, expresses $\mathcal{P}$. In [24], it was established that the centre-point property is not expressible in MSO and that there are properties expressible in μ-calculus and MSO that are not expressible in MSC (e.g., non-reachability), with the same proofs applying to GMSC.

Proposition 2.6. *Properties expressible in* GMSC *are expressible in* ω-GML*, but not vice versa.*

Proof. A property over Π expressed by a Π-program Γ of GMSC is expressible in ω-GML by the Π-formula $\bigvee_{X_i \in \mathcal{A},\, n \in \mathbb{N}} X_i^n$, where X_i^n is the nth iteration formula of appointed predicate X_i of Γ. Like GNN[$\mathbb{R}$]s, ω-GML can express undecidable properties (cf. Remark 2.2). Clearly GMSC-programs Λ cannot, as configurations defined by Λ in a finite graph eventually loop, i.e., the truth values of iteration formulae start repeating cyclically. $\square$

While GMSC is related to the graded modal μ-calculus (μGML), which originates from [22] and is used in [26] to characterize a recurrent GNN model, μGML and GMSC are orthogonal in expressivity. Iteration in GMSC need *not* be over a monotone function and does not necessarily yield a fixed point, and there are no syntactic restrictions that would, e.g., force schema variables to be used only positively as in μGML. The centre-point property from Example 2.5 is a simple property not expressible in μGML (as it is not even expressible in MSO, into which μGML translates). Conversely, GMSC offers neither greatest fixed points nor fixed point alternation. In particular, natural properties expressible in the ν-fragment of μGML such as non-reachability are not expressible in GMSC; this is proved in [24] for the non-graded version, and the same proof applies to GMSC. However, the μ-fragment of the graded modal μ-calculus translates into GMSC (by essentially the same argument as the one justifying Proposition 7 in [24]). We also note that GMSC translates into partial fixed-point logic with choice [28], but it is not clear whether the same holds without choice.

3 Connecting GNNs and logics via automata

In this section we establish exact matches between classes of GNNs and our logics. The first main theorem is Theorem 3.2, showing that GNN[F]s, R-simple aggregate-combine GNN[F]s and GMSC are equally expressive. Theorem 3.4 is the second main result, showing that GNN[$\mathbb{R}$]s and ω-GML are equally expressive. We begin by defining the concept of distributed automata which we will mainly use as a tool for our arguments but they also lead to nice additional characterizations. Informally, we consider a model of distributed automata called counting message passing automata and its variants which operate similarly to GNNs. These distributed automata update the state of each node according to the node's own state and the *multiset* of states of its out-neighbours.

Formally, given $\Pi \subseteq \mathrm{LAB}$, a **counting message passing automaton** (CMPA) over Π is a tuple (Q, π, δ, F) where Q is an at most countable set of *states* and π, δ and F are defined in a similar way as for GNNs: $\pi \colon \mathcal{P}(\Pi) \to Q$ is an *initialization function*, $\delta \colon Q \times \mathcal{M}(Q) \to Q$ a *transition function* and $F \subseteq Q$ a set of *accepting states*. Computation of a CMPA over Π is defined in a Π-labeled graph G analogously to GNNs: for each node w in G, the initialization function gives the initial state for w based on the node label symbols true in w, and the transition function is applied to the previous state of w and the multiset of states of out-neighbours of w. Acceptance is similar to GNNs: the CMPA **accepts** (G, w) if the CMPA visits (at least once) an accepting state at w in G. A **bounded** CMPA is a CMPA whose transition function can be written as $\delta \colon Q \times \mathcal{M}_k(Q) \to Q$ for some $k \in \mathbb{N}$ (i.e., $\delta(q, M) = \delta(q, M_{|k})$ for each multiset $M \in \mathcal{M}(Q)$ and state $q \in Q$). A **finite** CMPA (FCMPA) is a CMPA with finite Q. We define bounded FCMPAs similarly to bounded CMPAs. A CMPA $\mathcal{A}$ over Π **expresses** a node property $\mathcal{P}$ over Π if $\mathcal{A}$ accepts (G, w) iff $(G, w) \in \mathcal{P}$. We define whether a node property $\mathcal{P}$ is expressible by a CMPA in a way analogous to GNNs.

For any Π, a Π-object refers to a GNN over Π, a CMPA over Π, a Π-formula of ω-GML or a Π-program of GMSC. Let $\mathcal{C}$ be the class of all Π-objects for all Π. Two Π-objects in $\mathcal{C}$ are **equivalent** if they express the same node property over Π. Subsets $A, B \subseteq \mathcal{C}$ **have the same expressive power**, if each $x \in A$ has an equivalent $y \in B$ and vice versa. It is easy to obtain the following.

Proposition 3.1. *Bounded* FCMPA*s have the same expressive power as* GMSC.

Proof. (Sketch) Details in Appendix B.2. To obtain a bounded FCMPA equivalent to a GMSC-program Λ, we first turn Λ into an equivalent program Γ where the modal depth of terminal clauses (resp., iteration clauses) is 0 (resp., at most 1). Then from Γ with head predicate set $\mathcal{T}'$, we construct an equivalent FCMPA $\mathcal{A}$ as follows. The set of states of $\mathcal{A}$ is $\mathcal{P}(\Pi \cup \mathcal{T}')$ and $\mathcal{A}$ enters in round $n \in \mathbb{N}$ in node w into a state that contains precisely the node label symbols true in w and the predicates X whose iteration formula X^n is true at w. For the converse, we create a head predicate for each state in $\mathcal{A}$, and let predicates for accepting states be appointed. The terminal clauses simulate π using disjunctions of conjunctions of non-negated and negated node label symbols. The iteration clauses simulate δ using, for each pair (q, q'), a subschema specifying the multisets that take q to q'. □

We are ready to show equiexpressivity of GMSC and GNN[F]s. This applies *without relativising to any background logic*. The direction from GNN[F]s to GMSC is trivial. The other direction is more challenging, in particular when going all the way to R-simple GNN[F]s. While size issues were not a concern in this work, we observe that the translation from GNN[F]s to GMSC involves only polynomial blow-up in size; the related definitions and proofs are in appendices A.1, B.2 and B.3. We also conjecture that a polynomial translation from GMSC to R-simple GNN[F]s is possible by the results and techniques in [3], taking into account differences between GNN[F]s and R-simple GNN[F]s w.r.t. the definition of size. A more serious examination of blow-ups would require a case-by-case analysis taking other such details into account.

Theorem 3.2. *The following have the same expressive power:* GNN[F]*s,* GMSC*, and R-simple aggregate-combine* GNN[F]*s.*

Proof. (Sketch) Details in Appendix B.3. By definition, a GNN[F] is just a bounded FCMPA and translates to a GMSC-program by Proposition 3.1. To construct an R-simple aggregate-combine GNN[F] $\mathcal{G}$ for a GMSC-program Λ with formula depth D, we first turn Λ into an equivalent program Γ, where each terminal clause has the body $\perp$, the formula depth of each body of an iteration clause is D' (linear in D) and for each subschema of Γ that is a conjunction, both conjuncts have the same formula depth if neither conjunct is $\top$. We choose a floating-point system that can express all integers up to the width of Γ. We define binary feature vectors that are split into two halves: the 1st half intuitively calculates the truth values of the head predicates and subschemata of Γ one formula depth at a time in the style of Barceló et al. [5]. The 2nd half records the current formula depth under evaluation. $\mathcal{G}$ simulates one round of Γ in $D'+1$ rounds, using the 2nd half of the features to accept nodes only every $D'+1$ rounds: the truth values of head predicates are correct in those rounds. Note that the choice of floating-point system in $\mathcal{G}$ depends on Λ and thus no single floating-point system is used by all GNNs resulting from the translation. In fact, fixing a single floating-point system would trivialize the computing model as only finitely many functions could be defined. □

To show that GNN[$\mathbb{R}$]s and ω-GML are equally expressive, we first prove a useful theorem.

Theorem 3.3. CMPA*s have the same expressive power as* ω-GML.

Proof. (Sketch) Details in Appendix B.4. To construct a CMPA for each ω-GML-formula χ, we define a GML-formula called the "*full graded type of modal depth* n" for each pointed graph (G, w), which expresses all the local information of the neighbourhood of w up to depth n (with the maximum out-degree of G plus 1 sufficing for width). We show that each ω-GML-formula χ is equivalent to an infinite disjunction ψ_χ of types. We then define a CMPA that computes the type of modal depth n of each node in round n. Its accepting states are the types in the type-disjunction ψ_χ. For the converse, to show that each CMPA has an equivalent ω-GML-formula, we first show that two pointed graphs satisfying the same full graded type of modal depth n have identical states in each round $\ell \leq n$ in each CMPA. The ω-GML-formula is the disjunction containing every type T such that some (G, w) satisfying T is accepted by the automaton in round n, where n is the depth of T. □

Next we characterize GNN[$\mathbb{R}$]s with ω-GML *without relativising to a background logic*. As the theorems above and below imply that GNN[$\mathbb{R}$]s and CMPAs are equally expressive, it follows that GNN[$\mathbb{R}$]s can be discretized to CMPAs having—by definition—an only countable set of states.

Theorem 3.4. GNN[$\mathbb{R}$]*s have the same expressive power as* ω-GML.

Proof. (Sketch) Details in Appendix B.5. For any GNN[$\mathbb{R}$], we build an equivalent ω-GML-formula using the same method as in the proof of Theorem 3.3, where we show that for each CMPA, we can find an equivalent ω-GML-formula. For the converse, we first translate an ω-GML-formula to a CMPA $\mathcal{A}$ by Theorem 3.3 and then translate $\mathcal{A}$ to an equivalent CMPA $\mathcal{A}'$ with maximal ability to distinguish nodes. Then we build an equivalent GNN[$\mathbb{R}$] for $\mathcal{A}'$ by encoding states into integers. □

Remark 3.5. It is easy to show that unrestricted GNN[F]s have the same expressive power as FCMPAs. Moreover, the proof of Theorem 3.4 is easily modified to show that bounded GNN[$\mathbb{R}$]s have the same expressive power as width-bounded ω-GML. See Appendix B.6 for the proofs.

In Appendix B.7 we show that our model of constant-iteration GNN[$\mathbb{R}$]s is equivalent to the one in Barceló et al. [5]. Thus Theorem 3.6 (proven in Appendix B.8) generalizes the result in [5] saying that any property $\mathcal{P}$ expressible by FO is expressible as a constant-iteration GNN[$\mathbb{R}$] iff it is expressible in GML. Furthermore, any such $\mathcal{P}$ is expressible in GML iff it is expressible in ω-GML.

Theorem 3.6. *Constant-iteration* GNN[$\mathbb{R}$]*s have the same expressive power as depth-bounded* ω-GML.

4 Characterizing GNNs over MSO-expressible properties

In this section we consider properties expressible in MSO. The first main result is Theorem 4.1, where we show that for properties expressible in MSO, the expressive power of GNN[$\mathbb{R}$]s is captured by a *finitary* logic. In fact, this logic is GMSC and by Theorem 3.2, it follows that relative to MSO, GNN[$\mathbb{R}$]s have the same expressive power as GNN[F]s (Theorem 4.3 below). Our arguments in this section work uniformly for any finite set Σ_N of node label symbols.

Theorem 4.1. *Let $\mathcal{P}$ be a property expressible in* MSO. *Then $\mathcal{P}$ is expressible as a* GNN[$\mathbb{R}$] *if and only if it is expressible in* GMSC.

Theorem 4.1 is proved later in this section. The recent work [26] relates to Theorem 4.1; see the introduction for a discussion. The proof of Theorem 4.1 can easily be adapted to show that a property $\mathcal{P}$ expressible in MSO is expressible as a constant-iteration GNN[$\mathbb{R}$] iff it is expressible in GML. This relates to [5] which shows the same for FO in place of MSO. However, in this particular case the MSO version is not an actual generalization as based on [10], we show the following in Appendix C.1.

Lemma 4.2. *Any property expressible in* MSO *and as a constant-iteration* GNN[$\mathbb{R}$] *is also* FO-*expressible.*

Uniting Theorems 4.1 and 3.2, we see that (recurrent) GNN[$\mathbb{R}$]s and GNN[F]s coincide relative to MSO. It is easy to get a similar result for constant-iteration GNNs (the details are in Appendix C.3).

Theorem 4.3. *Let $\mathcal{P}$ be a property expressible in* MSO. *Then $\mathcal{P}$ is expressible as a* GNN[$\mathbb{R}$] *if and only if it is expressible as a* GNN[F]. *The same is true for constant-iteration* GNN*s.*

To put Theorem 4.3 into perspective, we note that by Example 2.5, the centre-point property is expressible in GMSC, but not in MSO. From Theorem 3.2, we thus obtain the following.

Proposition 4.4. *There is a property $\mathcal{P}$ that is expressible as a* GNN[F] *but not in* MSO.

Theorem 3.6 shows that constant-iteration GNN[$\mathbb{R}$]s and depth-bounded ω-GML are equally expressive. The proof of Proposition 2.6 shows that already depth-bounded ω-GML can express properties that GMSC cannot, in particular undecidable ones. Thus, by Theorem 3.2 we obtain the following.

Proposition 4.5. *There is a property expressible as a constant-iteration* GNN[$\mathbb{R}$] *but not as a* GNN[F].

We next discuss the proof of Theorem 4.1. We build upon results due to Janin and Walukiewicz [31, 20], reusing an automaton model from [31] that captures the expressivity of MSO on tree-shaped graphs. With a tree-shaped graph we mean a *potentially infinite* graph that is a directed tree. The out-degree of nodes is unrestricted (but nevertheless finite), different nodes may have different degree, and both leaves and infinite paths are admitted in the same tree.

We next introduce the mentioned automaton model. Although we are only going to use it on tree-shaped Σ_N-labeled graphs, in its full generality it is actually defined on unrestricted such graphs. We nevertheless call them **tree automata** as they belong to the tradition of more classical forms of such automata. In particular, a run of an automaton is tree-shaped, even if the input graph is not. Formally, a **parity tree automaton (PTA)** is a tuple $\mathcal{A} = (Q, \Sigma_N, q_0, \Delta, \Omega)$, where Q is a finite set of **states**, $\Sigma_N \subseteq \mathrm{LAB}$ is a finite set of node label symbols, $q_0 \in Q$ is an **initial state**, $\Delta\colon Q \times \mathcal{P}(\Sigma_N) \to \mathcal{F}$ is a transition relation with $\mathcal{F}$ being the set of all **transition formulas** for $\mathcal{A}$ defined below, and $\Omega : Q \to \mathbb{N}$ is a **priority function**. A transition formula for $\mathcal{A}$ is a disjunction of FO-formulas of the form

$$\exists x_1 \cdots \exists x_k \left(\mathsf{diff}(x_1, \ldots, x_k) \wedge q_1(x_1) \wedge \cdots \wedge q_k(x_k) \wedge \forall z(\, \mathsf{diff}(z, x_1, \ldots, x_k) \to \psi)\right)$$

where $k \geq 0$, $\mathsf{diff}(y_1, \ldots, y_n)$ shortens an FO-formula declaring $y_1, \ldots, y_n$ as pairwise distinct, $q_i \in Q$ are states used as unary predicates and ψ is a disjunction of conjunctions of atoms $q(z)$, with $q \in Q$. A PTA $\mathcal{A}$ accepts a language $L(\mathcal{A})$ consisting of (possibly infinite) graphs G. We have $G \in L(\mathcal{A})$ if there is an accepting **run** of $\mathcal{A}$ on G, and runs are defined in the spirit of alternating automata. While details are in Appendix C.2, we mention that transition formulas govern transitions in the run: a transition of a PTA currently visiting node v in state q consists of sending copies of itself to out-neighbours of v, potentially in states other than q. It is not required that a copy is sent to every out-neighbour, and multiple copies (in different states) can be sent to the same out-neighbour. However, we must find some $(q, \lambda(v), \vartheta) \in \Delta$ such that the transition satisfies ϑ in the sense that ϑ is satisfied in the graph with one element for each out-neighbour of v and unary predicates (states of $\mathcal{A}$) are interpreted according to the transition. This specific form of PTAs is interesting to us due to the following.

Theorem 4.6. *Let $\mathcal{P}$ be a property expressible in* MSO *and in* ω-GML. *Then there is a PTA $\mathcal{A}$ such that for any graph G: $(G, w) \in \mathcal{P}$ iff the unraveling of G at w is in $L(\mathcal{A})$.*

Proof. Let $\varphi(x)$ be the MSO-formula over Σ_N that expresses $\mathcal{P}$. Theorem 9 in [20] states that for every MSO-sentence ψ, there is a PTA $\mathcal{A}$ such that for every tree-shaped Σ_N-labeled graph G we have that $G \in L(\mathcal{A})$ iff $G \models \psi$. To obtain a PTA for $\varphi(x)$, we start from the MSO-sentence $\psi := \exists x\,(\varphi(x) \wedge \neg\exists y E(y,x))$ and build the corresponding PTA $\mathcal{A}$. Then $\mathcal{A}$ is as desired. In fact, $(G,w) \in \mathcal{P}$ iff $G \models \varphi(w)$ iff $U \models \varphi(w)$ with U the unraveling of G at w since $\mathcal{P}$ is expressible by an ω-GML-formula which is invariant under unraveling (defined in Appendix C.1). Moreover, $U \models \varphi(w)$ iff $U \models \psi$ iff $U \in L(\mathcal{A})$. □

Since GMSC-programs are invariant under unraveling, we may now prove Theorem 4.1 by considering PTAs obtained by Theorem 4.6 and constructing a Σ_N-program Λ of GMSC for each such PTA $\mathcal{A} = (Q, \Sigma_N, q_0, \Delta, \Omega)$ so that the following holds: For every tree-shaped Σ_N-labeled graph G with root w, we have $G \in L(\mathcal{A})$ iff $G, w \models \Lambda$. We then say that $\mathcal{A}$ and Λ are **tree equivalent**.

For a state $q \in Q$, let $\mathcal{A}_q$ be the PTA defined like $\mathcal{A}$ but with q as its initial state. For a tree-shaped graph T, set $Q_T = \{q \in Q \mid T \in L(\mathcal{A}_q)\}$. We say that $\mathcal{Q} \subseteq \mathcal{P}(Q)$ is **the universal set for** $P \subseteq \Sigma_N$ if $\mathcal{Q}$ consists precisely of the sets Q_T, for all tree-shaped graphs T, whose root is labeled exactly with the node label symbols in P. Let $T = (V, E, \lambda)$ be a tree-shaped graph with root w and $k \in \mathbb{N}$, and let V_k be the restriction of V to the nodes on level at most k (the root being on level 0). A k**-prefix decoration** of T is a mapping $\mu : V_k \to \mathcal{P}(\mathcal{P}(Q))$ such that the following conditions hold: **(1)** for each $S \in \mu(w)$: $q_0 \in S$; **(2)** for all $v \in V$ on the level k, $\mu(v)$ is the universal set for $\lambda(v)$; **(3)** for each $v \in V$ on some level smaller than k that has out-neighbours $u_1, \ldots, u_n$ and all $S_1 \in \mu(u_1), \ldots, S_n \in \mu(u_n)$, $\mu(v)$ contains the set S that contains a state $q \in Q$ iff we have $\Delta(q, \lambda(v)) = \vartheta$ such that ϑ is satisfied in the following graph: the universe is $\{u_1, \ldots, u_n\}$ and each unary predicate $q' \in Q$ is interpreted as $\{u_i \mid q' \in S_i\}$. Intuitively, a k-prefix decoration of T represents a set of accepting runs of $\mathcal{A}$ on the prefix T_k of T. As these runs start in universal sets, for each extension of T_k obtained by attaching trees to nodes on level k, we can find a run among the represented ones that can be extended to an accepting run of $\mathcal{A}$ on that extension. In fact T is such an extension. The following crucial lemma is proved in Appendix C.2.

