On the Expressive Power of Message-Passing Neural Networks as Global Feature Map Transformers

Floris Geerts[1], Jasper Steegmans[2](✉), and Jan Van den Bussche[2]

[1] University of Antwerp, Antwerp, Belgium
[2] Hasselt University, Hasselt, Belgium
jasper.steegmans@uhasselt.be

Abstract. We investigate the power of message-passing neural networks (MPNNs) in their capacity to transform the numerical features stored in the nodes of their input graphs. Our focus is on global expressive power, uniformly over all input graphs, or over graphs of bounded degree with features from a bounded domain. Accordingly, we introduce the notion of a global feature map transformer (GFMT). As a yardstick for expressiveness, we use a basic language for GFMTs, which we call MPLang. Every MPNN can be expressed in MPLang, and our results clarify to which extent the converse inclusion holds. We consider exact versus approximate expressiveness; the use of arbitrary activation functions; and the case where only the ReLU activation function is allowed.

Keywords: Closure under concatenation · Semiring provenance semantics for modal logic · Query languages for numerical data

1 Introduction

An important issue in machine learning is the choice of formalism to represent the functions to be learned [24,25]. For example, feedforward neural networks with hidden layers are a popular formalism for representing functions from $\mathbb{R}^n$ to $\mathbb{R}^p$. When considering functions over graphs, graph neural networks (GNNs) have come to the fore [18]. GNNs come in many variants; in this paper, specifically, we will work with the variant known as message-passing neural networks (MPNNs) [12].

MPNNs compute numerical values on the nodes of an input graph, where, initially, the nodes already store vectors of numerical values, known as *features*. Such an assignment of features to nodes may be referred to as a *feature map* on the graph [15]. We can thus view an MPNN as representing a function that maps a graph, together with a feature map, to a new feature map on that graph. We refer to such functions as *global feature map transformers (GFMTs)*.

Of course, MPNNs are not intended to be directly specified by human designers, but rather to be learned automatically from input–output examples. Still, MPNNs do form a language for GFMTs. Thus the question naturally arises: what is the expressive power of this language?

We believe GFMTs provide a suitable basis for investigating this question rigorously. The G for 'global' here is borrowed from the terminology of *global function* introduced by Gurevich [16,17]. Gurevich was interested in defining functions in structures (over some fixed vocabulary) *uniformly*, over all input structures. Likewise, here we are interested in expressing GFMTs uniformly over *all* input graphs. We also consider infinite subclasses of all graphs, notably, the class of all graphs with a fixed bound on the degree.

As a concrete handle on our question about the expressive power of MPNNs, in this paper we define the language MPLang. This language serves as a yardstick for expressing GFMTs, in analogy to the way Codd's relational algebra serves as a yardstick for relational database queries [2]. Expressions in MPLang can define features built arbitrarily from the input features using three basic operations also found in MPNNs:

1. Summing a feature over all neighbors in the graph, which provides the message-passing aspect;
2. Applying an activation function, which can be an arbitrary continuous function;
3. Performing arbitrary affine transformations (built using constants, addition, and scalar multiplication).

The difference between MPLang-expressions and MPNNs is that the latter must apply the above three operations in a rigid order, whereas the operations can be combined arbitrarily in MPLang. In particular, every MPNN is readily expressible in MPLang.

Our research question can now be made concrete: is, conversely, every GFMT expressible in MPLang also expressible by an MPNN? We offer the following answers.

1. We begin by considering the case of the popular activation function ReLU [3,13]. In this case, we show that every MPLang expression can indeed be converted into an MPNN (Theorem 1).
2. When arbitrary activation functions are allowed, we show that Theorem 1 still holds in restriction to any class of graphs of bounded degree, equipped with features taken from a bounded domain (Theorem 2).
3. Finally, when the MPNN is required to use the ReLU activation function, we show that every MPLang expression can still be *approximated* by an MPNN; for this result we again restrict to graphs of bounded degree, and moreover to features taken from a compact domain (Theorem 3).

This paper is organized as follows. Section 2 discusses related work. Section 3 defines GFMTs, MPNNs and MPLang formally. Sections 4, 5 and 6 develop our Theorems 1, 2 and 3, respectively. We conclude in Sect. 7.

In this paper, proofs of some lemmas and theorems are only sketched. Detailed proofs will be given in the journal version of this paper. Certain concepts and arguments assume some familiarity with real analysis [23].

2 Related Work

The expressive power of GNNs has received a great deal of attention in recent years. A very nice introduction, highlighting the connections with finite model theory and database theory, has been given by Grohe [15].

One important line of research is focused on characterizing the distinguishing power (also called separating power) of GNNs, in their many variants. There, one is interested in the question: given two graphs, when can they be distinguished by a GNN? This question is closely related to strong methods for graph isomorphism checking, and more specifically, the Weisfeiler-Leman algorithm. A recent overview has been given by Morris et al. [21].

Another line of research has as goal to extend classical results on the "universality" of neural networks [22] to graphs [1,4]. (There are close connections between this line of research and the one just mentioned on distinguishing power [11].) These results consider graphs with a fixed number n of nodes; functions on graphs are shown to be approximable by appropriate variants of GNNs, which, however, may depend on n.

