Interpretable Proof Generation via Iterative Backward Reasoning

Hanhao Qu[1] **Yu Cao**[3] **Jun Gao**[1] **Liang Ding**[3,4] **Ruifeng Xu**[1,2*]
[1]Harbin Institute of Technology, Shenzhen [2]Peng Cheng Laboratory
{hhqu0917,jgao95}@stu.hit.edu.cn xuruifeng@hit.edu.cn
[3]The University of Sydney, Australia [4]JD Explore Academy
{ycao8647,ldin3097}@uni.sydney.edu.au

Abstract

We present IBR, an Iterative Backward Reasoning model to solve the proof generation tasks on rule-based Question Answering (QA), where models are required to reason over a series of textual rules and facts to find out the related proof path and derive the final answer. We handle the limitations of existed works in two folds: 1) enhance the interpretability of reasoning procedures with detailed tracking, by predicting nodes and edges in the proof path iteratively backward from the question; 2) promote the efficiency and accuracy via reasoning on the elaborate representations of nodes and history paths, without any intermediate texts that may introduce external noise during proof generation. There are three main modules in IBR, *QA and proof strategy prediction* to obtain the answer and offer guidance for the following procedure; *parent node prediction* to determine a node in the existing proof that a new child node will link to; *child node prediction* to find out which new node will be added to the proof. Experiments on both synthetic and paraphrased datasets demonstrate that IBR has better in-domain performance as well as cross-domain transferability than several strong baselines. Our code and models are available at https://github.com/find-knowledge/IBR.

1 Introduction

Endowing machines with reasoning capabilities is a longstanding problem (Newell and Simon, 1956) in the field of AI. Though existing tasks such as multi-hop QA (Yang et al., 2018; Welbl et al., 2018) or logical-reasoning QA (Yu et al., 2020; Dua et al., 2019) impose a higher requirement on the reasoning capabilities, they usually just request for an answer without the reasoning procedure that would make it interpretable. Recently, Clark et al. (2020) proposed new datasets and tasks for interpretable reasoning. Given a question, coupling with a set of facts (plain statements) and rules (implication relationships) that are expressed in natural language, there are two tasks: 1) predicting the binary answer; 2) generating the proof path behind this answer. Large-scale pretrained models have shown strong performance on the first subtask in the early work (Liu et al., 2019), but there still remain challenges for the second one. These proof paths are usually more complicated than those involved in multi-hop QA tasks, as there are more nodes and branches rather than a single-directed chain.

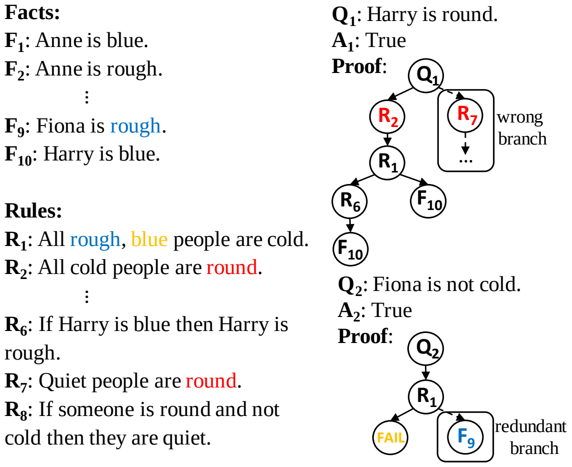


Figure 1: Illustration of generating proof iteratively. Regarding the proof path as a graph, and using the question as the initial node, other nodes and edges will be added step by step. (The gold proof is the obtained path in a reverse order exclude the question). The main challenges are wrong (cannot derive the answer) or redundant (can derive the answer, but the path is longer than the optimal one) branches may be involved.

