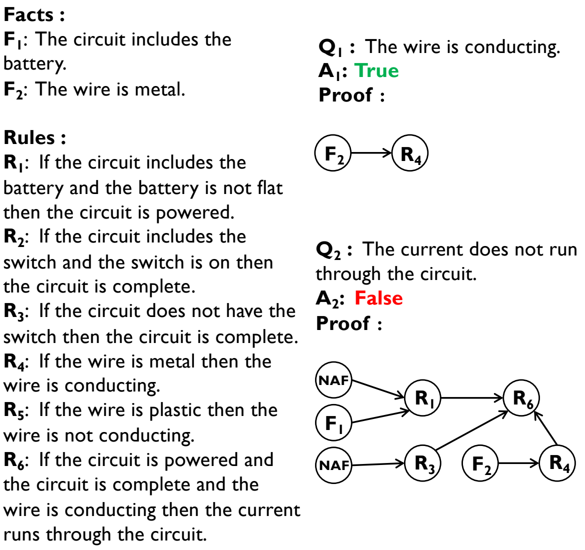
Probabilistic Graph Reasoning for Natural Proof Generation

Changzhi Sun[†*], Xinbo Zhang[†*], Jiangjie Chen[†§], Chun Gan[†¶],
Yuanbin Wu[‡], Jiaze Chen[†], Hao Zhou[†], and Lei Li[†]

[†]ByteDance AI Lab
[§]School of Computer Science, Fudan University
[¶]Math Department, University of Wisconsin–Madison
[‡]School of Computer Science and Technology, East China Normal University
{sunchangzhi,zhangxinbo.freya}@bytedance.com
{chenjiaze,zhouhao.nlp,lileilab}@bytedance.com
jjchen19@fudan.edu.cn, cgan5@wisc.edu, ybwu@cs.ecnu.edu.cn

Abstract

In this paper, we investigate the problem of reasoning over natural language statements. Prior neural based approaches do not explicitly consider the inter-dependency among answers and their proofs. In this paper, we propose PRobr, a novel approach for joint answer prediction and proof generation. PRobr defines a joint probabilistic distribution over all possible proof graphs and answers via an induced graphical model. We then optimize the model using variational approximation on top of neural textual representation. Experiments on multiple datasets under diverse settings (fully supervised, few-shot and zero-shot evaluation) verify the effectiveness of PRobr, e.g., achieving 10%-30% improvement on QA accuracy in few/zero-shot evaluation. Our codes and models can be found at https://github.com/changzhisun/PRobr/.



Facts :
$\mathbf{F_1}$: The circuit includes the battery.
$\mathbf{F_2}$: The wire is metal.

Rules :
$\mathbf{R_1}$: If the circuit includes the battery and the battery is not flat then the circuit is powered.
$\mathbf{R_2}$: If the circuit includes the switch and the switch is on then the circuit is complete.
$\mathbf{R_3}$: If the circuit does not have the switch then the circuit is complete.
$\mathbf{R_4}$: If the wire is metal then the wire is conducting.
$\mathbf{R_5}$: If the wire is plastic then the wire is not conducting.
$\mathbf{R_6}$: If the circuit is powered and the circuit is complete and the wire is conducting then the current runs through the circuit.

$\mathbf{Q_1}$ **:** The wire is conducting.
$\mathbf{A_1}$: **True**
Proof :

$\mathbf{Q_2}$ **:** The current does not run through the circuit.
$\mathbf{A_2}$: **False**
Proof :

Figure 1: An example of reasoning over natural language statements. The goal is to predict the answer (true/false) and generate the proof graph.

1 Introduction

Automatic reasoning over explicitly provided knowledge has been a persistent goal of AI (Newell and Simon, 1956; McCarthy et al., 1960). Early approaches focus on reasoning over formal (logical or probabilistic) representations. However, automatically constructing and reasoning over formal representations remain challenging. To bypass these challenges, in this work, we investigate reasoning over natural language statements instead of formal representations.

Given a set of facts and rules and a query (expressed in natural language), we aim to predict the answer and provide proof to prove or disprove the query. For example, in Figure 1, there are two facts, six rules and two queries, each of which is expressed by natural language. To predict the true/false of each query, starting from the facts, we need to reason deductively by applying given rules until we can derive the truth value of the query. The process of deduction can be represented as a graph, whose node is either a fact, rule or special NAF node (explained in the Section 2.1). Generating answer and proof together makes a system easier to interpret and diagnose.

Recent work by PRover (Saha et al., 2020) first explored this problem through two modules: question answering and proof generation. It trains these two modules through implicit parameter sharing, and then uses integer linear programming (ILP) to enforce consistency constraints (only test time). It is difficult to ensure that the proof generation module contributes to the question answering module, because the proof is not explicitly involved in the answer prediction. Parameter sharing becomes more limited under few/zero-shot settings, as demonstrated in our experiments. We expect the proof to enhance the capability of question answering, especially under few/zero-shot settings. One

[*]Equal contribution.

promising solution is to explicitly exploit more interaction between question answering and proof generation.

In this paper, we propose PROBR, a novel **prob**abilistic graph **r**easoning framework for joint question answering and proof generation. PROBR defines a joint distribution over all possible proof graphs and answers with an undirected *probabilistic graphical model* (PGM). It directly characterizes the interaction between proofs and answers. PGMs generally incur intractable learning and inference for the complex graph (Koller and Friedman, 2009). For example, computing normalization constant in PGMs using traditional probabilistic propagation algorithm (e.g.sum-product algorithm (Kschischang et al., 2001)) requires large time complexity. Therefore, we propose a variational approach to maximize the pseudolikelihood of joint distribution to optimize the model more efficiently. First, a variational distribution was introduced based on mean-field assumption. Then we maximize the pseudolikelihood of joint distribution given the output of variational distribution. At the same time, we align these two distributions using the training data. PROBR can be efficiently trained by stochastic gradient descent. Our contributions are summarized as follows[1]:

- We propose PROBR for joint question answering and proof generation, which defines a joint distribution over all possible proofs and answers with an undirected PGM to capture more dependencies.

- We present an efficient variational approximation method to learn PROBR.

