Expressive Power of Graph Transformers via Logic

Veeti Ahvonen[1], **Maurice Funk**[2], **Damian Heiman**[1], **Antti Kuusisto**[1], **Carsten Lutz**[2]

[1]Tampere University,
[2]Leipzig University and ScaDS.AI Center Dresden/Leipzig
veeti.ahvonen@tuni.fi, maurice.funk@uni-leipzig.de, damian.heiman@tuni.fi, antti.kuusisto@tuni.fi, carsten.lutz@uni-leipzig.de

Abstract

Transformers are the basis of modern large language models, but relatively little is known about their precise expressive power on graphs. We study the expressive power of graph transformers (GTs) by Dwivedi and Bresson (2020) and GPS-networks by Rampásek et al. (2022), both under soft-attention and average hard-attention. Our study covers two scenarios: the theoretical setting with real numbers and the more practical case with floats. With reals, we show that in restriction to vertex properties definable in first-order logic (FO), GPS-networks have the same expressive power as graded modal logic (GML) with the global modality. With floats, GPS-networks turn out to be equally expressive as GML with the counting global modality. The latter result is absolute, not restricting to properties definable in a background logic. We also obtain similar characterizations for GTs in terms of propositional logic with the global modality (for reals) and the counting global modality (for floats).

1 Introduction

Transformers have emerged as a powerful machine learning architecture serving as the basis of modern large language models such as GPTs (Vaswani et al. 2017) and finding success also, e.g., in computer vision (Dosovitskiy et al. 2021). Recently, transformers have received significant attention in the field of graph learning, traditionally dominated by graph neural networks (GNNs) (Scarselli et al. 2009) and related formalisms like graph convolutional networks (Kipf and Welling 2017). This shift is driven by well-known challenges GNNs face in handling long-range interactions, including issues such as over-squashing (Alon and Yahav 2021) and over-smoothing (Li, Han, and Wu 2018). Whereas GNNs rely primarily on local message passing, transformers can attend globally to any vertex in the graph. The literature now includes many graph learning models incorporating transformers. An important distinction is between 'pure' transformer models, which ignore the graph structure and result in 'bags-of-vertices' models (Yun et al. 2019; Kreuzer et al. 2021), and hybrids that combine transformers and GNN-style message passing (Rampásek et al. 2022).

To understand the limitations of learning models and their relationships, an expanding literature characterizes the expressive power of such models using logical formalisms. While transformers on words as used in GPTs connect to versions of linear temporal logic and first-order logic with counting (Li and Cotterell 2025; Chiang, Cholak, and Pillay 2023), GNNs relate to variants of graded modal logic (GML) (Barceló et al. 2020; Benedikt et al. 2024).

In this article, we provide logical characterizations of graph learning models that incorporate transformers. Our characterizations are uniform in that we do not impose a constant bound on graph size. We are primarily interested in models that combine GNN message passing layers with transformer layers, and focus in particular on the rather general GPS-networks of Rampásek et al. (2022). In addition, we also consider pure bags-of-vertices graph transformers (GTs) (Yun et al. 2019; Dwivedi and Bresson 2020). For both models, we study the case where features are vectors of real numbers, as in most theoretical studies, and also the case where they are floats, as in real-life implementations. We study both soft-attention and average hard-attention in the transformer layers. We focus on these models in their 'naked' form, without positional (or structural) encodings. Such encodings—often based on the graph Laplacian, homomorphism counts, and notions of graph centrality—enrich each vertex with information regarding its position in the graph. While they play an important role in graph learning with transformers, there is an uncomfortably large zoo of them. Therefore, we believe that to characterize expressive power, it is natural to begin with the naked case, providing a foundation for analyzing models with encodings. We focus on vertex classification as a basic learning task, but many of our results also generalize to graph classification tasks, see the end of Section 4.3 and the technical report (Ahvonen et al. 2025).

To survey our results, we start with the case of real numbers. Our first main result is that in restriction to vertex properties expressible in first-order logic (FO), GPS-networks based on reals have the same expressive power as GML with the (non-counting) global modality ($\mathrm{GML} + \mathrm{G}$), stated in Theorem 4. As all our results, this applies to both soft-attention and average hard-attention, assuming sum aggregation in message-passing layers as in (Barceló et al. 2020). While it is unsurprising that adding transformer layers to a GNN corresponds to adding a global feature to the logic GML, it was far from clear that this feature is the non-

counting global modality, rather than, say, its counting version. Our result implies that GPS-networks cannot globally count in an absolute way, as in 'the graph contains at least 10 vertices labeled p'. In contrast, they can globally count in a relative way, as in 'the graph contains more vertices labeled p than vertices labeled q'. This, however, is not expressible in FO. The proof of our result is non-trivial and requires the introduction of a new type of bisimilarity (global-ratio graded bisimilarity $\sim_{G\%}$) and the proof of a new van Benthem/Rosen-style result that essentially states: if an FO-formula $\varphi(x)$ is invariant under $\sim_{G\%}$, then it is equivalent to a $\mathrm{GML} + \mathrm{G}$-formula. We also prove that, relative to FO, real-based GTs have the same expressive power as propositional logic PL with only the non-counting global modality ($\mathrm{PL} + \mathrm{G}$), stated as Theorem 10.

We next discuss our results regarding floating-point numbers. Our main result for this case is that float-based GPS-networks have the same expressive power as GML with the counting global modality ($\mathrm{GML} + \mathrm{GC}$), stated as Theorem 16. In contrast to the case of the reals, this characterization is absolute rather than relative to FO. It applies to any reasonable aggregation function including sum, max and mean. We consider it remarkable that transitioning from reals to floats results in incomparable expressive power: while relative global counting is no longer possible, absolute global counting becomes expressible. Our proof techniques leverage the underflow phenomenon of floats. Via techniques from (Ahvonen et al. 2024), it follows that with *floats*, GPS-networks have the same expressive power as GNNs with counting global readout ($\mathrm{GNN} + \mathrm{GC}$). We also show that float-based GTs have the same expressive power as PL with the counting global modality ($\mathrm{PL} + \mathrm{GC}$), stated in Theorem 14. We note that the logic $\mathrm{PL} + \mathrm{GC}$ was recently used to study links between entropy and description complexity (Jaakkola, Kuusisto, and Vilander 2025). We also briefly discuss our characterizations with floats in restriction to word-shaped graphs, how the float results generalize for graph classification and non-Boolean classification tasks, and how they could be modified to cover positional encodings. Finally, we emphasize that the paper focuses exclusively on theoretical analysis and does not involve any experimental evaluation.

Related Work. To our knowledge, the expressive power of graph transformers and GPS-networks has not yet been studied from the perspective of logic. Regarding transformers *over words*, the closest to our work are (Jerad et al. 2025; Li and Cotterell 2025), which characterized fixed-precision transformers that use 'causal masking' with soft-attention, average hard-attention and unique hard-attention, via the past fragment of linear temporal logic (LTL). In contrast, our characterizations exclude masking and focus on transformers on graphs rather than words. Similar characterizations via variations of LTL are given, for instance, in (Yang, Chiang, and Angluin 2024; Yang, Cadilhac, and Chiang 2025).

Regarding logic-based lower and upper bounds for the expressive power of transformers over words, we mention (Chiang, Cholak, and Pillay 2023), which established the logic $\mathsf{FOC}[+;\mathsf{MOD}]$ as an upper bound for the expressivity of fixed-precision transformer encoders and as a lower bound for transformer encoders working with reals. Merrill and Sabharwal (2023) showed that first-order logic with majority quantifiers is an upper bound for log-precision transformers. Yang and Chiang (2024) gave a lower bound for future-masked soft-attention transformers with unbounded input size via the past fragment of Minimal Tense Logic with counting terms ($\mathsf{K}_t[\#]$). Barceló et al. (2024) established two lower bounds for hard-attention transformers working with reals: first-order logic extended with unary numerical predicates (FO(Mon)) in the case of unique hard-attention and linear temporal logic extended with unary numerical predicates and counting formulas (LTL(C,+)) for average hard-attention. For a more comprehensive study of precise characterizations and upper/lower bounds on the expressive power of different transformer architectures on words, see the survey (Strobl et al. 2024).

