Symbol Grounding for Discrete Graphical Models through Data Imputation

Marianne Defresne [1] , **Romain Gambardella** [2] , **Sophie Barbe** [3] and **Thomas Schiex** [4]
[1] Department of Computer Science, KU Leuven, Leuven, Belgium
[2] Télécom-Paris 91120 Palaiseau, France
[3] TBI, Université de Toulouse, CNRS, INRAE, INSA, ANITI, 31077 Toulouse, France
[4] Université Fédérale de Toulouse, ANITI, INRAE, UR 875, 31326 Toulouse, France
marianne.defresne@kuleuven.be, thomas.schiex@inrae.fr

Abstract

Neurosymbolic architectures hybridize discrete reasoning with neural networks to solve discrete optimization problems from natural inputs. One key challenge for such architectures is the symbol grounding problem, *i.e.,* learning how to map visual inputs to discrete variables without explicit supervision. We extend a previously-proposed neurosymbolic architecture, that uses discrete graphical models for reasoning, to tackle symbol grounding. This architecture is trained efficiently using the E-NPLL, a probabilistic loss that requires to observe a complete assignment of the variables. We propose a new approach based on imputation of unobserved variables to enable the use of the E-NPLL for the symbol grounding problem. Experimentally, we first show the efficiency, both in time and data, and interpretability of the E-NPLL on the task on learning the rules of Sudoku from solution examples. We then assess our imputation strategy on the standard Visual Sudoku benchmark.

1 Introduction

In recent years, several neurosymbolic architectures have been proposed to integrate discrete reasoning or optimization within neural networks. The main motivation is the ability to process natural inputs while simultaneously exhibiting extensive logical reasoning capabilities. This is a promising direction towards Artificial General Intelligence (AGI).

One of the pioneer neurosymbolic architectures, SATNet [Wang *et al.*, 2019], illustrated those capacities on the Visual Sudoku task, where the architecture must simultaneously learn how to solve Sudoku puzzles and how to recognize the hand-written digits providing initial hints (see Figure 1). If the original paper reported convincing performance, later work [Chang *et al.*, 2020] showed the architecture 'cheated' by using direct supervision to learn the mapping from the visual inputs to symbolic numbers. Without this direct supervision, performances collapsed. Indeed, learning this mapping – the symbol grounding task – is a much more challenging task and subsequent works improving SATNet for symbol grounding required heavy machinery, such as a large neural net for unsupervised clustering [Topan *et al.*, 2021].

In this paper, we aim for a light-weight and efficient architecture for symbol grounding. We build upon a previously proposed architecture learning discrete graphical models from examples of solutions [Defresne *et al.*, 2023]. For instance, it is able to learn the rules of Sudoku, from a set of filled grids. We chose this approach because it is data-efficient, has low-training time and has been proved efficient on both benchmark and real-life problems. This architecture is trained using the Emmental-NPLL (E-NPLL), a probabilistic loss that requires to observe all the variables of the target solution. In the case of symbol grounding, some variables are unobserved to prevent cheating from direct supervision.

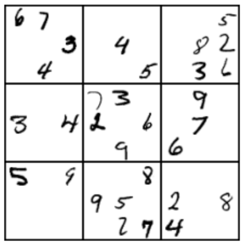
To extend the E-NPLL for symbol grounding, we propose a new approach using data imputation to infer the missing variables and obtain a complete assignment. Then, the generic method based on the E-NPLL [Defresne *et al.*, 2023] can be applied. We first experimentally evaluate the data-efficiency and parameter-efficiency of the hybrid architecture trained under the E-NPLL on the standard Sudoku benchmark (further than done in [Defresne *et al.*, 2023]). We also show the interpretability of the approach by retrieving the learned logical constraints to show the exact rules have been learned. We then demonstrate on the Visual Sudoku benchmark that data imputation enables the hybrid architecture to solve the associated symbol grounding problem. Our approach is competitive with state-of-the-art approaches while being 32% faster.

2 Related work

In the wake of the seminal work of SATNet [Wang *et al.*, 2019], proven to fail at symbol grounding [Chang *et al.*, 2020] and improved accordingly [Topan *et al.*, 2021], several directions were explored for symbol grounding: constraint satisfaction problems with a recurrent Transformer [Yang *et al.*, 2023], grounding with Satisfiability Modulo Theory [Wang *et al.*, 2023], casting symbol grounding as a game with two optimization problems (neural and symbolic learning) [Li *et al.*, 2023a] or inspired by policy from Reinforcement Learning [Daniele *et al.*, 2023]. Except for the last one, they all used Visual Sudoku as a benchmark.