- Experiments on several datasets verify the effectiveness of PROBR under multiple settings (supervised, few-shot, and zero-shot evaluation).

2 Task Definition

To reason over natural language statements, we design to answer the query and generate corresponding proof generation jointly. Figure 1 shows an example. Given a declarative query Q, and given relevant facts and rules (expressed in natural language), the task aims to predict the answer A (true/false) to the query Q based on the closed-world assumption (described in 2.1). Meanwhile, it generates a proof P (described in 2.2) to prove or disprove Q.

2.1 Semantics

We adopt the semantics of Datalog (Ceri et al., 1989) in this work. Following prior work (Clark et al., 2020), we make a closed-world assumption (CWA), which means a fact is true if it can be deduced based on a given context, and any fact not provable is assumed false. And we use negation as failure (NAF) (Clark, 1978), a rule of inference which allows one to deduce that `NOT S` is true if all possible proofs of a statement `S` fail. For example, in Figure 1, the `NAF` node before $\mathtt{R}_3$ represents “the circuit does not have the switch”. Note that under this semantics, negative facts and negative rules are not allowed because of redundancy under the CWA assumption.

2.2 Formulations

A proof is a directed acyclic graph (Figure 1). Each node is either a fact, rule or special `NAF` node. Each edge directs from either a fact (or `NAF`) to a rule or a rule to another rule, which indicates that a fact is consumed by a rule, or another rule consumes a rule, respectively. For simplicity, let context $C = \{s_1, \ldots, s_n\}$ denote the collection of sentence, each of which is a fact or rule.

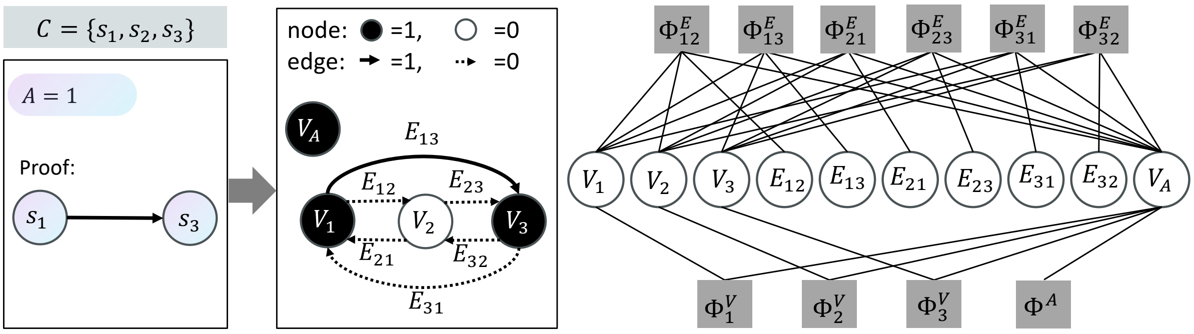
Proof Formulation We assign an indicator variable (0/1) for each possible node and edge, to vectorize the structure of a given proof P. Specifically, we introduce the indicator variables $V = \{V_i\}_{i=1}^n$ for each element s_i in the context C, and an indicator variable $E = \{E_{ij}\}_{i,j=1}^n$ $(i \neq j)$ for a possible edge connecting from node s_i to node s_j, where:

- $V_i = 1$ indicates s_i is in the proof P, while $V_i = 0$ means s_i is absent.

- $E_{ij} = 1$ indicates there is an edge directing from s_i to s_j, while $E_{ij} = 0$ means s_i cannot direct to s_j in the proof by an edge.

In addition, we assign a binary answer variable A to indicate the true value of the query. Figure 2a shows a simplified example, where context $C = \{s_1, s_2, s_3\}$, and the query can be decided as true by a very simple proof, consisting of only two nodes (s_1 and s_3) and a single edge (from s_1 to s_3). The proof can be represented by the following variables: $A = V_1 = V_3 = E_{13} = 1, V_2 = E_{12} = E_{21} = E_{23} = E_{31} = E_{32} = 0$.

[1] Our codes and models can be found at https://github.com/changzhisun/PRobr/.



$C = \{s_1, s_2, s_3\}$
$A = 1$
Proof:
s_1
s_3
node: ● =1, ○ =0
edge: → =1, ⇢ =0
V_A
E_{13}
E_{12}
E_{23}
V_1
V_2
V_3
E_{21}
E_{32}
E_{31}
Φ^E_{12} Φ^E_{13} Φ^E_{21} Φ^E_{23} Φ^E_{31} Φ^E_{32}
V_1 V_2 V_3 E_{12} E_{13} E_{21} E_{23} E_{31} E_{32} V_A
Φ^V_1 Φ^V_2 Φ^V_3 Φ^A

(a) Proof graph and its induced random variables.

(b) Factor graph induced by the proof graph.

Figure 2: Joint probabilistic distribution by assigning indicator variables for the answer and proof. The solid circles and lines in 2a indicate that these corresponding statements and edges are in the final proof graph.

3 Approach

We introduce the proposed framework PROBR, which jointly provides the answer to the given query over natural contexts and generates corresponding proof. Different from PROVER that makes independence assumption, PROBR can capture more dependencies between the proof and answer. PROBR defines a joint distribution over all possible proofs and answers with an undirected graphical model (Section 3.1), and we use neural networks to parameterize each component (Section 3.2). To optimize PROBR efficiently, we adopt a variational approach to maximize the pseudolikelihood of joint distribution (Section 3.3). Finally, we introduce the strategy during inference (Section 3.4).

3.1 Overview

We start by formalizing joint question-answering module and proof-generation module in a probabilistic way. We clarify some notations as follows:

- A context $C = \{s_1, \dots, s_n\}$, s_i is a sentence.
- A query Q.
- An answer variable A, it can take any value a in $\{0, 1\}$.
- Node variables $\mathcal{V} = \{V_i\}_{i=1}^{n}$, each V_i can take any value v_i in $\{0, 1\}$.
- Edge variables $\mathcal{E} = \{E_{ij}\}_{i,j=1}^{n}$ $(i \neq j)$, each E_{ij} can take any value e_{ij} in $\{0, 1\}$.
- Let $Y \triangleq (A, \mathcal{E}, \mathcal{V})$ denote all output variables.

