GRAPH TRANSFORMERS EXPRESS MONADIC SECOND-ORDER LOGIC

Tamara Drucks*
Research Unit Machine Learning
TU Wien
Vienna, Austria
tamara.drucks@tuwien.ac.at

Mahito Sugiyama
National Institute of Informatics
SOKENDAI
Tokyo, Japan
mahito@nii.ac.jp

ABSTRACT

We quantify the expressive power of graph transformers by establishing a formal connection to *monadic second-order logic* (MSO). Expressivity analysis for graph learning algorithms commonly focuses on their ability to produce distinct embeddings for non-isomorphic graphs. This line of research is well established for message-passing graph neural networks and their tight connection to the Weisfeiler-Leman color refinement algorithm. In contrast, graph transformers have mostly been evaluated empirically, with little theoretical investigation thus far. The expressive power of transformers can be quantified by their ability to simulate certain formal languages in the context of natural language processing. Here, we focus on graph learning and MSO and show that transformers can produce distinct embeddings for graphs that differ in MSO-definable properties. MSO is a powerful logic for graph-related tasks, as it allows to decide decision problems for graphs with bounded treewidth in linear time.

1 INTRODUCTION

Recently, there have been several attempts to capture the expressivity of transformers in the context of natural language processing through their ability to express certain formal languages (Chiang et al., 2023) (please see (Strobl et al., 2024) for an overview). We extend this analysis to graph-structured data. The expressivity analysis of graph learning algorithms is commonly conducted through the lens of Weisfeiler-Leman (WL) tests, a hierarchy of increasingly expressive color refinement algorithms which test graph isomorphism. Xu et al. (2019) and Morris et al. (2019) have shown that the expressive power of message-passing graph neural networks (GNNs) can be characterized by the 1-dimensional WL test. Transformers can, very broadly, be seen as GNNs with some attention mechanism as an aggregation function (Groß et al., 2023; Sáez de Ocáriz Borde, 2024). However, since graph transformer architectures often use additional tools such as positional or structural encodings which increase their expressive power, it becomes challenging to directly analyze the expressive power of graph transformers, leading to limited research in this area.

To address this challenge, we establish a formal connection between transformers and *monadic second-order logic* (MSO), following the line of research that investigates the ability of general transformers, including large language models (LLMs), to reason on graph-structured data (de Luca & Fountoulakis, 2024; Sanford et al., 2024; Ye et al., 2024). We show that transformers can accept all sentences definable in MSO. This implies that transformers are able to distinguish non-isomorphic graphs that differ in any graph property which is definable in MSO. Graph properties that are definable in MSO include p-colorability for fixed p, connectivity, and planarity. MSO is particularly interesting as there exists a procedure to transform an MSO formula into a finite automaton which decides many generally NP-hard decision problems in linear time for graphs of bounded treewidth. Bounded treewidth graphs include a wide variety of natural graph classes such as outerplanar graphs, which comprise the majority of molecular benchmark datasets (Bause et al., 2024, Table 1). Thus, our contribution can provide a theoretical basis for the success of transformer-based models for, e.g., drug design (Cofala & Kramer, 2021; Mao et al., 2024; Lu et al., 2023; Pirnay et al., 2024).

*Work done during an internship at NII

2 RELATED WORK

Graph transformers (GT) have been shown to be successful for many graph classification tasks, often surpassing standard message-passing neural networks (MPNNs) (Müller et al., 2024a). In general, graph transformers without additional positional encodings (PEs) have expressive power equivalent to set-based architectures (cf. DeepSets (Zaheer et al., 2017) and are thus less expressive than standard MPNNs. There has been a number of recently proposed architectures with provable expressive power guarantees related to the WL-hierarchy. Ying et al. (2021), Kreuzer et al. (2021) and Rampášek et al. (2022) propose GT with structural encodings and PEs that surpass MPNN expressivity. Black et al. (2024) provide a theoretical investigation into the discriminative power of different PEs. Kim et al. (2022) proposed a hierarchy of GT that aligns with k-IGNs (Maron et al., 2019), thus establishing a connection between GT and the k-WL hierarchy. Müller et al. (2024b) recently showed that GTs (more specifically the EdgeTransformer (Bergen et al., 2021)) are at least as expressive as 3-WL, creating another direct link to classical GNN expressivity analysis. Cai et al. (2023) show that MPNNs with virtual nodes can simulate certain GT. With sufficiently expressive structural encodings GT can become universally expressive (Müller et al., 2024a). Rosenbluth et al. (2024) showed that MPNNs endowed with the same structural information have equivalent expressive power. All of the results above hold in the non-uniform setting, i.e., for fixed-size graphs. In the uniform setting, i.e., one network for graphs of all sizes, Rosenbluth et al. (2024) provide impossibility results for MPNNs and GT. They further show that in the uniform setting, MPNNs and GT are incomparable in expressive power.