Lemma 4.7. *For every tree-shaped Σ_N-labeled graph T: $T \in L(\mathcal{A})$ if and only if there is a k-prefix decoration of T, for some $k \in \mathbb{N}$.*

Using the above, we sketch the proof of Theorem 4.1; the full proof is in Appendix C.2.

Proof of Theorem 4.1. By Lemma 4.7, given a PTA $\mathcal{A}$ obtained from Lemma 4.6, we get a tree equivalent GMSC-program Λ by building Λ to accept the root of a tree-shaped graph G iff G has a k-prefix decoration for some k. This requires care, but is possible; the details are in Appendix C.2. A crucial part of the proof is transferring the set of transition formulas of $\mathcal{A}$ into the rules of Λ. □

5 Conclusion

We have characterized the expressivity of recurrent GNNs with floats and reals in terms of modal logics, both in general and relative to MSO. Particularly, in restriction to MSO, GNNs with floats and with reals have the same expressiveness. We mention two interesting directions for future research. The first one is to extend our GNN model, e.g., with global readouts. On the GML side, these correspond to the (counting) global modality [7]. Interestingly, it can be proved—via showing that fixed points can be defined—that this extension makes non-reachability expressible, cf. Example 2.5. The second direction is to consider other acceptance conditions for recurrent GNNs. Intuitively, it should be possible to establish Theorem 3.2 for virtually any GNN acceptance condition, as long as the acceptance condition of GMSC is changed accordingly. Both of these directions are explored in [4]. In particular, it is shown there (and in Appendix D.1) that Theorem 3.2 also holds for recurrent GNNs where termination is based on fixed points, studied under the name "RecGNN" in [26]. These fixed points require only that a round is reached where the examined node does not exit the set of accepting states which is closely related to but different from the fixed point condition in the inaugural works [30, 11] which requires all nodes in the network to reach a stable state.

Limitations: For our acceptance condition (and for many others such as those based on fixed points), training and applying a recurrent GNN brings questions of termination. These are very important from a practical perspective, but not studied in this paper. A particularly interesting question is whether termination can be learned in a natural way during the training phase.

Acknowledgements

The list of authors on the first page is given based on the alphabetical order. Veeti Ahvonen was supported by the Vilho, Yrjö and Kalle Väisälä Foundation of the Finnish Academy of Science and Letters. Damian Heiman was supported by the Magnus Ehrnrooth Foundation. Antti Kuusisto was supported by the Research Council of Finland consortium project Explaining AI via Logic (XAILOG), grant number 345612, and the Research Council of Finland project Theory of computational logics, grant numbers 352419, 352420, 353027, 324435, 328987; Damian Heiman was also supported by the same project, grant number 353027. Carsten Lutz was supported by the Federal Ministry of Education and Research of Germany (BMBF) and by Sächsisches Staatsministerium für Wissenschaft, Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI. Lutz is also supported by BMBF in DAAD project 57616814 (SECAI, School of Embedded Composite AI) as part of the program Konrad Zuse Schools of Excellence in Artificial Intelligence.

References

[1] Veeti Ahvonen, Damian Heiman, Lauri Hella, and Antti Kuusisto. Descriptive complexity for distributed computing with circuits. In Jérôme Leroux, Sylvain Lombardy, and David Peleg, editors, *48th International Symposium on Mathematical Foundations of Computer Science, MFCS 2023, August 28 to September 1, 2023, Bordeaux, France*, volume 272 of *LIPIcs*, pages 9:1–9:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.MFCS.2023.9.

[2] Veeti Ahvonen, Damian Heiman, Lauri Hella, and Antti Kuusisto. Descriptive complexity for distributed computing with circuits, 2023. arXiv:2303.04735.

[3] Veeti Ahvonen, Damian Heiman, and Antti Kuusisto. Descriptive Complexity for Neural Networks via Boolean Networks. In Aniello Murano and Alexandra Silva, editors, *32nd EACSL Annual Conference on Computer Science Logic (CSL 2024)*, volume 288 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:22, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CSL.2024.9.

[4] Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. Logical characterizations of recurrent graph neural networks with reals and floats, 2024. arXiv:2405.14606.

[5] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[6] Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan. Decidability of graph neural networks via logical characterizations. *CoRR*, abs/2404.18151, 2024. arXiv:2404.18151.

[7] Patrick Blackburn, J. F. A. K. van Benthem, and Frank Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in logic and practical reasoning*. North-Holland, 2007.

[8] David Tena Cucala, Bernardo Cuenca Grau, Boris Motik, and Egor V. Kostylev. On the correspondence between monotonic max-sum GNNs and datalog. In Pierre Marquis, Tran Cao Son, and Gabriele Kern-Isberner, editors, *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pages 658–667, 2023. doi:10.24963/KR.2023/64.

[9] H.D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer New York, 1996.

[10] Michael Elberfeld, Martin Grohe, and Till Tantau. Where first-order and monadic second-order logic coincide. *ACM Trans. Comput. Log.*, 17(4):25, 2016. doi:10.1145/2946799.

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2:729–734 vol. 2, 2005.

[12] Martin Grohe. The logic of graph neural networks. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–17. IEEE, 2021. doi:10.1109/LICS52264.2021.9470677.

[13] Martin Grohe. The descriptive complexity of graph neural networks. In *LICS*, pages 1–14, 2023. doi:10.1109/LICS56636.2023.10175735.

[14] Erich Grädel, Wolfgang Thomas, and Thomas Wilke. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer Nature, Berlin, Heidelberg, 1 edition, 2003.

[15] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 1024–1034, 2017.

[16] Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. In *Proceedings of the 2012 ACM Symposium on Principles of distributed computing*, pages 185–194, 2012.

[17] Lauri Hella, Matti Järvisalo, Antti Kuusisto, Juhana Laurinharju, Tuomo Lempiäinen, Kerkko Luosto, Jukka Suomela, and Jonni Virtema. Weak models of distributed computing, with connections to modal logic. *Distributed Comput.*, 28(1):31–53, 2015.

[18] Nicholas J. Higham. The accuracy of floating point summation. *SIAM J. Sci. Comput.*, 14(4):783–799, 1993. doi:10.1137/0914050.

[19] Nicholas J. Higham. *Accuracy and stability of numerical algorithms, Second Edition*. SIAM, 2002. doi:10.1137/1.9780898718027.

[20] David Janin and Igor Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In Ugo Montanari and Vladimiro Sassone, editors, *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996. doi:10.1007/3-540-61604-7_60.

[21] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Syst. Appl.*, 207:117921, 2022. doi:10.1016/J.ESWA.2022.117921.

[22] Orna Kupferman, Ulrike Sattler, and Moshe Y. Vardi. The complexity of the graded μ-calculus. In Andrei Voronkov, editor, *Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings*, volume 2392 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 2002. doi:10.1007/3-540-45620-1_34.

[23] Orna Kupferman, Moshe Y. Vardi, and Pierre Wolper. An automata-theoretic approach to branching-time model checking. *J. ACM*, 47(2):312–360, mar 2000. doi:10.1145/333979.333987.

[24] Antti Kuusisto. Modal Logic and Distributed Message Passing Automata. In *Computer Science Logic 2013 (CSL 2013)*, volume 23 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 452–468, 2013.

[25] 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 *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 4602–4609. AAAI Press, 2019.

[26] Maximilian Pfluger, David Tena Cucala, and Egor V. Kostylev. Recurrent graph neural networks and their connections to bisimulation and logic. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(13):14608–14616, 2024. `doi:10.1609/aaai.v38i13.29377`.

[27] Fabian Reiter. Asynchronous distributed automata: A characterization of the modal mu-fragment. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 100:1–100:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. `doi:10.4230/LIPICS.ICALP.2017.100`.

[28] David Richerby. Logical characterizations of PSPACE. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 370–384. Springer, 2004. `doi:10.1007/978-3-540-30124-0\_29`.

[29] Thomas G. Robertazzi and Stuart C. Schwartz. Best "ordering" for floating-point addition. *ACM Trans. Math. Softw.*, 14(1):101–110, 1988. `doi:10.1145/42288.42343`.

[30] 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. `doi:10.1109/TNN.2008.2005605`.

[31] Igor Walukiewicz. Monadic second order logic on tree-like structures. In Claude Puech and Rüdiger Reischuk, editors, *STACS 96, 13th Annual Symposium on Theoretical Aspects of Computer Science, Grenoble, France, February 22-24, 1996, Proceedings*, volume 1046 of *Lecture Notes in Computer Science*, pages 401–413. Springer, 1996. `doi:10.1007/3-540-60922-9\_33`.

[32] James Hardy Wilkinson. Rounding errors in algebraic processes. In *Information Processing, Proceedings of the 1st International Conference on Information Processing, UNESCO, Paris 15-20 June 1959*, pages 44–53. UNESCO (Paris), 1959.

[33] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5):97:1–97:37, 2023. `doi:10.1145/3535101`.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1):4–24, 2021. `doi:10.1109/TNNLS.2020.2978386`.

[35] Zhang XM, Liang L, Liu L, and Tang MJ. Graph neural networks and their current applications in bioinformatics. *Frontiers in Genetics*, 21, 2021. `doi:10.3389/fgene.2021.690049`.

[36] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

A Appendix: Preliminaries

A.1 Notions of size

We start this appendix by presenting definitions for the sizes of GMSC-programs, GNN[F]s and bounded FCMPAs.

The **size** of a Π-program of GMSC is here defined as the size of Π plus the number of occurrences of node label symbols, head predicates and logical operators in the program, with each $\Diamond_{\geq k}$ adding k to the sum instead of just 1.

The **size** of a GNN[F] or bounded FCMPA over Π is defined here as follows. For FCMPAs define $U := Q$, and for GNN[F]s define $U := S^d$ where S is the floating-point system and d the dimension of the GNN[F]. In both cases, let $n := |U|$ and let k be the bound. Note that the cardinality of a multiset $\mathcal{M}_k(U)$ is the number $(k+1)^n$ of different functions of type $U \to [0; k]$. Then the size of the GNN[F] or bounded FCMPA is $|\mathcal{P}(\Pi)|$ plus $n \cdot (k+1)^n$, i.e., the sum of the cardinality $|\mathcal{P}(\Pi)|$ of its initialization function $\mathcal{P}(\Pi) \to U$ and the cardinality $n \cdot (k+1)^n$ of its transition function $U \times \mathcal{M}_k(U) \to U$ as look-up tables.

We note that there is no single way to define the sizes of the objects above. For example, counting each node label symbol, head predicate or state into the size just once may be naïve, as there may not be enough distinct symbols in practice. To account for this, we could also define that the size of each node label symbol, head predicate and state is instead $\log(k)$, where k is the number of node label symbols, head predicates or states respectively. Likewise, the size of a diamond $\Diamond_{\geq \ell}$ could be considered $\log(k)$, where k is the maximum ℓ appearing in a diamond in the program. However, this happens to not affect the size of our translation from GNN[F]s to GMSC. As another example, defining the size of a single floating-point number as one may be naïve; one could define that the size of a single floating-point number is the length of the string that represents it in the floating-point system instead of one.

A.2 Proof of Proposition 2.3

We recall Proposition 2.3 and give a more detailed proof.

Proposition 2.3. *For all floating-point systems S, there exists a $k \in \mathbb{N}$ such that for all multisets M over floats in S, we have $\mathrm{SUM}_S(M) = \mathrm{SUM}_S(M_{|k})$.*

Proof. Consider a floating-point system $S = ((p, n, \beta), +, \cdot)$ and a multiset $M \in \mathcal{M}(S)$. We assume $f = 0.d_1 \cdots d_p \times \beta^e \in S$ (the case where $f = -0.d_1 \cdots d_p \times \beta^e$ is symmetric). Assume that the sum of all numbers smaller than f in M in increasing order is $s \in S$. Let $s + f^0$ denote s and let $s + f^{\ell+1}$ denote $(s + f^\ell) + f$ for all $\ell \in \mathbb{N}$. It is clear that $s + f^{\ell+1} \geq s + f^\ell$, and furthermore, if $s + f^{\ell+1} = s + f^\ell$, then $s + f^{\ell'} = s + f^\ell$ for all $\ell' \geq \ell$. If $s + f^{\ell+1} = s + f^\ell$, then we say that the sum $s + f^\ell$ has *stabilized*.

Let $\beta' := \beta - 1$. We may assume that $s \geq -0.\beta' \cdots \beta' \times \beta^n$. Since $s + f^{\ell+1} \geq s + f^\ell$ for each $\ell \in \mathbb{N}$ and S is finite, there must exist some $k \in \mathbb{Z}_+$ such that either $s + f^k = s + f^{k-1}$ or $s + f^k = 0.\beta' \cdots \beta' \times \beta^n$ but then $s + f^{k+1} = s + f^k$. Clearly, $k = |S|$ is sufficient for each s and f, and thus satisfies the condition of the proposition.

A smaller bound can be found as follows. Assume without loss of generality that $d_1 \neq 0$ (almost all floating-point numbers in S can be represented in this form and the representation does not affect the outcome of a sum; we consider the case for floats not representable in this way separately). Now, consider the case where $s = -0.\beta' \cdots \beta' \times \beta^{e'}$ for some $e' \in [-n, n]$, as some choice of e' clearly maximizes the number of times f can be added. Now it is quite easy to see that that $s + f^{\beta^p - \beta^{p-1}} \geq -0.10 \cdots 0 \times \beta^{e'}$, unless the sum stabilizes sooner. This is because there are exactly $\beta^p - \beta^{p-1}$ numbers in S from $-0.\beta' \cdots \beta' \times \beta^{e'}$ to $-0.10 \cdots 0 \times \beta^{e'}$. After this another $2\beta^{p-1}$ additions ensures that $s + f^{\beta^p + \beta^{p-1}} \geq 0$. (Consider the case where $p = 3$ and $\beta = 10$ and we add the number $f = 0.501 \times 10^0$ to -0.100×10^3 repeatedly. We first get -0.995×10^2, then -0.990×10^2, etc. Each two additions of f is more than adding $f' = 0.001 \times 10^3$ once, and f' can be added exactly β^{p-1} times to -0.100×10^3 until the sum surpasses 0, meaning that f can be added at most $2\beta^{p-1}$ times. On the other hand, if f were 0.500×10^0 or smaller, then the sum would have never reached -0.100×10^3 in the first place.)

Let $\ell := \beta^p + \beta^{p-1}$. Let $s' := s + f^{\ell}$ and assume the sum s' has not yet stabilized. Next, it is clear that for any $i \in [p]$ we have $s' + f^{\beta^i} \geq 0.10\cdots0 \times \beta^{e+i}$ because $d_1 \geq 1$. For $d_1 \leq \frac{\beta}{2}$, we have that $s' + f^{\beta^p} = 0.10\cdots0 \times \beta^{e+p}$ which is where the sum stabilizes. If on the other hand $d_1 > \frac{\beta}{2}$, we have that $s' + f^{\beta^{p+1}} = 0.10\cdots0 \times \beta^{e+p+1}$, which is where the sum stabilizes (this is because $d_1 > \frac{\beta}{2}$ causes the sum to round upwards until the exponent reaches the value $e+p+1$). Thus, we get a threshold of $\beta^{p+1} + \beta^p + \beta^{p-1}$.

The case where $d_1 = 0$ for all representations of f is analogous; we simply choose one representation and shift our focus to the first i such that $d_i \neq 0$ and perform the same examination. The position of i does not affect the overall analysis.

We conjecture that a smaller bound than $\beta^{p+1} + \beta^p + \beta^{p-1}$ is possible, but involves a closer analysis. □

B Appendix: Connecting GNNs and logics via automata

B.1 An extended definition for distributed automata

We give a more detailed definition for distributed automata. Given $\Pi \subseteq \mathrm{LAB}$, a **counting message passing automaton** (or CMPA) over Π is a tuple (Q, π, δ, F), where

- Q is an at most countable non-empty set of **states**,
- $\pi \colon \mathcal{P}(\Pi) \to Q$ is an **initialization function**,
- $\delta \colon Q \times \mathcal{M}(Q) \to Q$ is a **transition function**, and
- $F \subseteq Q$ is a set of **accepting states**.

If the set of states is finite, then we say that the counting message passing automaton is finite and call it an FCMPA. Note that in the context of FCMPAs, finiteness refers specifically to the number of states. We also define a subclass of counting message passing automata: a **k-counting message passing automaton** (k-CMPA) is a tuple (Q, π, δ, F), where Q, π and F are defined analogously to CMPAs and the transition function δ can be written in the form $Q \times \mathcal{M}_k(Q) \to Q$, i.e., for all multisets $M \in \mathcal{M}(Q)$ we have $\delta(q, M) = \delta(q, M_{|k})$ for all $q \in Q$. A k-FCMPA naturally refers to a k-CMPA whose set of states is finite; this is truly a finite automaton. For any $k \in \mathbb{N}$, each k-CMPA (respectively, each k-FCMPA) is called a **bounded** CMPA (resp., a **bounded** FCMPA).

Let $G = (V, E, \lambda)$ be a Π-labeled graph. We define the computation of a CMPA formally. A CMPA (or resp. k-CMPA) (Q, π, δ, F) over Π and a Π-labeled graph $G = (V, E, \lambda)$ define a distributed system, which executes in ω-rounds as follows. Each **round** $n \in \mathbb{N}$ defines a **global configuration** $g_n \colon V \to Q$ which essentially tells in which state each node is in round n. If $n = 0$ and $w \in V$, then $g_0(w) = \pi(\lambda(w))$. Now assume that we have defined g_n. Informally, a new state for w in round $n+1$ is computed by δ based on the previous state of w and the multiset of previous states of its immediate out-neighbours. More formally, let w be a node and $v_1, \ldots, v_m$ its immediate out-neighbours (i.e., the nodes $v \in V$ s.t. $(w, v) \in E$). Let S denote the corresponding multiset of the states $g_n(v_1), \ldots, g_n(v_m)$. We define $g_{n+1}(w) = \delta(g_n(w), S)$. We say that $g_n(w)$ is the **state of w at round n**.

A CMPA (or resp. k-CMPA) **accepts** a pointed graph (G, w) if and only if $g_n(w) \in F$ for some $n \in \mathbb{N}$. If $g_n(w) \in F$, then we say that the CMPA **accepts** (G, w) **in round** n. The computation for FCMPAs is defined analogously (since each FCMPA is a CMPA).