In our notation, we use uppercase letters for variables (e.g., Y, A, V_i, E_{ij}) and lowercase letters for variables that take values (e.g., y, a, v_i, e_{ij}).

Given a context C and a query Q, PROBR tries to assign true/false values for all variables, including answer variable A, node variables V and edge variables E. We define a joint distribution over all possible Y, officially denoted as $p(Y)$: [2]

$$p(Y = y) \propto \Phi^A(a) \prod_i \Phi^V_i(v_i, a) \prod_{i,j} \Phi^E_{ij}(v_i, v_j, e_{ij}, a) \quad (1)$$

Different from PROVER that makes independent assumption, such a factorization of Equation 1 can characterize the interaction between the variables V_i, V_j, E_{ij} and A. Figure 2b shows the factor graph of joint distribution $p(Y)$ for the example in Figure 2a. Theoretically, when we have the ground truth y^*, [3] we can minimize the following objective:

$$\mathcal{L}_{\text{joint}} = -\log p(Y = y^*) \quad (2)$$

However, the normalization constant of $p(Y)$ is hard to calculate due to high-order factors of large size (RHS of Equation 1). In this paper, we provide a variational-based solution to optimize objective $\mathcal{L}_{\text{joint}}$ (Section 3.3).

3.2 Parameterization

We use neural networks to parameterize each potential function of Equation 1: Φ^A, Φ^V_i and Φ^E_{ij}.

Text Representation Network Given a context C and a query Q, to obtain a contextual representations, we use RoBERTa (Liu et al., 2019) as our backbone network following (Clark et al., 2020; Saha et al., 2020). The input to RoBERTa is the concatenation of C and Q, separated by [SEP] tokens, denoted as: [CLS], C, [SEP], [SEP], Q, [SEP].

[2] We drop the input variables for clarity.

[3] We use $*$ to indicate the ground truth in the text.

Potential Function for the Answer (Φ^A) After the RoBERTa encoding, we can get the global representation of the entire input through the first token $[\texttt{CLS}]$, denoted as $h_{[\texttt{CLS}]}$. To score the possible values of variable A, i.e. 0 or 1, we use a multilayer perceptron (MLP) as a nonlinear transformation:

$$\begin{bmatrix} \Phi^A(A=0) \\ \Phi^A(A=1) \end{bmatrix} = \mathrm{MLP}_1(h_{[\texttt{CLS}]}) \in \mathbb{R}^2$$

Potential Function for Statements (Φ_i^V) For each sentence s_i (a fact or a rule), we compute the sentence representation h_{s_i} by performing a mean pool of the all token representation based on the output of RoBERTa. It is worth noting that $\texttt{NAF}$ is a special fact, we calculate $h_{\texttt{NAF}}$ through linear transformation on $h_{\texttt{CLS}}$. To score the possible values of variables (V_i, A), we also use another MLP as a score function:

$$\begin{bmatrix} \Phi_i^V(V_i=0, A=0) \\ \Phi_i^V(V_i=0, A=0) \\ \Phi_i^V(V_i=0, A=0) \\ \Phi_i^V(V_i=1, A=1) \end{bmatrix} = \mathrm{MLP}_2(h_{s_i}) \in \mathbb{R}^4,$$

where the dimension 4 indicates the number of possible values for the combination of variables V_i and A. We share the parameters of MLP_2 across all sentences.

Potential Function for Statement Relations (Φ_{ij}^E) For each sentence pair (s_i, s_j), we obtain the sentence pair representation h_{s_i,s_j}, by concatenating h_{s_i} and h_{s_j} with their element-wise difference (directionality). To score four variables (V_i, V_j, E_{ij}, A) simultaneously, similarly, we use a new MLP as score function:

$$\begin{bmatrix} \Phi_{ij}^E\begin{pmatrix} V_i=0, V_j=0, \\ E_{ij}=0, A=0 \end{pmatrix} \\ \vdots \\ \Phi_{ij}^E\begin{pmatrix} V_i=1, V_j=1, \\ E_{ij}=1, A=1 \end{pmatrix} \end{bmatrix} \begin{matrix} = \mathrm{MLP}_3(h_{s_i,s_j}) \\ \in \mathbb{R}^{16}, \end{matrix}$$

$$h_{s_i,s_j} = h_{s_i} \oplus h_{s_j} \oplus (h_{s_i} - h_{s_j}),$$

where $\oplus$ is the vector concatenation, and the dimension 16 indicates the number of possible values for the combination of four variables (V_i, V_j, E_{ij}, A). We also share the parameters of MLP_3 across all sentence pairs.

3.3 Learning the Model

To tackle the challenge of optimizing $\mathcal{L}_{\text{joint}}$ (Equation 2), we adopt the widely used pseudolikelihood as an alternative objective for optimization (Richardson and Domingos, 2006), bypassing the calculation of the normalization constant.

Pseudolikelihood Given a set of variable Y, the pseudolikelihood of Y is defined as:

$$p_{\text{pseduo}}(Y) = \prod_{y \in Y} p(y|Y_{-y}) =$$
$$p(A|\mathcal{E}, \mathcal{V}) \prod_i p(V_i|Y_{-V_i}) \prod_{i,j} p(E_{ij}|Y_{-E_{ij}})$$

When we have the ground truth y^*, we can minimize the following objective:

$$\mathcal{L}_{\text{pseudo}} = -\log p_{\text{pseudo}}(Y = y^*)$$

However, it is difficult to decode the optimal assignments based on the pseudolikelihood (Equation 3). There is a rich body of literature on how to decoding in a sampling way (Chapter 12 (Salakhutdinov, 2014)). In this paper, however, we choose a modern approach using variational approximation.