An equivalent notion to 1-WL is the two-variable fragment of first order logic with counting, denoted by $\mathcal{C}^2$ (Cai et al., 1992): Two graphs are 1-WL-equivalent iff they satisfy the same set of sentences in $\mathcal{C}^2$ (Barceló et al., 2020)[1]. Benedikt et al. (2024) build upon these results and show a link between GNNs and Presburger logics, a decidable extension of first-order logic. Grohe (2023) analyze GNNs through the lens of Boolean circuits and show that there is a TC^0 upper bound for GNNs. Barlag et al. (2024) showed that the expressive power of GNNs (not limited to message-passing) is equivalent to that of arithmetic circuits over the reals (cf. AC^0). Müller et al. (2024b) utilize the connection of the EdgeTransformer and 3-WL to formalize systematic generalization through first-order logic statements. Inspired by the zero-one convergence law for formal languages, a new line of research studies the expressive power of probabilistic graph classifiers (Adam-Day & Ceylan, 2023; Adam-Day et al., 2024). For their expressivity analysis, (Adam-Day et al., 2024) develop an aggregate term language, which captures several GT architectures and for which they prove asymptotically almost sure convergence (i.e., the output of the probabilistic graph classifiers which can be expressed in the term language converges to a constant function as the size of the graphs increases).

Sanford et al. (2024); de Luca & Fountoulakis (2024); Ouyang et al. (2024) investigate the abilities of transformers, not limited to graph transformers, on graph reasoning tasks, which is aligned with our approach. In particular, Sanford et al. (2024) is the work closest to our analysis. Whereas Sanford et al. (2024) focus on global graph properties relevant for graph reasoning tasks, we focus on properties definable in MSO useful to distinguish between non-isomorphic graphs. Furthermore, Sanford et al. (2024) provide bounds dependent on the arboricity of a graph while our theoretical results are for bounded treewidth graphs. Thanks to Courcelle's theorem, this enables us to efficiently compute these properties for bounded treewidth graphs.

3 PRELIMINARIES

Graph transformers. A transformer is a type of neural network that consists of alternating blocks of *attention heads* and multi-layer perceptrons (MLPs) and is often used for sequence-to-sequence tasks. *Attention* usually refers to the softmax function applied to a linear projection of *queries*, *keys*, and *values*. More formally, a *single attention head* for a feature matrix $\mathbf{X}^{(t)} \in \mathbb{R}^{n \times d}$ is defined as follows

$$f(\mathbf{X}^{(t)}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d^k}}\right)\mathbf{V}, \tag{1}$$

[1]This is a bit of a simplification, please refer to Barceló et al. (2020) for more details on the relationship between MPNNs and $\mathcal{C}^2$

where $\mathbf{Q} = \mathbf{X}^{(t)}\mathbf{W}_\mathbf{Q}$, $\mathbf{K} = \mathbf{X}^{(t)}\mathbf{W}_\mathbf{K}$, and $\mathbf{V} = \mathbf{X}^{(t)}\mathbf{W}_\mathbf{V}$ with learnable weight matrices $\mathbf{W}_\mathbf{Q}, \mathbf{W}_\mathbf{K} \in \mathbb{R}^{d \times d^K}$ and $\mathbf{W}_\mathbf{V} \in \mathbb{R}^{d \times d}$.