B.2 Proof of Proposition 3.1

Note that extended definitions for automata are in Appendix B.1.

First we recall Proposition 3.1.

Proposition 3.1. *Bounded* FCMPA*s have the same expressive power as* GMSC.

To prove Proposition 3.1 (in the end of this subsection), we first prove an auxiliary Lemma B.1 that informally shows that we can translate any GMSC-program into an equivalent program, where the modal depth of the terminal clauses is zero and the modal depth of the iteration clauses is at most one. Then in Lemma B.2 we show with the help of Lemma B.1 that for each GMSC-program we can construct an equivalent bounded FCMPA. Finally, we show in Lemma B.3 that for each bounded FCMPA we can construct an equivalent GMSC-program.

We start with an auxiliary result.

Lemma B.1. *For every* Π*-program of* GMSC*, we can construct an equivalent* Π*-program* Γ *of* GMSC *such that the modal depth of each terminal clause of* Γ *is* 0*, and the modal depth of each iteration clause of* Γ *is at most* 1.

Proof. The proof is similar to the proof of Theorem 11 in [1] (or the proof of Theorem 5.4 in [2]). We consider an example and conclude that the strategy mentioned in the cited papers can be generalized to our framework.

For simplicity, in the example below we consider a program where the modal depth of each terminal clause is zero, since it is easy to translate any program of GMSC into an equivalent GMSC-program, where the modal depth of the terminal clauses is zero. Then from the program of GMSC, where the modal depth of each terminal clause is zero, it is easy to obtain an equivalent GMSC-program where the modal depth of each terminal clause is zero and the modal depth of each iteration clause is at most one.

Consider a $\{q\}$-program Λ, with the rules $X(0) :- \bot$, $X :- \Diamond_{\geq 3}(\neg\Diamond_{\geq 2}\Diamond_{\geq 1}X \wedge \Diamond_{\geq 3}q)$. The modal depth of the iteration clause of Λ is 3. The program Λ can be translated into an equivalent program of GMSC where the modal depth of the iteration clauses is at most one as follows. First, we define the following subprogram called a "clock":

$$\begin{array}{ll} T_1(0) :- \top & T_1 :- T_3 \\ T_2(0) :- \bot & T_2 :- T_1 \\ T_3(0) :- \bot & T_3 :- T_2. \end{array}$$

We split the evaluation of each subformula between corresponding head predicates $X_{1,1}$, $X_{1,2}$, X_2 and X_3 and define their terminal clauses and iteration clauses as follows. The body for each terminal clause is $\bot$ and the iteration clauses are defined by

- $X_{1,1} :- (T_1 \wedge \Diamond_{\geq 1}X_3) \vee (\neg T_1 \wedge X_{1,1})$,
- $X_{1,2} :- (T_1 \wedge \Diamond_{\geq 3}q) \vee (\neg T_1 \wedge X_{1,2})$,
- $X_2 :- (T_2 \wedge \Diamond_{\geq 2}X_{1,1}) \vee (\neg T_2 \wedge X_2)$,
- $X_3 :- (T_3 \wedge \Diamond_{\geq 3}(\neg X_2 \wedge X_{1,1})) \vee (\neg T_3 \wedge X_3)$.

The appointed predicate of the program is X_3.

Let us analyze how the obtained program works. The program works in a periodic fashion: a single iteration round of Λ is simulated in a 3 step period by the obtained program. The clock (i.e. the head predicates T_1, T_2 and T_3) makes sure that each level of the modal depth is evaluated once during each period in the correct order. For example, if T_2 is true, then the truth of X_2 depends on the truth of $\Diamond_{\geq 2}X_{1,1}$ and when T_2 is false, then the truth of X_2 stays the same. The head predicate X_3 essentially simulates the truth of X in Λ in the last round of each period. It is easy to show by induction on $n \in \mathbb{N}$ that for all pointed $\{q\}$-labeled graphs (G, w) and for all $k \in [0; 2]$: $G, w \models X^n$ iff $G, w \models X_3^{n3+k}$. $\square$

With Lemma B.1, it is easy to construct an equivalent bounded FCMPA for any GMSC-program.

Lemma B.2. *For each* Π*-program of* GMSC*, we can construct an equivalent bounded* FCMPA *over* Π.

Proof. Let Λ be a $(\Pi, \mathcal{T})$-program of GMSC. By Lemma B.1 we obtain an equivalent $(\Pi, \mathcal{T}')$-program Γ where the modal depth is 0 for each terminal clause and at most 1 for each iteration clause. Let k be the width of Γ. We create an equivalent k-FCMPA $\mathcal{A}_\Gamma$ as follows.

We create a state for each subset $q \subseteq \Pi \cup \mathcal{T}'$; these form the state set $Q = \mathcal{P}(\Pi \cup \mathcal{T}')$.

We formulate the initialization function π as follows: $\pi(P) = q$ if $\Pi \cap q = P$ and for each head predicate $X \in \mathcal{T}'$ (with the terminal clause $X(0) :- \varphi$) we have that $X \in q$ if and only if φ is true when exactly the node label symbols in P are true. (Here it is vital that the terminal clauses have modal depth 0.)

Before we define the transition function, we define an auxiliary relation $\Vdash$ for schemata with modal depth at most 1. Let $N \in \mathcal{M}(Q)$ be a multiset of states and let q be a state. Given a $(\Pi, \mathcal{T})$-schema ψ with modal depth at most 1, we define the relation $(q, N) \Vdash \psi$ recursively as follows: $(q, N) \Vdash \top$ always, $(q, N) \Vdash p$ iff $p \in q$, $(q, N) \Vdash X$ iff $X \in q$, $(q, N) \Vdash \neg\varphi$ iff $(q, N) \nVdash \varphi$, $(q, N) \Vdash \varphi \wedge \theta$ iff $(q, N) \Vdash \varphi$ and $(q, N) \Vdash \theta$, and $(q, N) \Vdash \Diamond_{\geq \ell}\varphi$ iff there is a set $Q' \subseteq Q$ of states such that $\sum_{q' \in Q'} N(q') \geq \ell$ and $(q', \emptyset) \Vdash \varphi$ for all $q' \in Q'$ (note that φ has modal depth 0).

Now we shall formulate the transition function δ as follows. Let $N \in \mathcal{M}(Q)$ be a multiset of states and let q be a state. For each node label symbol $p \in \Pi$, $p \in \delta(q, N)$ iff $(q, N) \Vdash p$ (i.e. $p \in q$). For each head predicate $X \in \mathcal{T}'$ with the iteration clause $X :- \psi$ we have that $X \in \delta(q, N)$ if and only if $(q, N) \Vdash \psi$. (Here it is crucial that the iteration clauses have modal depth at most 1.)

The set of accepting states is defined as follows. If $X \in q$ for some appointed predicate X of Γ then $q \in F$; otherwise $q \notin F$.

If G is Π-labeled graph and v is a node in G, then we let v^n denote the state of $\mathcal{A}_\Gamma$ at v in round n. Moreover, if N is the set of out-neighbours of v, we let N_v^n denote the multiset $\{\{u^n \mid u \in N\}\}$. Then we prove by induction on $n \in \mathbb{N}$ that for any $(\Pi, \mathcal{T}')$-schema φ of modal depth 1 and for each pointed Π-labeled graph (G, w), we have $(w^n, N_w^n) \Vdash \varphi$ iff $G, w \models \varphi^n$.

If $n = 0$ we prove by induction on the structure of φ that $(w^0, N_w^0) \Vdash \varphi$ iff $G, w \models \varphi^0$.

- Case $\varphi = p \in \Pi$: Trivial, since $(w^0, N_w^0) \Vdash p$ iff $p \in w^0$ iff $G, w \models p$ iff $G, w \models p^0$.
- Case $\varphi = X \in \mathcal{T}'$: Let ψ_X be the body of the terminal clause of X. Now, $(w^0, N_w^0) \Vdash X$ iff $X \in w^0$ iff $G, w \models \psi_X$ iff $G, w \models X^0$.

Now, assume that the induction hypothesis holds for $(\Pi, \mathcal{T}')$-schemata ψ and θ with modal depth at most 1.

- Case $\varphi := \neg\psi$: Now, $(w^0, N_w^0) \Vdash \neg\psi$ iff $(w^0, N_w^0) \nVdash \psi$. By the induction hypothesis, this is equivalent to $G, w \not\models \psi^0$ which is equivalent to $G, w \models \neg\psi^0$.
- Case $\varphi := \psi \wedge \theta$: Now, $(w^0, N_w^0) \Vdash \psi \wedge \theta$ iff $(w^0, N_w^0) \Vdash \psi$ and $(w^0, N_w^0) \Vdash \theta$. By the induction hypothesis, this is equivalent to $G, w \models \psi^0$ and $G, w \models \theta^0$, which is equivalent to $G, w \models \psi^0 \wedge \theta^0$.
- Case $\varphi := \Diamond_{\geq \ell}\psi$: First it is easy to show that for every $(\Pi, \mathcal{T}')$-schema ψ of modal depth 0 and for every state q of $\mathcal{A}_\Gamma$ that $(q, \emptyset) \Vdash \psi$ iff $(q, N) \Vdash \psi$ for every multiset N of states of $\mathcal{A}_\Gamma$. Now, $(w^0, N_w^0) \Vdash \Diamond_{\geq \ell}\psi$ iff there is a set $Q' \subseteq Q$ of states such that $\sum_{q' \in Q'} N_w^0(q') \geq \ell$ and $(q', \emptyset) \Vdash \psi$ for every $q' \in Q'$ iff there is a set N of out-neighbours of w such that $|N| \geq \ell$ and $(u^0, N_u^0) \Vdash \psi$ for every $u \in N$. By the induction hypothesis this is true iff there is a set N of out-neighbours of w such that $|N| \geq \ell$ and $G, u \models \psi^0$ for every $u \in N$, which is equivalent to $G, w \models \Diamond_{\geq \ell}\psi^0$.

Now, assume that the induction hypothesis holds for n and let us prove the case for $n+1$. Similarly to the case $n = 0$, we prove by induction on structure of φ that $(w^{n+1}, N_w^{n+1}) \Vdash \varphi$ iff $G, w \models \varphi^{n+1}$. The proof by induction is almost identical, except in the case where $\varphi = X \in \mathcal{T}'$. Let χ_X be the body of the iteration clause of X. Then $(w^{n+1}, N_w^{n+1}) \Vdash X$ iff $X \in w^{n+1}$ iff $(w^n, N_w^n) \Vdash \chi_X$. By the induction hypothesis, this is equivalent to $G, w \models \chi_X^n$, which is equivalent to $G, w \models X^{n+1}$.

Thus, the result above implies that for all pointed Π-labeled graphs (G, w), the automaton $\mathcal{A}_\Gamma$ is in the state q at w in round n iff $G, w \models X^n$ and $G, w \models p$ for every $X, p \in q$, and $G, w \not\models X^n$ and $G, w \not\models p$ for every $p, X \notin q$. Therefore, by the definition of the set of accepting states we see that $\mathcal{A}_\Gamma$ is equivalent to Γ. □

The converse direction is easier. Note that the definitions for sizes of GMSC-programs and bounded FCMPAs can be found in Appendix A.1.

Lemma B.3. *For each bounded* FCMPA *over* Π*, we can construct an equivalent* Π*-program of* GMSC*. The size of the constructed* GMSC*-program is polynomial in the size of the* FCMPA*.*

Proof. The proof below is analogous to the proof of Theorem 1 in [24]. Let Π be a set of node label symbols, and let $\mathcal{A} = (Q, \pi, \delta, F)$ be a k-FCMPA over Π. For each state $q \in Q$, we define a head predicate X_q and the corresponding rules as follows. The terminal clause for X_q is defined by

$$X_q(0) \coloneq \bigvee_{P \subseteq \Pi,\, \pi(P)=q} \Big(\bigwedge_{p \in P} p \wedge \bigwedge_{p \in \Pi \setminus P} \neg p \Big).$$

To define the iteration clause for each X_q we first define some auxiliary formulae. Given $q, q' \in Q$, we let $M_k(q, q') = \{ S \in \mathcal{M}_k(Q) \mid \delta(q, S) = q' \}$. Notice that the number of multisets S specified by the set $M_k(q, q')$ is finite. Now, for each $S \in M_k(q, q')$ we define

$$\varphi_S \coloneqq \bigwedge_{q \in Q,\, S(q)=n,\, n<k} \Diamond_{=n} X_q \wedge \bigwedge_{q \in Q,\, S(q)=k} \Diamond_{\geq k} X_q.$$

Then the iteration clause for X_q is defined by $X_q \coloneq \bigwedge_{q' \in Q} \Big(X_{q'} \to \bigvee_{S \in M_k(q',q)} \varphi_S \Big)$. It is easy to show by induction on $n \in \mathbb{N}$ that for every pointed Π-labeled graph (G, w) it holds that $G, w \models X_q^n$ if and only if w is in the state q in round n (see [24] for the details; the only difference is swapping sets for multisets).

The size of $\mathcal{A}$ is $|\mathcal{P}(\Pi)| + |Q|(k+1)^{|Q|}$ by definition. The size of all terminal clauses of the constructed program is altogether $|Q| + \mathcal{O}(|\Pi| \cdot |\mathcal{P}(\Pi)|)$, as there are $|Q|$ terminal clauses that altogether encode each element of $\mathcal{P}(\Pi)$ and each encoding is of size $\mathcal{O}(|\Pi|)$. The size of all iteration clauses of the constructed program is altogether $\mathcal{O}(|Q|^2 + k|Q|^2(k+1)^{|Q|}) = \mathcal{O}(k|Q|^2(k+1)^{|Q|})$, as there are $|Q|$ iteration clauses with $|Q|$ conjuncts each, and they altogether encode every element of $\mathcal{M}_k(Q)$ exactly $|Q|$ times and each encoding is of size $\mathcal{O}(k|Q|)$. Note that the cardinality of $\mathcal{M}_k(Q)$ is $(k+1)^{|Q|}$, i.e., the number of functions of type $Q \to [0; k]$. Thus, the size of the program is

$$|\Pi| + |Q| + \mathcal{O}(|\Pi| \cdot |\mathcal{P}(\Pi)|) + \mathcal{O}(k|Q|^2(k+1)^{|Q|}) = \mathcal{O}(|\Pi| \cdot |\mathcal{P}(\Pi)| + k|Q|^2(k+1)^{|Q|})$$

which is clearly less than $\mathcal{O}((|\mathcal{P}(\Pi)| + |Q|(k+1)^{|Q|})^2)$ and therefore polynomial in the size of $\mathcal{A}$. If head predicates and states were encoded in binary, the result would still be polynomial, as it would simply add a factor of $\log(|\Pi|)$ and $\log(|Q|)$ respectively to the sizes of terminal and iteration clauses. □

Proof of Proposition 3.1. Note that the proof uses auxiliary results that are introduced in this subsection. Lemma B.2 shows that for every Π-program of GMSC we can construct an equivalent bounded FCMPA over Π. Lemma B.3 shows that for every bounded FCMPA over Π we can construct an equivalent Π-program of GMSC. Thus, we conclude that GMSC has the same expressive power as bounded FCMPAs. □

B.3 Proof of Theorem 3.2

First we recall Theorem 3.2.

Theorem 3.2. *The following have the same expressive power:* GNN[F]*s,* GMSC*, and R-simple aggregate-combine* GNN[F]*s.*

In the end of this appendix section, we formally prove Theorem 3.2. Informally, this is done in the following steps. First, we note that it is easy to obtain an equivalent bounded FCMPA for a given GNN[F] and by Proposition 3.1 we can translate the bounded FCMPA into an equivalent GMSC-program. The converse direction is much more interesting. In Lemma B.5 we show that for each GMSC-program we can construct an equivalent R-simple aggregate-combine GNN[F].

Before we show how to obtain an equivalent R-simple GNN[F] for each GMSC-program, we first want to modify the programs so that they are easier to handle via R-simple GNN[F]s. In particular, the terminal clauses of a program may involve diamonds, the rules of a program may have differing amounts of nested logical operators, and so may the two conjuncts of each conjunction. Thus we prove a lemma which intuitively shows how to "balance" GMSC-programs such that this is not the case. We let $\mathrm{fdepth}(\varphi)$ denote the formula depth of a schema φ (see the definition of formula depth in Section 2.2). The formula depth of a terminal clause (resp., of an iteration clause) refers to the formula depth of the body of the clause. The formula depth of a GMSC-program is the maximum formula depth of its clauses.

Lemma B.4. *For each Π-program Λ of* GMSC *with formula depth D, we can construct an equivalent Π-program Γ of* GMSC *which has the following properties.*

1. *Each terminal clause is of the form $X(0) :- \bot$.*
2. *Each iteration clause has the same formula depth $\max(3, D+2)$.*
3. *If $\varphi \wedge \theta$ is a subschema of Λ such that neither φ nor θ is $\top$, then φ and θ have the same formula depth.*

Proof. First we define a fresh auxiliary predicate I with the rules $I(0) :- \bot$ and $I :- \top$. Informally, we will modify the rules of Λ such that the terminal clauses are simulated by the iteration clauses with the help of I. Then for each head predicate of Λ with terminal clause $X(0) :- \varphi$ and iteration clause $X :- \psi$ we define new rules in Γ as follows: $X(0) :- \bot$ and $X :- (\neg I \wedge \varphi) \vee (I \wedge \psi)$. Now, the formula depth of Γ is $\max(3, D+2)$.

First, we "balance" the formula depth of each iteration clause in Γ according to the second point in the statement of the lemma. Let d be the formula depth of an iteration clause $X :- \psi$, let $D' = \max(3, D+2)$ and let $n = D' - d$. If n is even, then the new iteration clause for X is $X :- (\neg)^n \psi$, where $(\neg)^n = \neg \cdots \neg$ denotes n nested negations. If n is odd, then the new iteration clause for X is $X :- (\neg)^{n-1}(\psi \wedge \top)$. In either case, the new iteration clause for X has formula depth D'.

Then we show how to "balance" each subschema of Γ according to the third point in the statement of the lemma as follows. Let $\varphi \wedge \psi$ be a subschema of Γ such that neither φ nor ψ is $\top$, and w.l.o.g. assume that $\mathrm{fdepth}(\varphi) < \mathrm{fdepth}(\psi)$ and let $n = \mathrm{fdepth}(\psi) - \mathrm{fdepth}(\varphi)$. If n is even, then we replace $\varphi \wedge \psi$ in Γ with $(\neg)^n \varphi \wedge \psi$. If n is odd, then we replace $\varphi \wedge \psi$ in Γ with $(\neg)^{n-1}(\varphi \wedge \top) \wedge \psi$.

We have now obtained the desired Γ. It is easy to see that Γ is equivalent to Λ as follows. If X is a head predicate that appears in both Λ and Γ, then for each $n \in \mathbb{N}$ and for each pointed Π-labeled graph, we have $G, w \models X^n$ w.r.t. Λ iff $G, w \models X^{n+1}$ w.r.t. Γ. In Γ, the auxiliary predicate I makes sure that each head predicate X in Γ that appears in Λ simulates in round 1 the corresponding terminal clause of X in Λ and in subsequent rounds simulates the corresponding iteration clause of X in Λ. Furthermore, for each head predicate X in Γ and for each pointed Π-labeled graph (G, w), we have $G, w \not\models X^0$ w.r.t. Γ. Thus, Γ and Λ are equivalent. $\square$

Now we are ready to show the translation from GMSC to R-simple aggregate-combine GNN[F]s.