Several approaches have been proposed to simultaneously address the two subtasks. PROVER (Saha et al., 2020) and PROBR (Sun et al., 2021) try to construct the reasoning path at once, where two classifiers are used to determine whether each node or edge is involved in the proof path respectively based on corresponding encoded

[*] Corresponding author

representations. But they lack interpretability on tracking the detailed reason for selecting each step. To make proof generation more interpretable, Proofwriter (Tafjord et al., 2021) and EVR (Liang et al., 2021) decompose complex reasoning over the question into multiple simple procedures, resulting in iterative and interpretable processes with the help of intermediate texts. Nevertheless, both of them suffer from efficiency and external errors issues. The reason is that they both require a large searching space, as they perform on the whole inferable texts and ignore the structure information from the history path that has been obtained. Moreover, the generation of intermediate text is costly and may introduce extra noise propagation.

Inspired by the top-down AMR parsing (Cai and Lam, 2019), where a sentence is divided into sub-meanings iteratively, we present **I**terative **B**ackward **R**easoning (IBR) for better proof generation. It generates a proof path iteratively starting from the core component for QA, i.e. the question, making the process interpretable with trackable intermediate states. Regarding a higher efficiency and accuracy, and two challenges mentioned in Figure 1, the proof generation module of IBR simplifies the intermediate process of reasoning as well as avoids the unnecessary search for a possible unsuitable branch. To add a new node and edge to the path, there are two steps in IBR for each iteration: 1) finding out the next parent node, i.e. one existing rule or fact in the parsed history path that a new node will become its child; 2) determine which rule or fact that will be the new child node and added to the path. Equipped with question-aware representations from a pre-trained encoder, along with structure-aware node and path features, our model can choose the optimal endpoint. It accomplishes reasoning with the highest possibility to obtain a correct subsequent proof path based on relevant features, getting rid of intermediate texts while avoiding redundancy on all possible texts than previous iterative works.

In addition, to make IBR applicable for samples with incomplete proof paths, which are abandoned in the former backward iterative model EVR (Liang et al., 2021), we employ a proof strategy predictor to output a proof type. This prediction is then integrated into the later proof generation actions, making the process more controllable under different conditions.

We validate our approach on several datasets that are widely used in previous studies (i.e. DU0-DU5, Birds-Electricity, and ParaRules) spanning different settings (i.e. fully-supervised, fewer training data, and out-of-domain). Experimental results show that, compared to existing strong baselines including both non-iterative and iterative ones, IBR can achieve the best overall performance of proof generation and comparable answer prediction accuracy, along with noticeable generalization capability. Extensive analyses show that 1) the improvements come from our elaborately designed iterative and simplified proof generation modules, and 2) both the reasoning ability and latency could be significantly improved compared to former iterative models, making a better trade-off considering its reasonable interpretability.

2 Related Work

Question answering and reasoning. Endowing machines to do reasoning over explicit knowledge is a primitive task (Newell and Simon, 1956). Early works tried to solve it by converting texts into logic forms (Newell and Simon, 1956; Musen and Lei, 1988). But such kinds of approaches can be affected by the error propagation caused by semantic parsing (Zettlemoyer and Collins, 2012; Berant et al., 2013; Berant and Liang, 2014).

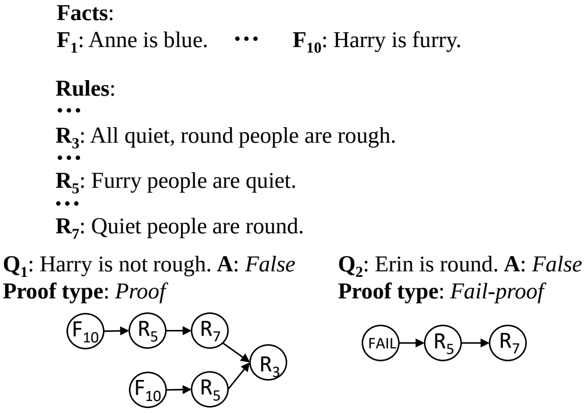
Lately, question answering (QA) is employed as an important task for machine reasoning. Numerous datasets were proposed, including synthesized data (Weston et al., 2016), comprehension on natural texts (Rajpurkar et al., 2016; Joshi et al., 2017; Fisch et al., 2019) or more complex relationship reasoning (Tafjord et al., 2019; Lin et al., 2019). There are also multi-hop QA tasks like HotpotQA (Yang et al., 2018) or QAngaroo (Welbl et al., 2018), and logical QA datasets such as ReClor (Yu et al., 2020) and LogiQA (Liu et al., 2020), in which textual rules need to be inferred implicitly from a long supporting context. Plenty of studies try to solve these problems via neural networks and achieve remarkable performance (Joshi et al., 2020; Yu et al., 2018; Shao et al., 2020). Nevertheless, nearly all of them only focus on the prediction of final answers and neglect the acquisition of interpretable proofs. Although some datasets provide proof paths for better interpretability, these paths are only short chains with very few entities and cannot teach models to generate complex proofs.