Multi-head attention is defined as the concatenation of multiple independent copies of a single attention head followed by a linear layer, which we will denote as f_{attn}. An L-layer transformer f_{tf} is defined as the composition of attention heads and MLPs, i.e.,

$$f_{\text{tf}} = \left(\mathrm{id} + f_{\text{mlp}}^{(L)}\right) \circ \left(\mathrm{id} + f_{attn}^{(L)}\right) \circ \left(\mathrm{id} + f_{\text{mlp}}^{(L-1)}\right) \circ \ldots \circ \left(\mathrm{id} + f_{attn}^{(1)}\right) \circ (\mathrm{id} + P)\,, \tag{2}$$

where id is the identity function representing residual connections and P is a *positional encoding* of the sequence elements. In the context of graph learning, graph transformers are transformers that are adapted to operate on graph-structured input data. This usually entails encoding the structure of the graph as positional encodings. Please refer to Müller et al. (2024a); Black et al. (2024) for an overview of commonly used positional encodings.

Monadic second-order logic. Monadic second-order logic (MSO) is a fragment of second-order logic which allows quantification over sets. We assume familiarity with first-order logic (FO).

MSO extends FO with second-order variables of arity 1. An MSO formula can contain both first-order and second-order free variables. We write $\varphi(x_1, \ldots, x_m, X_1, \ldots, X_\ell)$ to denote that x_i for $i = [m] = \{1, \ldots, m\}$ are free first-order variables and X_j for $j = [\ell]$ are free second-order variables. Please refer to Section A.1.1 for the full definition. In the context of graphs, we often distinguish between MSO_1, which only allows quantification over vertices, and MSO_2, which additionally allows quantification over edges. Note that MSO_2 is more powerful than MSO_1, e.g., we can express the existence of a Hamiltonian cycle in MSO_2, but not in MSO_1.

For instance, we can express 3-colorability in MSO_1 as follows:

$$\varphi = \exists X_1 \exists X_2 \exists X_3 \; part(X_1, X_2, X_3) \wedge \forall v \forall w Evw \implies \bigwedge_{i=1}^{3} (X_i v \wedge X_i w),$$

where Exy means that there is an edge between vertices x and y and

$$part(X_1, X_2, X_3) = \forall v X_1 v \vee X_2 v \vee X_3 v \wedge \forall v \neg(X_1 v \wedge X_2 v) \wedge \neg(X_1 v \wedge X_3 v) \wedge \neg(X_2 v \wedge X_3 v)$$

denotes that X_1, X_2, X_3 are disjoint partitions of the vertex set.

Fixed-parameter tractability and treewidth. We briefly introduce the concepts of fixed-parameter tractability and treewidth. We assume some familiarity with graph theory and complexity theory.

Definition 3.1 (FPT algorithm)**.** An algorithm is *fixed-parameter tractable* (FPT) with respect to some parameter k, if its runtime is bounded by $f(k)\,\mathrm{poly}(n)$ for some computable function f.

The *treewidth* of a graph G, denoted by $\mathrm{tw}(G)$, can be defined in terms of a tree decomposition.

Definition 3.2 (Robertson & Seymour (1986))**.** A *tree decomposition* of a graph $G = (\mathcal{V}, \mathcal{E})$ is a pair $(\mathcal{B}, \mathcal{D})$, where $\mathcal{D} = (B, A)$ is a tree, and $\mathcal{B} = \{\mathcal{B}_i \mid i \in B\}$ is a family of subsets of $\mathcal{V}$, such that (i) $\bigcup_{i \in B} \mathcal{B}_i = \mathcal{V}$, (ii) every edge of G has both of its endpoints in some $\mathcal{B}_i$ with $i \in B$, and (iii) for all $i, j, k \in I$, if j lies on the path from i to k in $\mathcal{D}$, then $\mathcal{B}_i \cap \mathcal{B}_k \subseteq \mathcal{B}_j$.

Definition 3.3. The *treewidth* of a tree decomposition is $\max_{i \in B} |\mathcal{B}_i| - 1$. The treewidth of G is the minimum treewidth taken over all possible tree decompositions of G.

Intuitively, the treewidth of a graph specifies how tree-like it is; e.g., trees have treewidth 1, cycles have treewidth 2. Many NP-complete decision problems can be decided in linear time for graphs with bounded treewidth, as shown in the famous meta-theorem by Courcelle (1990).

Theorem 3.4 (Courcelle 1990)**.** *For an MSO_1 sentence φ and a graph G, one can decide whether $G \models \varphi$ in time $f(\mathrm{tw}(G), |\varphi|)n$ for some function f.*

Note that Theorem 3.4 can also be extended to MSO_2. In fact, there are multiple variations of Courcelle's theorem, including optimization problems on graphs (Arnborg et al., 1991; Courcelle et al., 2000) and modulo counting (Courcelle, 1990).