Lemma B.5. *For each Π-program of* GMSC *we can construct an equivalent R-simple aggregate-combine* GNN[F] *over* Π.

Proof. Let Λ be a Π-program of GMSC. Informally, an equivalent R-simple aggregate-combine GNN[F] for Λ is constructed as follows. First from Λ we construct an equivalent Π-program Γ of GMSC with Lemma B.4. Let D be the formula depth of Γ.

Intuitively, we build an R-simple aggregate-combine GNN[F] $\mathcal{G}_\Gamma$ for Γ which periodically computes a single iteration round of Γ in $D+1$ rounds. The feature vectors used by $\mathcal{G}_\Gamma$ are *binary* vectors $\mathbf{v} = \mathbf{u}\mathbf{w}$ where:

- $\mathbf{u}$ has one bit per each (distinct) subschema of a body of an iteration clause in Γ as well as for each head predicate, and $\mathbf{v}_1$ keeps track of their truth values, and
- $\mathbf{w}$ has $D+1$ bits and it keeps track of the formula depth that is currently being evaluated.

Therefore, $\mathcal{G}_\Gamma$ is a GNN[F] over $(\Pi, N+D+1)$, where N is the number of (distinct) subschemata of the bodies of the iteration clauses of Γ as well as head predicates, and D is the maximum formula depth of the bodies of the iteration clauses. In order to be able to compute values, the floating-point system for $\mathcal{G}_\Gamma$ is chosen to be high enough. More precisely, we choose a floating-point system S which express all integers from 0 at least up to $K_{\max}$, where $K_{\max}$ is the width of Γ. Note that 0s and 1s are also represented in the floating-point system S. Note that although feature vectors exist that have elements other than 1s and 0s, they are not used.

Next we define the functions π and COM, and the set F of accepting states for $\mathcal{G}_\Gamma$. (Note that AGG is just the sum in increasing order.) We assume an enumeration $\mathrm{SUB}(\Gamma) := (\varphi_1, \ldots, \varphi_N)$ of subschemata and head predicates in Γ such that if φ_k is a subschema of φ_ℓ, then $k \leq \ell$. The initialization function π of $\mathcal{G}_\Gamma$ with input $P \subseteq \Pi$ outputs a feature vector $\mathbf{v} \in \{0,1\}^{N+D+1}$, where the value of each component corresponding to a node label symbol is defined as follows: the component for p is 1 iff $p \in P$. The other components are 0s (excluding the possible subschema $\top$ which is assigned 1, as well as the very last $(N+D+1)$th bit which is also assigned 1).

Recall that S is the floating-point system for $\mathcal{G}_\Gamma$. The combination function (as per the definition of R-simple GNNs) is $\mathrm{COM}(\mathbf{x}, \mathbf{y}) = \sigma(\mathbf{x} \cdot C + \mathbf{y} \cdot A + \mathbf{b})$ where σ is the truncated ReLU (ReLU^*) defined by $\mathrm{ReLU}^*(x) = \min(\max(0,x),1)$, $\mathbf{b} \in S^{N+D+1}$ and $C, A \in S^{(N+D+1)\times(N+D+1)}$, where $\mathbf{b}, C$ and A are defined as follows. For $k, \ell \leq N+D+1$, we let $C_{k,\ell}$ (resp. $A_{k,\ell}$) denote the element of C (resp. A) at the kth row and ℓth column. Similarly for $\ell \leq N+D+1$ and for any vector $\mathbf{v} \in S^{N+D+1}$ including $\mathbf{b}$, we let $\mathbf{v}_\ell$ denote the ℓth value of $\mathbf{v}$. Now, we define the top-left $N \times N$ submatrices of C and A and the first N elements of $\mathbf{b}$ in the same way as Barceló et al. [5]. For all $\ell \leq N$ we define as follows.

- If $\varphi_\ell \in \Pi \cup \{\top\}$, then $C_{\ell,\ell} = 1$.
- If φ_ℓ is a head predicate X with the iteration clause $X :- \varphi_k$, then $C_{k,\ell} = 1$.
- If $\varphi_\ell = \varphi_j \wedge \varphi_k$, then $C_{j,\ell} = C_{k,\ell} = 1$ and $\mathbf{b}_\ell = -1$.
- If $\varphi_\ell = \neg\varphi_k$, then $C_{k,\ell} = -1$ and $\mathbf{b}_\ell = 1$.
- If $\varphi_\ell = \Diamond_{\geq K}\varphi_k$, then $A_{k,\ell} = 1$ and $\mathbf{b}_\ell = -K+1$.

Next, we define that the bottom-right $(D+1)\times(D+1)$ submatrix of C is the $(D+1)\times(D+1)$ identity matrix, except that the rightmost column is moved to be the leftmost column. More formally, for all $N+1 \leq \ell < N+D+1$ we have $C_{\ell,\ell+1} = 1$ and also $C_{N+D+1,N+1} = 1$. Lastly, we define that all other elements in C, A and $\mathbf{b}$ are 0s. Finally, we define that $\mathbf{v} \in F$ if and only if $\mathbf{v}_\ell = 1$ (i.e., the ℓth value of $\mathbf{v}$ is 1) for some appointed predicate φ_ℓ and also $\mathbf{v}_{N+D+1} = 1$.

Recall that the formula depth of Γ is D. Let $\mathbf{v}(w)_i^t$ denote the value of the ith component of the feature vector in round t at node w. It is easy to show by induction that for all $n \in \mathbb{N}$, for all pointed Π-labeled graphs (G, w), and for every schema φ_ℓ in $\mathrm{SUB}(\Gamma)$ of formula depth d, we have

$$\mathbf{v}(w)_\ell^{n(D+1)+d} = 1 \text{ if } G, w \models \varphi_\ell^n \text{ and } \mathbf{v}(w)_\ell^{n(D+1)+d} = 0 \text{ if } G, w \not\models \varphi_\ell^n.$$

Most of the work is already done by Barceló et al. (see [5] for the details), but we go over the proof as there are some additional considerations related to recurrence.

For the base case, let $n = 0$. We prove the case by induction over the formula depth d of φ_ℓ. First, let $d = 0$.

- Case 1: $\varphi_\ell \in \Pi \cup \{\top\}$. Now $\mathbf{v}_\ell^0 = 1$ and $G, w \models \top^0$ if $\varphi_\ell = \top$. If $\varphi_\ell = p \in \Pi$, then by the definition of the initialization function π we have $\mathbf{v}(w)_\ell^0 = 1$ iff $p \in \lambda(w)$ iff $G, w \models p^0$, and $\mathbf{v}(w)_\ell^0 = 0$ iff $p \notin \lambda(w)$ iff $G, w \not\models p^0$.
- Case 2: $\varphi_\ell = X$, where X is a head predicate of Γ. Now $\mathbf{v}(w)_\ell^0 = 0$ due to the definition of the initialization function π, and $G, w \not\models X^0$ because each head predicate of Γ has the terminal clause $\bot$.

Now, assume the claim holds for $n = 0$ for any formulae φ_j, φ_k with formula depth $d-1$. We show that the claim holds for $n = 0$ for formulae φ_ℓ of formula depth d.

- Case 3: $\varphi_\ell = \varphi_j \wedge \varphi_k$ for some φ_j, φ_k (recall that Γ only contains conjunctions where both conjuncts have the same formula depth if neither of them are $\top$). This means we have $C_{j,\ell} = C_{k,\ell} = 1$ and $\mathbf{b}_\ell = -1$. Moreover, $C_{m,\ell} = 0$ for all $m \neq j, k$ and $A_{m,\ell} = 0$ for all m. Now
$$\mathbf{v}(w)^d_\ell = \mathrm{ReLU}^* \left(\mathbf{v}(w)^{d-1}_j + \mathbf{v}(w)^{d-1}_k - 1\right).$$
Thus $\mathbf{v}(w)^d_\ell = 1$ iff $\mathbf{v}(w)^{d-1}_j = \mathbf{v}(w)^{d-1}_k = 1$. By the induction hypothesis this is equivalent to $G, w \models \varphi^0_j$ and $G, w \models \varphi^0_k$ which is equivalent to $G, w \models (\varphi_j \wedge \varphi_k)^0$. Likewise, $\mathbf{v}(w)^d_\ell = 0$ iff $\mathbf{v}(w)^{d-1}_j = 0$ or $\mathbf{v}(w)^{d-1}_k = 0$. By the induction hypothesis this is equivalent to $G, w \not\models \varphi^0_j$ or $G, w \not\models \varphi^0_k$ which is equivalent to $G, w \not\models (\varphi_j \wedge \varphi_k)^0$.

- Case 4: $\varphi_\ell = \neg\varphi_k$ for some φ_k. This means we have $C_{k,\ell} = -1$ and $\mathbf{b}_\ell = 1$. Moreover, $C_{m,\ell} = 0$ for all $m \neq k$ and $A_{m,\ell} = 0$ for all m. Now
$$\mathbf{v}(w)^d_\ell = \mathrm{ReLU}^* \left(-\mathbf{v}(w)^{d-1}_k + 1\right).$$
Thus $\mathbf{v}(w)^d_\ell = 1$ iff $\mathbf{v}(w)^{d-1}_k = 0$. By the induction hypothesis this is equivalent to $G, w \not\models \varphi^0_k$ which is further equivalent to $G, w \models (\neg\varphi_k)^0$. Likewise, $\mathbf{v}(w)^d_\ell = 0$ iff $\mathbf{v}(w)^{d-1}_k = 1$. By the induction hypothesis this is equivalent to $G, w \models \varphi^0_k$ which is equivalent to $G, w \not\models (\neg\varphi_k)^0$.

- Case 5: $\varphi_\ell = \Diamond_{\geq K}\varphi_k$ for some φ_k. This means we have $A_{k,\ell} = 1$ and $\mathbf{b}_\ell = -K + 1$. Moreover, $C_{m,\ell} = 0$ for all m and $A_{m,\ell} = 0$ for all $m \neq k$. Now
$$\mathbf{v}(w)^d_\ell = \mathrm{ReLU}^* \left(\mathrm{SUM}_S \left(\{\{\, \mathbf{v}(v)^{d-1}_k \mid (w, v) \in E \,\}\}\right) - K + 1\right),$$
where $\mathrm{SUM}_S \colon \mathcal{M}(S) \to S$ is the sum of floating-point numbers in S in increasing order (see Section 2.1 for more details) and E is the set of edges of G. Thus $\mathbf{v}(w)^d_\ell = 1$ iff there are at least K out-neighbours v of w such that $\mathbf{v}(v)^{d-1}_k = 1$. By the induction hypothesis, this is equivalent to there being at least K out-neighbours v of w such that $G, v \models \varphi^0_k$ which is further equivalent to $G, w \models (\Diamond_{\geq K}\varphi_k)^0$. Likewise, $\mathbf{v}(w)^d_\ell = 0$ iff there are fewer than K out-neighbours v of w such that $\mathbf{v}(v)^{d-1}_k = 1$. By the induction hypothesis, this is equivalent to there being fewer than K out-neighbours v of w such that $G, v \models \varphi^0_k$ which is equivalent to $G, w \not\models (\Diamond_{\geq K}\varphi_k)^0$.

Next, assume the claim holds for n for all formulae of any formula depth. We show that it also holds for $n + 1$. We once again prove the claim by structure of φ_ℓ. Cases 3, 4 and 5 are handled analogously to how they were handled in the case $n = 0$, so we only consider cases 1 and 2.

- Case 1: $\varphi_\ell \in \Pi \cup \{\top\}$. This means we have $C_{\ell,\ell} = 1$ and $\mathbf{b}_\ell = 0$. Moreover, $C_{m,\ell} = 0$ for all $m \neq \ell$ and $A_{m,\ell} = 0$ for all m. Now
$$\mathbf{v}(w)^{(n+1)(D+1)}_\ell = \mathrm{ReLU}^* \left(\mathbf{v}(w)^{n(D+1)+D}_\ell\right).$$
Thus we see that $\mathbf{v}(w)^{(n+1)(D+1)}_\ell = \mathbf{v}(w)^{n(D+1)+D}_\ell$ and a trivial induction shows that also $\mathbf{v}(w)^{(n+1)(D+1)}_\ell = \mathbf{v}(w)^{n(D+1)}_\ell$. Now $\mathbf{v}(w)^{(n+1)(D+1)}_\ell = 1$ iff $\mathbf{v}(w)^{n(D+1)}_\ell = 1$. By the induction hypothesis this is equivalent to $G, w \models \varphi^n_\ell$ which is equivalent to $G, w \models \varphi^{n+1}_\ell$ because $\varphi^{n+1}_\ell = \varphi^n_\ell$. Likewise, $\mathbf{v}(w)^{(n+1)(D+1)}_\ell = 0$ iff $\mathbf{v}(w)^{n(D+1)}_\ell = 0$. By the induction hypothesis this is equivalent to $G, w \not\models \varphi^n_\ell$ which is equivalent to $G, w \not\models \varphi^{n+1}_\ell$.

- Case 2: $\varphi_\ell = X$, where X is a head predicate of Γ with the iteration clause φ_k of formula depth D (recall that in Γ, each iteration clause has formula depth D). This means we have $C_{k,\ell} = 1$ and $\mathbf{b}_\ell = 0$. Moreover, $C_{m,\ell} = 0$ for all $m \neq k$ and $A_{m,\ell} = 0$ for all m. Now
$$\mathbf{v}(w)^{(n+1)(D+1)}_\ell = \mathrm{ReLU}^* \left(\mathbf{v}(w)^{n(D+1)+D}_k\right).$$
Thus $\mathbf{v}(w)^{(n+1)(D+1)}_\ell = 1$ iff $\mathbf{v}(w)^{n(D+1)+D}_k = 1$. By the induction hypothesis this is equivalent to $G, w \models \varphi^n_k$ which is equivalent to $G, w \models X^{n+1}$. Likewise, $\mathbf{v}(w)^{(n+1)(D+1)}_\ell = 0$ iff $\mathbf{v}(w)^{n(D+1)+D}_k = 0$. By the induction hypothesis this is equivalent to $G, w \not\models \varphi^n_k$ which is equivalent to $G, w \not\models X^{n+1}$.

This concludes the induction.

We also know for all $n \geq 1$ and all $N+1 \leq \ell < N+D+1$ that $\mathbf{v}(w)^n_{\ell+1} = \mathbf{v}(w)^{n-1}_\ell$ and $\mathbf{v}(w)^n_{N+1} = \mathbf{v}(w)^{n-1}_{N+D+1}$. This is because $C_{\ell,\ell+1} = 1$, $C_{\ell',\ell+1} = 0$ for all $\ell' \neq \ell$ and $\mathbf{b}_{\ell+1} = 0$ and thus

$$\mathbf{v}(w)^n_{\ell+1} = \mathrm{ReLU}^*\left(\mathbf{v}(w)^{n-1}_\ell\right),$$

and also $C_{N+D+1,N+1} = 1$, $C_{\ell',N+1} = 0$ for all $\ell' \neq N+D+1$ and $\mathbf{b}_{N+1} = 0$ and thus

$$\mathbf{v}(w)^n_{N+1} = \mathrm{ReLU}^*\left(\mathbf{v}(w)^{n-1}_{N+D+1}\right).$$

By the initialization this means for all $1 \leq \ell, \ell' \leq D+1$ and all $n \in \mathbb{N}$ that $\mathbf{v}(w)^{n(D+1)+\ell'}_{N+\ell} = 1$ iff $\ell' = \ell$ and 0 otherwise. Specifically, this means that $\mathbf{v}(w)^{n(D+1)+\ell'}_{N+D+1} = 1$ iff $\ell' = D+1$ and $\mathbf{v}(w)^{n(D+1)+\ell'}_{N+D+1} = 0$ otherwise.

Thus if φ_ℓ is an appointed predicate X of Γ, then we know for all $n \in \mathbb{N}$ that $G, w \models X^n$ iff $\mathbf{v}(w)^{n(D+1)}_\ell = 1$ and we also know that $\mathbf{v}(w)^{n(D+1)}_{N+D+1} = 1$ and thereby $\mathbf{v}(w)^{n(D+1)} \in F$. Thus $\mathcal{G}_\Gamma$ is equivalent to Γ.

Since $\mathcal{G}_\Gamma$ is equivalent to Γ which is equivalent to Λ, we are done. □

We note that the proof of Lemma B.5 generalizes for other types of GNN[F]s, such as the type described below. Let $S = ((p, n, \beta), +, \cdot)$ be the floating-point system, where $p = 1$, $n = 1$ and $\beta = K_{\max} + 1$, where $K_{\max}$ is again the maximum width of the rules of Γ in the proof above. We believe that the proof generalizes for other floating-point systems which can represent all non-negative integers up to $K_{\max}$. Recall that $K_{\max}$ is the width of Γ in the proof of Lemma B.5. Consider a class of GNN[F]s over S with dimension $2(N+D+1)$ (twice that in the proof above) where the aggregation function is $\mathrm{SUM}_S\colon \mathcal{M}(S) \to S$ (the sum of floating-point numbers in S in increasing order applied separately to each element of feature vectors) and whose combination function $\mathrm{COM}\colon S^{2(N+D+1)} \times S^{2(N+D+1)} \to S^{2(N+D+1)}$ is defined by

$$\mathrm{COM}(x, y) = \mathrm{ReLU}(x \cdot C + y \cdot A + \mathbf{b}),$$

where $C, A \in S^{2(N+D+1)\times 2(N+D+1)}$ are matrices, $\mathbf{b} \in S^{2(N+D+1)}$ is a bias vector and ReLU is the rectified linear unit defined by $\mathrm{ReLU}(x) := \max(0, x)$ as opposed to ReLU^*. The last $2(D+1)$ elements of feature vectors function as they do above, counting the $2(D+1)$ steps required each time an iteration of Γ is simulated (here the computation takes twice as long compared to above). The first $2N$ elements calculate the truth values of subformulae as before, but now each calculation takes two steps instead of one as each subschema is assigned two elements instead of one. The first of these elements calculates the truth value of the subschema as before. However, due to the use of ReLU instead of ReLU^* the value of the element might be more than 1 if the subschema is of the form $\Diamond_{\geq K}\varphi_k$. Thus, the second element normalizes the values by assigning a weight of $K_{\max}$ ensuring that each positive value becomes $K_{\max}$ and then assigning the bias $-K_{\max}+1$ to bring them all down to 1. More formally, for each subschema φ_ℓ, let ℓ be the first and $N+\ell$ the second element associated with φ_ℓ. In the initialization step we define that if $\varphi_\ell \in P \subseteq \Pi$ or $\varphi_\ell = \top$, then $\pi(P)_{N+\ell} = 1$ and other elements of $\pi(P)$ are 0s. Now we define as follows for all $1 \leq \ell \leq N$.