A notable exception is the work by Barceló et al. [6,7], which inspired our present work. Barceló et al. were the first to consider expressiveness of GNNs uniformly over all graphs (note, however, the earlier work of Hella et al. [19] on similar message-passing distributed computation models). Barceló et al. focus on MPNNs, which they fit in a more general framework named AC-GNNs, and they also consider extensions of MPNNs. They further focus on *node classifiers*, which, in our terminology, are GFMTs where the input and output features are boolean values. Using the truncated ReLU activation function, they show that MPNNs can express every node classifiers expressible in *graded modal logic* (the converse inclusion holds as well).

In a way, our work can be viewed as generalizing the boolean setting considered by Barceló et al. to the numerical setting. Indeed, the language MPLang can be viewed as giving a numerical semantics to positive modal logic without conjunction, following the established methodology of semiring provenance semantics for query languages [9,14], and extending the logic with application of arbitrary activation functions. By focusing on boolean inputs and outputs, Barceló et al. are able to capture a stronger logic than our positive modal logic, notably, by expressing negation and counting.

We note that MPLang is a sublanguage of the Tensor Language defined recently by one of us and Reutter [11]. That language serves to unify several GNN variants and clarify their separating power and universality (cf. the first two lines of research on GNN expressiveness mentioned above).

Finally, one can also take a matrix computation perspective, and view a graph on n nodes, together with a d-dimensional feature map, as an $n \times n$ adjacency matrix, together with d column vectors of dimension n. To express GFMTs, one may then simply use a general matrix query language such as MATLANG [8]. Indeed, results on the distinguishing power of MATLANG fragments [10] have been applied to analyze the distinguishing power of GNN variants [5]. Of course, the specific message-passing nature of computation with MPNNs is largely lost when performing general computations with the adjacency and feature matrices.

3 Models and Languages

In this section, we recall preliminaries on graphs; introduce the notion of global feature map transformer (GFMT); formally recall message-passing neural networks and define their semantics in terms of GFMTs; and define the language MPLang.

3.1 Graphs and Feature Maps

We define a *graph* as a pair $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the edge relation. We denote V and E of a particular graph G as $V(G)$ and $E(G)$ respectively. By default, we assume graphs to be finite, undirected, and without loops, so E is symmetric and antireflexive. If $(v, u) \in E(G)$ then we call u a neighbor of v in G. We denote the set of neighbors of v in G by $N(G)(v)$. The number of neighbors of a node is called the degree of that node, and the degree of a graph is the maximum degree of its nodes. We use $\mathbb{G}$ to denote the set of all graphs, and $\mathbb{G}_p$, for a natural number p, to denote the set of all graphs with degree at most p.

For a natural number d, a *d-dimensional feature map* on a graph G is a function $\chi : V(G) \to \mathbb{R}^d$, mapping the nodes to *feature vectors*. We use $\mathit{Feat}(G, d)$ to denote the set of all possible d-dimensional feature maps on G. Similarly, for a subset X of $\mathbb{R}^d$, we write $\mathit{Feat}(G, d, X)$ for the set of all feature maps from $\mathit{Feat}(G, d)$ whose image is contained in X.

3.2 Global Feature Map Transformers

Let d and r be natural numbers. We define a *global feature map transformer (GFMT) of type $d \to r$*, to be a function $T : \mathbb{G} \to (\mathit{Feat}(G, d) \to \mathit{Feat}(G, r))$, where $G \in \mathbb{G}$ is the input of T. Thus, if G is a graph and χ is a d-dimensional feature map on G, then $T(G)(\chi)$ is an r-dimensional feature map on G. We call d and r the *input* and *output arity* of T, respectively.

Example 1. We give a few simple examples, just to fix the notion, all with output arity 1. (GFMTs with higher output arities, after all, are just tuples of GFMTs with output arity 1.)

1. The GFMT T_1 of type $2 \to 1$ that assigns to every node the average of its two feature values. Formally, $T_1(G)(\chi)(v) = (x + y)/2$, where $\chi(v) = (x, y)$.
2. The GFMT T_2 defined like T_1, but taking the maximum instead of the average.
3. The GFMT T_3 of type $1 \to 1$ that assigns to every node the maximum of the features of its neighbors. Formally, $T_3(G)(\chi)(v) = \max\{\chi(u) \mid u \in N(G)(v)\}$.

4. The GFMT T_4 of type $1 \to 1$ that assigns to every node v the sum, over all paths of length two from v, of the feature values of the end nodes of the paths. Formally,

$$T_4(G)(\chi)(v) = \sum_{(v,u)\in E(G)} \sum_{(u,w)\in E(G)} \chi(w).$$

3.3 Operations on GFMTs

If $T_1, \dots, T_r$ are GFMTs of type $d \to 1$, then the tuple $(T_1, \dots, T_r)$ defines a GFMT T of type $d \to r$ in the obvious manner:

$$T(G)(\chi)(v) := (T_1(G)(\chi)(v), \dots, T_r(G)(\chi)(v)) \tag{1}$$

Conversely, it is clear that any T of type $d \to r$ can be expressed as a tuple $(T_1, \dots, T_r)$ as above, where $T_i(G)(\chi)(v)$ equals the i-th component in the tuple $T(G)(\chi)(v)$.