4 GRAPH TRANSFORMERS EXPRESS MSO

In this section, we present our main result: Graph transformers can express any graph property that is definable in MSO and are thus able to solve, e.g., p-colorability for fixed p and graph connectivity. To show this, we exploit the correspondence between MSO definability and tree automata. More specifically, we show that, given a graph G and an MSO formula φ, a graph transformer can decide whether $G \models \varphi$. We first define the concept of *expressibility*.

Definition 4.1. Let $\mathcal{G}^n$ be all graphs up to order n. We say that a function $f : \mathcal{G}^n \to \mathbb{R}^d$ *expresses* a graph property P if for each $G, G' \in \mathcal{G}^n$, $f(G) = f(G')$ iff $P(G) = P(G')$.

For instance, the function $f(G) = \mathbb{1}[|V| \equiv 1 \bmod 2]$ expresses the graph property *odd number of vertices*. Next, we show that transformers are able to produce distinct embeddings for graphs with different MSO-definable properties.

Theorem 4.2. *For every $n \in \mathbb{N}$ and every MSO-definable property P, there exists a graph transformer $GT : \mathcal{G}^n \to \mathbb{R}^d$ that can express P for each $G \in \mathcal{G}^n$.*

Proof sketch. The underlying idea of the proof for Theorem 4.2 is the following: Given a bounded treewidth graph G and an MSO formula φ, we can build a tree automaton $\mathcal{A}$ that accepts G if $G \models \varphi$ and rejects G if $G \not\models \varphi$. As an intermediate step, we construct a tree decomposition $(\mathcal{B}, \mathcal{D})$ of G and map G and $(\mathcal{B}, \mathcal{D})$ to a labeled ordered binary tree t, which we can input into $\mathcal{A}$. The proof itself then consists of showing how to simulate this construction with a transformer. We consider the computation of the tree decomposition and the labeled ordered binary tree as linear-time preprocessing steps. It remains to show that a graph transformer can simulate a finite tree automaton $\mathcal{A}$ that accepts or rejects t. This largely follows from Rizvi-Martel et al. (2024, Corollary 1), who show that transformers can simulate weighted tree automata. See Theorem A.3 in the appendix for the full proof. □

Note that our proof of Theorem 4.2 depends on the computation of a tree decomposition of our input graph. While we can compute a trivial tree decomposition of width $\mathcal{O}(n)$ for any graph, in practice, it makes sense to focus on bounded treewidth graphs as model checking, i.e., verifying whether a graph satisfies a formula, is FPT with respect to the treewidth of the graph and the length of the formula.

In Corollary 4.1, we summarize some MSO-definable properties useful for graph classification tasks.

Corollary 4.1. *Graph transformers can express each of the following properties: (i) connectivity, (ii) p-colorability for fixed p, (iii) minor inclusion, (iv) planarity, (v) Hamiltonicity, and (vi) perfect matching.*

Corollary 4.1 provides a theoretical explanation for the experimental results conducted on connectivity conducted in Sanford et al. (2024) and on connectivity, Hamiltonian paths and perfect matchings in Ouyang et al. (2024).

5 CONCLUSION

We have characterized the expressive power of graph transformers by establishing a formal connection to MSO. MSO is a powerful fragment of second-order logic and can be used to define many interesting graph properties, such as connectivity and 3-colorability. Our results imply that transformers can produce distinct embeddings for graphs with different MSO-definable properties. This might provide a theoretical explanation for the empirical success of many generative graph transformers. For future work, we plan to show that constructing tree decompositions and labeled binary trees can be simulated by transformers as well. Additionally, we would like to investigate whether there exist shortcut solutions similar to (Liu et al., 2022) and use our theoretical insights to design novel GT architectures.

ACKNOWLEDGMENTS

We thank Maximilian Thiessen and Sagar Malhotra for valuable discussions and feedback. This work was supported by JST, CREST Grant Number JPMJCR22D3, Japan (MS).

REFERENCES

Sam Adam-Day and Ismail Ceylan. Zero-one laws of graph neural networks. *Advances in Neural Information Processing Systems*, 36:70733–70756, 2023.