- If $\varphi_\ell \in \Pi \cup \{\top\}$, then $C_{N+\ell,\ell} = 1$.
- If φ_ℓ is a head predicate X with the iteration clause $X :- \varphi_k$, then $C_{N+k,\ell} = 1$.
- If $\varphi_\ell = \varphi_j \wedge \varphi_k$, then $C_{N+j,\ell} = C_{N+k,\ell} = 1$ and $\mathbf{b}_\ell = -1$.
- If $\varphi_\ell = \neg\varphi_k$, then $C_{N+k,\ell} = -1$ and $\mathbf{b}_\ell = 1$.
- If $\varphi_\ell = \Diamond_{\geq K}\varphi_k$, then $A_{N+k,\ell} = 1$ and $\mathbf{b}_\ell = -K+1$.

In each of the above cases we also define that $C_{\ell,N+\ell} = K_{\max}$ and $\mathbf{b}_{N+\ell} = -K_{\max}+1$. Lastly for all $2N+1 \leq \ell < 2(N+D+1)$ we define (as before) that $C_{\ell,\ell+1} = 1$ and also $C_{2(N+D+1),2D+1} = 1$. All other elements of C, A and $\mathbf{b}$ are 0s. The set F of accepting feature vectors is defined such that $\mathbf{v} \in F$ iff $\mathbf{v}_{N+\ell} = 1$ for some appointed predicate φ_ℓ and also $\mathbf{v}_{2(N+D+1)} = 1$. It is then straightforward to prove for all $n \in \mathbb{N}$, for all pointed Π-labeled graphs (G, w) and for every schema $\varphi_\ell \in \mathrm{SUB}(\Gamma)$ of formula depth d that

$$\mathbf{v}(w)^{2n(D+1)+2d-1}_\ell \geq 1 \text{ and } \mathbf{v}(w)^{2n(D+1)+2d}_{N+\ell} = 1 \text{ if } G, w \models \varphi^n_\ell$$

and

$$\mathbf{v}(w)_{\ell}^{2n(D+1)+2d-1} = \mathbf{v}(w)_{N+\ell}^{2n(D+1)+2d} = 0 \text{ if } G, w \not\models \varphi_{\ell}^{n}.$$

The proof is similar to the above, but we also need to show that normalization works as intended, i.e., $\mathbf{v}(w)_{\ell}^{2n(D+1)+2d-1} \geq 1$ implies $\mathbf{v}(w)_{N+\ell}^{2n(D+1)+2d} = 1$ and $\mathbf{v}(w)_{\ell}^{2n(D+1)+2d-1} = 0$ implies $\mathbf{v}(w)_{N+\ell}^{2n(D+1)+2d} = 0$. To see this, observe that $C_{\ell,N+\ell} = K_{\max}$ and $\mathbf{b}_{N+\ell} = -K_{\max} + 1$ and also $C_{\ell',N+\ell} = 0$ for all $\ell' \neq \ell$ and $A_{\ell',N+\ell} = 0$ for all ℓ'. Thus

$$\mathbf{v}(w)_{N+\ell}^{2n(D+1)+2d} = \mathrm{ReLU}\left(K_{\max} \cdot \mathbf{v}(w)_{\ell}^{2n(D+1)+2d-1} - K_{\max} + 1\right),$$

which is 0 if $\mathbf{v}(w)_{\ell}^{2n(D+1)+2d-1} = 0$ and 1 if $\mathbf{v}(w)_{\ell}^{2n(D+1)+2d-1} \geq 1$ (because by the choice of S we have $K_{\max} \cdot x = K_{\max}$ for all $x \in S$, $x \geq 1$).

Having proved Lemma B.5, we are now ready to conclude the main theorem of this appendix section expressed below.

Proof of Theorem 3.2. Note that the proof uses auxiliary results introduced in this subsection. Each GNN[F] is trivial to translate into an equivalent bounded FCMPA with linear blow-up in size by the fact that the definitions of GNN[F]s and bounded FCMPAs are almost identical. Lemma B.5 shows that each GMSC-program can be translated into an equivalent R-simple aggregate-combine GNN[F]. The translation from GNN[F]s to GMSC-programs causes only polynomial blow-up in size by Lemma B.3. ☐

B.4 Proof of Theorem 3.3

Note that k-CMPAs are defined in Appendix B.1. First we recall Theorem 3.3.

Theorem 3.3. CMPA*s have the same expressive power as* ω-GML.

The proof of Theorem 3.3 is in the end of this subsection but we need some auxiliary results first. We show in Lemma B.7 that we can construct an equivalent counting type automaton over Π for each Π-formula of ω-GML. Informally, to do this, we first define a Π-formula of GML called the "full graded type of modal depth n" for each pointed graph, which expresses all the local information of its neighbourhood up to depth n. We show in Proposition B.6 that for each ω-GML-formula there is a logically equivalent disjunction of types. We also define counting type automata that compute the type of modal depth n of each node in every round n. The accepting states of the resulting automaton are exactly those types that appear in the disjunction of types.

Then we show in Lemma B.9 that for each CMPA over Π we can construct an equivalent Π-formula of ω-GML. Informally, to do this we first prove in Lemma B.8 that two pointed graphs that satisfy the same full graded type of modal depth n also have identical states in each round $\ell \leq n$ in each CMPA. For each $n \in \mathbb{N}$, we consider exactly those full graded types of modal depth n which are satisfied by some pointed graph that is accepted by the automaton in round n. We obtain the desired ω-GML-formula by taking the disjunction of all these types across all $n \in \mathbb{N}$.

By similar arguments, we also obtain Theorem B.10 which is analogous to Theorem 3.3 but restricted to bounded CMPAs and width-bounded ω-GML, which involves defining analogous width-bounded concepts.

Now, we start formalizing the proof of Theorem 3.3. To show that for each ω-GML-formula we can construct an equivalent CMPA, we need to define the concepts of graded Π-types, full graded Π-types and counting type automata. The graded Π-type of modal depth n and width k of a pointed Π-labeled graph (G, w) is a Π-formula of GML that contains all the information from the n-neighbourhood of w (according to outgoing edges), with the exception that at each distance from w we can only distinguish between at most k identical branches. The full graded Π-type of modal depth n lifts this limitation and contains all the information from the neighbourhood of depth n of w. A counting type automaton of width k is a k-CMPA that calculates the graded Π-type of modal depth n and width k of a node in each round n. Likewise, we define counting type automata which calculate the full graded Π-type of modal depth n of a node in each round n.

Let Π be a set of node label symbols, let (G, w) be a pointed Π-labeled graph and let $k, n \in \mathbb{N}$. The **graded Π-type of width k and modal depth** n of (G, w) (denoted $\tau_{k,n}^{(G,w)}$) is defined recursively as follows. For $n = 0$ we define that

$$\tau_{k,0}^{(G,w)} := \bigwedge_{p_i \in \lambda(w)} p_i \wedge \bigwedge_{p_i \notin \lambda(w)} \neg p_i.$$

Assume we have defined the graded Π-type of width k and modal depth n of all pointed Π-labeled graphs, and let $T_{k,n}$ denote the set of such types. The graded Π-type of (G, w) of width k and modal depth $n + 1$ is defined as follows:

$$\tau_{k,n+1}^{(G,w)} := \tau_{k,0}^{(G,w)} \wedge \bigwedge_{\ell=0}^{k-1} \{ \Diamond_{=\ell}\tau \mid \tau \in T_{k,n} \text{ and } G, w \models \Diamond_{=\ell}\tau \}$$
$$\wedge \{ \Diamond_{\geq k}\tau \mid \tau \in T_{k,n} \text{ and } G, w \models \Diamond_{\geq k}\tau \}.$$

Canonical bracketing and ordering is used to ensure that no two types are logically equivalent, and thus each pointed Π-labeled graph has exactly one graded Π-type of each modal depth and width.

A **counting type automaton of width** k over Π is a k-CMPA defined as follows. The set Q of states is the set $\bigcup_{n\in\mathbb{N}} T_{k,n}$ of all graded Π-types of width k (of any modal depth). The initialization function $\pi\colon \mathcal{P}(\Pi) \to Q$ is defined such that $\pi(P) = \tau_{k,0}^{(G,w)}$ where (G, w) is any pointed Π-labeled graph satisfying exactly the node label symbols in $P \subseteq \Pi$. Let $N\colon T_{k,n} \to \mathbb{N}$ be a multiset of graded Π-types of width k and modal depth n, and let τ be one such type (note that $T_{k,n}$ is finite for any $k, n \in \mathbb{N}$). Let τ_0 be the unique type of modal depth 0 in $T_{k,n}$ that does not contradict τ. We define the transition function $\delta\colon Q \times \mathcal{M}(Q) \to Q$ such that

$$\delta(\tau, N) = \tau_0 \wedge \bigwedge_{\ell=0}^{k-1} \{ \Diamond_{=\ell}\sigma \mid N(\sigma) = \ell \} \wedge \{ \Diamond_{\geq k}\sigma \mid N(\sigma) \geq k \},$$

For other N and τ that do not all share the same modal depth, we may define the transition as we please such that $\delta(q, N) = \delta(q, N_{|k})$ for all q and N.

We similarly define the **full graded** Π**-type of modal depth** n of a pointed Π-labeled graph (G, w) (denoted by $\tau_n^{(G,w)}$) which contains all the local information of the n-depth neighbourhood of (G, w) with no bound on width. For $n = 0$, we define that $\tau_0^{(G,w)} = \tau_{k,0}^{(G,w)}$ for any $k \in \mathbb{N}$. Assume that we have defined the full graded Π-type of modal depth n of all pointed Π-labeled graphs, and let T_n be the set of such full types. The full graded Π-type of modal depth $n + 1$ of (G, w) is defined as follows:

$$\tau_{n+1}^{(G,w)} := \tau_0^{(G,w)} \wedge \bigwedge_{\ell\geq 1} \{ \Diamond_{=\ell}\tau \mid \tau \in T_n, (G, w) \models \Diamond_{=\ell}\tau \} \wedge \Diamond_{=|\mathcal{N}(w)|}\top,$$

where $\mathcal{N}(w)$ is the set of out-neighbours of w. The formula tells exactly how many out-neighbours of a node satisfy each full graded type of the previous modal depth; to keep the formulae finite (over finite graphs), instead of containing conjuncts $\Diamond_{=0}\tau$ it tells exactly how many out-neighbours a node has.

A **counting type automaton** over Π is a CMPA defined as follows. The set of states is the set of all full graded Π-types. The initialization function $\pi\colon \mathcal{P}(\Pi) \to Q$ is defined such that $\pi(P) = \tau_0^{(G,w)}$ where (G, w) is any pointed Π-labeled graph satisfying exactly the node label symbols in P. Let $N\colon T_n \to \mathbb{N}$ be a multiset of full graded Π-types of some modal depth n, and let τ be one such type. Let τ_0 be the unique full type of modal depth 0 in T_n that does not contradict τ. We define the transition function $\delta\colon Q \times \mathcal{M}(Q) \to Q$ such that

$$\delta(\tau, N) = \tau_0 \wedge \bigwedge_{\ell\geq 1} \{ \Diamond_{=\ell}\sigma \mid N(\sigma) = \ell \} \wedge \Diamond_{=|N|}\top,$$

For other N and τ that do not all share the same modal depth, we may define the transition as we please.

We prove the following useful property.

Proposition B.6. *Each* Π*-formula* φ *of modal depth* n *and width* k *of* GML *has*

1. *a logically equivalent countably infinite disjunction of full graded* Π*-types of modal depth* n *and*
2. *a logically equivalent finite disjunction of graded* Π*-types of width* k *and modal depth* n*.*

Proof. First we prove the case for graded types of width k, then we prove the case for full graded types. Let $T_{k,n}$, as above, denote the set of all graded Π-types of width k and modal depth n. Let $\Phi = \{\, \tau \in T_{k,n} \mid \tau \models \varphi \,\}$ and $\neg\Phi = \{\, \tau \in T_{k,n} \mid \tau \models \neg\varphi \,\}$, and let $\bigvee \Phi$ denote the disjunction of the types in Φ. Note that this disjunction is finite since the set $T_{k,n}$ is finite. Obviously we have that $\Phi \cap \neg\Phi = \emptyset$ and $\bigvee \Phi \models \varphi$. To show that $\varphi \models \bigvee \Phi$, it suffices to show that $\Phi \cup \neg\Phi = T_{k,n}$. Assume instead that $\tau \in T_{k,n} \setminus (\Phi \cup \neg\Phi)$. Then there exist pointed Π-labeled graphs (G, w) and (H, v) that satisfy τ such that $G, w \models \varphi$ and $H, v \models \neg\varphi$. Since (G, w) and (H, v) satisfy the same graded Π-type of modal depth n and width k, there can be no Π-formula of GML of modal depth at most n and width at most k that distinguishes (G, w) and (H, v), but φ is such a formula, which is a contradiction. Ergo, $\bigvee \Phi$ and φ are logically equivalent.

In the case of full graded types we first observe that the set T_n of full graded Π-types of modal depth n is countable. Thus the sets $\Phi = \{\, \tau \in T_n \mid \tau \models \varphi \,\}$ and $\neg\Phi = \{\, \tau \in T_n \mid \tau \models \neg\varphi \,\}$ are also countable. Now, with the same proof as for graded types of width k, we can show that $\bigvee \Phi$ and φ are logically equivalent. Therefore, φ is also logically equivalent to a countably infinite disjunction of full graded Π-types of modal depth n. □

Now, we are ready to show the translation from ω-GML to CMPAs.

Lemma B.7. *For each* Π*-formula of* ω-GML*, we can construct an equivalent counting type automaton over* Π*. If the formula has finite width k, we can also construct an equivalent counting type automaton of width k.*

Proof. Assume the class $\mathcal{K}$ of pointed Π-labeled graphs is expressed by the countable disjunction $\psi := \bigvee_{\varphi \in S} \varphi$ of Π-formulae φ of GML.

First, we prove the case without the width bound. By Proposition B.6, each $\varphi \in S$ is logically equivalent with a countably infinite disjunction φ^* of full graded Π-types of GML such that the modal depth of φ^* is the same as the modal depth of φ. We define a counting type automaton $\mathcal{A}$ whose set F of accepting states is the set of types that appear as disjuncts of φ^* for any $\varphi \in S$. Now $(G, w) \in \mathcal{K}$ if and only if $G, w \models \psi$ if and only if $G, w \models \tau$ for some $\tau \in F$ if and only if the state of (G, w) is τ in round n in $\mathcal{A}$, where n is the modal depth of τ. Ergo, $\mathcal{A}$ accepts exactly the pointed Π-labeled graphs in $\mathcal{K}$.

The case for the width bound is analogous. First with Proposition B.6 we transform each disjunct $\varphi \in S$ to a finite disjunction φ^+ of graded Π-types with the same width and modal depth as φ. Then, for ψ we construct an equivalent type automaton of width k, where the set of accepting states is the set of graded types of width k that appear as disjuncts of φ^+ for any $\varphi \in S$. □

Before we prove Lemma B.9 we prove another helpful (and quite obvious) lemma.

Lemma B.8. *Two pointed* Π*-labeled graphs* (G, w) *and* (H, v) *satisfy exactly the same full graded* Π*-type of modal depth* n *(respectively, graded* Π*-type of width* k *and modal depth* n*) if and only if they share the same state in each round up to* n *for each* CMPA *(resp., each* k-CMPA*) over* Π.

Proof. We prove the claim by induction over n without the width bound, since again the case for the width bound is analogous. Let $n = 0$. Two pointed Π-labeled graphs (G, w) and (H, v) share the same full graded Π-type of modal depth 0 if and only if they satisfy the exact same node label symbols if and only if each initialization function π assigns them the same initial state.

Now assume the claim holds for n. Two pointed Π-labeled graphs (G, w) and (H, v) satisfy the same full graded Π-type of modal depth $n + 1$ if and only if **1)** they satisfy the same full graded Π-type of modal depth 0 and **2)** for each full graded Π-type τ of modal depth n, they have the same number of neighbors that satisfy τ. Now **1)** holds if and only if (G, w) and (H, v) satisfy the same node label symbols, and by the induction hypothesis **2)** is equivalent to the neighbors of (G, w) and (H, v) sharing (pair-wise) the same state in each round (up to n) for each CMPA. This is equivalent to (G, w) and (H, v) satisfying the same full graded Π-type of modal depth 0, and receiving the same multiset of states as messages in round n in each CMPA. By the definition of the transition function, this is equivalent to (G, w) and (H, v) sharing the same state in round $n + 1$ for each CMPA. ☐

Now, we are able to show the translation from CMPAs to ω-GML.

Lemma B.9. *For each* CMPA *(respectively, each k-*CMPA*) over* Π*, we can construct an equivalent* Π*-formula of* ω-GML *(resp., of width k).*

Proof. We prove the claim without the width bound, since the case with the width bound is analogous. Assume that the class $\mathcal{K}$ of pointed Π-labeled graphs is expressed by the counting message passing automaton $\mathcal{A}$ over Π. Let $\mathcal{T}$ be the set of all full graded Π-types and let

$$\Phi = \{\, \tau_n^{(G,w)} \in \mathcal{T} \mid \mathcal{A} \text{ accepts the pointed } \Pi\text{-labeled graph } (G, w) \in \mathcal{K} \text{ in round } n \,\}.$$

We define the countable disjunction $\bigvee_{\tau \in \Phi} \tau$ and show that $G, w \models \bigvee_{\tau \in \Phi} \tau$ if and only if $\mathcal{A}$ accepts (G, w). Note that Φ is countable since $\mathcal{T}$ is countable.

If $G, w \models \bigvee_{\tau \in \Phi} \tau$, then $G, w \models \tau_n^{(H,v)}$ for some pointed Π-labeled graph (H, v) accepted by $\mathcal{A}$ in round n. This means that (G, w) and (H, v) satisfy the same full graded Π-type of modal depth n. By Lemma B.8, this means that (G, w) and (H, v) share the same state in $\mathcal{A}$ in each round $\ell \leq n$. Since $\mathcal{A}$ accepts (H, v) in round n, $\mathcal{A}$ also accepts (G, w) in round n. Conversely, if $\mathcal{A}$ accepts (G, w) in round n, then $\tau_n^{(G,w)} \in \Phi$ and thus $G, w \models \bigvee_{\tau \in \Phi} \tau$. ☐

Finally, we prove Theorem 3.3.

Proof of Theorem 3.3. Note that the proof uses auxiliary results that are introduced in this subsection. Lemma B.7 shows that for each Π-formula of ω-GML we can construct an equivalent CMPA over Π and Lemma B.9 shows that for each CMPA over Π we can construct an equivalent Π-formula of ω-GML. ☐

By Lemma B.7 and Lemma B.9, it is also straightforward to conclude a similar result for bounded CMPAs and width-bounded ω-GML-formulae.

Theorem B.10. *Bounded* CMPA*s have the same expressive power as width-bounded* ω-GML-*formulae.*

B.5 Proof of Theorem 3.4

First we recall Theorem 3.4.

Theorem 3.4. GNN$[\mathbb{R}]$*s have the same expressive power as* ω-GML.