Regarding non-logic-based studies of the expressive power of graph transformers, Kreuzer et al. (2021) showed that graph transformers with positional encodings are universal function approximators in the non-uniform setting. In the uniform setting, Rosenbluth et al. (2024) proved that $\mathrm{GNN} + \mathrm{GCs}$ and GPS-networks using reals have incomparable expressive power w.r.t. function approximation. By contrast, this paper proves that with floats, $\mathrm{GNN} + \mathrm{GCs}$ and GPS-networks are equally expressive w.r.t. both graph and vertex properties, and also w.r.t. non-Boolean graph classification which amounts to expressing functions over floating-point numbers on graphs. Recently, Sälzer, Schwarzentruber, and Troquard (2025) studied the complexity of verifying float-based GNNs, and showed that it is PSPACE-complete.

Barceló et al. (2020) pioneered work on the expressive power of graph neural networks by characterizing aggregate-combine GNNs with reals in restriction to FO via graded modal logic, and the same GNNs extended with a global readout mechanism (essentially the same as our $\mathrm{GNN} + \mathrm{GCs}$) with the two-variable fragment of FO with counting quantifiers. Grohe (2023) connected GNNs to the circuit complexity class TC^0, utilizing dyadic rationals. Ahvonen et al. (2024) gave logical characterizations of recurrent and constant-iteration GNNs with both reals and floats, making similar assumptions to ours on float operations such as summation. Pfluger, Tena Cucala, and Kostylev (2024) characterized recurrent GNNs that use reals in terms of the graded two-way μ-calculus relative to a logic LocMMFP. We also mention (Benedikt et al. 2024), which characterized GNNs with bounded activation functions via logics involving Presburger quantifiers.

2 Preliminaries

We let $\mathbb{Z}_+$ denote the set of positive integers and $\mathbb{N}$ the set of non-negative integers. For $n \in \mathbb{Z}_+$, define $[n] := \{1, \ldots, n\}$. Also, set $\mathbb{B} := \{0, 1\}$. For a set S, we let $\mathcal{M}(S)$ denote the set of multisets over S, i.e., the set of functions $S \to \mathbb{N}$. For a multiset M, $M_{|k}$ denotes the **k-restriction** of M, i.e., the multiset given by $M_{|k}(x) = \min\{M(x), k\}$.

For $\mathbf{x} \in X^n$ and $i \in [n]$, let $\mathbf{x}_i$ denote the ith component of $\mathbf{x}$. For a matrix $M \in X^{n \times m}$, we use $M_{i,*}$, $M_{*,j}$ and

$M_{i,j}$ to denote, respectively, the ith row (from the top), the jth column (from the left), and the jth entry in the ith row of M. For a sequence $(M^{(1)}, \dots, M^{(k)})$ of matrices in $X^{n\times m}$, their **concatenation** is the matrix $M \in X^{n\times km}$ such that $M^{(\ell)}_{i,j} = M_{i,(\ell-1)m+j}$ for all $\ell \in [k]$. For non-empty sets X and Y, let X^+ denote the set of non-empty sequences over X, while $f\colon X^+ \to^{|\cdot|} Y^+$ is a notation for functions that map each sequence in X^+ to a sequence of the same length in Y^+.

2.1 Graphs and Feature Maps

For a finite domain $D \neq \emptyset$, a dimension $d \in \mathbb{N}$ and a non-empty set W, a **feature map** is a function $f\colon D \to W^d$ that maps each $x \in D$ to a **feature vector** $f_x \in W^d$. Typically, D consists of graph vertices and W is $\mathbb{R}$ or a set of floating-point numbers. If D is ordered by $<^D$, then we can identify f with a **feature matrix** $M \in W^{|D|\times d}$, where the row $M_{i,*}$ contains the feature vector of the ith element of D w.r.t. $<^D$, the column $M_{*,j}$ containing the jth vector components.

We work with vertex-labeled directed graphs, allowing self-loops, and simply refer to them as *graphs*. Let LAB denote a countably infinite set of **(vertex) label symbols**. We assume an ordering $<^{\mathrm{LAB}}$ of LAB, also inducing an ordering $<^L$ of every $L \subseteq \mathrm{LAB}$. Finite subsets of LAB are denoted by Π. Given $\Pi \subseteq \mathrm{LAB}$, a Π**-labeled graph** is a tuple $\mathcal{G} = (V, E, \lambda)$, where V is a finite non-empty set of **vertices**, $E \subseteq V \times V$ is a set of **edges** and $\lambda\colon V \to 2^\Pi$ a **vertex labeling function**. Here, 2^Π denotes the power set of Π. For convenience, we set $V(\mathcal{G}) := V$, $E(\mathcal{G}) := E$, and $\lambda(\mathcal{G}) := \lambda$. A **pointed graph** is a pair $(\mathcal{G}, v)$ with $v \in V$. The set of **out-neighbors** of $v \in V(\mathcal{G})$ is $\mathrm{Neigh}_{\mathcal{G}}(v) := \{\, u \mid (v,u) \in E \,\}$. We may identify λ with a feature map $\lambda'\colon V \to \mathbb{B}^{|\Pi|}$ where $\lambda'(v)_i = 1$ if the ith vertex label symbol (w.r.t. $<^\Pi$) is in $\lambda(v)$ and else $\lambda'(v)_i = 0$. Thus, Π-labeled graphs can be seen as graphs where each vertex is labeled with a single vector from $\mathbb{B}^{|\Pi|}$. We assume w.l.o.g. that for any graph $\mathcal{G}$, $V(\mathcal{G}) = [n]$ for some $n \in \mathbb{Z}_+$. Hence, we can identify feature maps of graphs with feature matrices and use labeling functions, feature maps and feature matrices interchangeably.

2.2 Graph Transformers and GNNs

We next discuss the computing architectures relevant to this article: graph transformers, GPS-networks, and GNNs. We view them as vertex classifiers that produce Boolean classifications. For this section, fix an arbitrary Π-labeled graph $\mathcal{G} = (V, E, \lambda)$ with $|\Pi| = \ell$ and $|V| = n$. In what follows, we will often speak of the input/hidden/output dimension of learning models and their components. For brevity, we abbreviate these to I/H/O dimension.

Basic Components. A multilayer perceptron (or feedforward neural network) can be intuitively understood as a structure that propagates an input through a series of layers, where in each layer every "neuron" computes a weighted sum of its incoming signals, applies an activation function, and passes the result forward to the next layer. The formal details, along with some auxiliary notions, are given below.

A **perceptron layer** P of I/O dimension (d_I, d_O) consists of a **weight matrix** $W \in \mathbb{R}^{d_O\times d_I}$, a **bias term** $b \in \mathbb{R}^{d_O\times 1}$ and an **activation function** $\alpha\colon \mathbb{R} \to \mathbb{R}$. Given an input vector $\mathbf{x} \in \mathbb{R}^{d_I}$, P computes the vector $P(\mathbf{x}) := \alpha\big(b + W\mathbf{x}\big)$, where α is applied element-wise. A **multilayer perceptron** (MLP) F of I/O dimension (d_I, d_O) is a sequence $(P^{(1)}, \dots, P^{(m)})$ of perceptron layers, where each $P^{(i)}$ has I/O dimension (d_{i-1}, d_i), where $d_0 = d_I$ and $d_m = d_O$ and $P^{(m)}$ uses the identity activation function. Given a vector $\mathbf{x} \in \mathbb{R}^{d_I}$, F computes the vector $F(\mathbf{x}) := P^{(m)}(\cdots P^{(2)}(P^{(1)}(\mathbf{x}))\cdots)$. For a matrix $X \in \mathbb{R}^{n\times d_I}$, we let $F(X)$ denote the $\mathbb{R}^{n\times d_O}$-matrix, where F is applied row-wise for X. An MLP is α**-activated** if every layer uses α, except the last, which always uses the identity function. Unless otherwise stated, MLPs are ReLU-activated, where $\mathrm{ReLU}(x) = \max(0, x)$. An MLP is **simple** if it is ReLU-activated and has only two perceptron layers.

Next, we introduce aggregation functions and readout gadgets, which are essential components of graph transformers, GPS-networks and graph neural networks. An **aggregation function** of dimension d_I is a function $\mathrm{AGG}\colon \mathcal{M}(\mathbb{R}^{d_I}) \to \mathbb{R}^{d_I}$ which typically is (pointwise) sum, max or mean. It is **set-based** if $\mathrm{AGG}(M) = \mathrm{AGG}(M_{|1})$ for all $M \in \mathcal{M}(\mathbb{R}^{d_I})$. A **readout gadget** of I/O dimension (d_I, d_O) is a tuple $R := (F, \mathrm{AGG})$, where F and AGG are as above. Given a matrix $X \in \mathbb{R}^{n\times d_I}$, it computes the matrix $R(X) \in \mathbb{R}^{n\times d_O}$ where each row is the same, defined by $R(X)_{i,*} := F\big(\mathrm{AGG}(\{\!\{\, X_{j,*} \mid j \in [n] \,\}\!\})\big)$.