Sam Adam-Day, Michael Benedikt, Ismail Ilkan Ceylan, and Ben Finkelshtein. Almost surely asymptotically constant graph neural networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991. ISSN 0196-6774. doi: https://doi.org/10.1016/0196-6774(91)90006-K.

Pablo Barceló, Egor V. Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations*, 2020.

Timon Barlag, Vivian Holzapfel, Laura Strieker, Jonni Virtema, and Heribert Vollmer. Graph neural networks and arithmetic circuits. *arXiv preprint arXiv:2402.17805*, 2024.

Franka Bause, Fabian Jogl, Patrick Indri, Tamara Drucks, Nils Morten Kriege, Thomas Gärtner, Pascal Welke, and Maximilian Thiessen. Maximally expressive GNNs for outerplanar graphs. In *TMLR*, 2024.

Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan. Decidability of graph neural networks via logical characterizations, 2024.

Leon Bergen, Timothy O'Donnell, and Dzmitry Bahdanau. Systematic generalization with edge transformers. *Advances in Neural Information Processing Systems*, 34:1390–1402, 2021.

Mitchell Black, Zhengchao Wan, Gal Mishne, Amir Nayyeri, and Yusu Wang. Comparing graph transformers via positional encodings, 2024.

Hans L Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pp. 226–234, 1993.

Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between mpnn and graph transformer, 2023.

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

David Chiang, Peter Cholak, and Anand Pillay. Tighter bounds on the expressivity of transformer encoders. In *International Conference on Machine Learning*, pp. 5544–5562. PMLR, 2023.

Tim Cofala and Oliver Kramer. Transformers for molecular graph generation. In *ESANN*, 2021.

Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. ISSN 0890-5401. doi: https://doi.org/10.1016/0890-5401(90)90043-H.

Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

Bruno Courcelle, Johann A Makowsky, and Udi Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

Artur Back de Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers, 2024.

Rodney G Downey, Michael R Fellows, et al. *Fundamentals of parameterized complexity*, volume 4. Springer, 2013.

Manfred Droste and Heiko Vogler. Weighted tree automata and weighted logics. *Theoretical Computer Science*, 366(3):228–247, 2006.

Martin Grohe. The descriptive complexity of graph neural networks. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pp. 1–14, 2023. doi: 10.1109/LICS56636.2023.10175735.

Joschka Groß, Gerrit Großmann, and Verena Wolf. Elucidating the relationship between transformers and gnns, 2023.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners, 2022.

Stephan Kreutzer. Algorithmic meta-theorems. *Finite and algorithmic model theory*, 379:177–270, 2011.

Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 21618–21629. Curran Associates, Inc., 2021.

Leonid Libkin. *Elements of finite model theory*, volume 41. Springer, 2004.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Hao Lu, Zhiqiang Wei, Xuze Wang, Kun Zhang, and Hao Liu. Graphgpt: A graph enhanced generative pretrained transformer for conditioned molecular generation. *International Journal of Molecular Sciences*, 24(23), 2023. ISSN 1422-0067.

Jiashun Mao, Jianmin Wang, Amir Zeb, Kwang-Hwi Cho, Haiyan Jin, Jongwan Kim, Onju Lee, Yunyun Wang, and Kyoung Tai No. Transformer-based molecular generative model for antiviral drug design. *Journal of Chemical Information and Modeling*, 64(7):2733–2745, 2024. doi: 10.1021/acs.jcim.3c00536. PMID: 37366644.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence*, pp. 4602–4609. AAAI Press, 2019.

Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers, 2024a.

Luis Müller, Daniel Kusuma, Blai Bonet, and Christopher Morris. Towards principled graph transformers, 2024b.

Sheng Ouyang, Yulan Hu, Ge Chen, and Yong Liu. Gundam: Aligning large language models with graph understanding, 2024.

Jonathan Pirnay, Jan G. Rittig, Alexander B. Wolf, Martin Grohe, Jakob Burger, Alexander Mitsos, and Dominik G. Grimm. Graphxform: Graph transformer for computer-aided molecular design with application to extraction, 2024.

Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Michael Rizvi-Martel, Maude Lizaire, Clara Lacroce, and Guillaume Rabusseau. Simulating weighted automata over sequences and trees with transformers. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2368–2376. PMLR, 02–04 May 2024.

Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of algorithms*, 7(3):309–322, 1986.