Related to the above tupling operation is concatenation. Let T_1 and T_2 be GFMTs of type $d \to r_1$ and $d \to r_2$, respectively. Their *concatenation* $T_1 \mid T_2$ is the GFMT T of type $d \to r_1 + r_2$ defined by $T(G)(\chi)(v) = T_1(G)(\chi)(v) \mid T_2(G)(\chi)(v))$, where $\mid$ denotes concatenation of vectors. Concatenation is associative. Thus, we could write the previously defined $(T_1, \dots, T_r)$ also as $T_1 \mid \cdots \mid T_r$.

We also define the *parallel composition* $T_1 \parallel T_2$ of two GFMTs T_1 and T_2, of type $d_1 \to r_1$ and $d_2 \to r_2$, respectively. It is the GFMT T of type $(d_1 + d_2) \to (r_1 + r_2)$ defined by $T(G)(\chi)(v) = T_1(G)(\chi_1)(v) \mid T_2(G)(\chi_2)(v)$, where χ_1 (χ_2) is the feature map that assigns to any node w the projection of $\chi(w)$ to its first (last) d_1 (d_2) components.

In contrast, the *sequential composition* $T_1;T_2$ of two GFMTs T_1 and T_2, of type $d_1 \to d_2$ and $d_2 \to d_3$ respectively, is the GFMT T of type $d_1 \to d_3$ that maps every graph G to $T_2(G) \circ T_1(G)$. In other words, $(T_1;T_2)(G)(\chi)(v) = T_2(G)(T_1(G)(\chi))(v)$.

Finally, for two GFMTS T_1 and T_2 of type $d \to r$, we naturally define their sum $T_1 + T_2$ by $(T_1 + T_2)(G)(\chi)(v) := T_1(G)(\chi)(v) + T_2(G)(\chi)(v)$ (addition of r-dimensional vectors). The difference $T_1 - T_2$ is defined similarly.

Example 2. Recall T_1 and T_4 from Example 1, and consider the following simple GFMTs:

- For $j = 1, 2$, the GFMT P_j of type $2 \to 1$ defined by $P_j(G)(\chi)(v) = x_j$, where $\chi(v) = (x_1, x_2)$.
- The GFMT T_{half} of type $1 \to 1$ defined by $T_{\text{half}}(G)(\chi)(v) = \chi(v)/2$.
- The GFMT T_{sum} of type $1 \to 1$ defined by $T_{\text{sum}}(G)(\chi)(v) = \sum_{u\in N(G)(v)} \chi(u)$.

Then T_1 equals $(P_1 + P_2);T_{\text{half}}$, and T_4 equals $T_{\text{sum}};T_{\text{sum}}$.

3.4 Message-passing Neural Networks

A *message-passing neural network (MPNN)* consists of layers. Formally, let d and r be natural numbers. An *MPNN layer of type* $d \to r$ is a 4-tuple $L = (W_1, W_2, b, \sigma)$, where $\sigma : \mathbb{R} \to \mathbb{R}$ is a continuous function, and W_1, W_2 and b are real matrices of dimensions $r \times d$, $r \times d$ and $r \times 1$, respectively. We call σ the *activation function* of the layer; we also refer to L as a *σ-layer.*

An MPNN layer L as above defines a GFMT of type $d \to r$ as follows:

$$L(G)(\chi)(v) := \sigma\big(W_1\chi(v) + W_2 \sum_{u \in N(G)(v)} \chi(u) + b\big). \tag{2}$$

In the above formula, feature vectors are used as *column vectors*, i.e., $d \times 1$ matrices. The matrix multiplications involving W_1 and W_2 then produce $r \times 1$ matrices, i.e., r-dimensional feature vectors as desired. We see that matrix W_1 transforms the feature vector of the current node from a d-dimensional vector to an r-dimensional vector. Matrix W_2 does a similar transformation but for the sum of the feature vectors of the neighbors. Vector b serves as a *bias.* The application of σ is performed component-wise on the resulting vector.

We now define an MPNN as a finite, nonempty sequence $L_1, \dots, L_p$ of MPNN layers, such that the input arity of each layer, except the first, equals the output arity of the previous layer. Such an MPNN naturally defines a GFMT that is simply the sequential composition $L_1; \dots; L_p$ of its layers. Thus, the input arity of the first layer serves as the input arity, and the output arity of the last layer serves as the output arity. Next we shall give examples of MPNNs that express commonly known functions.

Example 3. Recall the "rectified linear unit" function $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R} : z \mapsto \max(0, z)$. Observe that $\max(x, y) = \mathrm{ReLU}(y - x) + x$, and also that $x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x)$. Hence, T_2 from Example 1 can be expressed by a two-layer MPNN, where the first layer L_1 transforms input feature vectors (x, y) to feature vectors $(y - x, x, -x)$ and then applies ReLU, and the second layer L_2 transforms the feature vector (a, b, c) produced by L_1 to the final result $a + b - c$. Formally, $L_1 = (A, 0^{3\times 2}, 0^{3\times 1}, \mathrm{ReLU})$, with

$$A = \begin{pmatrix} -1 & 1 \\ 1 & 0 \\ -1 & 0 \end{pmatrix},$$

and $L_2 = ((1, 1, -1), (0, 0, 0), 0, \mathrm{id})$, with id the identity function.