Variational Approximation We approximate pseudolikelihood of Y with a mean-field (Opper and Saad, 2001) variational distribution $q(Y)$, in which $y \in Y$ is independent of each other. Similarly, we parameterize each independent distribution with a neural network. Formally, $q(Y)$ is formulated as below:

$$q(Y) = q(A) \prod_i q(V_i) \prod_{i,j} q(E_{ij}),$$
$$\begin{bmatrix} q(A=0) \\ q(A=1) \end{bmatrix} = \mathrm{Softmax}\left(\mathrm{MLP}_4(h_{[\texttt{CLS}]})\right) \in \mathbb{R}^2,$$
$$\begin{bmatrix} q(V_i=0) \\ q(V_i=1) \end{bmatrix} = \mathrm{Softmax}\left(\mathrm{MLP}_5(h_{s_i})\right) \in \mathbb{R}^2,$$
$$\begin{bmatrix} q(E_{ij}=0) \\ q(E_{ij}=1) \end{bmatrix} = \mathrm{Softmax}\left(\mathrm{MLP}_6(h_{s_i,s_j})\right) \in \mathbb{R}^2.$$

Once the variational distribution $q(Y)$ is obtained, it can provide conditions for pseudolikelihood $p(y|Y_{-y})$, thus avoiding the sampling process to obtain the optimal assignments. In the optimization process, we adopt the simple strategy to update the parameters of p and q.

- For node and edge variables, we optimize

$$\mathcal{L}_{\text{node}} = -\sum_i \log q(V_i = v_i^*),$$
$$\mathcal{L}_{\text{edge}} = -\sum_{i,j} \log q(E_{ij} = e_{ij}^*).$$

- For the answer variable, we optimize

$$\mathcal{L}_{\text{qa}} = -\log p(A = a^* | \hat{\mathcal{E}}, \hat{\mathcal{V}}),$$

where $\hat{\mathcal{E}} = \{\hat{e}_{ij}\}, \hat{\mathcal{V}} = \{\hat{v}_i\}$ are the predictions of variational model [4].

The final objective is to minimize:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{qa}} + \mathcal{L}_{\text{node}} + \mathcal{L}_{\text{edge}}$$

Overall, PROBR is a mixture of independent (variational) model and undirected graphical model through some reasonable approximations. Our final optimized distribution can be decomposed as directed graphical model $q(\mathcal{V})q(\mathcal{E})p(A|\mathcal{E},\mathcal{V})$, where $q(\mathcal{V}), q(\mathcal{E})$ adopts the independent factorized probability, and $p(A|\mathcal{E},\mathcal{V})$ is implied by the undirected graphical model (Equation 1). In this way, PROBR enjoys the advantage of global normalization (undirected graphical model) and is easier to optimize (directed graphical model).

Discussion Another way to achieve consensus between $q(Y)$ and $p_{\text{pseudo}}(Y)$ is to directly optimize the KL divergence:

$$\mathcal{L}_{\text{kl}} = \sum_{y \in Y} \text{KL}\left(q(y) || p(y|Y_{-y})\right)$$

However, $\mathcal{L}_{\text{kl}}$ does not bring any improvement for supervised learning (Section 4.6), hence we exclude it during training. PROBR can be easily extended to semi-supervised learning scenario by using this $\mathcal{L}_{\text{kl}}$ term. Specifically, minimize the $\mathcal{L}_{\text{final}}$ for the labeled data; and minimize the $\mathcal{L}_{\text{kl}}$ for the unlabeled data. We save this for future work.

3.4 Inference

After training, for nodes and edges, we choose the predictions of the variational model, and for answers, we choose the prediction of the joint model based on the output of variational model. In addition, we also employ the Integer Linear Programming (ILP) to enforce consistency constraints following (Saha et al., 2020).

4 Experiments

To evaluate the effectiveness and generality of our PROBR model, we conduct both fully supervised learning, few-shot learning, and zero-shot learning over several datasets[5] against two baselines: RuleTakers and PROVER[6].

4.1 Datasets and Metrics

We use three datasets (DU0-DU5, Birds-Electricity, ParaRules) introduced by (Clark et al., 2020).

DU0-DU5 DUd (d=0,1,2,3,5) are five synthetic datasets, each containing 100k queries with theories expressed in templated English, proof graphs expressed in natural language, and answers described as *True/False*. Answers require reasoning up to depth d for queries in DUd.

Birds-Electricity This dataset is a test-only dataset of 5k samples in total. It describes birds and electric circuit, which was used to evaluate the out-of-distribution performance of the models.

ParaRules ParaRules is a dataset generated and paraphrased from sampled theories (facts + rules). It contains 40k queries against ≈2k theories, where the original templated English facts and rules are creatively paraphrased into more diverse natural language by crowdsourcing. For example, the fact "Dave is cold" can be rephrased as"After Dave got wet in the rain, he feels cold"; the rule "If someone is nice then they are young" can be rephrased into "A person described as being nice will certainly be young". Different from DUd and Birds-Electricity dataset composed of synthetic language, ParaRules can better test models' reasoning ability over human-like language.

Metrics We evaluate the performance considering both answers and proofs. For answers, we evaluate the **QA Accuracy** (**QA**). For proofs, we evaluate the **Proof Accuracy** (**PA**), and PA refers to the fraction of examples where generated proof matches exactly with the gold proof. We also report **Full Accuracy** (**FA**) to denote the faction of examples where both the answer and the proof are exactly correct.

4.2 Fully Supervised Learning

For the supervised setting, we train PROBR on the training split of the DU5 dataset with gold answer

[4] We also experiment with the gold proof to optimize $\mathcal{L}_{\text{qa}}$ (Section 4.6).

[5] https://allenai.org/data/ruletaker

[6] Please refer to supplementary materials for our hyper-parameter and computing infrastructure. For RuleTakers and PROVER, we directly adopt results reported in their papers if exist, and for extra setting beyond papers, we reproduce the baselines using provided codes and parameters:https://github.com/swarnaHub/PRover.

D	Cnt	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
0	6299	100	100	**100**	98.4	**98.4**	98.4	**98.4**
1	4434	98.4	99.0	**99.9**	93.2	**94.3**	93.1	**94.3**
2	2915	98.4	98.8	**99.9**	84.8	**86.1**	84.8	**86.1**
3	2396	98.8	99.1	**100**	80.5	**82**	80.5	**82**
4	2134	99.2	98.8	**100**	72.5	**76.1**	72.4	**76.1**
5	2003	99.8	99.3	**100**	65.1	**72.2**	65.1	**72.2**
All	20192	99.2	99.3	**99.9**	87.1	**88.8**	87.1	**88.8**

Table 1: Fully supervised learning performance compared among RuleTakers (RT), PROVER (PV) and PROBR (PB) on test split of DU5 after training on training split of DU5, reported in varying depth.

