A Graph Neural Network Reasoner for Game Description Language

Extended Abstract

Alvaro Gunawan
Auckland University of Technology
Auckland, New Zealand
alvaro.gunawan@aut.ac.nz

Ji Ruan
Auckland University of Technology
Auckland, New Zealand
ji.ruan@aut.ac.nz

Xiaowei Huang
University of Liverpool
Liverpool, United Kingdom
xiaowei.huang@liverpool.ac.uk

ABSTRACT

General Game Playing (GGP) aims to develop agents that are able to play any game with only rules given. The game rules are encoded in the Game Description Language (GDL). A GGP player processes the game rules to obtain game states and expand the game tree search for an optimal move. The recent accomplishments of AlphaGo and AlphaZero have triggered new works in extending neural network approaches to GGP. In these works, the neural networks are used only for optimal move selection, while the components dealing with GDL still use logic-based methods. This motivates us to explore if a neural network based method would be able to approximate the logical inference in GDL with a high accuracy. The structured nature of logic tends to be a difficulty for neural networks, which rely heavily on statistical features. Inspired by the recent works on neural network learning for logical entailments, we propose a neural network based reasoner that is able to learn logical inferences for GDL. We present three key contributions: (i) a general, game-agnostic graph-based representation for game states described in GDL, (ii) methods for generating samples and datasets to frame the GDL inference task as a neural network based machine learning problem and (iii) a GNN based *neural reasoner* that is able to learn and infer various game states with a high accuracy and has some capability of transfer learning across games.

KEYWORDS

Game Description Language; Inference; Graph Neural Network

ACM Reference Format:
Alvaro Gunawan, Ji Ruan, and Xiaowei Huang. 2022. A Graph Neural Network Reasoner for Game Description Language: Extended Abstract. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 3 pages.

1 INTRODUCTION

General Game Playing (GGP) [1] aims to develop agents that are able to play any game with only rules given. The game rules are encoded in Game Description Language (GDL). In order to play games, the players in GGP parse the game rules using logic-based inferences to obtain game states and expand the game tree search for an optimal move. These inferences are typically done through Prolog or PropNet (propositional networks) [2] implementations.

The recent accomplishments of AlphaGo [3, 4] and AlphaZero [5] use Monte-Carlo Tree Search, deep neural networks and learning through self-play. These methods are limited in that they do not automatically extend to play other games and the game types are assumed to be two-player, turn-taking and zero-sum. In the domain of GGP, an agent must be able to account for games with any number of players, simultaneous action and non-zero-sum. A few recent works [6, 7] have shown the effectiveness of extending the methods of AlphaZero to the domain of GGP with some limitations.

Evans et al. [8] introduces a dataset of logical entailment examples and a newly proposed PossibleWorldNet model. Rawson and Reger [9] extends this by applying a Graph Neural Network-based architecture to a graph based encoding for the logical statements. Beyond these two works, some other researchers have applied graph neural networks to the logical domain [10–15]. This motivates us to explore if a neural network based method would be able to approximate the logical inference in GDL with a high accuracy.

We present three key contributions: (i) a general, game-agnostic graph-based representation for game states described in GDL, (ii) methods for generating samples and datasets to frame the GDL inference task as a neural network based machine learning problem and (iii) a GNN based *neural reasoner* that is able to learn and infer various game states with a high accuracy and has some capability of transfer learning across games.

2 OVERVIEW OF THE NEURAL REASONER

Two key inference tasks in GDL are to find out *(1) what are the legal actions for the players at the current state and (2) what is the next state if the agents make a joint move.* Task (1) can be formalized as the following: to find out in game G for each player i, its legal moves m at the current state s, as in a logical representation $G \wedge s \models legal(i, m)$. Task (2) can be formalized as the following: to find out all the fluents f that holds in the next state, given each player $i \in [1, k]$ doing a move m_i at the current state s, as in a logical representation $G \wedge s \wedge does(1, m_1) \wedge ... \wedge does(k, m_k) \models next(f)$.