Graph Neural Networks. A (constant-iteration) graph neural network can be intuitively viewed as a distributed system in which vertices of the input graph exchange information synchronously for a fixed number of rounds. In each round, every vertex updates its feature vector based on its previous feature vector and an aggregated representation of the feature vectors of its out-neighbors. The formal details follow below.

A **message-passing layer** of dimension d is a pair $L = (\mathrm{COM}, \mathrm{AGG})$, where COM is an MLP of I/O dimension $(2d, d)$ and AGG is an aggregation function of dimension d. A **message-passing layer with counting global readout** of dimension d is a pair (L, R), where L is defined as above and R is a readout gadget of I/O dimension (d, d). A **message-passing layer with non-counting global readout** (L, R) of dimension d is defined analogously, but the aggregation function of R is set-based.

A **graph neural network** (GNN) over (Π, d) is a tuple $G = (P, L^{(1)}, \dots, L^{(k)}, C)$ where P is an MLP of I/O dimension (ℓ, d), each $L^{(i)} = (\mathrm{COM}^{(i)}, \mathrm{AGG}^{(i)})$ is a message passing layer of dimension d, and C is an MLP of I/O dimension $(d, 1)$ that induces a function $\mathbb{R}^d \to \mathbb{B}$ called a **(Boolean vertex) classification head**. The MLP C does not have to be ReLU-activated, and can use, e.g., the Heaviside function σ, defined such that $\sigma(x) = 1$ if $x > 0$ and $\sigma(x) = 0$ if $x \leq 0$. Over a graph $\mathcal{G}$, G computes a sequence $\lambda^{(0)}, \dots, \lambda^{(k)}$ of feature maps and a final feature map $G(\mathcal{G})$ as follows: $\lambda^{(0)} := P(\lambda)$ and $\lambda^{(i+1)} := \lambda^{(i)} + L^{(i+1)}\big(\lambda^{(i)}\big)$, where for each $v \in V$, $L^{(i+1)}\big(\lambda^{(i)}\big)$ maps v to the feature

vector

$$\mathrm{COM}^{(i+1)}\Big(\lambda_v^{(i)}, \mathrm{AGG}^{(i+1)}\big(\{\{\lambda_u^{(i)} \mid (v,u) \in E\}\}\big)\Big).$$

Finally, $G(\mathcal{G}) := C(\lambda^{(k)})$. The final feature map $G(\mathcal{G})$ labels each vertex with 0 or 1, and we say that G **accepts** a vertex v of $\mathcal{G}$ if $G(\mathcal{G})$ maps v to 1. Note that we follow the convention of (Rampásek et al. 2022; Rosenbluth et al. 2024) by including skip connections around message-passing layers, which refers to the fact that $\lambda^{(i+1)}$ is not simply defined as $L^{(i+1)}\big(\lambda^{(i)}\big)$. It is easy to see that GNNs have the same expressive power with and without skip connections.

We define **graph neural networks with counting global readout** ($\mathrm{GNN} + \mathrm{GCs}$) and **graph neural networks with non-counting global readout** ($\mathrm{GNN} + \mathrm{Gs}$) analogously, except each $L^{(i)}$ is a message-passing layer $(\hat{L}^{(i)}, R^{(i)})$ with counting and non-counting global readout, respectively. They behave analogously to GNNs, except that $\lambda^{(i+1)} := \hat{\lambda}^{(i+1)} + R^{(i+1)}(\hat{\lambda}^{(i+1)})$ where $\hat{\lambda}^{(i+1)} := \lambda^{(i)} + \hat{L}^{(i+1)}(\lambda^{(i)})$. In other words, each vertex's feature vector additionally reflects the global information aggregated from all vertices in the graph.

Example 1. We can define a GNN over $(\{p\}, 1)$ that accepts precisely the vertices that satisfy the following property $\mathcal{P}$: 'a vertex has at least 5 out-neighbors that have the label p'. The initial MLP assigns the feature vector (1) to each vertex labeled p, and (0) to others. The first (and only) message-passing layer uses summation as the aggregation function; if the sum is at least 5, the vertex has the property $\mathcal{P}$. Similarly, a $\mathrm{GNN} + \mathrm{G}$ can express that a graph contains a vertex with the property $\mathcal{P}$ by adding a second layer that sums over all vertices in the graph, and a $\mathrm{GNN} + \mathrm{GC}$ can express, e.g., that a graph contains exactly 3 vertices with the property $\mathcal{P}$.

Self-Attention and Graph Transformers. A graph transformer is intuitively quite similar to a GNN; each vertex of a graph updates its feature vector in synchronous rounds. The key difference is that instead of aggregation and combination functions, a global attention mechanism is used to obtain information from all vertices. A GPS-network is then a variation of graph transformers that combines the attention mechanism with the update mechanism of GNNs. The formal details are given below.

A **self-attention head** H of I/H dimension (d, d_h) over $\mathbb{R}$ is defined w.r.t. an **attention-function** $\alpha\colon \mathbb{R}^+ \to^{|\cdot|} \mathbb{R}^+$ and three $\mathbb{R}^{d \times d_h}$-matrices: the **query-matrix** W_Q, the **key-matrix** W_K and the **value-matrix** W_V. Given a matrix $X \in \mathbb{R}^{n \times d}$, it computes the $n \times d_h$-matrix

$$H(X) := \alpha\left(\frac{(XW_Q)(XW_K)^\mathsf{T}}{\sqrt{d_h}}\right)(XW_V),$$

where α is applied row-wise. A **self-attention module** of dimension d over $\mathbb{R}$ is a tuple $\mathrm{SA} := (H^{(1)}, \ldots, H^{(k)}, W_O)$, where each $H^{(i)}$ is a self-attention head of I/H dimension (d, d_h) and $W_O \in \mathbb{R}^{kd_h \times d}$ is an **output matrix**. Let $\mathcal{H}(X)$ denote the concatenation of $H^{(1)}(X), \ldots, H^{(k)}(X)$. Now, SA computes the matrix $\mathrm{SA}(X) := \mathcal{H}(X)W_O$. For brevity, we may omit 'self' from 'self-attention'.

A **transformer layer** of dimension d is a pair $(\mathrm{SA}, \mathrm{FF})$, where SA is an attention module and FF is an MLP of dimension d. A **GPS-layer** of dimension d is a tuple $(\mathrm{SA}, \mathrm{MP}, \mathrm{FF})$, where MP is a message-passing layer.

A **graph transformer** (GT) over (Π, d) is a tuple $T = (P, L^{(1)}, \ldots, L^{(k)}, C)$, where P and C are as for GNNs and each $L^{(i)}$ is a transformer layer $(\mathrm{SA}^{(i)}, \mathrm{FF}^{(i)})$ of dimension d. A **GPS-network** N over (Π, d) is defined like a GT except that each $L^{(i)}$ is a GPS-layer $(\mathrm{SA}^{(i)}, \mathrm{MP}^{(i)}, \mathrm{FF}^{(i)})$ of dimension d. This definition does not include the optional normalization layers. For more about normalization layers, see e.g. (Rampásek et al. 2022). Analogously to a GNN, a GPS-network N computes over a graph $\mathcal{G}$ a sequence of feature maps and a final feature map $N(\mathcal{G})$ as follows: $\lambda^{(0)} := P(\lambda)$,

$$\begin{aligned}
\lambda_B^{(i+1)} &:= \lambda^{(i)} + B^{(i+1)}\big(\lambda^{(i)}\big), \text{ where } B \in \{\mathrm{SA}, \mathrm{MP}\},\\
\lambda_{\mathrm{SA+MP}}^{(i+1)} &:= \lambda_{\mathrm{SA}}^{(i+1)} + \lambda_{\mathrm{MP}}^{(i+1)},\\
\lambda^{(i+1)} &:= \lambda_{\mathrm{SA+MP}}^{(i+1)} + \mathrm{FF}^{(i+1)}\big(\lambda_{\mathrm{SA+MP}}^{(i+1)}\big).
\end{aligned}$$

Finally, $N(\mathcal{G}) := C(\lambda^{(k)})$. A GT T computes a feature map $T(\mathcal{G})$ analogously to N, but without the modules $\mathrm{MP}^{(i)}$:

$$\lambda^{(i+1)} := \lambda_{\mathrm{SA}}^{(i+1)} + \mathrm{FF}^{(i+1)}\big(\lambda_{\mathrm{SA}}^{(i+1)}\big).$$

Acceptance for GTs and GPS-networks is defined analogously to GNNs.