Eran Rosenbluth, Jan Tönshoff, Martin Ritzert, Berke Kisin, and Martin Grohe. Distinguished in uniform: Self attention vs. virtual nodes, 2024.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms, 2024.

Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Trans. Assoc. Comput. Linguistics*, 12:543–561, 2024. doi: 10.1162/TACL_A_00663.

Haitz Sáez de Ocáriz Borde. Elucidating graph neural networks, transformers, and graph transformers, 2024.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations*, 2019.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. Language is all a graph needs, 2024.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation?, 2021.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A Appendix

A.1 Preliminaries

We briefly introduce monadic-second order logic (MSO) and tree automata. Please refer to (Libkin, 2004) and (Droste & Vogler, 2006) for more details.

A.1.1 Monadic second-order logic

Definition A.1 (MSO)**.** MSO extends FO with second-order variables of arity 1. An MSO formula can contain both first-order and second-order free variables. We write $\varphi(x_1, \ldots, x_m, X_1, \ldots, X_\ell)$ to denote that x_i for $i = [m] = \{1, \ldots, m\}$ are free first-order variables and X_j for $j = [\ell]$ are free second-order variables. Given a vocabulary σ consisting of relation and constant symbols, we define MSO terms and formulae as follows:

- Every first-order variable x and every constant symbol c are first-order terms.
- There are three kinds of atomic formulae:
  - FO atomic formulae:
    - $t = t'$, where t, t' are terms,
    - $R(t_1, \ldots, t_k)$ where $R \in \sigma$, and
  - $X(t)$, where X is a second-order variable of arity 1. The free first-order variables of this formula are free first-order variables of t, and the free second order variable is X.
- The formulae of MSO are closed under the Boolean connectives $\wedge, \vee, \neg$ and first-order quantification, with the usual rules for free variables.

- If $\varphi(x_1, \ldots, x_m, Y, X_1, \ldots, X_\ell)$ is a formula, then $\exists Y \varphi(x_1, \ldots, x_m, Y, X_1, \ldots, X_\ell)$ and $\forall Y \varphi(x_1, \ldots, x_m, Y, X_1, \ldots, X_\ell)$ are formulae with free variables $x_1, \ldots, x_m$ and $X_1, \ldots, X_\ell$.

A.1.2 TREE AUTOMATA

In the following, we define *tree automata*, which accept or reject *labeled ordered binary trees* and operate in a bottom-up manner.

Let Σ be a finite nonempty alphabet, i.e., a finite set of symbols. A binary tree domain dom is a prefix closed subset of $\{0,1\}^*$ such that if $si \in \mathrm{dom}$ for $s \in \{0,1\}^*$ and $i \in \{0,1\}$ then $sj \in \mathrm{dom}$ for all $j < i$. Every nonempty tree domain has node ϵ, which is the root node. A tree t over Σ is a mapping $t : \mathrm{dom} \to \Sigma$. We denote by $\mathcal{T}_\Sigma$ the set of all Σ-trees. $\mathcal{T}_\Sigma$ is the smallest set such that $\Sigma \subset \mathcal{T}_\Sigma$ and $(t_0, t_1) \in \mathcal{T}_\Sigma$ for all $t_0, t_1 \in \mathcal{T}_\Sigma$.

Definition A.2 (Deterministic finite tree automaton (DTA))**.** A DTA $\mathcal{A}$ is a tuple (Q, Σ, F, δ), where Q is a finite set of states, Σ is a finite alphabet, $F \subseteq Q$ is the set of final states and $\delta : Q^2 \times \Sigma \to Q$ is a transition function. A DTA computes a function $f_\mathcal{A} : \mathcal{T}_\Sigma \to Q$ defined by $f_\mathcal{A}(t) = \mu(\epsilon)$, where the mapping $\mu : \mathcal{T}_\Sigma \to Q$ is recursively defined as $\mu(\sigma) = q_\sigma \in Q$ for all $\sigma \in \Sigma$ and $\mu(t) = \delta(\mu(t_0), \mu(t_1), \sigma)$ for all $t \in \mathcal{T}_\Sigma$ with children $t_0, t_1 \in \mathcal{T}_\Sigma$.