Train Data		QA		PA		FA	
		PV	PB	PV	PB	PV	PB
	100%	99.3	**99.9**	87.1	**88.8**	87.1	**88.8**
RC	10%	94.5	**99.9**	**63.6**	60.4	**63.3**	60.4
	5%	80.6	**99.7**	34.0	**44.2**	32.1	**44.2**
	1%	70.2	**88.2**	20.0	**21.6**	15.1	**20.3**
RQ	30k	97.8	**99.9**	72.5	**86.8**	72.4	**86.8**
	10k	87.1	**99.9**	44.0	**72.4**	42.7	**72.3**
	1k	51.3	**82.1**	**28.0**	21.1	15.0	**18.4**

Table 2: Few-shot performance comparison among PROVER and PROBR on test split of DU5 after training on partial DU5 samples. (Two types of training samples, RC: queries from randomly reserved contexts; RQ: randomly reserved queries.)

and gold proof and evaluate on the test split of DU5. We evaluate above metrics of varying depths d against two state-of-the-art baselines: RuleTakers (Clark et al., 2020) and PROVER (Saha et al., 2020), showed in Table 1. For RuleTakers and PROVER, we directly adopt the results reported in their paper. Note that RuleTakers can not generate a proof, so we only report the PA and FA on PROVER and PROBR. The corresponding validation set results can be found in the supplementary materials .

Overall, at each depth, PROBR generates comparable or superior QA accuracy to baselines. And for 88.8% of test examples, PROBR can generate exact proofs and answers. Similar to PROVER, the full accuracy matches the proof accuracy for PROBR, showing that in this fully supervised setting, full accuracy depends on proof accuracy at each depth. The predicted answer is always correct when the corresponding proof is correct. Actually, answering predicting is much easier than a proof generation.

When increasing depth, PROBR provides accurate answers without any loss in QA performance. It becomes harder to generate correct proofs for both PROVER and PROBR, while PROBR outperforms PROVER by 7 points of proof accuracy (65.1% → 72.2%) at depth 5.

4.3 Few-shot Learning

We explore the few-shot learning ability of PROBR against PROVER by reducing training data size. For the sake of comparison, we follow the same setting in (Saha et al., 2020), that is, randomly reserve 30k, 10k, 1k queries of overall 69762 training queries to train the model, denoted as "RQ".

It's worth noting that in the DU5 training dataset, several queries can be asked from a shared context. To better explore the ability when varying the amount of training data, we conduct another set of experiments, denoted as "RC". Specifically, we first randomly select context that appeared in the DU5 training dataset by a varying percentage, i.e., 10%, 5%, 1%, and then reserve training samples where the query is asked from the selected context. Results of both "RQ" and "RC" are showed in Table 2.

Generally speaking, proof generation is harder to improve with increased training data, while QA performance improves rapidly by enlarging the training size. PROBR widely defeats PROVER on QA accuracy in each setting in Table 2. Surprisingly, PROBR achieves 88.2% QA accuracy when training with only 700 samples (RC-1%). Overall, PROBR has a more stable ability for question answering when varying training data; however, PROVER's QA accuracy drops sharply when lacking training data. This is because that PROBR considers the joint distribution over all possible proofs and answers, and can better learn to reason over natural language statements. While as for proof accuracy, even if in some settings, PROBR loses to PROVER (RC-1%), we will soon discover that PROVER overfits to the small training data (Section 4.4 and 4.5).

Another interesting observation is that the full accuracy is not always consistent with the proof accuracy in few-shot learning, which is different from the observation in Section 4.2. Furthermore, we find that the gap between **PA** and **FA** when using PROBR is much smaller than that of PROVER. This is because PROVER trains in a multi-task way, where the question answering module and proof generation module could make independent errors, especially when training data is not enough. But PROBR can better utilize limited data to reason, which again verifies the effectiveness of PROBR.

Test	Cnt	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
B1	40	97.5	95.0	**100.0**	92.5	**100.0**	92.5	**100.0**
B2	40	100	95.0	**100.0**	95.0	**100.0**	95.0	**100.0**
E1	162	96.9	100	**100.0**	95.1	**97.5**	95.1	**97.5**
E2	180	98.3	100	**100.0**	91.7	**93.3**	91.7	**93.3**
E3	624	91.8	89.7	**98.2**	72.3	**79.3**	71.8	**79.3**
E4	4224	76.7	84.8	**95.6**	**80.6**	77.7	**80.6**	77.7
All	5270	80.1	86.5	**96.3**	**80.7**	79.3	**80.5**	79.3

Table 3: Zero-shot performance comparison among RuleTakers, PRover, and PRobr on Birds-Electricity dataset after training on DU5.

Train Data		QA		PA		FA	
		PV	PB	PV	PB	PV	PB
100%		86.5	**96.3**	**80.7**	79.3	**80.5**	79.3
RC	10%	71.2	**99.9**	**59.4**	55.4	**59.2**	55.4
	5%	59.4	**99.5**	55.0	**69.1**	46.6	**69.0**
	1%	47.1	**60.6**	15.1	**34.6**	10.6	**24.4**
RQ	30k	83.3	**99.9**	76.79	**76.91**	76.72	**76.91**
	10k	78.2	**99.7**	54.3	**56.6**	54.3	**56.6**
	1k	50.4	**51.3**	**59.5**	34.6	**29.9**	17.3

Table 4: Zero-shot performance comparison between PRover and PRobr after few-shot learning. Test on Birds-Electricity after training on DU5 or partial DU5 (RC-k and RQ-k) training partitions.

4.4 Zero-shot Evaluation

Following previous work (Clark et al., 2020; Saha et al., 2020), we evaluate the out-of-distribution (OOD) performance of PRobr against baselines on six sub-datasets of Birds-Electricity. We conduct zero-shot experiments using DU5-trained models, which means that the model does not see any bird-domain or any electricity-domain samples during training. Results are showed in Table 3.