We focus on the two most commonly used attention functions. For $\mathbf{x} \in \mathbb{R}^\ell$, let $\mathcal{I}_\mathbf{x} = \{\, i \in [\ell] \mid \mathbf{x}_i = \max(\mathbf{x}) \,\}$ where $\max$ returns the largest entry in vector $\mathbf{x}$. We define the **average hard** (AH) and **softmax** functions:

1. $\mathrm{AH}(\mathbf{x})_i := \frac{1}{|\mathcal{I}_\mathbf{x}|}$ if $i \in \mathcal{I}_\mathbf{x}$ and $\mathrm{AH}(\mathbf{x})_i := 0$ otherwise,
2. $\mathrm{softmax}(\mathbf{x})_i := \frac{e^{x_i - b}}{\sum_{j \in [\ell]} e^{x_j - b}}$, where $b = \max(\mathbf{x})$.[1]

Example 2. For $\mathbf{x} = (5, 7, 1, 7) \in \mathbb{R}^4$, $\mathrm{AH}(\mathbf{x}) = (0, \frac{1}{2}, 0, \frac{1}{2})$ and $\mathrm{softmax}(\mathbf{x}) \approx (0.063, 0.468, 0.001, 0.468)$.

Attention heads that use AH or softmax are called average hard-attention heads and soft-attention heads, respectively. The same naming applies to attention modules, transformer layers, graph transformers, GPS-layers and GPS-networks. Later in Example 5 in Section 3, we show that there exists a soft-attention graph transformer that expresses the property: 'at least half of the vertices in the studied graph have the label symbol p'.

2.3 Logics

We define the logics used in this paper. Let Π be a finite set of vertex label symbols. With a first-order (FO) formula φ over Π, we mean a formula of first-order logic over the vocabulary that contains a unary relation symbol for each $p \in \Pi$ and a binary edge relation symbol E (equality is included). A Π-**formula of graded modal logic with the counting global modality** ($\mathrm{GML} + \mathrm{GC}$) is defined by the

[1] This is also known as the "stable" or "safe" softmax due to its numerical stability (Blanchard, Higham, and Higham 2019), in contrast to the version of softmax without the biases $-b$.

grammar $\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond_{\geq k}\varphi \mid \langle \mathsf{G} \rangle_{\geq k}\varphi$, where $p \in \Pi$ and $k \in \mathbb{N}$. We use $\vee$, $\rightarrow$ and $\leftrightarrow$ as abbreviations in the usual way, and for $D \in \{\Diamond, \langle \mathsf{G} \rangle\}$, we define that $D_{<k}\varphi := \neg D_{\geq k}\varphi$ and $D_{=k}\varphi := D_{\geq k}\varphi \wedge D_{<k+1}\varphi$.

The semantics of $\mathrm{GML} + \mathrm{GC}$ is defined over pointed graphs. In the field of modal logic, Π-labeled graphs are often called Kripke models. For a Π-formula φ of $\mathrm{GML} + \mathrm{GC}$ and a pointed Π-labeled graph $(\mathcal{G}, v)$, the truth of φ in $(\mathcal{G}, v)$ (denoted by $\mathcal{G}, v \models \varphi$) is defined as follows. $\mathcal{G}, v \models \top$ holds always. For $p \in \Pi$, $\mathcal{G}, v \models p$ iff $p \in \lambda(v)$. The cases $\neg$ and $\wedge$ are defined in the usual way. For diamonds,

$$\begin{array}{rcl} \mathcal{G}, v \models \Diamond_{\geq k}\varphi & \text{iff} & |\{u \in \mathrm{Neigh}_{\mathcal{G}}(v) \mid \mathcal{G}, u \models \varphi\}| \geq k \\ \mathcal{G}, v \models \langle \mathsf{G} \rangle_{\geq k}\varphi & \text{iff} & |\{u \in V(\mathcal{G}) \mid \mathcal{G}, u \models \varphi\}| \geq k. \end{array}$$

Graded modal logic with global modality ($\mathrm{GML} + \mathrm{G}$) is the fragment of $\mathrm{GML} + \mathrm{GC}$ where diamonds $\langle \mathsf{G} \rangle_{\geq k}$ are allowed only if $k = 1$. For simplicity, we let $\langle \mathsf{G} \rangle := \langle \mathsf{G} \rangle_{\geq 1}$. **Graded modal logic** (GML) is the fragment of $\mathrm{GML} + \mathrm{G}$ without diamonds $\langle \mathsf{G} \rangle$. **Modal logic** (ML) is the fragment of GML where we allow diamonds $\Diamond_{\geq k}$ only if $k = 1$, and we let $\Diamond := \Diamond_{\geq 1}$. **Propositional logic** (PL) is the fragment of ML without diamonds $\Diamond$. The logics $\mathrm{ML} + \mathrm{GC}$, $\mathrm{ML} + \mathrm{G}$, $\mathrm{PL} + \mathrm{GC}$ and $\mathrm{PL} + \mathrm{G}$ are defined in the expected way.

Example 3. The property 'no vertex is a dead-end' is expressed by the $\mathrm{GML} + \mathrm{GC}$-formula $\langle \mathsf{G} \rangle_{=0} \Diamond_{=0} \top$. No $\mathrm{GML} + \mathrm{GC}$-formula can express the property 'at least half of the vertices in the graph have label p'. A formal proof via graded bisimulations is straightforward.

2.4 Equivalence of Vertex Classifiers

A **(vertex) property over** Π is a mapping f that assigns to each Π-labeled graph $\mathcal{G}$ a feature map $\lambda' \colon V(\mathcal{G}) \to \{0, 1\}$ and is invariant under isomorphisms. A **vertex classifier** is any object C that defines a vertex property. Note that each of our computing models is a vertex classifier. Each Π-formula φ of any logic introduced above also corresponds to a vertex classifier (where for FO, φ must have a single free variable) which maps each Π-labeled graph $\mathcal{G}$ to the feature map λ_φ with $\lambda_\varphi(v) = 1$ if $\mathcal{G}, v \models \varphi$ and $\lambda_\varphi(v) = 0$ otherwise.

We say that vertex classifiers C_1 and C_2 are **equivalent** if they define the same vertex property. Two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ of vertex classifiers **have the same expressive power** if for each $C_1 \in \mathcal{C}_1$ there is an equivalent $C_2 \in \mathcal{C}_2$, and vice versa. We say that $\mathcal{C}_1$ and $\mathcal{C}_2$ **have the same expressive power relative to** FO, if for each vertex property f definable by a formula $\varphi(x) \in \mathrm{FO}$, there is some $C_1 \in \mathcal{C}_1$ that defines f if and only if there is some $C_2 \in \mathcal{C}_2$ that defines f.

3 Characterizing Real-Based Transformers

We provide characterizations of the expressive power of GPS-networks and of GTs, over the reals and relative to FO, in terms of the logics $\mathrm{GML} + \mathrm{G}$ and $\mathrm{PL} + \mathrm{G}$, respectively. The characterizations apply to both soft-attention and average hard-attention. We start with GPS-networks.

Theorem 4. *Relative to* FO*, the following have the same expressive power:* $\mathrm{GML} + \mathrm{G}$*, soft-attention* GPS*-networks, and average hard-attention* GPS*-networks.*

We discuss Theorem 4 giving a proof. An interesting comparison is to the results of Barceló et al. (2020), who prove that relative to FO, GNNs without transformer layers and without global readout have the same expressive power as GML. They also show that, when counting global readout is admitted, GNNs can express all of $\mathrm{GML} + \mathrm{GC}$.[2] Relative to FO, GPS-networks thus sit properly in the middle between GNNs and GNN + GCs: they can express global properties such as P_1 = 'the graph contains a vertex labeled p', but cannot express absolute global counting, such as P_2 = 'the graph contains at least 2 vertices labeled p'.

Let us also discuss absolute expressive power, dropping FO as a background logic. GNNs are by definition a special case of GPS-networks. Conversely, Property P_1 witnesses that GPS-networks are strictly more expressive than GNNs, also in an absolute sense. Likewise, Property P_2 shows that some GNN + GCs do not have an equivalent GPS-network. It remains open whether every GPS-network is equivalent to a GNN + GC. It is proved in (Rosenbluth et al. 2024) that this is not the case for graph classification, but the result does not immediately transfer to vertex classification.