For another, simple, example, T_{sum} from Example 2 is expressed by the single layer $(0, 1, 0, \mathrm{id})$.

Same activation function If, for a particular MPNN, and an activation function σ, all layers except the last one are σ-layers, and the last layer is either also a σ-layer, or has the identity function as activation function, we refer to the MPNN as a σ-MPNN. Thus, the two MPNNs in the above example are ReLU-MPNNs.

3.5 MPLang

We introduce a basic language for expressing GFMTs. The syntax of expressions e in MPLang is given by the following grammar:

$$e ::= 1 \mid P_i \mid ae \mid e + e \mid f(e) \mid \Diamond\, e$$

where i is a non-zero natural number, $a \in \mathbb{R}$ is a constant, and $f : \mathbb{R} \to \mathbb{R}$ is continuous.

An expression e is called *appropriate for input arity* d if all subexpressions of e of the form P_i satisfy $1 \leq i \leq d$. In this case, e defines a GFMT of type $d \to 1$, as follows:

- if $e = 1$, then $e(G)(\chi)(v) := 1$
- if $e = P_i$, then $e(G)(\chi)(v) :=$ the i-th component of $\chi(v)$
- if $e = ae_1$, then $e(G)(\chi)(v) := ae_1(G)(\chi)(v)$
- if $e = e_1 + e_2$, then $e(G)(\chi)(v) := e_1(G)(\chi)(v) + e_2(G)(\chi)(v)$
- if $e = f(e_1)$, then $e(G)(\chi)(v) := f(e_1(G)(\chi)(v))$
- if $e = \Diamond\, e_1$, then $e(G)(\chi)(v) := \sum_{u \in N(G)(v)} e_1(G)(\chi)(u)$

Notice how there is no concatenation operator since the output arity of an expression is always 1. To express higher output arities, we agree that a GFMT T of type $d \to r$ is expressible in MPLang if there exists a tuple $(e_1, \dots, e_r)$ of expressions that defines T in the sense of Eq. 1. We further agree:

- The constant a will be used as a shorthand for the expression $a1$, i.e., the scalar multiplication of expression 1 by a.
- For any fixed function f, we denote by f-MPLang the language fragment of MPLang where all function applications apply f.

Example 4. Continuing Example 3, we can also express T_2 and T_{sum} in MPLang, namely, T_2 as $\text{ReLU}(P_2 - P_1) + P_1$, and T_{sum} as $\Diamond\, P_1$.

3.6 Equivalence

Let T_1 and T_2 be MPNNs, or tuples of MPLang expressions, of the same type $d \to r$.

- We say that T_1 and T_2 are *equivalent* if they express the same GFMT.
- For a class $\mathbf{G}$ of graphs and a subset X of $\mathbb{R}^d$, we say that T_1 and T_2 are *equivalent over* $\mathbf{G}$ *and* X if the GFMTs expressed by T_1 and T_2 are equal on every graph G in $\mathbf{G}$ and every $\chi \in \mathit{Feat}(G, d, X)$ (see Sect. 3.1).

Example 4 illustrates the following general observation:

Proposition 1. *For every MPNN T there is an equivalent MPLang-expression that applies, in function applications, only activation functions used in T.*

Proof. (Sketch.) Since we can always substitute subexpressions of the form P_i by more complex expressions, MPLang is certainly closed under sequential composition. It thus suffices to verify that single MPNN layers L are expressible in MPLang. For each output component of L we devise a separate MPLang expression. Inspecting Eq. 2, we must argue for linear transformation; summation over neighbors; addition of a constant (component from the bias vector); and application of an activation function. Linear transformation, and addition of a constant, are expressible using the addition and scalar multiplication operators of MPLang. Summation over neighbors is provided by the $\Diamond$ operator. Application of an activation function is provided by function application in MPLang.

4 From MPLang to MPNN Under ReLU

In Proposition 1 we observed that MPLang readily provides all the operators that are implicitly present in MPNNs. MPLang, however, allows these operators to be combined arbitrarily in expressions, whereas MPNNs have a more rigid architecture. Nevertheless, at least under the ReLU activation function, we have the following strong result:

Theorem 1. *Every GFMT expressible in ReLU-MPLang is also expressible as a ReLU-MPNN.*

Crucial to proving results of this kind will be that the MPNN architecture allows the construction of concatenations of MPNNs. We begin by noting:

Lemma 1. *Let σ be an activation function. The class of GFMTs expressible as a single σ-MPNN layer is closed under concatenation and under parallel composition.*

Proof. (Sketch.) For parallel composition, we construct block-diagonal matrices from the matrices provided by the two layers. For concatenation, we can simply stack the matrices vertically. □

For $\sigma = \text{ReLU}$, we can extend the above Lemma to multi-layer MPNNs:

Lemma 2. *ReLU-MPNNs are closed under concatenation.*

Proof. Let L and K be two ReLU-MPNNs. Since ReLU is idempotent, every n-layer ReLU-MPNN is equivalent to an $n+1$-layer ReLU-MPNN. Hence we may assume that $L = L_1; \dots, L_n$ and $K = K_1; \dots; K_n$ have the same number of layers. Now $L \mid K = (L_1 \mid K_1); (L_2 \parallel K_2); \dots; (L_n \parallel K_n)$ if $n \geq 2$; if $n = 1$, clearly $L \mid K = L_1 \mid K_1$. Hence, the claim follows from Lemma 1. □

Note that a ReLU-MPNN layer can only output positive numeric values, since the result of ReLU is always positive. This explains why we must allow the identity function (id) in the last layer of a ReLU-MPNN (see the end of Sect. 3.4). Moreover, we can simulate intermediate id-layers in a ReLU-MPNN, thanks to the identity $x = \text{ReLU}(x) - \text{ReLU}(-x)$. Specifically, we have:

Lemma 3. *Let L be an id-layer and let K be a σ-layer. Then there exists a ReLU-layer L' and a σ-layer K' such that $L; K$ is equivalent to $L'; K'$.*

Proof. Let $L = (W_1, W_2, b, \mathrm{id})$. We put

$$L' = (W_1, W_2, b, \mathrm{ReLU}) \mid (-W_1, -W_2, -b, \mathrm{ReLU})$$

which corresponds to a ReLU-layer by Lemma 1. Let $K = (A, B, c, \sigma)$. Consider the block matrices $A' = (A|-A)$ and $B' = (B|-B)$ (single-row block matrices, with two matrices stacked horizontally, not vertically). Now for K' we use (A', B', c, σ). $\square$

We are now ready to prove Theorem 1. By Lemma 2, it suffices to focus on MPLang expressions, i.e., GFMTs of output arity one. So, our task is to construct, for every expression e in ReLU-MPLang, an equivalent ReLU-MPNN E. However, by Lemma 3, we are free to use intermediate id-layers in the construction of E. We proceed by induction on the structure of e. We skip the base cases and consider the inductive cases where e is of one of the forms ae_1, $e_1 + e_2$, $f(e_1)$ (with $f = \mathrm{ReLU}$), or $\Diamond\, e_1$. By induction, we have MPNNs E_1 and E_2 for e_1 and e_2.

- If e is of the form ae_1, we set $E = E_1; (a, 0, 0, \mathrm{id})$.
- If e is of the form $e_1 + e_2$, we set $E = (E_1 \mid E_2); ((1, 1), (0, 0), 0, \mathrm{id})$. Here, $E_1 \mid E_2$ corresponds to a ReLU-MPNN by Lemma 2.
- If e is of the form $f(e_1)$, we set $E = E_1; (1, 0, 0, f)$.
- If e is of the form $\Diamond\, e_1$, we set $E = E_1; (0, 1, 0, \mathrm{id})$.

5 Arbitrary Activation Functions

Theorem 1 only supports the ReLU function in MPLang expressions. On the other hand, the equivalent MPNN then only uses ReLU as well. If we allow arbitrary activation functions in MPNNs, can they then simulate also MPLang expressions that apply arbitrary functions? We can answer this question affirmatively, under the assumption that graphs have bounded degree and feature vectors come from a bounded domain. The proof of our Lemma 4 explains how we rely on the bounded-domain assumption. Moreover, also the degree has to be bounded, for otherwise we can still create unbounded values using $\Diamond(P_i)$.

Theorem 2. *Let p and d be natural numbers, let $\mathbb{G}_p$ be the class of graphs of degree at most p, and let $X \subseteq \mathbb{R}^d$ be bounded. For every GFMT T expressible in MPLang there exists an MPNN that is equivalent to T over $\mathbb{G}_p$ and X.*

The above theorem can be proven exactly as Theorem 1, once we can deal with the concatenation of two MPNN layers with possibly different activation functions. The following result addresses this task:

Lemma 4. *Let L and K be MPNN layers of type $d_L \to r_L$ and $d_K \to r_k$, respectively. Let $X_L \subseteq \mathbb{R}^{d_L}$ and $X_K \subseteq \mathbb{R}^{d_K}$ be bounded, and let p be a natural number. There exist two MPNN layers L' and K' such that*

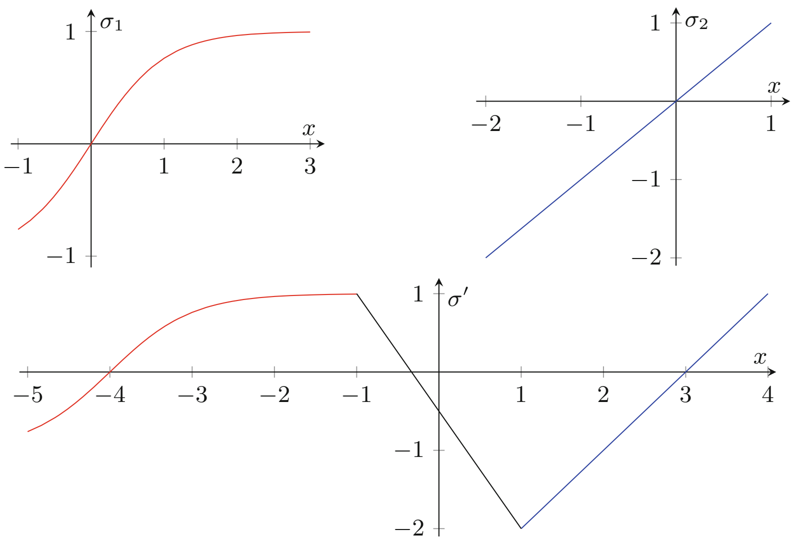

Fig. 1. Illustration of the proof of Lemma 4.