For QA accuracy, PRobr outperforms PRover and RuleTakers obviously in all of sub-datasets. As for proof accuracy, PRobr performs better when the depth of the out-of-domain sample ≤ 3, while there is a PA drop compared to PRover when testing on E4. This is a very interesting thing: superficially, proof accuracy drops for complicated unseen queries, but the QA accuracy for out-of-domain queries improves a lot (11 points on E4: $84.8\% \rightarrow 95.6\%$). We save it for future work to explore the portability of the proof and how an out-of-domain proof can help with question answering.

Moreover, we evaluate the zero-shot performance after few-shot learning. In Table 4, we report the results when testing on Birds-Electricity after training the model only on partial DU5 (RC-k and RQ-k, described in 4.3) training partitions. As shown in Table 4, when testing zero-shot performance after few-shot learning, PRobr is well ahead of PRover on QA accuracy. However, as for proof accuracy, PRobr seems worth than PRover on the zero-shot test. Again we point out this amazing observation. This indicates that data from different domains might have different proof form. The well-learned proof from one domain might not be directly adopted to another, but, by training with PRobr, the well-learned proof from one domain can help answer out-of-distribution queries.

Train Data	QA			PA		FA	
	RT	PV	PB	PV	PB	PV	PB
DU0	53.5	**68.7**	56.9	44.4	**50.7**	**42.8**	41.3
DU1	63.5	73.7	**97.7**	63.8	**63.9**	61.9	**63.9**
DU2	83.9	89.6	**99.9**	72.6	**74.5**	72.3	**74.4**
DU3	98.9	98.6	**99.9**	79.1	**83.2**	79.1	**83.2**
DU5	99.2	99.3	**99.9**	87.1	**88.8**	87.1	**88.8**

Table 5: Performance comparison between RuleTakers, PRover, and PRobr when testing on DU5 after training on DU0, DU1, DU2, DU3, respectively.

4.5 Generalization Ability

Generalize to Unseen Depth We conduct experiments to explore how well PRobr can generate proofs and provide answers at depths unseen during training. Following PRover, we train the model on the training splits of DU0, DU1, DU2, and DU3, respectively, and test the QA performance and proof performance on the overall DU5 test set. As DU5 contains queries with higher depth than those seen during training, we can evaluate the model's ability when generalized to higher depth.

As shown in Table 5, PRobr performs better than RuleTakers and PRover on all of QA/PA/FA performance when training on D1, D2 and D3, especially a significant improvement on QA performance. PRobr shows a high and comparable QA performance when training only on depth=1 (97.7%), which demonstrates PRobr's superior generalization ability on depth. This means PRobr can perfectly answer complicated queries using only simple training samples, which reduces the cost of constructing training data.

Generalize to Complex Language We also evaluate the robustness of PRobr when generalized to more diverse natural language. Following (Clark et al., 2020; Saha et al., 2020), we train our model on the combined training partitions of DU3

D	Cnt	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
0	2968	99.8	99.7	**99.8**	99.5	**99.5**	99.4	**99.4**
1	2406	99.3	98.6	**99.7**	98.0	**98.0**	97.3	**98.0**
2	1443	98.2	98.2	**99.9**	88.9	**88.9**	88.7	**88.9**
3	1036	96.7	96.5	**99.8**	90.0	**90.1**	89.9	**90.1**
4	142	90.1	88.0	**100**	76.1	**82.4**	76.1	**82.4**
All	8008	98.8	98.4	**99.8**	95.4	**95.6**	95.1	**95.5**

Table 6: Performance comparison between PRover and PRobr when testing on ParaRules test partitions after training on D3 + ParaRules training partitions.

Train Data		QA		PA		FA	
		PV	PB	PV	PB	PV	PB
100%		53.6	**82.8**	40.0	**43.8**	38.4	**41.6**
RC	10%	64.4	**89.3**	**42.0**	41.3	**41.0**	40.3
	5%	73.6	**84.7**	33.1	**36.5**	29.3	**35.1**
	1%	**59.0**	56.3	**30.4**	25.0	18.1	**18.3**
RQ	30k	59.0	**85.8**	38.6	**43.2**	37.5	**41.7**
	10k	59.7	**87.7**	41.7	**42.3**	40.3	**41.3**
	1k	51.4	**56.3**	**35.0**	25.0	**18.4**	16.5

Table 7: Performance comparison between PRover and PRobr when testing on ParaRules test partitions after training on DU5 or partial DU5 (RC-k and RQ-k) training partitions.

and ParaRules, and then test on the ParaRules test partition. The results in Table 6 show that PRobr is more robust for human-like language.

To better test the generalization ability for complex natural language, we train the model only on DU5 or partial DU5 (RC-k and RQ-k, described in 4.3) training partitions and test on test split of ParaRules. This is a more convincing setup since the model will never see the human-like language but all templated language during training. Results are shown in Table 7. When testing on ParaRules after only training on DU5, PRobr outperforms PRover by nearly **30 points** on QA accuracy(53.6% $\rightarrow$ 82.8%). A similar trend is observed for training on RC-k and RQ-k datasets, where PRobr improves the QA accuracy when generalized to human-like natural language. And the change for proof accuracy is not significant between PRobr and PRover, which supports the observation in Section 4.3 and 4.4, that PRobr improves QA performance by joint question-answering and proof-generation learning, but not necessarily improve the proof performance.

4.6 Ablation Studies

We investigate the effect of training strategy and objective term $\mathcal{L}_{\text{kl}}$ for our model. Specifically we compare PRobr with the following three variants: 1) **PRobr + Gold**, that is, we replace predicted proofs with gold proofs when we optimize $\mathcal{L}_{\text{qa}}$ during training. 2) **PRobr + KL**, that is, we add $\mathcal{L}_{\text{kl}}$ between $q(Y)$ and $p_{\text{pseudo}}(Y)$ during training. 3)**PRobr + Gold + KL** means both. For PRobr and above three variants, we first train on DU5 or partial DU5 (RC-k) training splits respectively, and Figure 3 reports the QA accuracy on test split of DU5 (left), ParaRules test partitions (middle) and Birds-Electricity (right). We observe that PRobr always achieves the best QA accuracy on all of three test datasets (DU5, ParaRules, Birds-Electricity) after training on all of four datasets with varying size (RC-1%, RC-5%, RC-10%, RC-100%). And the other three model variants show inconsistent performance in different settings[7].