While GPS-networks cannot express global properties that involve absolute counting, they can express global properties with relative counting, and so can GTs. This is not visible in Theorem 4 because such properties do not fall within FO. We demonstrate relative counting in the example below.

Example 5. There is a 1-layer soft-attention GT (P, L, C) that accepts precisely the vertices of those graphs that satisfy the property: 'at least half of the vertices in the graph have the label p'. The initial MLP P maps each feature vector to a 2-dimensional feature vector, where the first component encodes the labeling by p and the other is 0. The soft-attention module of L then has $W_Q = W_K = [0, 0]^\mathsf{T}$, $W_V = [1, 0]^\mathsf{T}$ and $W_O = [0, 1]$ and the MLP outputs a zero matrix. After the last skip connection, the second column of the matrix consists of the value x, where x tells the ratio of how many vertices have the label p; the final classification head C outputs 1 if $x \geq 0.5$ and 0 otherwise.

We now survey the proof of Theorem 4, full details are in the technical report (Ahvonen et al. 2025). The easier direction is to show that every $\mathrm{GML} + \mathrm{G}$-formula can be translated into an equivalent GPS-network. We extend the corresponding construction of (Barceló et al. 2020) for GML, using self-attention heads to handle subformulae of the form $\langle \mathsf{G} \rangle \varphi$.

Lemma 6. *For every* $\mathrm{GML} + \mathrm{G}$*-formula, there is an equivalent* GPS*-network. This applies to both soft-attention and average hard-attention.*

A notable difference to the proof of (Barceló et al. 2020) is that we use a step function as an activation function, rather than truncated ReLU. Intuitively, this is because truth values are represented as 0 and 1 in feature vectors, but both

[2] They actually show that GNNs with counting global readout capture all of C_2—the two-variable fragment of FO with counting quantifiers—but only on undirected graphs; this fails for directed graphs, as GNN + GCs cannot express, for instance, the C_2-formula $\exists y\, E(x, y) \wedge E(y, x)$.

soft-attention and average hard-attention may deliver an arbitrarily small (positive) fractional value and there seems to be no way to 'rectify' this into a 1 without using a step function. This is similar to what happens in GNNs that use arithmetic mean as the aggregation function (Schönherr and Lutz 2026).

The difficult direction in the proof of Theorem 4 is to show that every GPS-network that expresses an FO-property is equivalent to a $\mathrm{GML} + \mathrm{G}$-formula. In (Barceló et al. 2020), this direction is proved by first showing that GNNs are invariant under graded bisimulation and then applying a van Benthem/Rosen-style result from finite model theory (Otto 2019) which says that every FO-formula invariant under graded bisimulation is equivalent to a GML-formula. GPS-networks, however, are not invariant under graded bisimulations because these do not preserve global properties. We thus introduce a stronger version of graded bisimilarity that also takes into account the multiplicities with which graded bisimulation types are realized, and prove a corresponding van Benthem/Rosen theorem.

Let Π be a finite set of vertex label symbols. A **graded bisimulation** between two Π-labeled graphs $\mathcal{G}_1 = (V_1, E_1, \lambda_1)$ and $\mathcal{G}_2 = (V_2, E_2, \lambda_2)$ is a binary relation $Z \subseteq V_1 \times V_2$ that satisfies the following conditions:

atom for all $(v_1, v_2) \in Z$, $\lambda_1(v_1) = \lambda_2(v_2)$.

graded forth for all $(u_1, u_2) \in Z$, for all $k \geq 1$: for all pairwise distinct $v_1, \ldots, v_k \in \mathrm{Neigh}_{\mathcal{G}_1}(u_1)$ there are pairwise distinct $v'_1, \ldots, v'_k \in \mathrm{Neigh}_{\mathcal{G}_2}(u_2)$ with $(v_1, v'_1), \ldots, (v_k, v'_k) \in Z$.

graded back for all $(u_1, u_2) \in Z$, for all $k \geq 1$: for all pairwise distinct $v'_1, \ldots, v'_k \in \mathrm{Neigh}_{\mathcal{G}_2}(u_2)$ there are pairwise distinct $v_1, \ldots, v_k \in \mathrm{Neigh}_{\mathcal{G}_1}(u_1)$ with $(v_1, v'_1), \ldots, (v_k, v'_k) \in Z$.

We write $(\mathcal{G}_1, v_1) \sim (\mathcal{G}_2, v_2)$ if there is a graded bisimulation Z between $\mathcal{G}_1$ and $\mathcal{G}_2$ with $(v_1, v_2) \in Z$.

A **graded bisimulation type over** Π is a maximal set t of Π-labeled pointed graphs such that $(\mathcal{G}_1, v_1) \sim (\mathcal{G}_2, v_2)$ for all $(\mathcal{G}_1, v_1), (\mathcal{G}_2, v_2) \in t$. For a pointed Π-labeled graph $(\mathcal{G}, v)$, we use $\mathsf{tp}_{\mathcal{G}}(v)$ to denote the unique graded bisimulation type t over Π such that $(\mathcal{G}, v) \in t$.

Pointed graphs $(\mathcal{G}_1, v_1)$, $(\mathcal{G}_2, v_2)$ are **global-ratio graded bisimilar**, written $(\mathcal{G}_1, v_1) \sim_{G\%} (\mathcal{G}_2, v_2)$, if $(\mathcal{G}_1, v_1) \sim (\mathcal{G}_2, v_2)$ and there exists a rational number $q > 0$ such that for every graded bisimulation type t,

$$|\{v \in V_1 \mid \mathsf{tp}_{\mathcal{G}_1}(v) = t\}| = q \cdot |\{v \in V_2 \mid \mathsf{tp}_{\mathcal{G}_2}(v) = t\}|.$$

Note that the ratios between graded bisimulation types above are closely related to relative counting as in Example 5.

Example 7. We illustrate that global-ratio graded bisimilarity provides a middle ground between graded bisimilarity and isomorphism.

First consider the $\{p\}$-labeled graphs $\mathcal{G}_1 = (\{v_1\}, \emptyset, \lambda_1)$ with $\lambda_1(v_1) = \{p\}$ and $\mathcal{G}_2 = (\{u_1, u_2\}, \emptyset, \lambda_2)$ with $\lambda_2(u_1) = \{p\}$ and $\lambda_2(u_2) = \emptyset$. Then $(\mathcal{G}_1, v_1) \sim (\mathcal{G}_2, u_1)$, but $(\mathcal{G}_1, v_1) \not\sim_{G\%} (\mathcal{G}_2, u_1)$ as the graded bisimulation type of $(\mathcal{G}_2, u_2)$ does not occur in $\mathcal{G}_1$.

Now consider the graph $\mathcal{G}_3 = (\{w_1, w_2\}, \emptyset, \lambda_3)$ with $\lambda_3(w_1) = \lambda_3(w_2) = \{p\}$. Then $(\mathcal{G}_1, v_1) \sim_{G\%} (\mathcal{G}_3, w_1)$, taking $q = \frac{1}{2}$, but $\mathcal{G}_1$ and $\mathcal{G}_3$ are not isomorphic.

A vertex classifier such as a GPS-network or an FO-formula $\varphi(x)$ is **invariant under** $\sim_{G\%}$ if for all pointed graphs $(\mathcal{G}_1, v_1)$ and $(\mathcal{G}_2, v_2)$, $(\mathcal{G}_1, v_1) \sim_{G\%} (\mathcal{G}_2, v_2)$ implies that $\mathcal{G}_1 \models \varphi(v_1)$ if and only if $\mathcal{G}_2 \models \varphi(v_2)$. A layer-by-layer analysis of GPS-networks shows the following.

Lemma 8. *Let N be a soft-attention or average hard-attention* GPS*-network. Then N is invariant under $\sim_{G\%}$.*

It follows that GPS-networks cannot distinguish (G_1, v_1) and (G_3, w_1) from Example 7. In contrast, (G_1, v_1) and (G_2, u_1) can be distinguished: by Lemma 6, one can employ a GPS-network equivalent to the $\mathrm{GML} + \mathrm{G}$-formula $\langle \mathsf{G} \rangle \neg p$.

We now give the announced van Benthem/Rosen theorem.