1. L' and K' use the same activation function;
2. L' is equivalent to L over $\mathbb{G}_p$ and X_L;
3. K' is equivalent to K over $\mathbb{G}_p$ and X_K.

Proof. Let $L = (W_{1L}, W_{2L}, b_L, \sigma_L)$ and $K = (W_{1K}, W_{2K}, b_K, \sigma_K)$. Let $w_{1,i}$, $w_{2,i}$ and b_i be the i-th row of W_{1L}, W_{2L} and b_L respectively. For each $i \in \{1, \dots, r_L\}$ and for any $k \in \{1, \dots, p\}$ consider the function

$$\lambda_i^k : \mathbb{R}^{(k+1)d_L} \to \mathbb{R} : (\vec{x_0}, \vec{x_1}, \dots, \vec{x_k}) \mapsto w_{1,i} \cdot \vec{x_0} + w_{2,i} \cdot \vec{x_1} + \cdots + w_{2,i} \cdot \vec{x_k} + b_i.$$

Then for any $G \in \mathbb{G}_p$, any $\chi \in \mathit{Feat}(G, d, X_L)$, and $v \in V(G)$, each component of $L(G)(\chi)(v)$ will belong to the image of some function λ_i^k on X_L^{k+1}, with k the degree of v. Since X_L^{k+1} is bounded and λ_i^k is continuous, these images are also bounded and their finite union over $i \in \{1, \dots, r\}$ and $k \in \{1, \dots, p\}$ is also bounded. Let Y_1 be this union and let $M = \max Y_1$.

For K we can similarly define the functions κ_i^k and arrive at a bounded set $Y_K \subseteq \mathbb{R}$. We then define $m = \min Y_2$.

We will now construct a new activation function σ'. First define the functions $\sigma'_L(x) := \sigma_L(x + M_i + 1)$ for $x \in]-\infty, -1]$ and $\sigma'_K(x) := \sigma_K(x - m_i - 1)$ for $x \in [1, \infty[$. Notice how σ'_L is simply σ_L shifted to the left so that its highest possible input value, which is M, aligns with -1. Similarly, σ'_K is simply σ_K shifted to the right so that its lowest possible input value, which is m, aligns with 1. We then define σ' to be any continuous function that extends both σ'_L and σ'_K. An example of this construction can be seen in Fig. 1 with $\sigma_1 = \tanh$, $M = 3$, σ_2 the identity, and $m = -2$.

We also construct new bias vectors, obtained by shifting b_L and b_K left and right respectively to provide appropriate inputs for σ'. Specifically, we define $b'_L := b_L - (M+1)^{r\times 1}$ and $b'_K := b_K + (m+1)^{r\times 1}$.

Finally, we can set $L' = (W_{1L}, W_{2L}, b'_L, \sigma')$ and $K' = (W_{1K}, W_{2K}, b'_K, \sigma')$ as desired. □

Thanks to the above lemma, Lemma 1 remains available to concatenate layers. The part of Lemma 1 that deals with parallel composition (which is needed to prove closure under concatenation for multi-layer MPNNs) must be slightly adapted as follows. It follows immediately from Lemma 4 above and the original Lemma 1.

Lemma 5. *Let L and K be MPNN layers of type $d_L \to r_L$ and $d_K \to r_k$, respectively. Let $X_L \subseteq \mathbb{R}^{d_L}$ and $X_K \subseteq \mathbb{R}^{d_K}$ be bounded, and let p be a natural number. Let $X = \{(\vec{x_L}, \vec{x_K}) \mid \vec{x_L} \in X_L \text{ and } \vec{x_K} \in X_K\} \subseteq \mathbb{R}^{d_L+d_K}$. There exists an MPNN layer that is equivalent to $L \parallel K$ over $\mathbb{G}_p$ and X.*

6 Approximation by ReLU-MPNNs

Theorem 2 allows the use of arbitrary activation functions in the MPNN simulating an MPLang expression; these activation functions may even be different from the ones applied in the expression (see the proof of Lemma 4). What if we insist on MPNNs using a fixed activation function? In this case we can still recover our result, if we allow approximation. Moreover, we must slightly strengthen our assumption of feature vectors coming from a bounded domain, to coming from a compact domain.[1]

We will rely on a classical result in the approximation theory of neural networks [20,22], to the effect that continuous functions can be approximated arbitrarily well by piecewise linear functions, which can be modeled using ReLU.[2] In order to recall this result, we recall that the uniform distance between two continuous functions g and h from $\mathbb{R}$ to $\mathbb{R}$ on a compact domain Y equals $\rho_Y(g,h) = \sup_{x\in Y} |g(x) - h(x)|$.