3 Background

Notations. We denote sequences, vectors, and tensors in bold. Variables are denoted in capitals with a given vari-



$$\text{values of variable } j\left\{\begin{array}{c|ccccccccc}
 & \multicolumn{9}{c}{\overbrace{\hspace{6cm}}^{\text{values of variable } i}} \\
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
1 & \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & \infty & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & \infty & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & \infty & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & \infty & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & \infty & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & 0 & 0 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \infty
\end{array}\right.$$

Figure 1: Left: a visual Sudoku grid. Right: the cost function stating that variables i and j must be different. The complete CFN representing Sudoku rules has one such cost function for each pair of variables on the same row, column or sub-square .

able $Y_i \in \mathbf{X}$ being the i^{th} variable in the sequence $\mathbf{X}$. An assignment of the variables in $\mathbf{X}$ is denoted $\mathbf{y}$ and y_i is the assignment of Y_i in $\mathbf{y}$.

For reasoning, we use discrete graphical models (GM). They concisely describe a complex function of many variables as the combination of many simple functions with few variables. GMs cover a large spectrum of NP-hard reasoning and optimization frameworks including Constraint Networks and Propositional Logic [Cooper *et al.*, 2020]. Here we use Cost Function Networks (CFN) for their ability to express both numerical functions (such as probability over digits) and logical ones (*e.g.,* rules of Sudoku).

Definition 1. *Given a sequence $\mathbf{X} = \{Y_1, \dots, Y_n\}$ of n finite domain variables, a cost function network $\mathcal{M}$ is defined as a set of elementary cost functions. It defines a joint cost function, also denoted $\mathcal{M}(\cdot) = \sum_{F \in \mathcal{M}} F$, involving all variables in $\mathbf{X}$. The optimization problem, known as the Weighted Constraint Satisfaction Problem (WCSP), is to find an assignment $\mathbf{y}$ that minimizes the joint function $\mathcal{M}(\mathbf{y})$. If $\mathcal{M}(\mathbf{y}) < \infty$, $\mathbf{y}$ is called a solution.*

CFNs are equivalent to Markov Random Fields (MRFs) through exponentiation of the joint cost function to define a joint probability distribution : $P^{\mathcal{M}} \propto \exp(-\mathcal{M}(\cdot)) = \prod_{F \in \mathcal{M}} \exp(-F)$. Intuitively, a high cost corresponds to a low probability.

Figure 1 illustrates one cost function representing the rules of Sudoku with CFN. The rule between each pair of cell is represented by a cost matrix of shape 9×9 (the size of the domains). All pairs of cells on the same row, column or 3 sub-square must be different, which is represented by a soft difference-like cost function (a matrix with a strictly positive diagonal and zeros) that prevents the use of the same value for the two variables. Other cost matrices are 0, indicating the absence of a pairwise constraint.

4 Method

4.1 Dataset and loss

We assume that we observe a dataset S of samples $(\omega, \mathbf{y})$ where ω is a natural input that can be described by a constrained optimization problem and $\mathbf{y}$ is one low-cost solution of this problem. In the example of Visual Sudoku, ω is the input grid with hand-written hints and $\mathbf{y}$ is the solution of the grid. From S we aim to train a neural network N to predict a pairwise CFN $\mathcal{M} = N(\omega)$ of which $\mathbf{y}$ is a low-cost solution, *i.e.,* $\mathbf{y} \in \arg\min_{\mathbf{y} \in D^{\mathbf{X}}} N(\omega)(\mathbf{y})$. At inference, a solution for a new input ω is proposed by predicting the CFN representing ω and solving it with a discrete solver [Hurley *et al.*, 2016].

The neural network N can be trained using the Emmental-NPLL (E-NPLL) [Defresne *et al.*, 2023], that relies on the probabilistic interpretation of a CFN.