Theorem 9. *For every* FO*-formula $\varphi(x)$ over Π, the following are equivalent:*

1. *φ is invariant under $\sim_{G\%}$;*
2. *φ is equivalent to a $\mathrm{GML} + \mathrm{G}$-formula over all (finite!) Π-labeled pointed graphs.*

The direction "2 $\Rightarrow$ 1" of Theorem 9 is easy to prove by a straightforward induction on the structure of $\mathrm{GML} + \mathrm{G}$-formulae. The difficult part is to show the "1 $\Rightarrow$ 2" direction, that is, every FO formula $\varphi(x)$ invariant under $\sim_{G\%}$ is equivalent to a $\mathrm{GML} + \mathrm{G}$-formula. To achieve this, we combine and extend techniques from (Otto 2004) and (Otto 2019). The former provides a van Benthem/Rosen theorem that links global (ungraded) bisimulation to $\mathrm{ML} + \mathrm{G}$, and the latter a theorem of the same kind that links graded bisimulation to GML.

Our proof consists of a sequence of results saying that if an FO-formula $\varphi(x)$ is invariant under $\sim_{G\%}$, then it is also invariant under certain other notions of bisimulation that become increasingly weaker.[3] We finally arrive at a notion of bisimulation that is called global c-graded ℓ-bisimulation, where c is a counting bound and ℓ a depth bound, both derived from $\varphi(x)$. Importantly, this notion of bisimulation has only a finite number of bisimulation types, and each type can be distinguished from the others using a characteristic $\mathrm{GML} + \mathrm{G}$-formula. We can thus construct the desired $\mathrm{GML} + \mathrm{G}$-formula by taking the disjunction of all characteristic formulae for bisimulation types in which $\varphi(x)$ is true. To make sure that the ratio-property of $\sim_{G\%}$ is respected, we replace several constructions from (Otto 2004) with more careful ones.

Combining Theorem 9 and Lemma 8 completes the proof sketch of Theorem 4. Moreover, as a special case, the construction in the proof of Lemma 6 shows that $\mathrm{PL} + \mathrm{G}$-formulae can be translated into GTs with both soft-attention and average hard-attention. A minor extension of our techniques used to prove Theorem 9, then shows the following.

Theorem 10. *Relative to* FO*, the following have the same expressive power:* $\mathrm{PL} + \mathrm{G}$*, soft-attention* GT*s, average hard-attention* GT*s.*

[3] While this provides a good intuition, it is not strictly true. For technical reasons, the intermediate notions of bisimulation sometimes get stronger in certain respects, e.g. they may use up-and-down features as used for modal logics with the converse modality.

4 Characterizing Float-Based Transformers

We give characterizations of GPS-networks and GTs based on floating-point numbers via the logics $\mathrm{GML} + \mathrm{GC}$ and $\mathrm{PL} + \mathrm{GC}$, respectively. Before the characterizations, we introduce floats and float-based GPS-networks and GTs.

4.1 Floating-Point Numbers and Arithmetic

We define the concepts of floating-point numbers based on the IEEE 754 standard (IEEE 2019). Let $p, q \in \mathbb{Z}_+$. A **floating-point number** (over p and q) is a string of the form

$$b_0 b_1 \cdots b_{p+q} \in \{0,1\}^{p+q+1}.$$

The bit b_0 is called the **sign**, the string $\mathbf{e} = b_1 \cdots b_q$ the **exponent** and $\mathbf{s} = b_{q+1} \cdots b_{p+q}$ the **significand**. Let $a = 2^{p-1}$, $b = 2^{q-1} - 1$, and let e and s be the non-negative integers represented in binary by $\mathbf{e}$ and $\mathbf{s}$. Then the above floating-point number is interpreted as the real number $(-1)^{b_0} \frac{s}{a} 2^{e-b}$. As an exception, the float with $\mathbf{s} = 0^p$, $\mathbf{e} = 1^q$ and $b_0 = 0$ (resp. $b_0 = 1$) corresponds to ∞ (resp. $-\infty$). When the context is clear, we identify a float with the real number (or ∞, $-\infty$) that it represents. A float is **normalized** if $b_{q+1} \neq 0$, and **subnormalized** if $b_{q+1} = 0$ and $\mathbf{e} = 0^q$. A **floating-point format** $\mathcal{F}(p, q)$ over p and q consists of all normalized and subnormalized floating-point numbers over p and q and the symbols ∞, $-\infty$, and NaN ('not-a-number'), and when clear we may write $\mathcal{F}$ instead of $\mathcal{F}(p, q)$.

Next, we discuss basic arithmetic operations over floating-point formats: addition $+$, subtraction $-$, multiplication $\cdot$, division $\div$ and square root $\sqrt{x}$. The definition of each of the operations is to first "compute" to unlimited precision in real arithmetic (extended with ∞ and $-\infty$) and then rounding to the nearest float in the format, with ties rounding to the float with an even least significant bit. Undefined results, such as $\frac{\infty}{\infty}$, are mapped to NaN. If any input is NaN, the output is NaN, i.e., our NaN is *silent* and propagated through the computation. The exponential function $\exp(x)$ over floats is not a basic operation and is implemented in a standard way, using basic operations, range reductions and polynomial approximations. Background and a discussion on these concepts is in (Ahvonen et al. 2025).

4.2 Float-Based Transformers

We introduce float-based GTs, GPS-networks and GNNs. To define them, we replace reals with floats, but we must also carefully specify how float operations are performed. One reason is that many float operations (e.g. sum) are not associative due to rounding errors between operations. Thus, switching the order of operations can affect the outcome.

Example 11. Consider the sum of the real numbers -1, 1 and 4 representable in the format $\mathcal{F}(2, 3)$: here we have $(-1 + 1) + 4 = 0 + 4 = 4$ but $-1 + (1 + 4) = -1 + 4 = 3$. Note that in the latter equation, the precise sum of 1 and 4 would be 5, which is not representable in the format $\mathcal{F}(2, 3)$ and is thereby rounded to the nearest number; both 4 and 6 are equally near, and 4 has the even least significant bit.

The $\mathrm{softmax}$ function, the sum aggregation function and some matrix multiplications in attention heads take a sum over the features of vertices in the studied graph, and are thus affected by this non-associativity issue. In the worst case, this can violate the isomorphism invariance of these learning models, which is undesirable. For example, in typical real-life implementations, the set V of vertices in the studied graph is associated with some implementation-related, implicit linear order $<^V$ (that is not part of the actual graph). Then isomorphism invariance can be violated if the sum aggregation sums in the order $<^V$. Hence, it is better to order the floats instead of the vertices. We make the natural assumption that floats are always summed in increasing order, which results in models that are isomorphism invariant. This is further justified by numerical stability (Wilkinson 1959; Robertazzi and Schwartz 1988; Higham 1993).

Given a floating-point format $\mathcal{F}$, we let $\mathrm{SUM}_{\mathcal{F}}$ denote the operation that maps a multiset N of floats to the sum $f_1 + \cdots + f_\ell$ where each f_i appears $N(f_i)$ times and the floats appear and are summed in increasing order. We recall from (Ahvonen et al. 2024) the following important result on *boundedness* of float sums.

Proposition 12. *For all floating-point formats $\mathcal{F}$, there exists a $k \in \mathbb{N}$ such that for all multisets M over floats in $\mathcal{F}$, we have $\mathrm{SUM}_{\mathcal{F}}(M) = \mathrm{SUM}_{\mathcal{F}}(M_{|k})$.*

To see that the above proposition holds, observe that $\mathrm{SUM}_{\mathcal{F}}$ repeatedly adds the same float the number of times it appears in the sum. In the format $\mathcal{F}(2, 2)$, the number $\frac{1}{2} \cdot 2^{-1} = 0.25$ is exactly representable. Summing 0.25 repeatedly in this format gives 0.25 after one addition, 0.50 after two additions, 0.75 after three additions, 1.0 after four additions, and 1.0 after five additions, since 1.25 is not in $\mathcal{F}(2, 2)$ and rounds to 1.0. Thus, the sum "saturates beyond a threshold". Using this phenomenon, it is easy to obtain a proof of Proposition 12

We say that an aggregation function AGG is **bounded** if there exists a $k \in \mathbb{N}$ such that $\mathrm{AGG}(M) = \mathrm{AGG}(M_{|k})$ for all M. Apart from sum, also mean aggregation is similarly bounded. Furthermore, we assume that $\mathrm{softmax}$ is implemented for a floating-point format $\mathcal{F}$ by using the above sum $\mathrm{SUM}_{\mathcal{F}}$ in the denominator, and the remaining operations are carried out in the natural order, i.e., we first calculate the bias b, then values $x_j - b$, then the exponents, and finally the division. Likewise, we assume AH is implemented for $\mathcal{F}$ by calculating the denominator in $\frac{1}{|\mathcal{I}_{\mathbf{x}}|}$ using the same approach as (Li and Cotterell 2025), i.e., calculating it as $\mathrm{SUM}_{\mathcal{F}}(M)$, where M is the multiset over $\mathcal{F}$ consisting of precisely $|\mathcal{I}_{\mathbf{x}}|$ instances of the float 1, and then performing the division.