5 Related Work

Text Reasoning over Formal Representation Early work employs a pipeline of methods that converts free text into logic form first (semantic parsing), and then uses formal logical reasoning (Musen and Van der Lei, 1988). Due to the serious error propagation caused by semantic parsing (Zettlemoyer and Collins, 2005; Berant et al., 2013; Berant and Liang, 2014), researchers focus on developing theorem provers by combining the symbolic techniques with the differentiable learning from neural networks (Reed and de Freitas, 2016; Abdelaziz et al., 2020; Abboud et al., 2020), such as NLProlog (Weber et al., 2019), SAT solving (Selsam et al., 2019) and Neural programme (Neelakantan et al., 2016). To bypass this expensive and error-prone intermediate logical representation, reasoning over natural language statements in an end-to-end manner is promising.

Text Reasoning over Natural Language Natural logic (MacCartney and Manning, 2009) focuses on semantic containment and monotonicity by incorporating semantic exclusion and implicativity. Subsequently, Clark et al. (2020) proposes to use a Transformer-based model to emulate deductive reasoning and achieves high accuracy on synthetically generated data. PRover (Saha et al., 2020)

[7] For more details, please refer to the supplementary materials.

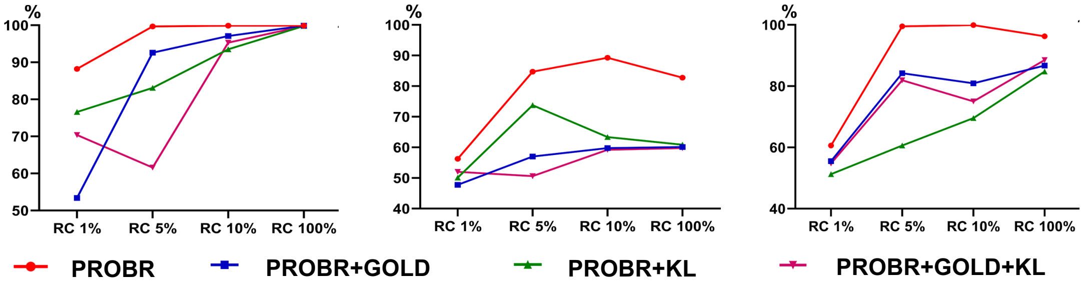


Figure 3: QA accuracy compared among PROBR, PROBR + Gold, PROBR + KL, and PROBR + Gold + KL on DU5 test partition (left), on ParaRules test partitions (middle) and on Birds-Electricity dataset (right), after training on DU5 or partial DU5 (RC-k) training splits.

points out that a reasoning system should not only answer queries but also generate a proof. However, PROVER adopts the multi-task learning framework in the training stage and cannot effectively capture the interactions between question answering and proof generation. Along this line, we explore more powerful joint models to achieve deep reasoning.

QA and NLI There are bAbI (Weston et al., 2016), QuaRTz (Tafjord et al., 2019), ROPES (Lin et al., 2019) and Hotpot QA (Yang et al., 2018) (QA datasets) involved in rule reasoning. However, for those datasets, implicit rules (i.e., which multi-hop chains are valid) need to be inferred from the training data. In our task, the rules of reasoning are given in advance. Compared with the Natural Language Inference (MacCartney and Manning, 2014), our task can be regarded as its deductive subset. In particular, NLI allows for unsupported inferences (Dagan et al., 2013).

6 Conclusion

In this work, we propose PROBR, a novel probabilistic graph reasoning framework for joint question answering and proof generation. PROBR defines a joint distribution over all possible answers and proofs, which can directly characterize the interaction between answers and proofs. Experiments prove the effectiveness of proposed PROBR.

Acknowledgement

The authors wish to thank the reviewers for their helpful comments and suggestions, the authors of RuleTakers for their datasets, and the authors of PROVER for providing the source code.

References

Ralph Abboud, İsmail İlkan Ceylan, and Thomas Lukasiewicz. 2020. Learning to reason: Leveraging neural networks for approximate DNF counting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3097–3104. AAAI Press.

Ibrahim Abdelaziz, Veronika Thost, Maxwell Crouse, and Achille Fokoue. 2020. An experimental study of formula embeddings for automated theorem proving in first-order logic. *arXiv preprint arXiv:2002.00423*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. 1989. What you always wanted to know about datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166.

Keith L. Clark. 1978. *Negation as Failure*, pages 293–322. Springer US, Boston, MA.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

F. R. Kschischang, B. J. Frey, and H. . Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

Bill MacCartney and Christopher D Manning. 2014. Natural logic and natural language inference. In *Computing meaning*, pages 129–147. Springer.

John McCarthy et al. 1960. *Programs with common sense*. RLE and MIT computation center.

Mark A Musen and Johan Van der Lei. 1988. Of brittleness and bottlenecks: Challenges in the creation of pattern-recognition and expert-system models. In *Machine Intelligence and Pattern Recognition*, volume 7, pages 335–352. Elsevier.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Allen Newell and Herbert Simon. 1956. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79.

Manfred Opper and David Saad. 2001. *Advanced mean field methods: Theory and practice*. MIT press.

Scott E. Reed and Nando de Freitas. 2016. Neural programmer-interpreters. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Ruslan Salakhutdinov. 2014. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a SAT solver from single-bit supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. QuaRTz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946, Hong Kong, China. Association for Computational Linguistics.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. Association for Computational Linguistics.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Luke S Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666. AUAI Press.

Supplementary Materials for *Probabilistic Graph Reasoning for Natural Proof Generation*

A Experimental Environment

Parameter	Value
Training Epochs	30
Optimizer	AdamW
Gradient Clipping	1.0
Batch Size	8
Dropout Rate	0.1
Learning rate	1e-5
Max Sequence Length	300

Table 8: Model configurations.