Informally, Theorem 3.4 is proved as follows (in the end of this section we give a formal proof). We first prove Lemma B.12 which shows that we can translate any GNN$[\mathbb{R}]$ into an equivalent ω-GML-formula. To prove Lemma B.12, we need to first prove an auxiliary result Lemma B.11, which shows that if two pointed graphs satisfy the same full graded type of modal depth n, then those two pointed graphs share the same feature vector in each round (up to n) with any GNN$[\mathbb{R}]$. For these results, we need graded types which were introduced in Section B.4.

Informally, the converse direction (Lemma B.13) is proved as follows. We first translate the Π-formula of ω-GML into an equivalent CMPA over Π by Theorem 3.3. Then we translate the equivalent CMPA into an equivalent counting type automaton (again, see Section B.4 for the definition of counting type automata). Then we prove that we can construct an equivalent GNN$[\mathbb{R}]$ for each counting type automaton. Informally, we encode each state of the counting type automaton into an integer and the GNN$[\mathbb{R}]$ can use them to mimic the type automaton in every step.

In both directions, we also consider the case where GNN$[\mathbb{R}]$s are bounded and the ω-GML-formulae are width-bounded.

Before proving Lemma B.12, we first establish the following useful lemma.

Lemma B.11. *Two pointed Π-labeled graphs (G, w) and (H, v) satisfy exactly the same full graded Π-type of modal depth n (respectively the same graded Π-type of width k and modal depth n) if and only if they share the same state in each round (up to n) for each unbounded* $\mathrm{GNN}[\mathbb{R}]$ *(resp.,* $\mathrm{GNN}[\mathbb{R}]$ *with bound k) over Π.*

Proof. The proof is analogous to that of Lemma B.8. □

With the above lemma, we are ready to prove Lemma B.12.

Lemma B.12. *For each* $\mathrm{GNN}[\mathbb{R}]$ *$\mathcal{G}$ over Π, we can construct an equivalent Π-formula of ω-*GML*. Moreover, if $\mathcal{G}$ is bounded with the bound k, we can construct an equivalent Π-formula of ω-*GML *of width k.*

Proof. Assume that $\mathcal{G}$ is a $\mathrm{GNN}[\mathbb{R}]$ over (Π, d). Let $\mathcal{K}$ be the class of pointed Π-labeled graphs expressed by $\mathcal{G}$. Now let

$$\Phi = \{\, \tau_n^{(G,w)} \mid \mathcal{G} \text{ accepts the pointed } \Pi\text{-labeled graph } (G, w) \in \mathcal{K} \text{ in round } n \,\},$$

where $\tau_n^{(G,w)}$ is the full graded Π-type of modal depth n of (G, w) (see Section B.4). Note that there are only countably many formulae in Φ since there are only countably many full graded Π-types. Consider the counting type automaton $\mathcal{A}$ over Π, where Φ is the set of accepting states. We will show that $\mathcal{A}$ accepts (G, w) if and only if $\mathcal{G}$ accepts (G, w).

If (G, w) is accepted by $\mathcal{A}$, then $G, w \models \tau_n^{(H,v)}$ for some pointed Π-labeled graph (H, v) accepted by $\mathcal{G}$ in round n. This means that (G, w) and (H, v) satisfy the same full graded Π-type of modal depth n. By Lemma B.11, this means that (G, w) and (H, v) share the same state in $\mathcal{G}$ in each round $\ell \leq n$. Since $\mathcal{G}$ accepts (H, v) in round n, $\mathcal{G}$ also accepts (G, w) in round n. Conversely, if $\mathcal{G}$ accepts (G, w) in round n, then $\tau_n^{(G,w)} \in \Phi$ and thus (G, w) is accepted by $\mathcal{A}$ by the definition of counting type automata. Thus $\mathcal{A}$ and $\mathcal{G}$ are equivalent. By Theorem 3.3 we can translate the type automaton $\mathcal{A}$ into an equivalent Π-formula of ω-GML. Therefore, for $\mathcal{G}$ we can obtain an equivalent ω-GML-formula.

Next consider the case where $\mathcal{G}$ is bounded. We follow the same steps as above with the following modification: instead of constructing a set of full graded types, we construct a set of graded types of width k: $\Phi = \{\, \tau_{k,n}^{(G,w)} \mid \mathcal{G}$ accepts the pointed Π-labeled graph $(G, w) \in \mathcal{K}$ in round $n \,\}$. The reasoning of the second paragraph is then modified to refer to graded types of width k and GNNs with the bound k respectively using Lemma B.11. By Theorem B.10 we obtain an equivalent width-bounded ω-GML-formula for the counting type automaton of width k. □

We then show the other direction of Theorem 3.4 in the next lemma. We first define a graph neural network model that is used as a tool in the proof that follows. A **recurrent graph neural network over natural numbers** $\mathrm{GNN}[\mathbb{N}]$ over (Π, d), is a $\mathrm{GNN}[\mathbb{R}]$ over (Π, d) where the feature vectors and the domains and co-domains of the functions are restricted to $\mathbb{N}^d$ instead of $\mathbb{R}^d$.

Lemma B.13. *For each Π-formula of ω-*GML*, we can construct an equivalent* $\mathrm{GNN}[\mathbb{R}]$ *over $(\Pi, 1)$. Moreover, for each Π-formula of ω-*GML *of width k, we can construct an equivalent bounded* $\mathrm{GNN}[\mathbb{R}]$ *over $(\Pi, 1)$ with the bound k.*

Proof. We first give an informal description, and then we give a formal proof. We show that each counting type automaton over Π can be translated into an equivalent $\mathrm{GNN}[\mathbb{N}]$ over $(\Pi, 1)$, which suffices since counting type automata have the same expressive power as ω-GML and CMPAs by Lemma B.9 and by Lemma B.7. Informally, each counting type automaton can be simulated by a graph neural network as follows. Each full type has a minimal tree graph that satisfies the full type. Each such tree can be encoded into a binary string (or more precisely into an integer) in a standard way. These binary strings are essentially used to simulate the computation of the counting type automaton. At each node and in each round, the $\mathrm{GNN}[\mathbb{N}]$ simulates the counting type automaton by combining the multiset of binary strings obtained from its out-neighbours and the local binary string into a new binary string that corresponds to the minimal tree that satisfies the type that would be obtained by the counting type automaton in the same round at the same node.

Now, we formally prove the statement. Let $\mathcal{A}$ be a counting type automaton. Now we create an equivalent GNN[$\mathbb{N}$] $\mathcal{G}$ over $(\Pi, 1)$ as follows. Each full graded type is converted into the unique finite rooted tree graph $(\mathcal{T}, r)$ of the same depth as the modal depth of the full type. This graph is encoded into a binary string (in a standard way) as follows.

- The first n bits of the string are 1s, telling the number of nodes in the tree; we choose some ordering $v_1, \dots, v_n$ for the nodes. (A natural ordering would be to start with the root, then list its children, then its grandchildren, etc.. The children of each node can be ordered in increasing order of magnitude according to the encodings of their generated subtrees. Another option is to assume that the domain of the tree is always $[n]$ for some $n \in \mathbb{N}$ and use the standard ordering of integers.) This is followed by a 0.
- The next nk bits are a bit string $\mathbf{b}$ that tells which node label symbol is true in which node; for each $\ell \in [n]$ and $i \in [k]$, the $((\ell - 1)k + i)$:th bit of $\mathbf{b}$ is 1 if and only if $\mathcal{T}, v_\ell \models p_i$.
- Finally, the last n^2 bits form a bit string $\mathbf{b}'$, where for each $i, j \in [n]$, the $((i-1)n + j)$:th bit of $\mathbf{b}'$ is 1 if and only if $(v_i, v_j) \in E$, i.e., there is an edge from v_i to v_j.

The GNN $\mathcal{G}$ operates on these binary strings (we can either use the integers encoded in binary by the binary strings, or interpret the binary strings as decimal strings). In round 0, the initialization function maps each set of node label symbols to the encoding of the unique one-node tree graph where the node satisfies exactly the node label symbols in question. In each subsequent round n, each node receives a multiset N of binary encodings of full graded types of modal depth $n - 1$. The aggregation function calculates a binary string that encodes the following graph:

- The graph contains a node r where every node label symbol is false.
- For each copy of each element in the multiset N, the graph contains a unique copy of the rooted tree encoded by that element.
- There is an edge from r to the root of each of these rooted trees; the result is itself a rooted tree where r is the root.

The combination function receives a binary string corresponding to the node's full graded type τ of modal depth $n - 1$, as well as the string constructed above. It takes the above string and modifies the bits corresponding to the node label symbols at the root; it changes them to be identical to how they were in the root in τ. The obtained binary encoding is the feature of the node in round n.

It is clear that the constructed GNN[$\mathbb{N}$] $\mathcal{G}$ calculates a node's full graded type of modal depth n in round n. This is because our construction is an embedding from full graded types to natural numbers. Any other such embedding would also suffice, as the images of the aggregation and combination function are always full graded types and GNN[$\mathbb{R}$]s do not limit the aggregation and combination functions. We can choose the accepting states to be the numbers that encode the accepting states in the corresponding counting type automaton $\mathcal{A}$, in which case the GNN accepts exactly the same pointed graphs as $\mathcal{A}$.

Now consider the case where $\mathcal{A}$ is a counting type automaton of width k. We follow similar steps as above with the following modifications.

- We convert each graded type of width k and modal depth n to a corresponding rooted tree. We choose the tree with depth n and the least amount of branches; in other words, each node has at most k identical out-neighbours.
- The aggregation function of $\mathcal{G}$ constructs a graph in the same way as above, except that it creates at most k copies of a graph encoded by an element in the multiset.

It is clear that the resulting GNN is bounded, as the aggregation function is bounded by k.

We have now shown that for each counting type automaton we can construct an equivalent GNN[$\mathbb{R}$] (or more precisely an equivalent GNN[$\mathbb{N}$]) and respectively for each counting type automaton of width k we can construct an equivalent bounded GNN[$\mathbb{R}$] (or more precisely an equivalent bounded

GNN[$\mathbb{N}$]). Therefore, by Lemma B.9 and by Lemma B.7 for each formula of ω-GML (and respectively for each CMPA) we can construct an equivalent GNN[$\mathbb{R}$]. Analogously, for each width-bounded formula of ω-GML (and resp., for each bounded CMPA) we can construct an equivalent bounded GNN[$\mathbb{R}$]. $\square$

We are now ready to formally prove Theorem 3.4.

Proof of Theorem 3.4. Note that the proof uses auxiliary results that are introduced in this subsection. By Lemma B.12 we can construct an equivalent Π-formula of ω-GML for each GNN[$\mathbb{R}$] over Π. By Lemma B.13 we can construct an equivalent GNN[$\mathbb{R}$] over Π for each Π-formula of ω-GML. $\square$

B.6 Proof of Remark 3.5

In this section we formally prove the claims in Remark 3.5, expressed in Theorems B.15 and B.16. Note that we do not require boundedness for FCMPAs in the statement of the lemma below.

First we prove an auxiliary result.

Lemma B.14. *For each* FCMPA *over* Π *with state set Q, we can construct an equivalent unrestricted* GNN[F] *over* $(\Pi, |Q|^2)$.

Proof. We encode the FCMPA fully into the GNN[F]. First we give an informal description. The feature vectors encode which state a node is in using one-hot encoding, i.e., only the bit corresponding to the occupied state is 1 and others are 0. For a received multiset M of feature vectors, the aggregation function encodes the function $\delta_M : Q \to Q, \delta_M(q) = \delta(q, M)$ into a vector, i.e., it tells for each state q_i which state q_j satisfies $\delta(q_i, M) = q_j$. To encode such a function we need $|Q|^2$ components in the feature vector. Finally, the combination function receives (the encodings of) q and δ_M and computes (the encoding of) $\delta_M(q)$.

Now we define the construction formally. Let Π be a set of node label symbols. Let $\mathcal{A} = (Q, \pi, \delta, F)$ be an FCMPA over Π. Assume some arbitrary ordering $<_Q$ between the states Q. Let $q_1, \ldots, q_{|Q|}$ enumerate the states of Q w.r.t. $<_Q$. Given a multiset $M \in \mathcal{M}(Q)$ the function $\delta_M : Q \to Q$, $\delta_M(q) = \delta(q, M)$ specified by M is possible to encode to the binary string $\mathbf{d}_M \in \{0,1\}^{|Q|^2}$ as follows. If $\delta_M(q_i) = q_j$, then the $((i-1)|Q| + j)$:th bit of $\mathbf{d}_M$ is 1; the other bits are 0. That is, $\mathbf{d}_M$ encodes the function δ_M in binary.

We construct a GNN over $(\Pi, |Q|^2)$ over a floating-point system S that includes at least 0 and 1 (that is to say, small floating-point systems suffice). For all $i \leq |Q|$, we let $\mathbf{d}_i \in S^{|Q|^2}$ denote the one-hot string where exactly the ith bit is 1 and the other bits are 0s.

- Let $P \subseteq \Pi$. The initialization function π' is defined as $\pi'(P) = \mathbf{d}_i$ where $\pi(P) = q_i$.
- The aggregation function AGG is defined as follows. Assume that the multiset M contains only one-hot strings $\mathbf{d}_i$, where $i \leq |Q|$. Now M corresponds to a multiset M^*, where each $\mathbf{d}_i$ is replaced by q_i. Then $\mathrm{AGG}(M)$ is $\mathbf{d}_{M^*}$. Otherwise, the aggregation function is defined in an arbitrary way.
- The combination function COM is defined as follows. Let $\mathbf{d}, \mathbf{d}' \in S^{|Q|^2}$ such that $\mathbf{d} = \mathbf{d}_i$ is a one-hot string for some $i \leq |Q|$ and $\mathbf{d}' = \mathbf{d}_M$ for some multiset $M \in \mathcal{M}(Q)$. Then we define that $\mathrm{COM}(\mathbf{d}, \mathbf{d}') = \mathrm{COM}(\mathbf{d}_i, \mathbf{d}_M) = \mathbf{d}_j$ if and only if $\delta_M(q_i) = q_j$. Otherwise COM is defined in an arbitrary way.
- The set F' of accepting states is defined as follows. If $\mathbf{d} = \mathbf{d}_j$ is a one-hot string, then $\mathbf{d}_j \in F'$ if and only if $q_j \in F$. Otherwise F' is defined in an arbitrary way.

It is easy to show by induction over $n \in \mathbb{N}$ that for any pointed Π-labeled graph (G, w), the state of $\mathcal{A}$ at w in round n is $q_i \in Q$ if and only if the state of $\mathcal{G}$ at w in round n is $\mathbf{d}_i$.

We note that this proof is based only on providing two embeddings to S^d where S is the floating-point system and d is the dimension of the GNN[F]. The first is an embedding from the set Q

and the second is an embedding from the set of functions $Q \to Q$, and the two embeddings need not be related in any way. Thus, any choice of floating-point system S and dimension d, such that there are at least $|Q|^2$ expressible feature vectors, would suffice and we would be able to define such embeddings and the subsequent aggregation and combination functions. In particular, dimension $|Q|^2$ suffices for any floating-point system S. □

The first claim in Remark 3.5 is stated as follows.

Theorem B.15. FCMPA*s have the same expressive power as unrestricted* GNN[F]*s.*

Proof. It is straightforward to translate an unrestricted GNN[F] into an equivalent FCMPA. By Lemma B.14 we can also construct an equivalent unrestricted GNN[F] for each FCMPA. □

The second claim in Remark 3.5 is stated as follows.

Theorem B.16. *Bounded* GNN[$\mathbb{R}$]*s have the same expressive power as width-bounded* ω-GML.

Proof. Note that this proof uses auxiliary results that are in Appendix B.5. By Lemma B.12 we can construct an equivalent width-bounded Π-formula of ω-GML for each bounded GNN[$\mathbb{R}$] over Π. By Lemma B.13 we can construct an equivalent bounded GNN[$\mathbb{R}$] over Π for each width-bounded Π-formula of ω-GML. □

B.7 Proof that multiple GNN layers can be simulated using a single layer

In the literature, GNNs are often defined as running for a constant number of iterations unlike our recurrent GNN model, see for example [5, 13, 12]. Each iteration of the GNN is considered its own layer, and each layer has its own aggregation and combination function. More formally for any $N \in \mathbb{N}$, an N-**layer** GNN[$\mathbb{R}$] $\mathcal{G}_N$ over (Π, d) is a tuple $(\mathbb{R}^d, \pi, (\delta^i)_{i\in[N]}, F)$, where $\pi\colon \mathcal{P}(\Pi) \to \mathbb{R}^d$ is the *initialization function*, $\delta^i\colon \mathbb{R}^d \times \mathcal{M}(\mathbb{R}^d) \to \mathbb{R}^d$ is the **transition function of layer** i of the form $\delta^{(i)}(q, M) = \mathrm{COM}^i(q, \mathrm{AGG}^i(M))$ (where $\mathrm{AGG}^i\colon \mathcal{M}(\mathbb{R}^d) \to \mathbb{R}^d$ is the **aggregation function of layer** i and $\mathrm{COM}^i\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is the **combination function of layer** i) and $F \subseteq \mathbb{R}^d$ is the set of *accepting feature vectors*. We define the computation of $\mathcal{G}_N$ in a Π-labeled graph $G = (V, E, \lambda)$ as follows. In round 0, the feature vector of a node v is $x_v^0 = \pi(\lambda(v))$. In round $i \in [N]$, the feature vector of a node is

$$x_v^i = \delta^i(x_v^{i-1}, \{\{\, x_u^{i-1} \mid (v, u) \in E \,\}\}) = \mathrm{COM}^i(x_v^{i-1}, \mathrm{AGG}^i(\{\{\, x_u^{i-1} \mid (v, u) \in E \,\}\})).$$

We say that $\mathcal{G}_N$ **accepts** a pointed graph (G, w) if and only if $x_w^N \in F$. Concepts concerning node properties, equivalence and same expressive power are defined as for other models of GNNs.

Proposition B.17. *For each N-layer* GNN[$\mathbb{R}$]*, we can construct an equivalent constant-iteration* GNN[$\mathbb{R}$].

Proof. Intuitively, we add a clock to the feature vectors of the N-layer GNN[$\mathbb{R}$] that tells the transition function of the constant-iteration GNN[$\mathbb{R}$] which layer to simulate.