Our neural reasoner is designed to approximate the $\models$ functions in the above tasks. Figure 1 gives an overview of the different components in our neural reasoner for GGP. We start with a game described as a set of rules in GDL. It is converted into a rule graph, which is built upon the dependency of the fluents or other predicates in such rules. The rule graph is then combined with a game state into an instantiated rule graph (IRG). We apply a node-vector embedding to the nodes of the IRG to prepare it as input of the Graph Neural Network component. The output of the GNN component is a rule graph with node probabilities. Node masking is then applied based on node type to extract the probabilities of the `legal` and `next` nodes, using information from the input rule graph. During training, the output probabilities are compared to the ground truth training

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

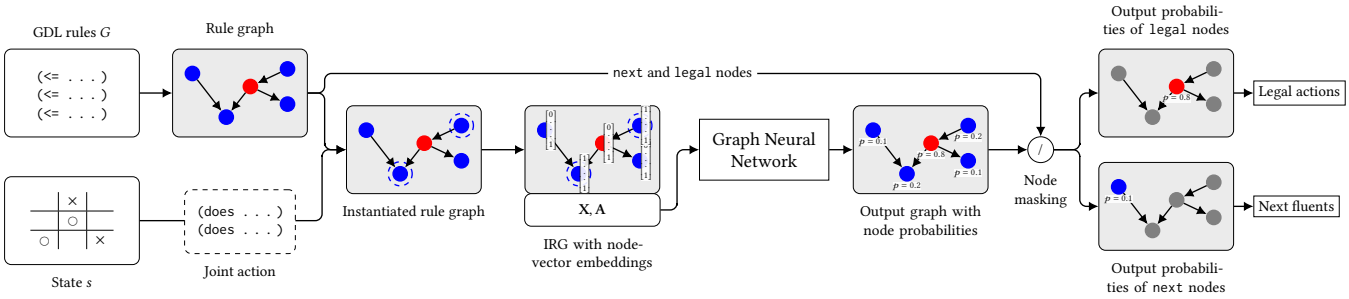

Figure 1: Overview of the Neural Reasoner

targets to calculate the error for backpropagation. Finally, the nodes with output probabilities greater than a threshold value are selected as the legal actions and next fluents.

Game State Representations

Rule graphs [16, 17] are a graph-based representation of game rules written in GDL. They consist of four types of nodes: keyword nodes, predicate nodes, label nodes and label argument nodes. Using rule graphs as a base, we present *instantiated rule graphs*, a general state representation generated by instantiating the rule graph of a game, localising it to a specific state by providing a unique node labelling for each state. Figure 2 shows an example.

Definition 2.1. Instantiated Rule Graph. Given a rule graph $R = (V, E)$ and a state $S = \{f_1, f_2, \ldots, f_n\}$ consisting of fluents f_i, an instantiated rule graph is a graph $I = (V, E, L)$ where V, E are the same as in rule graph R and $L_S : V \rightarrow \{false, true\}$ is a labelling function based on state S and the following requirements:

For node n in rule graph R and current state S:

- If node n is a predicate node that corresponds to a fluent in state S and parent node n_p is a `true` node with edge $(n_p, n) \in E$, then nodes $L_S(n) = true$, $L_S(n_p) = true$ and $L_S(n_c) = true$ for all children nodes given $(n, n_c) \in E$.
- For all other nodes, $L_S(n_o) = false$.

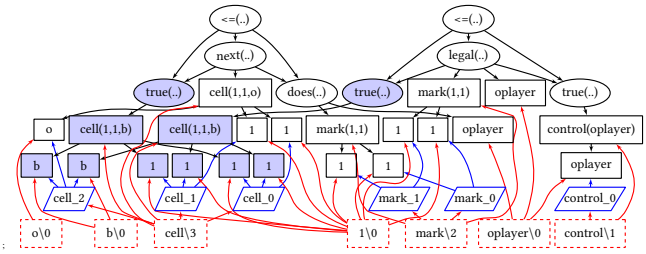

Figure 2: An example of instantiated rule graph

GNN Architecture and Training

As the game state representation we have discussed in the previous section is defined as a graph, we use a graph neural network. The overall architecture of our GNN consists of three distinct layers: an initial set of fully connected input embedding layers, a set of 3 GAT Block layers and finally a set of fully connected output layers. The GAT Blocks consist of two pairs of Graph Attention Networks (GAT) [18] acting as bi-directional edge layers and a skip connection. The Adam optimiser [19] is used with a base learning rate of 0.0001. PairNorm [20] is used between the graph neural network layers. The graph neural network components are implemented using the PyTorch Geometric library [21].

To generate a training dataset, the GDL description of the selected game G is used to play out random games. As new states are visited throughout the game plays, we store the states as instantiated rule graphs and use a Prolog based reasoner (implemented with the PySwip library [22]) to generate ground truth logical inferences for the states as the training target.

3 RESULTS

As the networks are trained to predict `legal` actions and `next` fluents, we measure the accuracy of the neural reasoner over 100 randomly played games. Table 1 shows the accuracy of the neural reasoner on 10 different games.