Density Property. *Let Y be a compact subset of $\mathbb{R}$, let $f : \mathbb{R} \to \mathbb{R}$ be continuous on Y, and let $\epsilon > 0$ be a real number. There exists a positive integer n and real coefficients a_i, b_i, c_i, for $i = 1, \dots, n$, such that $\rho_Y(f, f') \leq \epsilon$, where $f'(x) = \sum_{i=1}^{n} c_i \mathrm{ReLU}(a_i x - b_i)$.*

We want to extend the notion of uniform distance to GFMTs expressed in MPLang. For any MPLang expression e appropriate for input arity d, any class

[1] A subset of $\mathbb{R}$ or $\mathbb{R}^d$ is called compact if it is bounded and closed in the ordinary topology.

[2] The stated Density Property actually holds not just for ReLU, but for any nonpolynomial continuous function.

$\mathbf{G}$ of graphs, and any subset $X \subseteq \mathbb{R}^d$, the *image* of e over $\mathbf{G}$ and X is defined as the set

$$\{e(G)(\chi)(v) : G \in \mathbf{G} \ \& \ \chi \in \mathit{Feat}(G, d, X) \ \& \ v \in V(G)\}.$$

It is a subset of $\mathbb{R}$. We observe: (proof omitted)

Lemma 6. *For any natural number p and compact X, the image of e over $\mathbb{G}_p$ and X is a compact set.*

With p and X as in the lemma, and any two MPLang expression e_1 and e_2 appropriate for input arity d, the set

$$\{|e_1(G)(\chi)(v) - e_2(G)(\chi)(v)| : G \in \mathbb{G}_p \ \& \ \chi \in \mathit{Feat}(G, d, X) \ \& \ v \in V(G)\}$$

has a supremum. We define $\rho_{\mathbb{G}_p,X}(e_1, e_2)$, the uniform distance between e_1 and e_2 over $\mathbb{G}_p$ and X, to be that supremum.

The main result of this section can now be stated as follows. Note that we approximate MPLang expressions by ReLU-MPLang expressions. These can then be further converted to ReLU-MPNNs by Theorem 1.

Theorem 3. *Let p and d be natural numbers, and let $X \subseteq \mathbb{R}^d$ be compact. Let e be an MPLang expression appropriate for d, and let $\epsilon > 0$ be a real number. There exists a ReLU-MPLang expression e' such that $\rho_{\mathbb{G}_p,X}(e, e') \leq \epsilon$.*

Proof. By induction on the structure of e. We skip the base cases. In the inductive cases where e is of the form ae_1, $e_1 + e_2$, or $f(e_1)$, we consider any $G \in \mathbb{G}_p$, any $\chi \in \mathit{Feat}(G, d, X)$, and any $v \in V(G)$, but abbreviate $e(G)(\chi)(v)$ simply as e.

Let e be of the form ae_1. If $a = 0$ we set $e' = 0$. Otherwise, let e'_1 be the expression obtained by induction applied to e_1 and ϵ/a. We then set $e' = ae'_1$. The inequality $|e - e'| \leq \epsilon$ is readily verified.

Let e be of the form $e_1 + e_2$. For $j = 1, 2$, let e'_j be the expression obtained by induction applied to e_j and $\epsilon/2$. We then set $e' = e'_1 + e'_2$. The inequality $|e - e'| \leq \epsilon$ now follows from the triangle inequality.

Let e is of the form $f(e_1)$. By Lemma 6, the image of e_1 is a compact set $Y_1 \subseteq \mathbb{R}$. We define the closed interval $Y = [\min(Y_1) - \epsilon/2, \max(Y_1) + \epsilon/2]$. By the Density Property, there exists f' such that $\rho_Y(f, f') \leq \epsilon/2$. Since Y is compact, f' is uniformly continuous on Y. Thus there exists $\delta > 0$ such that $|f'(x) - f'(x')| < \epsilon/2$ whenever $|x - x'| < \delta$.

We now take e'_1 to be the expression obtained by induction applied to e_1 and $\min(\delta, \epsilon/2)$. We see that the image of e'_1 is contained in Y. Setting $e' = f(e'_1)$, we verify that $|e - e'| = |f(e_1) - f'(e'_1)| + |f'(e_1) - f'(e'_1)| \leq \epsilon$ as desired.

Our final inductive case is when e is of the form $\Diamond\, e_1$. We again consider any $G \in \mathbb{G}_p$, any $\chi \in \mathit{Feat}(G, d, X)$, and any $v \in V(G)$, but this time abbreviate $e(G)(\chi)(v)$ as $e(v)$. Let e'_1 be the expression obtained by induction applied to e_1 and ϵ/p. Setting $e' = \Diamond\, e'_1$, we verify, as desired:

$$\begin{aligned}|e(v) - e'(v)| &= |\sum_{u \in N(G)(v)} e_1(u) - \sum_{u \in N(G)(v)} e'_1(u)| \\ &\leq \sum_{u \in N(G)(v)} |e_1(u) - e'_1(u)| \\ &\leq p(\epsilon/p) \\ &= \epsilon.\end{aligned}$$

The penultimate step clearly uses that G has degree bound p. (This degree bound is also used in Lemma 6.)