Definition 2. *Given a GM $\mathcal{M}$ over variables $\mathbf{X}$ and $\mathbf{H} \subset \mathbf{X}$, we denote by $\mathcal{M} - \mathbf{H}$ the graphical model derived from $\mathcal{M}$ by replacing all cost functions involving a variable in $\mathbf{H}$ by a constant 0 function. The E-NPLL can then be defined as*

$$\textit{E-NPLL}(\mathbf{y}) = -\sum_{Y_i \in \mathbf{X}} \log(P^{(N(\omega) - \mathbf{H}_i)}(y_i | \mathbf{y}_{-i}))$$

where $\mathbf{H}_i$ is a random subset of $\{1, \dots, n\} \setminus \{i\}$ and $\mathbf{y}_{-i}$ is the sequence of variables $\mathbf{y}$ after removal of variable y_i.

To make the predicted optimization problem $N(\omega)$ easier to solve, we add to the E-NPLL loss a L1 regularization on $N(\omega)$ to encourage the GM to be sparse. Note the regularization is on the output of the neural network and not on its weights. The complete training loss is then:

$$\mathcal{L}(\mathbf{y}) = \text{E-NPLL}(\mathbf{y}) + ||N(\omega)||_1$$

4.2 Data imputation for symbol grounding

Symbol grounding consists in learning a mapping from visual inputs to symbols without explicit supervision. On neurosymbolic problems with a perception task and a reasoning task, solving the grounding problem is much more challenging than solving each task separately [Chang *et al.*, 2020]. To prevent any data leakage that would provide a direct supervision signal, the symbols corresponding to the visual inputs must be masked [Chang *et al.*, 2020], leading to samples $(\omega, \mathbf{y})$ with partial assignment $\mathbf{y}$. In the example of the Visual Sudoku benchmark, each image of a digit in the grid is known to represent the value of a single variable. To prevent direct supervision for digit recognition, and enforce simultaneous learning with Sudoku rules, initial hints are masked in the sequence $\mathbf{y}$, leading to unobserved variables.

The E-NPLL is defined only over complete assignments. To apply it for solving symbol grounding problems, the observed partial assignment $\mathbf{y}$ should be completed through data imputation. A usual strategy to deal with missing data is to rely on variants of the expectation-maximization (EM) algorithm [Qu *et al.*, 2019]. However, it is intractable in our case since it requires computing expensive marginals (#-P hard).

Instead, we define a simpler imputation procedure based on the GM solver. The values of the missing variables in $\mathbf{y}$ are obtained by solving the predicted CFN $N(\omega)$ with all variables observed in $\mathbf{y}$ being assigned to their values. The resulting complete assignment is then used instead of the partial $\mathbf{y}$ to compute the E-NPLL loss. During training, this imputation strategy requires one solver call (NP-hard) per sample with missing data. When the fraction of unobserved variables remains limited, the solved problems are simple, with just a few unassigned variables. In our experiments, we use an exact GM solver for imputation [Hurley *et al.*, 2016].

Type	Approach	Acc.	#hints	Train set	Param.	Train time (h)
DL	RRN [Palm *et al.*, 2018]	96.6%	17	180,000	200k	> 100
	Rec. Trans. [Yang *et al.*, 2023]	76.2–78.2%	17	180,000	211k	1.8
	DDPM [Ye *et al.*, 2025]	99.2–100%	33.8	100,000	6M	13.6
	DDPM	0.2%	17	-	-	-
Relax+DL	SATNet [Wang *et al.*, 2019]	95.1–99.8%	36.2	9,000	600k	2.9
	SATNet	86.1–86.2%	17	-	-	-
CO	[Bessiere *et al.*, 2023]	**100%**	-	200	-	0.01
CO + ML	[Brouard *et al.*, 2020]	**100%**	17	9,000	-	1.5
CO+DL	Hinge [Defresne *et al.*, 2023]	**100%**	17	1,000	180k	>50
	E-NPLL [Defresne *et al.*, 2023]	**100%**	17	200	180k	0.25
	E-NPLL (here)	**100%**	17	**100**	22k	0.05

Table 1: Accuracies of related works. They are sorted by type of approach: pure Deep Learning (DL), reasoning (CO for combinatorial optimization), and relaxation of reasoning (Relax), which can be combined with ML or DL. The '# hints' gives the average hardness of the test set. Param. is the number of parameters of the neural network.

5 Experiments

Reported training times have been measured using a Nvidia RTX-2070 Super GPU with 8GB of VRAM and a 4.2 GHz AMD Ryzen 9 5900X CPU with 32 GB of RAM. During our measures, we noticed that the reported training accuracies were often slightly below those in the original papers. We therefore reran them to collect 3 results (including the paper result) and we report the min/max accuracies observed. The only exception is RNN because of its training time. Our code is run with PyTorch 2.6 and PyToulbar2 0.0.0.4. We use the Adam optimizer with a weight decay of 10^{-4} and a learning rate of 10^{-3} (other parameters take default values). An L1 regularization with multiplier 2.10^{-4} is applied on the cost matrices $N(\omega)[i, j]$. Code and data will be made available.