Let $\mathcal{G}_N = (\mathbb{R}^d, \pi, (\delta^i)_{i\in[N]}, F)$ be an N-layer GNN[$\mathbb{R}$] over (Π, d). We construct a constant-iteration GNN[$\mathbb{R}$] $(\mathcal{G}, N) = ((\mathbb{R}^{d+N}, \pi', \delta', F'), N)$ over $(\Pi, d + N)$ as follows. In all feature vectors used, exactly one of the last N elements is 1 and the others are 0s. For the initialization function π' we define for all $P \subseteq \Pi$ that $\pi'(P) = (\pi(P)_1, \dots, \pi(P)_d, 1, 0, \dots, 0) \in \mathbb{R}^{d+N}$, where $\pi(P)_i$ denotes the ith element of $\pi(P) \in \mathbb{R}^d$. Before specifying the transition function, we define the following:

- For each feature vector $x = (x_1, \dots, x_{d+N}) \in \mathbb{R}^{d+N}$, we let $x' = (x_1, \dots, x_d)$ and $x'' = (x_{d+1}, \dots, x_{d+N})$.
- Likewise, for each $M \in \mathcal{M}(\mathbb{R}^{d+N})$, let M' be M where each $x \in M$ is replaced with x'.
- Let $f\colon \mathbb{R}^N \to \mathbb{R}^N$ be a function such that we have $f(1, 0, \dots, 0) = (0, 1, 0, \dots, 0)$, $f(0, 1, 0, \dots, 0) = (0, 0, 1, 0, \dots, 0)$, and so forth until $f(0, \dots, 0, 1) = (0, \dots, 0, 1)$.

Now, assuming that $x \in \mathbb{R}^{d+N}$ and $M \in \mathcal{M}(\mathbb{R}^{d+N})$ such that exactly the ith element of x'' and each y'' in M is 1 and others are 0s, we define that $\delta'(x, M) = (\delta^i(x', M'), f(x''))$ (i.e., we concatenate $\delta^i(x', M')$ and $f(x'')$). For other inputs, we define δ' arbitrarily. The set F' of accepting feature vectors is the set of feature vectors $(x_1, \dots, x_d, 0, \dots, 0, 1)$, where $(x_1, \dots, x_d)$ is an accepting feature vector of $\mathcal{G}_N$. It is easy to show that $(\mathcal{G}, N)$ accepts a pointed Π-labeled graph (G, w) if and only if $\mathcal{G}_N$ accepts (G, w) (note that for both N-layer GNN[$\mathbb{R}$]s and constant-iteration GNN[$\mathbb{R}$]s, only the feature vector of a node in round N counts for acceptance). □

B.8 Proof of Theorem 3.6

First we recall Theorem 3.6.

Theorem 3.6. *Constant-iteration* GNN[$\mathbb{R}$]*s have the same expressive power as depth-bounded* ω-GML.

Proof. Note that we *heavily* use the proofs of Lemma B.12 and Lemma B.13.

Assume that $(\mathcal{G}, N)$ is a constant-iteration GNN[$\mathbb{R}$] over (Π, d). Let $\mathcal{K}$ be the class of pointed Π-labeled graphs expressed by $(\mathcal{G}, N)$. Now let

$$\Phi = \{\, \tau_N^{(G,w)} \mid \mathcal{G} \text{ accepts } (G, w) \in \mathcal{K} \text{ in round } N \,\}$$

where $\tau_N^{(G,w)}$ is the full graded Π-type of modal depth N of (G, w) (see Section B.4). Consider the counting type automaton $\mathcal{A}$ over Π, where Φ is the set of accepting states. It is easy to show with an analogous argument as in the proof of Lemma B.12 that $\mathcal{A}$ accepts (G, w) if and only if $(\mathcal{G}, N)$ accepts (G, w).

For the converse, assume that ψ is a depth-bounded ω-GML-formula over Π of modal depth D. First by Lemma B.6 we translate ψ into an equivalent formula ψ^* which is a disjunction of full graded Π-types such that the modal depth of ψ^* is the same as ψ. By Lemma B.7, ψ^* is equivalent to a counting type automaton $\mathcal{A}$ over Π whose accepting states are the types that appear as disjuncts of ψ^*. Since the depth of each type is bounded by D, each pointed graph accepted by $\mathcal{A}$ is accepted in some round $r \leq D$. By the proof of Lemma B.13 we can construct an equivalent GNN[$\mathbb{R}$] $\mathcal{G}$ over Π such that for all pointed Π-labeled graphs (G, w) and for all $n \in \mathbb{N}$: $\mathcal{A}$ accepts (G, w) in round n iff $\mathcal{G}$ accepts (G, w) in round n. Now, it is easy to modify $\mathcal{G}$ such that if $\mathcal{G}$ accepts a pointed graph in round m then it also accepts that pointed graph in every round $m' > m$. Therefore, for all pointed graphs (G, w) we have that $(\mathcal{G}, D)$ accepts (G, w) iff $G, w \models \psi^*$ iff $G, w \models \psi$. □

C Appendix: Characterizing GNNs over MSO-expressible properties

C.1 Proof of Lemma 4.2

In the end of this subsection we give the formal proof of Lemma 4.2. First we give some preliminary definitions.

Let $G = (V, E, \lambda)$ be a graph and let $w_0 \in V$ be a node in G. A **walk in** G **starting at** w_0 is a sequence $p = w_0, \dots, w_n$ of elements of V such that $(w_i, w_{i+1}) \in E$ for all $i \leq n - 1$. We use $\mathsf{tail}(p)$ to denote w_n. Now, the **unraveling** of G at w_0 is the graph $U = (V', E', \lambda')$ defined as follows:

$$\begin{aligned} V' = {} & \text{the set of all walks in } G \text{ starting at } w_0 \\ E' = {} & \{\, (p, p') \in V' \times V' \mid p' = pw \text{ for some } w \in V \,\} \\ \lambda'(p) = {} & \lambda(\mathsf{tail}(p)) \text{ for all } p \in V'. \end{aligned}$$

We say that a formula $\varphi(x)$ is **invariant under unraveling** if for every graph $G = (V, E, \lambda)$ and every $w \in V$, we have $G \models \varphi(w)$ iff $U \models \varphi(w)$, with U the unraveling of G at w. Invariance under unraveling is defined in the same way also for GNNs. The following is easy to prove, see for example [7].

Lemma C.1. *The following are invariant under unraveling:* ω-GML*,* GMSC*,* GNN[F]*s,* GNN[$\mathbb{R}$]*s, and their constant iteration depth versions.*

We recall Lemma 4.2 and prove it.

Lemma 4.2. *Any property expressible in* MSO *and as a constant-iteration* GNN[$\mathbb{R}$] *is also* FO-*expressible.*

Proof. We present two different proofs. The first one is independent from Theorem 4.1 and the second one is not.

Assume that the MSO-formula $\varphi(x)$ over Σ_N expresses the same node property as the constant iteration depth GNN[$\mathbb{R}$] $(\mathcal{G}, k)$ over Σ_N, where $k \in \mathbb{N}$ is the iteration depth of $\mathcal{G}$. It is shown in [10] that on every class of graphs of bounded treedepth, MSO and FO have the same expressive power. The class $\mathcal{C}$ of all tree-shaped Σ_N-labeled graphs of depth at most k has bounded treedepth. We thus find an FO-formula $\vartheta(x)$ over Σ_N that is logically equivalent to $\varphi(x)$ on $\mathcal{C}$, i.e., for all $T \in \mathcal{C}$ with root w we have $T \models \vartheta(w)$ iff $T \models \varphi(w)$.

We may manipulate $\vartheta(x)$ into an FO-formula $\widehat{\vartheta}(x)$ such that for any pointed graph (G, w), we have $G \models \widehat{\vartheta}(w)$ if and only if $U_k \models \vartheta(w)$ with U_k the restriction of the unraveling of G at w to elements on level at most k. More precisely, to construct $\widehat{\vartheta}(x)$ we do the following:

- First we define an auxiliary formula

$$\psi_{\leq k}(x, y) := \bigvee_{0 \leq \ell \leq k} \exists y_0 \cdots \exists y_\ell \Big(y_0 = x \wedge y_\ell = y \wedge \bigwedge_{0 \leq m < \ell} E(y_m, y_{m+1}) \Big)$$

  which intuitively states that y lies at distance at most k from x.

- Then $\widehat{\vartheta}(x)$ is obtained from $\vartheta(x)$ by recursively replacing subformulae of type $\exists y \psi$ with $\exists y(\psi_{\leq k}(x, y) \wedge \psi)$ as follows. First, we simultaneously replace each subformula of quantifier depth 1. Having replaced subformulae of quantifier depth ℓ, we then simultaneously replace subformulae of quantifier depth $\ell + 1$.

We then have, for every graph G, the following where U is the unraveling of G at w and U_k denotes the restriction of U to elements on level at most k (the root being on level 0):

$$G \models \varphi(w) \text{ iff } U_k \models \varphi(w) \text{ iff } U_k \models \vartheta(w) \text{ iff } G \models \widehat{\vartheta}(w)$$

The first equivalence holds because φ is expresses the same property as $\mathcal{G}$, the second one by choice of ϑ, and the third one by construction of $\widehat{\vartheta}$.

We also present an alternative proof which takes advantage of Theorem 4.1. By Theorem 3.6 $(\mathcal{G}, k)$ is equivalent to some depth-bounded Σ_N-formula ψ of ω-GML. On the other hand, by Theorem 4.1 $\mathcal{G}$ is equivalent to some GMSC-program Λ, since the property expressed by $\mathcal{G}$ is expressible in MSO. By the proof of Proposition 2.6 Λ is equivalent to some width-bounded Σ_N-formula ψ' of ω-GML. Now, it is easy to show that $\psi \wedge \psi'$ is equivalent to some Σ_N-formula ψ^* of GML, since ψ is depth-bounded and ψ' is width-bounded. This can be seen by transforming ψ and ψ' into disjunctions of (non-full) graded types by applying Proposition B.6 (see also Appendix B.4 for the definition of graded types). Since $(\mathcal{G}, k)$ is equivalent to the Σ_N-formula ψ^* of GML, it expresses a node property also expressible in FO, since GML is a fragment of FO. □

C.2 Proof of Theorem 4.1

First we recall Theorem 4.1.

Theorem 4.1. *Let $\mathcal{P}$ be a property expressible in* MSO*. Then $\mathcal{P}$ is expressible as a* GNN[$\mathbb{R}$] *if and only if it is expressible in* GMSC.

We recall some details of the proof sketch of Theorem 4.1. We use an automaton model proposed in [31] that captures the expressive power of MSO on tree-shaped graphs. (Note that the automaton model is defined in Section 4.) We then show that the automaton for an MSO-formula φ that expresses the same property as a GNN[$\mathbb{R}$] (and thus as a formula of ω-GML) can be translated into a GMSC-program expressing the same property. To do this, we prove the important Lemma 4.7

which shows that for all tree-shaped graphs T: the automaton for φ accepts T iff there is a k-prefix decoration of T for some $k \in \mathbb{N}$. Intuitively, a k-prefix decoration of T represents a set of accepting runs of the automaton for φ on the prefix T_k of T (the formal definitions are in Section 4). Then we build a GMSC-program that accepts a tree-shaped graph T with root w iff there is a k-prefix decoration of T for some $k \in \mathbb{N}$.

We next define in a formal way the semantics of parity tree automata (the definition of a parity tree automaton (PTA) is in Section 4). For what follows, a **tree** T is a subset of $\mathbb{N}^*$, the set of all finite words over $\mathbb{N}$, that is closed under prefixes. We say that $y \in T$ is a **successor** of x in T if $y = xn$ for some $n \in \mathbb{N}$. Henceforth we will call successors **out-neighbours**. Note that the empty word ε is then the root of any tree T. A **Σ-labeled tree** is a pair (T, ℓ) with T a tree and $\ell : T \to \Sigma$ a node labeling function. A **maximal path** π in a tree T is a subset of T such that $\varepsilon \in \pi$ and for each $x \in \pi$ that is not a leaf in T, π contains one out-neighbour of x.

Definition C.2 (Run)**.** Let G be a Σ_N-labeled graph with $G = (V, E, \lambda)$ and $\mathcal{A} = (Q, \Sigma_N, q_0, \Delta, \Omega)$ a PTA. A **run**[3] of $\mathcal{A}$ on G is a $Q \times V$-labeled tree (T, ℓ) such that the following conditions are satisfied:

1. $\ell(\varepsilon) = (q_0, v)$ for some $v \in V$;
2. for every $x \in T$ with $\ell(x) = (q, v)$, the following graph satisfies the formula $\Delta(q, \lambda(v))$:[4]
   - the universe consists of all u with $(v, u) \in E$;
   - each unary predicate $q' \in Q$ is interpreted as the set

$$\{\, u \mid \text{ there is an out-neighbour } y \text{ of } x \text{ in } T \text{ such that } \ell(y) = (q', u) \,\}.$$

A run (T, ℓ) is **accepting** if for every infinite maximal path π of T, the maximal $i \in \mathbb{N}$, for which the set $\{x \in \pi \mid \ell(x) = (q, d) \text{ with } \Omega(q) = i\}$ is infinite, is even. We use $L(\mathcal{A})$ to denote the language accepted by $\mathcal{A}$, i.e., the set of Σ_N-labeled graphs G such that there is an accepting run of $\mathcal{A}$ on G.

We remark that, in contrast to the standard semantics of FO, the graph defined in Point 2 of the above definition may be empty and thus transition formulas may also be interpreted in the *empty graph*. A transition formula is true in this graph if and only if it does not contain any existential quantifiers, that is, $k = 0$. Note that a transition formula ϑ without existential quantifiers is a formula of the form $\forall z(\,\mathsf{diff}(z) \to \psi)$, where ψ is a disjunction of conjunctions of atoms $q(z)$ which are unary predicates for the states of the automaton. Such a formula may or may not be true in a non-empty graph. For example, if ψ in ϑ is a logical falsity (the empty disjunction), then ϑ is satisfied only in the empty graph.

Let $\mathcal{P}$ be a node property over Σ_N which is expressible in MSO and also in ω-GML. Let $\mathcal{A} = (Q, \Sigma_N, q_0, \Delta, \Omega)$ be a PTA that is obtained by Theorem 4.6 from $\mathcal{P}$. If ψ is the ω-GML-formula expressing $\mathcal{P}$, then we may simply say that $\mathcal{A}$ and ψ are equivalent. We identify a sequence $S_1, \ldots, S_n$ of subsets of Q with a graph $\mathsf{struct}(S_1, \ldots, S_n)$ defined as follows:

- the universe is $\{1, \ldots, n\}$;
- each unary predicate $q' \in Q$ is interpreted as the set $\{i \mid q' \in S_i\}$.

For a tree-shaped graph T, we let T_k denote the restriction of T to the nodes whose distance from the root is at most k. An **extension** of T_k is then any tree-shaped graph T' such that $T'_k = T_k$, that is, T' is obtained from T_k by extending the tree from the nodes at distance k from the root by attaching subtrees, but not from any node at distance $\ell < k$ from the root. Note that *k-prefix decorations of T* and *universal sets of states* are defined in Section 4.

Lemma 4.7. For every tree-shaped Σ_N-labeled graph T: $T \in L(\mathcal{A})$ if and only if there is a k-prefix decoration of T, for some $k \in \mathbb{N}$.

[3]Often semantics for parity tree automata are given with parity games, for the details see for example [14]. Informally, parity games are played by two players called Eloise and Abelard, where Eloise tries to show that the PTA accepts a given graph. Informally, the semantics introduced here represents a winning strategy of Eloise in parity games and similar semantics are used for example in [23].

[4]Note that the graph is empty if and only if v is a dead end in G.

Proof. "$\Rightarrow$". Assume that $T \in L(\mathcal{A})$. Let $\bigvee_i \psi_i$ be the Σ_N-formula of ω-GML that $\mathcal{A}$ is equivalent to. Then $T \models \psi_i$ for some i. Let k be the modal depth of ψ_i. Then $T' \models \psi_i$ for every extension T' of T_k.

Next we construct a mapping $\mu\colon V_k \to \mathcal{P}(\mathcal{P}(Q))$ and show that μ is a k-prefix decoration of (T, w). We construct μ as follows. For all $v \in V$ on the level k, we define $\mu(v)$ as the universal set for $\lambda(v)$. Then analogously to the third condition in the definition of k-prefix decorations, we define $\mu(v)$ for each node v on a level smaller than k. Now, all we have to do is show that the first condition in the definition of k-prefix decorations is satisfied, i.e., for each $S \in \mu(w)$, $q_0 \in S$. Then we may conclude that μ is a k-prefix decoration.

Assume by contradiction that there exists a set $S \in \mu(w)$ such that $q_0 \notin S$. (Note that $\mu(v) \neq \emptyset$ for all $v \in V_k$ by the definition of k-prefix decorations.) By Condition 3 of k-prefix decorations, there must exist sets $S^1_1 \in \mu(u^1_1), \ldots, S^1_{n_1} \in \mu(u^1_{n_1})$ (where $u^1_1, \ldots, u^1_{n_1}$ are the nodes on level 1) such that $q \in S$ iff the transition formula $\Delta(q, (\lambda(w)))$ is satisfied in the graph where each u^1_i is labeled S^1_i for each $1 \leq i \leq n_1$. By the same logic, we find such sets $S^m_1 \in \mu(u^m_1), \ldots, S^m_{n_m} \in \mu(u^m_{n_m})$ (where $u^m_1, \ldots, u^m_{n_m}$ are the nodes on level m) for each $m \leq k$ (note that these nodes do not necessarily share the same predecessor). Let T' be an extension of T_k that is obtained by attaching to each node u^k_i of T_k on level k some rooted tree T'' that is accepted by $\mathcal{A}$ precisely when starting from one of the states in the set we chose for that node, i.e., $Q_{T''} = S^k_i$.

Now, we can demonstrate that there is no accepting run $r_{T'} = (T_{T'}, \ell)$ of $\mathcal{A}$ on T'. Any such run has to begin with $\ell(\varepsilon) = (q_0, w)$. Note that the out-neighbours of ε cannot be labeled with exactly the labels (q, u^1_i) such that $q \in S^1_i$ because $\Delta(q_0, \lambda(w))$ is not satisfied in the graph consisting of the out-neighbours u^1_i of w labeled with S^1_i for each $1 \leq i \leq n_1$. In fact, the out-neighbours of ε cannot be labeled with any subset of such labels either, because by definition all transition formulae ϑ are monotonic in the sense that

$$(V, E, \lambda) \not\models \vartheta \implies (V, E, \lambda') \not\models \vartheta \text{ for all } \lambda' \subseteq \lambda.$$

Thus, there is an out-neighbour x_1 of ε in $T_{T'}$ such that $\ell(x_1) = (q, u^1_i)$ where $q \notin S^1_i$, and we may continue this examination starting from x_1 in the same way. Inductively, we see that for any level m we find a son x_m of x_{m-1} such that $\ell(x_m) = (q, u^m_i)$ where $q \notin S^m_i$, including the level $m = k$ where we let $\ell(x_k) = (q, u^k_i)$. Thus $q \notin S^k_i$. However, we have $S^k_i = Q_{T''}$ and it is witnessed by the run $r_{T'}$ that $\mathcal{A}$ accepts the tree T'' rooted at u^k_i when started in state q. Thus, $q \in S^k_i$, a contradiction.

"$\Leftarrow$". Let $\mu\colon V_k \to \mathcal{P}(\mathcal{P}(Q))$ be a k-prefix decoration of T, for some k. We may construct from μ an accepting run $r = (T', \ell)$ of $\mathcal{A}$ on T. For every node v in T_k on level k, let T_v denote the subtree of T rooted at v. Let a **semi-run** be defined like a run except that it needs not satisfy the first condition from the definition of runs (that is, it need not start in the initial state of the PTA). A semi-run being **accepting** is defined exactly as for runs.