7 Concluding Remarks

We believe that our approach has the advantage of modularity. For example, Theorem 1 is stated for ReLU, but holds for any activation function for which Lemmas 1 and 3 can be shown. We already noted that the Density Property holds not just for ReLU but for any nonpolynomial continuous activation function. It follows that for any activation function σ for which Lemmas 1 and 3 can be shown, every MPLang expression can be approximated by a σ-MPNN.

It would be interesting to see counterexamples that show that Theorems 2 and 3 do not hold without the restriction to bounded-degree graphs, or to features from a bounded or compact domain. Such counterexamples can probably be derived from known counterexamples in analysis or approximation theory.

Finally, in this work we have focused on the question whether MPLang can be simulated by MPNNs. However, it is also interesting to investigate the expressive power of MPLang by itself. For example, is the GFMT T_3 from Example 1 expressible in MPLang?

Acknowledgments. We thank the anonymous reviewers for suggesting improvements to the text. Jasper Steegmans is supported by the Special Research Fund (BOF) of Hasselt University.

References

1. Abboud, R., Ceylan, I., Grohe, M., Lukasiewicz, T.: The surprising power of graph neural networks with random node initialization. In: Zhou, Z.H. (ed.) Proceedings 30th International Joint Conference on Artificial Intelligence, pp. 2112–2118. ijcai.org (2021)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Boston (1995)
3. Arora, R., Basu, A., Mianjy, P., Mukherjee, A.: Understanding deep neural networks with rectified linear units. In: Proceedings 6th International Conference on Learning Representations. OpenReview.net (2018)

4. Azizian, W., Lelarge, M.: Expressive power of invariant and equivariant graph neural networks. In: Proceedings 9th International Conference on Learning Representations. OpenReview.net (2021)
5. Balcilar, M., Héroux, P., et al.: Breaking the limits of message passing graph neural networks. In: Meila, M., Zhang, T. (eds.) Proceedings 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 599–608 (2021)
6. Barceló, P., Kostylev, E., Monet, M., Pérez, J., Reutter, J., Silva, J.: The expressive power of graph neural networks as a query language. SIGMOD Rec. **49**(2), 6–17 (2020)
7. Barceló, P., Kostylev, E., Monet, M., Pérez, J., Reutter, J., Silva, J.: The logical expressiveness of graph neural networks. In: Proceedings 8th International Conference on Learning Representations. OpenReview.net (2020)
8. Brijder, R., Geerts, F., Van den Bussche, J., Weerwag, T.: On the expressive power of query languages for matrices. ACM Trans. Database Syst. **44**(4), 15:1–15:31 (2019)
9. Dannert, K.M., Grädel, E.: Semiring provenance for guarded logics. In: Madarász, J., Székely, G. (eds.) Hajnal Andréka and István Németi on Unity of Science. OCL, vol. 19, pp. 53–79. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-64187-0_3
10. Geerts, F.: On the expressive power of linear algebra on graphs. Theory Comput. Syst. **65**(1), 179–239 (2021)
11. Geerts, F., Reutter, J.: Expressiveness and approximation properties of graph neural networks. In: ICLR. OpenReview.net (2022). (to appear)
12. Gilmer, J., et al.: Neural message passing for quantum chemistry. In: Precup, D., Teh, Y. (eds.) Proceedings 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1263–1272 (2017)
13. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
14. Green, T., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings 26th ACM Symposium on Principles of Database Systems, pp. 31–40 (2007)
15. Grohe, M.: The logic of graph neural networks. In: Proceedings 36th Annual ACM/IEEE Symposium on Logic in Computer Science, pp. 1–17. IEEE (2021)
16. Gurevich, Y.: Algebras of feasible functions. In: Proceedings 24th Symposium on Foundations of Computer Science, pp. 210–214. IEEE Computer Society (1983)
17. Gurevich, Y.: Logic and the challenge of computer science. In: Börger, E. (ed.) Current Trends in Theoretical Computer Science, pp. 1–57. Computer Science Press (1988)
18. Hamilton, W.: Graph Representation Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool, San Rafael (2020)
19. Hella, L., et al.: Weak models of distributed computing, with connections to modal logic. Distrib. Comput. **28**(1), 31–53 (2013). https://doi.org/10.1007/s00446-013-0202-3
20. Leshno, M., Lin, V., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Netw. **6**(6), 861–867 (1993)
21. Morris, C., et al.: Weisfeiler and Leman go machine learning: the story so far. arXiv:2122.09992 (2021)
22. Pinkus, A.: Approximation theory of the MLP model in neural networks. Acta Numerica **8**, 143–195 (1999)

23. Rudin, W.: Principles of Mathematical Analysis, 3rd edn. McGraw Hill, New York (1976)
24. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 4th edn. Pearson, London (2022)
25. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, Cambridge (2014)