5.1 Efficiency on the Sudoku task

We first motivate the use of the E-NPLL by demonstrating its efficiency, both in terms of training time and data, on the NP-complete Sudoku problem. It is a classical logical reasoning problem that has been repeatedly used as a benchmark in a "learning to reason" context [Palm *et al.*, 2018; Wang *et al.*, 2019; Brouard *et al.*, 2020; Defresne *et al.*, 2023; Yang *et al.*, 2023; Li *et al.*, 2023a; Bessiere *et al.*, 2023; Ye *et al.*, 2025]. The task is to learn how to solve new Sudoku grids from a set of solved grids, without knowing the game rules. The E-NPLL has already been proven to be efficient in this task [Defresne *et al.*, 2023], both in terms of time and data. Here, we make the architecture even more efficient and compare it to the most recent approaches.

Task. Given samples $(\omega^\ell, \mathbf{y}^\ell)$ of initial and solved Sudoku grids, we want to learn how to solve new grids. Sudoku players know that Sudoku grids can be more or less challenging. As one could expect, it is also harder to train how to solve hard grids than easy grids [Brouard *et al.*, 2020]. We use the number of initially filled cells (hints) as a proxy for the problem's hardness, a grid with few hints being hard. The minimal number of hints required to define a single solution is 17, defining the hardest single-solution Sudoku grids. We use the RRN data set [Palm *et al.*, 2018], composed of single-solution grids with 17 to 34 hints. We train and validate on all-hardness grids. As in [Palm *et al.*, 2018], we test on the hardest 17-hints instances, $1,000$ in total.

Neural architecture. A 9×9 Sudoku grid is represented as 81 cell coordinates, possibly with a hint. Each cell is represented by a variable with domain $\{1, \dots, 9\}$. For N, we reuse the same architecture [Defresne *et al.*, 2023], but we drastically reduce the number of parameters. We use a Multi-Layer Perceptron (MLP) with 4 hidden layers of 64 neurons and residual connections [He *et al.*, 2016] every 2 layers. It results in 9 times fewer parameters. The neural net receives the pairs of coordinates of pairs of cells (Y_i, Y_j) and predicts all pairwise cost matrices $N(\omega)[i, j]$. Hints are used to set the values of their corresponding variable in $N(\omega)$. Performances are measured by the percentage of correctly filled grids; no partial credit is given for individual digits.

Test. In Table 1, we compare our results with related approaches that learn how to solve Sudoku. Pure Deep Learning methods, Recurrent Relational Network (RRN) [Palm *et al.*, 2018], Recurrent Transformer [Yang *et al.*, 2023], and Denoising Diffusion Probabilistic Models (DDPMs) [Ye *et al.*, 2025], require orders of magnitude more data and fail to solve some of the hardest puzzles. DDPMs [Ye *et al.*, 2025] solve simple Sudokus (from a Kaggle dataset) quite reliably, but completely fail on hard Sudokus. Adding a convex relaxation of Max-SAT reasoning optimization layer, SATNet [Wang *et al.*, 2019] becomes much more data-efficient. Still, it fails to solve all easy grids, and its accuracy drops significantly on hard grids, below that of pure DL approaches.

Reasoning alone [Bessiere *et al.*, 2023], or combined with ML [Brouard *et al.*, 2020], learns the proper rules, solves all test instances reliably, and offers far more efficient training than DL-based approaches. However, these non-end-to-end differentiable approaches cannot directly exploit natural inputs, as in the Visual Sudoku or protein design tasks described below. Finally, hybrid approaches combine both DL and exact reasoning. A follow-up work [Li *et al.*, 2023b] of SATNet,

Approach	MNIST accuracy	Percep.	Solved	Training (h)
Rec. Trans [Yang *et al.*, 2023]	99.4%	74.8%	75.6%	5.1
NeSy. Prog. [Li *et al.*, 2023a]	99.6–99.7%	90.7–93.1%	92.2–94.4%	4.7
E-PLL (here)	98.7–98.8%	69.3–72.9%	92.1–93.4%	3.2