Game	IRG size Nodes	Edges	Next fluents (%)	Legal actions (%)
parallelbuttonsandlights	1602	3245	84.35%	100.00%
tictactoe (TTT)	4810	11031	100.00%	100.00%
tictactoelarge (TTT_L)	3553	8185	100.00%	63.69%
doubletictactoe (TTT_D)	7379	16338	100.00%	91.96%
connectfour (C4)	10233	25791	93.58%	100.00%
connectfour3p ($C4_{3p}$)	17076	42623	95.50%	91.75%
blocker	5087	12201	88.94%	100.00%
knightstour	12291	30154	100.00%	100.00%
hamilton	14963	31516	94.94%	100.00%
hanoi6disks	28055	70237	87.55%	72.66%

Table 1: Neural reasoner accuracy over 100 games.

Additional experiments show the neural reasoner is capable of learning to reason multiple games simultaneously with high accuracy, as well as some zero-shot transfer capability across similar games. With multiple games, we have found that mixed training and sequential training produce neural reasoners with similar accuracy, with mixed training being more stable while sequential training providing a more practical alternative for training. We have also found that flattening certain rules in some game descriptions provide an increase in reasoning accuracy, e.g., achieving 100% accuracy for both *Next* and *Legal* in TTT_L, TTT_D and C4.

4 CONCLUSION

In this paper we have presented a method to approach GDL reasoning with neural networks in a general manner. The translation of the GDL rules and game states to a graph-based representation allows for the application of graph neural networks that are able take as input various games without resorting to game-specific networks. We have implemented a neural reasoner that is able to learn to infer the legal actions and next fluents in various games, with a high accuracy in most of them.

REFERENCES

[1] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth, "General game playing: Game description language specification," 2008.

[2] E. Schkufza, N. Love, and M. Genesereth, "Propositional automata and cell automata: Representational frameworks for discrete dynamic systems," in *Australasian Joint Conference on Artificial Intelligence*, pp. 56–66, Springer, 2008.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[6] A. Goldwaser and M. Thielscher, "Deep reinforcement learning for general game playing.," in *AAAI*, pp. 1701–1708, 2020.

[7] A. Gunawan, J. Ruan, M. Thielscher, and A. Narayanan, "Exploring a learning architecture for general game playing," in *Australasian Joint Conference on Artificial Intelligence*, pp. 294–306, Springer, 2020.

[8] R. Evans, D. Saxton, D. Amos, P. Kohli, and E. Grefenstette, "Can neural networks understand logical entailment?," *arXiv preprint arXiv:1802.08535*, 2018.

[9] M. Rawson and G. Reger, "Directed graph networks for logical entailment," tech. rep., EasyChair, 2020.

[10] V. Thost and J. Chen, "Directed acyclic graph neural networks," in *International Conference on Learning Representations*, 2021.

[11] I. Abdelaziz, M. Crouse, B. Makni, V. Austil, C. Cornelio, S. Ikbal, P. Kapanipathi, N. Makondo, K. Srinivas, M. Witbrock, *et al.*, "Learning to guide a saturation-based theorem prover," *arXiv preprint arXiv:2106.03906*, 2021.

[12] X. Glorot, A. Anand, E. Aygun, S. Mourad, P. Kohli, and D. Precup, "Learning representations of logical formulae using graph neural networks," in *Neural Information Processing Systems, Workshop on Graph Representation Learning*, 2019.

[13] A. Paliwal, S. Loos, M. Rabe, K. Bansal, and C. Szegedy, "Graph representations for higher-order logic and theorem proving," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 2967–2974, 2020.

[14] M. Crouse, I. Abdelaziz, C. Cornelio, V. Thost, L. Wu, K. Forbus, and A. Fokoue, "Improving graph neural network representations of logical formulae with subgraph pooling," *arXiv preprint arXiv:1911.06904*, 2019.

[15] M. Olšák, C. Kaliszyk, and J. Urban, "Property invariant embedding for automated reasoning," *arXiv preprint arXiv:1911.12073*, 2019.

[16] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," in *European Conference on Machine Learning*, pp. 188–200, Springer, 2007.

[17] S. Schiffel, "Symmetry detection in general game playing," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[20] L. Zhao and L. Akoglu, "Pairnorm: Tackling oversmoothing in gnns," in *International Conference on Learning Representations*, 2020.

[21] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *CoRR*, vol. abs/1903.02428, 2019.

[22] Y. Tekol and contributors, "PySwip v0.2.10," 2020.