Float-Based Learning Models. *Floating-point* GT*s, denoted by* $\mathrm{GT}[\mathrm{F}]$*, are defined in the same way as* GT*s based on reals, except that they use floats in feature vectors and float operations where the order of operations is as specified above. Likewise for* GPS*-networks,* GNN*s,* MLP*s, etc. We further assume that these models always use aggregation functions that are bounded. This is a natural assumption as sum, max and mean are all bounded on floats by the above findings.*

We call these learning models **simple** when the MLPs are

simple[4] and the aggregation functions are $\mathrm{SUM}_{\mathcal{F}}$. In fact, GTs and GPS-networks were originally defined based on simple MLPs (Dwivedi and Bresson 2020; Rampásek et al. 2022). We do not fix a single float format for all GT[F]s, GPS[F]-networks, GNN[F]s, etc.; instead, each of them is associated with *some* float format. In our translations, the format is assumed arbitrary when translating them into logics, but can be chosen freely in the other direction.

4.3 Characterizations

Next, we provide logical characterizations for GT[F]s and GPS[F]-networks with both soft and average hard-attention. The characterizations are absolute, i.e., they do not require relativizing to a background logic such as FO. Our float-based GT[F]s and GPS[F]-networks also do not require step function activated MLPs aside from the classification heads.

First, we make an observation about float-based multiplication relevant to our translation techniques. When multiplying two floating-point numbers that are very close to zero, **underflow** occurs: the exact result is so small that all significant bits are lost, and the output is 0. For instance, underflow can occur in attention heads in some matrix multiplications. The following proposition demonstrates this phenomenon. In the proposition and the proof that follows, we identify each float with the real number that it represents.

Proposition 13. *Let $\mathcal{F}$ be a floating-point format, let f be the smallest positive float in $\mathcal{F}$ and let k be some even integer such that $\frac{k}{2}$ is accurately representable in $\mathcal{F}$. For all $F \in \mathcal{F}$, $|F| \leq |\frac{1}{k}|$ if and only if $F \cdot (\frac{k}{2} f) = 0$.*

For the proof, note that since $\frac{k}{2}$ is accurately representable in $\mathcal{F}$, then so is also the precise product $\frac{k}{2}f$. Now, we observe that $|F| \leq |\frac{1}{k}|$ if and only if the precise product $F \cdot (\frac{k}{2} f)$ belongs to the closed interval $[-\frac{1}{2}f, \frac{1}{2}f]$. All numbers in this interval round to 0.

Now, we give our logical characterization for GT[F]s. Recall that float-based computing models by definition use bounded aggregation functions, and as explained, this is a natural assumption. By 'constant local aggregation functions', we intuitively mean that in message-passing layers, vertices cannot distinguish if a message was received from an out-neighbor or from any other vertex.

Theorem 14. *The following have the same expressive power: $\mathrm{PL} + \mathrm{GC}$, soft-attention $\mathrm{GT}[\mathrm{F}]$s and average hard-attention $\mathrm{GT}[\mathrm{F}]$s (and $\mathrm{GNN} + \mathrm{GC}[\mathrm{F}]$s with constant local aggregation functions). This also holds when the $\mathrm{GT}[\mathrm{F}]$s and $\mathrm{GNN} + \mathrm{GC}[\mathrm{F}]$s are simple.*

We provide a more detailed proof for Theorem 14 in the technical report (Ahvonen et al. 2025), but we *sketch* the proof here. In the direction from GT[F]s to logic, the general idea is that for each vertex v we simulate its feature vector $\mathbf{x}_v$ after each transformer layer by simulating each bit of $\mathbf{x}_v$ by a single formula, i.e., a sequence of formulae simulates the whole vector $\mathbf{x}_v$. As a last step, we combine these formulae recursively into a single formula that simulates the output of the classification head. There are two key insights for simulating bits. First, each 'local step' of a GT[F] where a vertex does not need to know the features of any other vertices (e.g. MLPs or matrix products XW_Q, XW_K and XW_V) can be expressed as a function $f_{\mathcal{F}} \colon \mathcal{F}^n \to \mathcal{F}^m$. As floats are bit strings, we can identify $f_{\mathcal{F}}$ with a *partial* function $f_{\mathbb{B}} \colon \{0,1\}^{kn} \to \{0,1\}^{km}$, where k is the number of bits in $\mathcal{F}$. PL is expressively complete for expressing Boolean combinations, i.e., each function $g \colon \{0,1\}^n \to \{0,1\}$ has an equivalent PL-formula as $g(\mathbf{x})$ is simply a Boolean combination of the values in $\mathbf{x}$. Thus, we can construct an equivalent PL-formula for each output bit of $f_{\mathbb{B}}$, and the full function $f_{\mathbb{B}}$ can be simulated by a sequence of formulae. Second, for the remaining 'non-local' steps, it suffices to know the features of other vertices in the 'global sense', i.e., the edges of the graph are not used. Due to Proposition 12, the float sums appearing in attention heads are bounded for some k, i.e., after k copies of a float F, further instances of F do not affect the sum. Since the attention heads sum over the features of all vertices, it suffices for a vertex to be able to distinguish a bounded number of each possible feature vector appearing in the graph, and we can count up to this bound with the counting global modality.

For the converse, to translate a $\mathrm{PL} + \mathrm{GC}$-formula φ into a simple GT[F], we use a similar strategy as with reals: we compute the truth values of the subformulae of φ one at a time, using multiple transformer layers per subformula. Again the truth value of each subformula is represented as 0 or 1 in the feature vector for each vertex, i.e., in the feature matrix there is a column for each subformula that is used to encode the truth value of the subformula in each vertex. There are also some auxiliary columns in the feature matrix. The translation involves making use of the properties of floats and floating-point operations. The operators $\neg$ and $\wedge$ are easy to handle by using the MLPs of the transformer layers. The hardest part is to simulate modalities $\langle \mathsf{G} \rangle_{\geq k}$ by using MLPs *and* attention modules. Assume that we are computing the truth value of a subformula of the form $\langle \mathsf{G} \rangle_{\geq k} \psi$ and the truth value of ψ has already been encoded into a column i of the current feature matrix. Then we build an attention head that checks if the number ℓ of 1s in the column i is at least k. Intuitively, we construct a query matrix W_Q and a key matrix W_K such that the matrix $\mathrm{softmax}\big((XW_Q)(XW_K)^{\mathsf{T}}/\sqrt{d_h}\big)$ has an entry in every row that has the value $\frac{1}{\ell}$ (or a suitable rounded value). A similar construction is also possible by using the average-hard function. Then due to Proposition 13, we can construct a value matrix which uses underflow to check if $\frac{1}{\ell} \leq \frac{1}{k}$. After that, by using MLPs, we can distinguish when $\ell \geq k$ and when $\ell < k$. This completes the proof sketch.

Before characterizing GPS[F]-networks, we prove a helpful characterization of float-based GNNs.

Theorem 15. *The following pairs have the same expressive power (denoted by $\equiv$): $\mathrm{GNN}[\mathrm{F}] \equiv \mathrm{GML}$, $\mathrm{GNN} + \mathrm{G}[\mathrm{F}] \equiv \mathrm{GML} + \mathrm{G}$ and $\mathrm{GNN} + \mathrm{GC}[\mathrm{F}] \equiv \mathrm{GML} + \mathrm{GC}$. This also holds when each type of $\mathrm{GNN}[\mathrm{F}]$ is simple.*

This theorem follows from the proof techniques of Theorem 3.2 of (Ahvonen et al. 2024), which showed that (sim-

[4] An exception is the final Boolean vertex classifier, which is otherwise a simple MLP but uses the Heaviside function.

ple) recurrent float GNNs are equally expressive as a recursive rule-based bisimulation invariant logic called the graded modal substitution calculus (GMSC). Unlike the float GNNs in that paper, our GNN[F]s are not recurrent, meaning they only scan the neighborhood of a vertex up to some fixed depth. The corresponding constant-iteration fragment of GMSC is GML. The techniques in (Ahvonen et al. 2024) generalize for global readouts and modalities, see (Ahvonen et al. 2025) for the technical details of the proof.