Take any node v in T_k on level k. Since $\mu(v)$ is universal for $\lambda(v)$, we find an $S_v \in \mu(v)$ such that $q \in S_v$ if and only if $T_v \in L(\mathcal{A}_q)$. (Note that $\mu(v) \neq \emptyset$ for all $v \in V_k$ by the definition of k-prefix decorations.) Consequently, for each $q \in S_v$ we find an accepting semi-run $r_{v,q} = (T'_{v,q}, \ell_{v,q})$ of $\mathcal{A}$ on T_v with $\ell_{v,q}(\varepsilon) = (q, v)$. We now proceed upwards across T_k, assembling all these semi-runs into a run. In particular, we choose a set $S_u \in \mu(u)$ for every node u in T_k and, for all $q \in S_u$, a semi-run $r_{u,q} = (T'_{u,q}, \ell_{u,q})$ with $\ell_{u,q}(\varepsilon) = (q, u)$.

Let v be a node in T_k that has not yet been treated and such that its out-neighbours $u_1, \ldots, u_n$ have already been treated, that is, we have already selected a set $S_{u_i} \in \mu(u_i)$ for $1 \leq i \leq n$ along with the associated semi-runs. Due to Condition 3 of k-prefix decorations, we find a set $S_v \in \mu(v)$ such that for each $q \in S_v$: $\Delta(q, \lambda(v))$ is satisfied by the graph $\mathsf{struct}(S_{u_1}, \ldots, S_{u_n})$. Choose this set S_v.[5] As for the semi-runs, let $q \in S_v$. Then $\Delta(q, \lambda(v))$ is satisfied by the above graph. We may thus choose as $r_{v,q}$ the semi-run that is obtained as follows:

1. start with a fresh root ε and set $\ell_{v,q}(\varepsilon) = (q, v)$;
2. for each $i \in \{1, \ldots, n\}$ and each $q' \in S_{u_i}$, add the semi-run $r_{u_i,q'}$ as a subtree, making the root of $r_{u_i,q'}$ an out-neighbour of the fresh root that we had chosen.

[5]Note that if v has no out-neighbours, then S_v is the set of states q such that $\Delta(q, \lambda(v))$ is satisfied in the empty graph.

Note in step 2 that for each $1 \leq i \leq n$, if $S_{u_i} = \emptyset$, then no semi-run $r_{u_i,q'}$ is added as a subtree. Let w be the root of T. In view of Condition 1 of k-prefix decorations, it is easy to verify that the semi-run r_{w,q_0} is in fact a run of $\mathcal{A}$ on T. Moreover, this run is accepting since we had started with accepting semi-runs at the nodes v in T_k on level k and the finite initial piece that r_{w,q_0} adds on top of those semi-runs has no impact on which states occur infinitely often in infinite paths. $\square$

By Lemma 4.7, we may finish the proof of Theorem 4.1 by constructing a GMSC-program Λ such that for every tree-shaped Σ_N-labeled graph T with root w, we have $T, w \models \Lambda$ iff there is a k-prefix decoration of T, for some k. The definition of Λ follows the definition of k-prefix decorations. This construction serves as the proof of Lemma C.4 below it.

We define a fresh schema variable X_S for all $S \subseteq Q$. First, a set $P \subseteq \Sigma_N$ of node label symbols can be specified with the formula

$$\varphi_P := \bigwedge_{p \in P} p \wedge \bigwedge_{p \in \Sigma_N \setminus P} \neg p,$$

which states that the node label symbols in P are true and all others are false. Let $\mathcal{Q}_P$ denote the universal set for P. For every X_S, the program Λ contains the following terminal clause, reflecting Condition 2 of k-prefix decorations:

$$X_S(0) :- \bigvee_{S \in \mathcal{Q}_P} \varphi_P.$$

Note that if the disjunction is empty, we have $X_S(0) :- \bot$.

We also define a special appointed predicate A that is true when all the head predicates X_S that do not contain q_0 are false, reflecting Condition 1 of k-prefix decorations. More formally A is the only appointed predicate of the program, and it is defined as follows: $A(0) :- \bot$ and

$$A :- \bigwedge_{q_0 \notin S} \neg X_S.$$

For the iteration clauses of head predicates X_S, we need some preliminaries. Let K be the maximum over all k such that Δ mentions a transition formula

$$\vartheta := \exists x_1 \cdots \exists x_k \left(\mathsf{diff}(x_1, \ldots, x_k) \wedge q_1(x_1) \wedge \cdots \wedge q_k(x_k) \wedge \forall z(\, \mathsf{diff}(z, x_1, \ldots, x_k) \to \psi)\right).$$

A *counting configuration* $c \in \mathcal{M}_{K+1}(\mathcal{P}(Q))$ is a multiset of sets of states that contains each set of states at most $K+1$ times. A sequence $S_1, \ldots, S_n$ of subsets of Q **realizes** the counting configuration c if for each $S \subseteq Q$, one of the following holds:

- $c(S) \leq K$ and the number of sets S_i among $S_1, \ldots, S_n$ with $S_i = S$ is $c(S)$;
- $c(S) = K+1$ and the number of sets S_i among $S_1, \ldots, S_n$ with $S_i = S$ exceeds K.

It is easy to prove the following lemma.

Lemma C.3. *Let ϑ be a transition formula mentioned in Δ and assume that $S_1, \ldots, S_n$ and $S'_1, \ldots, S'_{n'}$ realize the same counting configuration. Then* $\mathsf{struct}(S_1, \ldots, S_n)$ *satisfies ϑ iff* $\mathsf{struct}(S'_1, \ldots, S'_{n'})$ *satisfies ϑ.*

By the above lemma, we may write $c \models \vartheta$, meaning that $\mathsf{struct}(S_1, \ldots, S_n)$ satisfies ϑ for any (equivalently: all) $S_1, \ldots, S_n$ that realize c.

It is easy to see that we can specify a counting configuration $c \in \mathcal{M}_{K+1}(\mathcal{P}(Q))$ in GMSC because we can count out-neighbours; for example, we can write the following formula, which states that for each set S of states there are exactly $c(S)$ out-neighbours where the label includes S, unless $c(S) = K+1$, in which case it permits more such out-neighbours:

$$\psi_c := \bigwedge_{c(S) = \ell \leq K} \Diamond_{=k} X_S \wedge \bigwedge_{c(S) = K+1} \Diamond_{\geq K+1} X_S.$$

For each set P of node label symbols and set S of states, we can specify the set of counting configurations in $\mathcal{M}_{K+1}(\mathcal{P}(Q))$ that satisfy a formula $\Delta(q, P)$ if and only if $q \in S$ with the following disjunction:

$$\Psi_{S,P} := \bigvee_{\substack{c \in \mathcal{M}_{K+1}(\mathcal{P}(Q)) \\ c \models \Delta(q,P) \text{ iff } q \in S}} \psi_c.$$

Recall also that the formula φ_P specifies the set of node label symbols in P. The iteration clause for X_S must state that there is a set P of node label symbols that are true and for which $\Psi_{S,P}$ is true. This can be expressed as below:

$$X_S :- \bigvee_{P \subseteq \Sigma_N} \Big(\varphi_P \wedge \Psi_{S,P}\Big).$$

We prove that the GMSC-program Λ characterizes k-prefix decorations associated with $\mathcal{A}$. A **pseudo k-prefix decoration** μ is defined like a k-prefix decoration except that it needs not satisfy the first condition of the definition of k-prefix decorations (that is, every set in the root given by μ does not need to contain the initial state). Now, given $\ell \in \mathbb{N}$ and a tree-shaped Σ_N-labeled graph T, let μ_T^ℓ denote the pseudo ℓ-prefix decoration of T. For each $\ell \in \mathbb{N}$ and each set S of states of Λ, we show by induction on $n \in [0; \ell]$ that for every tree-shaped Σ_N-labeled graph T and for each node v in T on level $\ell - n$, we have

$$T, v \models X_S^n \iff S \in \mu_T^\ell(v).$$

In the case $n = 0$, the claim holds trivially for the nodes v on level ℓ by definition of Λ, since then $T, v \models X_S^0$ if and only if S is in the universal set for P, where P is the set of node label symbols that appear in v.

Assume that the claim holds for $n < \ell$; we will show that it also holds for $n + 1$. Let v be a node of T on the level $\ell - (n + 1)$.

First assume that $T, v \models X_S^{n+1}$, that is, for some $P \subseteq \Sigma_N$, we have $T, v \models \varphi_P^n$ and $T, v \models \Psi_{S,P}^n$. Therefore, there is a multiset $c \in \mathcal{M}_{K+1}(\mathcal{P}(Q))$ such that $c \models \Delta(q, P)$ iff $q \in S$ and $T, v \models \psi_c^n$. Thus, for every set S' of states of $\mathcal{A}$ we have that if $c(S') = m \leq K$, there are exactly m out-neighbours $v_1, \ldots, v_m$ of v such that $T, v_i \models X_{S'}^n$ for every $i \in [m]$, and if $c(S') = K + 1$, there are at least $K + 1$ out-neighbours $v_1, \ldots, v_{K+1}$ of v such that $T, v_i \models X_{S'}^n$ for every $i \in [K + 1]$. By the induction hypothesis, for every set S' of states of $\mathcal{A}$ we have that if $c(S') = m \leq K$, there are exactly m out-neighbours $v_1, \ldots, v_m$ of v such that $S' \in \mu_T^\ell(v_i)$ for every $i \in [m]$, and if $c(S') = K+1$, there are at least $K+1$ out-neighbours $v_1, \ldots, v_{K+1}$ of v such that $S' \in \mu_T^\ell(v_i)$ for every $i \in [K+1]$. Since $c \models \Delta(q, P)$ iff $q \in S$ and by the definition of pseudo k-prefix decorations, we have $S \in \mu_T^\ell(v)$.

Then assume that $S \in \mu_T^\ell(v)$. Let $\{v_1, \ldots, v_m\}$ be the set of out-neighbours of v. By the induction hypothesis for each $i \in [m]$ and each set S' of states of $\mathcal{A}$, it holds that $S' \in \mu_T^\ell(v_i)$ iff $T, v_i \models X_{S'}^n$. By the definition of pseudo k-prefix decorations there are sets $S_1 \in \mu_T^\ell(v_1), \ldots, S_m \in \mu_T^\ell(v_m)$ such that $q \in S$ iff $\mathsf{struct}(S_1, \ldots, S_m) \models \Delta(q, P)$, where P is the set of node label symbols that are true in v. Now, let $c \in \mathcal{M}_{K+1}(\mathcal{P}(Q))$ be a counting configuration realized by $S_1, \ldots, S_m$. Now since K is the maximum over all k such that Δ mentions a transition formula

$$\vartheta := \exists x_1 \cdots \exists x_k \big(\mathsf{diff}(x_1, \ldots, x_k) \wedge q_1(x_1) \wedge \cdots \wedge q_k(x_k) \wedge \forall z(\,\mathsf{diff}(z, x_1, \ldots, x_k) \to \psi)\big),$$

we have $c \models \Delta(q, P)$. Therefore, $T, v \models \psi_c^n$ and trivially $T, v \models \varphi_P^n$, and thus $T, v \models X_S^{n+1}$.

Now, if Λ accepts a Σ_N-labeled tree-shaped graph (T, w), then by the result above it means that there is a pseudo k-prefix decoration μ such that $T, w \models A^{k+1}$. Thus in round k we have $T, w \not\models X_S^k$ for all S where $q_0 \notin S$. Therefore $S \notin \mu(w)$ for all S where $q_0 \notin S$, i.e., μ is actually a k-prefix decoration. Similarly if a tree-shaped graph (T, w) has a k-prefix decoration, then $T, w \models A^{k+1}$ by the result above. Thus we have proved the lemma below.

Lemma C.4. *For every tree-shaped Σ_N-labeled graph T with root w: $T, w \models \Lambda$ iff there is a k-prefix decoration of T, for some k.*

Now we are ready to prove Theorem 4.1.

Proof of Theorem 4.1. Note that some of the results needed for this proof are given in this appendix section and some are given outside the appendix. Let $\mathcal{P}$ be a node property expressible in MSO by an MSO-formula $\varphi(x)$ over Σ_N.

Assume that $\mathcal{P}$ is expressible by a Σ_N-program of GMSC. By Proposition 2.6 $\mathcal{P}$ is also expressible Σ_N-formula of ω-GML and thus by Theorem 3.4 as a GNN[$\mathbb{R}$] over Σ_N. For the converse, assume that $\mathcal{P}$ is expressible as a GNN[$\mathbb{R}$] over Σ_N. Thus $\mathcal{P}$ is expressible by Σ_N-formula of ω-GML by Theorem 3.4. Therefore, there is a PTA $\mathcal{A}$ and a Σ_N-program Λ of GMSC such that for any Σ_N-labeled graph G with root w:

$$\begin{aligned} G \models \varphi(w) &\overset{\text{Theorem 4.6}}{\iff} \text{the unraveling } U \text{ of } G \text{ at } w \text{ is in } L(\mathcal{A}) \\ &\overset{\text{Lemma 4.7}}{\iff} \text{there is a } k\text{-prefix decoration of } U\text{, for some } k \in \mathbb{N} \\ &\overset{\text{Lemma C.4}}{\iff} U, w \models \Lambda \\ &\overset{\text{Lemma C.1}}{\iff} G, w \models \Lambda. \end{aligned}$$

Thus, we have proven Theorem 4.1. ☐

C.3 Proof of Theorem 4.3

We recall and prove Theorem 4.3.

Theorem 4.3. *Let $\mathcal{P}$ be a node property expressible in* MSO. *Then $\mathcal{P}$ is expressible as a* GNN[$\mathbb{R}$] *if and only if it is expressible as a* GNN[F]. *The same is true for constant-iteration* GNN*s.*

Proof. Let $\mathcal{P}$ be an MSO-expressible property over Σ_N. By Theorem 4.1, $\mathcal{P}$ is expressible as a GNN[$\mathbb{R}$] iff it is expressible in GMSC. Thus by Theorem 3.2 $\mathcal{P}$ is expressible as a GNN[$\mathbb{R}$] iff it is expressible as a GNN[F].

For constant-iteration GNNs we work as follows. First assume that $\mathcal{P}$ is expressible as a constant-iteration GNN[$\mathbb{R}$] over Σ_N. Then by Lemma 4.2, $\mathcal{P}$ is also expressible in FO. Thus, by Theorem 4.2 in [5] $\mathcal{P}$ is expressible by a Σ_N-formula of GML (trivially, each constant-iteration GNN $(\mathcal{G}, L)$ is trivial to translate to an L-layer GNN; see Appendix B.7 for the definition of L-layer GNNs). For a GML-formula it is easy to construct an equivalent constant-iteration GNN[F]; the technique is essentially the same as in the proof of Lemma B.5 (omit the clock in the construction) and in the proof of Theorem 4.2 in [5]. The converse direction is trivial. ☐

D Appendix: Conclusion

D.1 A note on other termination conditions

A variant of our proofs gives a counterpart of Theorem 3.2 for recurrent GNNs with a termination condition based on fixed points, studied under the name of "RecGNN" in [26] and closely related to the termination condition in [30, 11].

We start by defining a fixed point acceptance condition for GNNs and for GMSC. Let $\mathcal{G}$ be a GNN over Π, let (G, w) be a pointed Π-labeled graph, and let x_w^t denote the feature vector of w in round t. We say that $\mathcal{G}$ **fixed point accepts** (G, w) if there is a round $t \in \mathbb{N}$ such that $x_w^T \in F$ for all $T \geq t$.

We define the same accepting condition for GMSC analogously. Let Λ be a Π-program of GMSC, let $\mathcal{A}$ be the set of appointed predicates of Λ and let (G, w) be a pointed Π-labeled graph. We say that Λ **fixed point accepts** (G, w) if there is a round $t \in \mathbb{N}$ such that for all $T \geq t$ we have $G, w \models X^T$ for some $X \in \mathcal{A}$.

Next we discuss how the proof of Theorem 3.2 is modified such that the same result holds with fixed point acceptance.

Theorem D.1. *The following three have the same expressive power relative to fixed point acceptance:* GNN[F]*s,* GMSC*, and R-simple aggregate-combine* GNN[F]*s.*

Proof. We start by showing that each GNN[F] can be translated into an equivalent GMSC-program relative to fixed point acceptance. First we note that it is easy to translate a GNN[F] $\mathcal{G}$ over (Π, d)

into an equivalent bounded FCMPA $\mathcal{A}$ over Π such that the following holds. Let x_w^t denote the feature vector of $\mathcal{G}$ in round t at node w and respectively let y_w^t denote the state of $\mathcal{A}$ in round t at node w. Let $F_{\mathcal{G}}$ and $F_{\mathcal{A}}$ denote the accepting states of $\mathcal{G}$ and $\mathcal{A}$ respectively. Then for all pointed Π-labeled graphs (G, w) and for all $t \in \mathbb{N}$ it holds that

$$x_w^t \in F_{\mathcal{G}} \iff y_w^t \in F_{\mathcal{A}}.$$

Moreover, the construction in the proof of Lemma B.3 shows that we can construct a GMSC-program Λ over Π for $\mathcal{A}$ with the set A of appointed predicates such that for all pointed Π-labeled graphs (G, w) and for all $t \in \mathbb{N}$ we have

$$G, w \models X^t \text{ for some } X \in A \iff y_w^t \in F_{\mathcal{A}}.$$

Clearly, for all pointed Π-labeled graphs (G, w): $\mathcal{G}$ fixed point accepts (G, w) iff Λ fixed point accepts (G, w).

Then we show that each GMSC-program Λ can be translated into an equivalent R-simple aggregate-combine GNN[F] $\mathcal{G}$ relative to fixed point acceptance. We simply modify the proof of Lemma B.5 as follows: we add to the set of accepting feature vectors all feature vectors $\mathbf{v}$ where $\mathbf{v}_{N+D+1} \neq 1$. Now it is clear for all pointed graphs (G, w) that Λ fixed point accepts (G, w) if and only if $\mathcal{G}$ fixed points accepts (G, w). $\square$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: This is a theoretical paper and all results listed in the abstract and intro are ultimately justified via proofs, occasionally with proof sketches with full proofs in the appendix. We also discuss limitations of our results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are introduced in the "Limitations" section in the conclusion and also outside that in the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All claims are formally and fully proven, but some of the proofs are in the appendix. For proofs in the appendix, sketches are provided in the main text, with the exception of three results relating to constant-iteration GNNs, two of which have highly corresponding, more general results in the paper, and one of which follows mainly from an external article.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [NA]

Justification: The paper does not include experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [NA]

   Justification: The paper does not include experiments requiring code.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [NA]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [NA]

   Justification: The paper does not include experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The code of ethics is not violated, as the paper does not involve datasets and is theoretical so that it is practically impossible to predict any harmful consequences. We have discussed all related work to the best of our knowledge and pointed out the limitations of our results, with even a short limitations-section included.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The article is a theory paper, and (sufficiently direct) paths to specific positive or negative applications are difficult to conceive.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The article does not involve any datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.