Table 2: Grounded Visual Sudoku performance (balanced RNN test set). Percep. refers to the percentage of grids correctly filled without modifying the recognized digits, while Solved is the percentage of correct grids after correction of misclassified digits. The minimum and maximum values observed of 3 runs are indicated (when they differ).

extracts explicit logical rules from SATNet, enabling reliable solving on 4×4 grids. On 9×9 grids, it learns hundreds of thousands of clauses, leading to unsolvable instances. Approaches that embed an exact solver during training, as the structured Perceptron/Hinge losses, solve any grids [Defresne *et al.*, 2023], but at the cost of an excruciatingly long training time. Overall, the E-NPLL stands out for its perfect reliability, its data-efficiency, and its low training time. Compared to the first-proposed architecture [Defresne *et al.*, 2023], ours offers a five-time reduced training time, training in less than 3 minutes on a single CPU, and requires only half of the training grids (100 in total).

Retrieving exact constraints After learning, logical constraints can be retrieved with two alternative strategies. When constraints only need to be learned, a threshold can be applied : learnt costs below the threshold are set to 0 and costs above are set to ∞ (hard constraint). The second strategy is cost function hardening [Brouard *et al.*, 2020], which has the advantage of requiring no parameter and also of preserving the learned criterion (if needed). Non-zero learned costs are considered in decreasing order and set to ∞ if the corresponding value combination is not observed in any of the training set. This is repeated until a contradiction is found. When applying cost function hardening to the output of the neural network trained for the Sudoku task, all of the 810 pairwise constraints are predicted, with no additional constraints. Therefore, the exact rules are learnt and we can be confident that the test accuracy of 100% extends to any Sudoku instance.

5.2 Visual Sudoku Grounding

Task. In the Visual Sudoku problem, hints are images of hand-written digits. The goal is to simultaneously learn how to recognize digits and how to play Sudoku. Our data set is obtained from the symbolic Sudoku data set by replacing hints with arbitrarily selected MNIST images, as in [Brouard *et al.*, 2020; Topan *et al.*, 2021]. We use grids from the RRN dataset ($9,000$ for training, 64 for validation), as they are much more challenging than the SATNet dataset (average 36.2 hints) [Wang *et al.*, 2019]. The test set contains 100 grids of each difficulty (from 17 to 34 initial hints). For training samples, all labels corresponding to an input hint image are masked in the ground-truth sequence $\mathbf{y}$. Thus, between 17 and 34 variables are masked for each solution.

Neural architecture. To recognize hand-written digits, we add an untrained convolutional neural network (CNN, the same as in [Wang *et al.*, 2019]) to our previous architecture. The logits predicted by the CNN are negated and interpreted as a unary cost function on each variable Y_i with a hint. The GM produced by $N(\omega)$ comprises the pairwise cost functions predicted by the ResMLP (as before) and the unary cost functions. In order to compute the E-NPLL, missing labels in $\mathbf{y}$ are imputed by solving the predicted GM – only on unobserved variables – using the exact solver toulbar2 [Hurley *et al.*, 2016]. Finally, the regularized E-NPLL is computed over the completed solution and back-propagated. During the first epochs, the predicted rules are mostly random, making the imputation long. Therefore, we first restrict training to grids with few initial hints: only 17 on first epoch, then less than 20 on the second, 30 on the third and so on until all training grids are included. In average, imputation takes 0.05s per Sudoku grid. Finally, we divide the learning rate of both neural nets by 10 at epochs 6 and 8 (out of 20).

Test. We compare our architecture with the two methods [Yang *et al.*, 2023; Li *et al.*, 2023a] applying grounding to the RRN dataset as they outperformed other competitors [Topan *et al.*, 2021; Wang *et al.*, 2023] on the simpler SATNet dataset. After visual Sudoku training, we assessed the accuracy of the CNN alone on MNIST digits, the percentage of correctly-filled grids when no correction of MNIST classification is allowed, and the final percentage of correct grids with corrections by the solver. We only compare methods tackling the ungrounded problem, where no label is available for hints. The resulting perception architecture is not as accurate as competitors, with a 1% lower accuracy, leading to far fewer properly filled grids. Yet, the Sudoku rules are learnt properly, enabling the solver to correct 20% of the grids eventually reaching a competitive 93.3% of visual grids solved, with a training time reduced by 32%.