We now characterize float-based GPS-networks.

Theorem 16. *The following have the same expressive power:* GML + GC*, soft-attention* GPS[F]*-networks, average hard-attention* GPS[F]*-networks and* GNN + GC[F]*s. This also holds in the case where the* GPS[F]*-networks and* GNN + GC[F]*s are simple.*

The result follows from Theorems 14 and 15. Importantly, any transformer layer and message-passing layer can be simulated by a GPS-layer of a higher dimension by appending the inputs and outputs of the transformer and message-passing layer with zeros on the GPS side. The technical details of the proof are in (Ahvonen et al. 2025).

We make some final observations. As seen in Example 5, 'relative global counting' is expressible by real-based GTs. However, the same construction does not work for GT[F]s as, due to Proposition 12, the softmax-function and average hard function lose accuracy in a drastic way with large graphs. For the same reason, Lemma 8 fails with floats; GT[F]s and GPS[F]-networks are not invariant under the bisimilarity $\sim_{G\%}$. However, Theorems 14, 15 and 16 show that with float-based GTs, GPS-networks and GNNs, 'absolute counting' is possible (locally or globally depending on the model), since the matching logics can count.

We note that our results with floats immediately hold when restricted to word-shaped graphs, i.e., graphs where the domain is a prefix $[n]$ of positive integers, the edge relation is the successor relation over $[n]$, and $\lambda_{\mathbf{w}}(v)$ is a singleton for each vertex. For example, a GT over word-shaped graphs is just an 'encoder-only transformer without causal masking'. A popular example is BERT (Devlin et al. 2019), which is such a model inspired by (Vaswani et al. 2017). In the technical report (Ahvonen et al. 2025), we study unique hard-attention graph transformers over word-shaped graphs. In the same technical report, we also consider generalizations of our float results for graph classification tasks and non-Boolean classification tasks.

We also briefly discuss how our float results could be modified to cover positional encodings. Often, each GNN[F], GT[F], or GPS[F]-network $A = (P, L^{(1)}, \ldots, L^{(k)}, C)$ (with input dimension ℓ) is associated with a **positional encoding** (or PE) π over $\mathcal{F}$, i.e., an isomorphism-invariant mapping that assigns to each graph $\mathcal{G}$ a function $\pi(\mathcal{G})\colon V(\mathcal{G}) \to \mathcal{F}^{\ell}$. For example, a popular PE is LapPE (Rampásek et al. 2022). Now, A with π computes over $\mathcal{G}$ a sequence of feature maps similarly to A without π (see Section 2.2), but for each vertex v in $\mathcal{G}$, we define $\lambda_v^{(0)} := P(\lambda)_v + \pi(\mathcal{G})_v$. Our characterizations with floats can be modified to cover PEs by simply adding proposition symbols to the logic that encode the PE of each vertex, see the technical report (Ahvonen et al. 2025) for more details. Positional encodings are an important future topic that warrants further study.

5 Conclusion

We have given logical characterizations for GPS-networks and graph transformers, based on reals and on floats. As future work, it would be interesting to lift all our characterizations from vertex to graph classification, and to more comprehensively study the expressive power of GPS-networks and GTs enriched with common forms of positional encodings such as graph Laplacians. Our results in the float case in fact already lift to graph classification tasks and also to non-Boolean classification, and they also hold when restricted to word-shaped graphs; this is covered in the technical report (Ahvonen et al. 2025). Another interesting open question is whether, in the case of the reals, every GPS-network can be expressed as a GNN + GC.

6 Acknowledgments

Veeti Ahvonen was supported by the Vilho, Yrjö and Kalle Väisälä Foundation. Damian Heiman was supported by the Magnus Ehrnrooth Foundation. Antti Kuusisto was supported by the project *Perspectives on computational logic*, funded by the Research Council of Finland, project number 369424. Carsten Lutz was supported by DFG project LU 1417/4-1.

References

Ahvonen, V.; Funk, M.; Heiman, D.; Kuusisto, A.; and Lutz, C. 2025. Expressive Power of Graph Transformers via Logic. arXiv:2508.01067.

Ahvonen, V.; Heiman, D.; Kuusisto, A.; and Lutz, C. 2024. Logical characterizations of recurrent graph neural networks with reals and floats. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Alon, U.; and Yahav, E. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J. L.; and Silva, J. P. 2020. 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.

Barceló, P.; Kozachinskiy, A.; Lin, A. W.; and Podolskii, V. V. 2024. Logical Languages Accepted by Transformer Encoders with Hard Attention. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Benedikt, M.; Lu, C.; Motik, B.; and Tan, T. 2024. Decidability of Graph Neural Networks via Logical Characterizations. In Bringmann, K.; Grohe, M.; Puppis, G.; and Svensson, O., eds., *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, 127:1–127:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Blanchard, P.; Higham, D. J.; and Higham, N. J. 2019. Accurate Computation of the Log-Sum-Exp and Softmax Functions. arXiv:1909.03469.

Chiang, D.; Cholak, P.; and Pillay, A. 2023. Tighter Bounds on the Expressivity of Transformer Encoders. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of ICML 2023*, volume 202 of *Proceedings of Machine Learning Research*, 5544–5562. PMLR.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Dwivedi, V. P.; and Bresson, X. 2020. A Generalization of Transformer Networks to Graphs. *CoRR*, abs/2012.09699.

Grohe, M. 2023. The Descriptive Complexity of Graph Neural Networks. In *LICS*, 1–14.

Higham, N. J. 1993. The Accuracy of Floating Point Summation. *SIAM J. Sci. Comput.*, 14(4): 783–799.

IEEE. 2019. IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, 1–84.

Jaakkola, R.; Kuusisto, A.; and Vilander, M. 2025. Relating description complexity to entropy. *Journal of Computer and System Sciences*, 149: 103615.

Jerad, S.; Svete, A.; Li, J.; and Cotterell, R. 2025. Unique Hard Attention: A Tale of Two Sides. arXiv:2503.14615.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kreuzer, D.; Beaini, D.; Hamilton, W. L.; Létourneau, V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 21618–21629.

Li, J.; and Cotterell, R. 2025. Characterizing the Expressivity of Transformer Language Models. arXiv:2505.23623.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, 3538–3545. AAAI Press.

Merrill, W.; and Sabharwal, A. 2023. A Logic for Expressing Log-Precision Transformers. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Otto, M. 2004. Modal and Guarded Characterisation Theorems over Finite Transition Systems. *Annals of Pure and Applied Logic*, 130(1-3): 173–205.

Otto, M. 2019. Graded modal logic and counting bisimulation. *CoRR*, abs/1910.00039.

Pfluger, M.; Tena Cucala, D.; and Kostylev, E. V. 2024. Recurrent Graph Neural Networks and Their Connections to Bisimulation and Logic. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(13): 14608–14616.

Rampásek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Robertazzi, T. G.; and Schwartz, S. C. 1988. Best "ordering" for floating-point addition. *ACM Trans. Math. Softw.*, 14(1): 101–110.

Rosenbluth, E.; Tönshoff, J.; Ritzert, M.; Kisin, B.; and Grohe, M. 2024. Distinguished In Uniform: Self-Attention Vs. Virtual Nodes. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Sälzer, M.; Schwarzentruber, F.; and Troquard, N. 2025. Verifying Quantized Graph Neural Networks is PSPACE-complete. *CoRR*, abs/2502.16244.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Trans. Neural Networks*, 20(1): 61–80.

Schönherr, M.; and Lutz, C. 2026. Logical Characterizations of GNNs with Mean Aggregation. In *Proc. of AAAI*. AAAI Press.

Strobl, L.; Merrill, W.; Weiss, G.; Chiang, D.; and Angluin, D. 2024. What Formal Languages Can Transformers Express? A Survey. *Trans. Assoc. Comput. Linguistics*, 12: 543–561.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Yang, A.; Cadilhac, M.; and Chiang, D. 2025. Knee-Deep in C-RASP: A Transformer Depth Hierarchy. arXiv:2506.16055.

Yang, A.; and Chiang, D. 2024. Counting Like Transformers: Compiling Temporal Counting Logic Into Softmax Transformers. In *First Conference on Language Modeling*.

Yang, A.; Chiang, D.; and Angluin, D. 2024. Masked Hard-Attention Transformers Recognize Exactly the Star-Free Languages. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph Transformer Networks. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.