Table 8 lists the default model configurations. We produce PROBR on 8 NVIDIA Tesla-V100 GPUs. We implement PROBR with PyTorch, using RoBERTa (Liu et al., 2019) as pre-trained language model.

B Results on Development Set

Table 9 shows the results on development set under the fully supervised setting, and the corresponding test set results are shown in Table 1. Overall, it is consistent with the findings of the test set results. PROBR achieves the best performance in three accuracy metrics (**QA**, **PA** and **FA**).

D	Cnt	QA			PA		FA	
		RT	PV	PB	PV	PB	PV	PB
0	6299	100	100	**100**	98.5	**98.7**	98.5	**98.7**
1	4434	98.4	98.8	**100**	92.2	**93.6**	92.2	**93.6**
2	2915	98.4	99.2	**99.0**	85.6	**87.3**	85.6	**87.3**
3	2396	98.8	98.7	**100**	82.8	**85.4**	82.8	**85.4**
4	2134	99.2	98.8	**99.7**	76.9	**80.8**	76.9	**80.8**
5	2003	99.8	99.3	**99.9**	67.4	**74.6**	67.4	**74.6**
All	20192	99.2	99.3	**99.9**	88.0	**90.0**	88.0	**90.0**

Table 9: Fully supervised learning performance compared among RuleTakers (RT), PROVER (PV) and PROBR (PB) on development split of DU5 after training on training split of DU5, reported in varying depth.

Table 10 lists the results on development set of DU5 after training on DU0, DU1, DU2, DU3, respective. The corresponding test set resluts are shown in Table 5. Comparing with Table 5, each number is very close (fluctuates within 1%) and similar conclusions can be drawn.

Train Data	QA		PA		FA	
	PV	PB	PV	PB	PV	PB
DU0	**68.3**	57.0	43.8	**50.7**	**42.3**	41.7
DU1	73.2	**98.5**	63.9	**64.3**	61.8	**64.3**
DU2	89.3	**99.9**	72.6	**74.3**	72.3	**74.3**
DU3	98.3	**99.9**	79.4	**83.2**	79.4	**83.2**
DU5	99.3	**99.9**	88.0	**90.0**	88.0	**90.0**

Table 10: Performance comparison between PROVER and PROBR development split of DU5 after training on DU0, DU1, DU2, DU3, respectively.

C Results of Ablation Studies

Table 11 lists the comparison results of PROBR and the three variants of PROBR (mentioned in Section 4.6). Regarding the Table 11, we have three observations:

1. In all ablation experiments, PROBR achieved the best **QA** performance, demonstrating that PROBR can capture critical information for question answering in a variety of settings. However, since some of the dataset are artificially synthesized, it is difficult to guarantee that PROBR will work in the real dataset as well. We leave it as future work.

2. In some cases, variant d) (**PROBR + Gold + KL**) outperforms PROBR in **PA** and **FA**. It shows the potential advantages of the KL term. In the future, we will explore proof generation in a semi-supervised learning scenario through this KL term.

3. When we compare the performance of the two models PROBR and **PROBR + Gold**, we can see that whether the predicted proof or the correct proof is used during training significantly affects the final performance. Applying some heuristic strategies may give better results, such as scheduled sampling (Bengio et al., 2015). We will try it in the future.

	Test on DU5			Test on ParaRules			Test on Birds-Electricity		
	QA	PA	FA	QA	PA	FA	QA	PA	FA
RC-1%	a)**88.2** b)53.4 c)76.6 d)70.4	a)**21.6** b)**21.6** c)**21.6** d)**21.6**	a)20.3 b)**21.3** c)14.8 d)20.0	a)**56.3** b)47.8 c)50.1 d)52.0	a)**25.0** b)**25.0** c)**25.0** d)**25.0**	a)**18.3** b)14.1 c)12.9 d)14.5	a)**60.6** b)55.5 c)51.3 d)54.8	a)**34.6** b)**34.6** c)**34.6** d)**34.6**	a)**24.4** b)17.4 c)17.6 d)19.9
RC-5%	a)**99.7** b)92.6 c)83.1 d)61.6	a)44.2 b)43.7 c)31.4 d)**47.9**	a)44.2 b)43.6 c)30.1 d)**45.9**	a)**84.7** b)57.0 c)73.8 d)50.6	a)36.5 b)37.1 c)36.6 d)**38.9**	a)35.1 b)35.4 c)27.6 d)**37.2**	a)**99.5** b)84.2 c)60.6 d)81.9	a)69.1 b)62.0 c)45.3 d)**73.4**	a)**69.0** b)62.0 c)42.0 d)65.4
RC-10%	a)**99.9** b)97.1 c)93.5 d)95.3	a)60.4 b)57.9 c)52.2 d)**69.3**	a)60.4 b)57.9 c)51.8 d)**69.0**	a)**89.3** b)59.8 c)63.4 d)59.2	a)**41.3** b)40.0 c)40.7 d)37.5	a)**40.3** b)38.6 c)39.2 d)35.9	a)**99.9** b)80.9 c)69.6 d)75.0	a)55.4 b)59.6 c)53.3 d)**66.2**	a)55.4 b)59.6 c)53.2 d)**66.1**
RC-100%	a)**99.9** b)**99.9** c)99.8 d)99.8	a)88.8 b)88.7 c)89.1 d)**89.6**	a)88.8 b)88.7 c)89.1 d)**89.6**	a)**82.8** b)60.1 c)60.9 d)59.8	a)**43.8** b)42.7 c)42.9 d)43.2	a)**41.6** b)39.8 c)41.2 d)41.5	a)**96.3** b)86.7 c)84.8 d)88.5	a)79.3 b)**81.4** c)78.6 d)80.9	a)79.3 b)**81.3** c)78.6 d)80.9

Table 11: QA accuracy, proof accuracy and full accuracy compared among PROBR, PROBR + Gold, PROBR + KL, and PROBR + Gold + KL on DU5 test partition (left), on ParaRules test partitions (middle) and on Birds-Electricity dataset (right), after training on DU5 or partial DU5 (RC-k) training splits, where a)–**PROBR**, b)–**PROBR + Gold**, c)–**PROBR + KL**, d)–**PROBR + Gold + KL**.