6 Conclusion

Symbol grounding, or the ability to map visual inputs to symbols, is a crucial ability for neurosymbolic architectures to progress towards AGI. We build upon such an architecture, that relies on discrete graphical models and the E-NPLL for training [Defresne *et al.*, 2023], and propose a data imputation scheme to enable it to tackle symbol grounding problems, where some variables are unobserved. We show on the Sudoku benchmark that this hybrid architecture is still the most time-and-data efficient while being interpretable in terms of rules learnt. With the data imputation, our approach is competitive on the grounded Visual Sudoku while faster.

For the data imputation, we used an exact solver, which can be time-consuming if many variables are unobserved. Heuristic or approximate solvers [Durante *et al.*, 2022] could be used instead, with multiple imputations [Little and Rubin, 2019] if necessary.

References

[Bessiere *et al.*, 2023] Christian Bessiere, Clément Carbonnel, and Areski Himeur. Learning constraint networks over unknown constraint languages. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 1876–1883. International Joint Conferences on Artificial Intelligence Organization, 8 2023. Main Track.

[Brouard *et al.*, 2020] Céline Brouard, Simon de Givry, and Thomas Schiex. Pushing data into CP models using graphical model learning and solving. In *International Conference on Principles and Practice of Constraint Programming*, pages 811–827. Springer, 2020.

[Chang *et al.*, 2020] Oscar Chang, Lampros Flokas, Hod Lipson, and Michael Spranger. Assessing SATNet's ability to solve the symbol grounding problem. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1428–1439. Curran Associates, Inc., 2020.

[Cooper *et al.*, 2020] Martin Cooper, Simon de Givry, and Thomas Schiex. Graphical models: queries, complexity, algorithms. In *Symposium on Theoretical Aspects of Computer Science, Leibniz International Proceedings in Informatics*, volume 154, pages 4–1, 2020.

[Daniele *et al.*, 2023] Alessandro Daniele, Tommaso Campari, Sagar Malhotra, and Luciano Serafini. Deep symbolic learning: discovering symbols and rules from perceptions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 3597–3605, 2023.

[Defresne *et al.*, 2023] Marianne Defresne, Sophie Barbe, and Thomas Schiex. Scalable Coupling of Deep Learning with Logical Reasoning. In *Thirty-second International Joint Conference on Artificial Intelligence, IJCAI'2023*, 2023.

[Durante *et al.*, 2022] Valentin Durante, George Katsirelos, and Thomas Schiex. Efficient low rank convex bounds for pairwise discrete Graphical Models. In *Thirty-ninth International Conference on Machine Learning*, Baltimore, United States, July 2022.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[Hurley *et al.*, 2016] Barry Hurley, Barry O'sullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias Zytnicki, and Simon de Givry. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*, 21:413–434, 2016.

[Li *et al.*, 2023a] Zenan Li, Yunpeng Huang, Zhaoyu Li, Yuan Yao, Jingwei Xu, Taolue Chen, Xiaoxing Ma, and Jian Lu. Neuro-symbolic learning yielding logical constraints. *Advances in Neural Information Processing Systems*, 36:21635–21657, 2023.

[Li *et al.*, 2023b] Zhaoyu Li, Jinpei Guo, Yuhe Jiang, and Xujie Si. Learning reliable logical rules with satnet. *Advances in Neural Information Processing Systems*, 36:14837–14847, 2023.

[Little and Rubin, 2019] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2019.

[Palm *et al.*, 2018] Rasmus Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[Qu *et al.*, 2019] Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: Graph markov neural networks. In *International conference on machine learning*, pages 5241–5250. PMLR, 2019.

[Topan *et al.*, 2021] Sever Topan, David Rolnick, and Xujie Si. Techniques for symbol grounding with SATnet. *Advances in Neural Information Processing Systems*, 34:20733–20744, 2021.

[Wang *et al.*, 2019] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6545–6554. PMLR, 09–15 Jun 2019.

[Wang *et al.*, 2023] Zifan Wang, Saranya Vijayakumar, Kaiji Lu, Vijay Ganesh, Somesh Jha, and Matt Fredrikson. Grounding neural inference with satisfiability modulo theories. *Advances in Neural Information Processing Systems*, 36:22794–22806, 2023.

[Yang *et al.*, 2023] Zhun Yang, Adam Ishay, and Joohyung Lee. Learning to solve constraint satisfaction problems with recurrent transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[Ye *et al.*, 2025] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *The Thirteenth International Conference on Learning Representations*, 2025.