We define a *run* of $\mathcal{A}$ on a tree $t \in \mathcal{T}_\Sigma$ as mapping $\rho : \mathrm{dom} \to Q$. A run ρ is successful whenever $\rho(\epsilon) \in F$ and a tree is accepted if a successful run exists on it. The tree language recognized by $\mathcal{A}$ consists of the trees accepted by $\mathcal{A}$.

A.2 PROOFS

Theorem A.3 (Theorem 4.2)**.** *For every $n \in \mathbb{N}$ and every MSO-definable property P, there exists a graph transformer $GT : \mathcal{G}^n \to \mathbb{R}^d$ that can express P for each $G \in \mathcal{G}^n$.*

Proof. Given a graph $G = (V, E)$ and an arbitrary MSO formula φ, we perform the following linear-time preprocessing steps: First, we compute the tree decomposition $(\mathcal{B}, \mathcal{D})$ of G (Bodlaender, 1993). Next, given G and $(\mathcal{B}, \mathcal{D})$, we compute the labeled ordered binary (LOB) tree t (Downey et al., 2013, Thm. 12.7.1). Finally, we translate φ into a new formula φ^* such that $G \models \varphi$ iff $t \models \varphi^*$ (Kreutzer, 2011). Given LOB t and φ^*, we can construct a tree automaton $\mathcal{A} = (\Sigma, Q, F, \delta)$, which accepts t iff $G \models \varphi$ (Libkin, 2004, Thm. 7.30). Recall that $\mathcal{A}$ accepts t if there exists a successful run ρ of $\mathcal{A}$ on t. A run ρ is successful if $\rho(\epsilon) \in F$, i.e., if the label of the root node ϵ is one of the accepting states. We show in Theorem A.5 that a transformer can simulate $\mathcal{A}$. It remains to be shown is that we can map all accepting states $q_f \in F$ to the same output vector, and all other states $q \in Q \setminus F$ to different output vectors. This can be done by adding an additional MLP after the last layer of the transformer which can approximate arbitrary functions (Hornik et al., 1989). □

Definition A.4 (Simulation)**.** Given a DTA $\mathcal{A} = (Q, \Sigma, F, \delta)$, we say that a function $f : \mathcal{T}_\Sigma^T \to Q^T$ simulates $\mathcal{A}$ at length T if for all trees $t \in \mathcal{T}_\Sigma$ such that $\mathrm{depth}(t) \leq T$, $f(t)_i = \mu(\tau_i)$ for all subtrees τ_i. Furthermore, we say that a family of functions $\mathcal{F}$ simulates DTAs with n states at length T if for any DTA $\mathcal{A}$ with n states there exists a function $f \in \mathcal{F}$ that simulates $\mathcal{A}$ at length T.

Rizvi-Martel et al. (2024) define tree automata and simulation slightly differently; however, this can be adapted to our setting. First of all, our tree automata are non-weighted and we thus omitted the initial weight vectors $\alpha \in \mathbb{R}^n$ in our definition. A classical tree automaton is equivalent to a weighted tree automaton with weights in the Boolean semi-ring, which we can simulate with real weights by interpreting non-zero weights as true and zero weights as false. Secondly, states are represented as vectors $\mathbf{v}_\sigma \in \mathbb{R}^n$; we can encode each $q \in Q$ as, e.g., one-hot encoding.

Theorem A.5 (Rizvi-Martel et al. 2024, Corollary 1, slightly re-written)**.** *Given a tree automaton $\mathcal{A}$ with n states and depth T, there exists a transformer with $\mathcal{O}(n)$ embedding dimension, $\mathcal{O}(n)$ attention width, $\mathcal{O}(n^3)$ MLP width, $\mathcal{O}(1)$ attention heads and $\mathcal{O}(T)$ depth that simulates $\mathcal{A}$ up to arbitrary precision $\varepsilon > 0$.*

Proof. See (Rizvi-Martel et al., 2024, Appendix C) for the proof of Theorem A.5. □

Corollary A.1 (Corollary 4.1)**.** *Graph transformers can express each of the following properties:* (i) *connectivity,* (ii) p*-colorability for fixed* p*,* (iii) *minor inclusion,* (iv) *planarity,* (v) *Hamiltonicity, and* (vi) *perfect matching.*

Proof. This follows from the fact that connectivity, p-colorability for fixed p, minor inclusion (and thus planarity), Hamiltonicity, and perfect matching are MSO-definable properties (Courcelle & Engelfriet, 2012). □