Proof generation. NLProlog (Weber et al., 2019) first employs logic programming to search for a proof and then predicts the answer in multi-hop QA. Recently, Clark et al. (2020) propose new rule-based QA datasets for this line of research that include more complex proof paths, and present RuleTaker to answer questions. Saha et al. (2020) argue that producing answer proofs makes models more reliable and propose PROVER, a transformer-based model that enumerates all possible nodes and edges of a proof path and predicts whether each one exists at once based on their embeddings. PROBR (Sun et al., 2021) further improves this framework using the probabilistic graph to model more variables. There has been also an increasing interest in solving proof generation iteratively. EVR (Liang et al., 2021) splits the question into sub-questions, using generated intermediate texts to guide proof generation step by step. ProofWriter (Tafjord et al., 2021) shares a similar idea but uses intermediate textual conclusions instead and a more powerful T5-11B model (Raffel et al., 2020) for generation, which makes it hard to reproduce. IBR is also an iterative model, being more interpretable than at-once models. Despite getting rid of intermediate texts and directly using various representations to finish each step, it improves efficiency and effectiveness.

3 Methodology

3.1 Task Definition

We first formulate the proof generation task as follows. Given a tuple (C, Q, A, P), where $C = \{RF_i\}$ is the contexts containing several textual rules and facts RF, Q is the question, $A \in \{$*True*, *False*$\}$ is the answer, and P indicates the proof path for the detailed reasoning procedure to derive A, our goal is twofold: 1) predicting the answer A, and 2) generating the proof path P. Taking **DU0-DU5** (Clark et al., 2020) dataset as example, P is a single-directed acyclic graph having the shortest path to derive A. P can start from one or multiple nodes but must end in one node that directly entails or contradicts Q. A node in P can be a fact, a rule, or a special NAF (Negation As Failure) node[1]. Edges between nodes indicate that the start nodes can be used to prove the end nodes during reasoning. Proofs in the dataset can be roughly classified into two types according to their **strategies** S to prove the question: (1)*Proof*: the question can be directly proven to be True or False using the given C and NAF; (2) *Fail-Proof*: the question cannot be explicitly deduced barely using C and NAF as some key information is missed, hence a positive statement is judged as **False** while a negative statement as **True** in such cases (Figure 2).


Facts:
$\mathbf{F_1}$: Anne is blue. ··· $\mathbf{F_{10}}$: Harry is furry.

Rules:
···
$\mathbf{R_3}$: All quiet, round people are rough.
···
$\mathbf{R_5}$: Furry people are quiet.
···
$\mathbf{R_7}$: Quiet people are round.

$\mathbf{Q_1}$: Harry is not rough. **A**: *False*
Proof type: *Proof*

$\mathbf{Q_2}$: Erin is round. **A**: *False*
Proof type: *Fail-proof*

Figure 2: Examples of *Proof* and *Fail-proof* strategies.

3.2 Overview

The proposed Iterative Backward Reasoning (IBR) model takes Q as the initial node and produce a proof path P backward, from the end node to the start node. Two actions are included at each iteration: (1) **Predicting the new parent node**, i.e. a node in the derived proof path where a child node will be added (except the first step that only Q exists); (2) **Predicting the child node**, i.e. the fact or rule in C that will be the child for the selected parent node. After each iteration, a new node and an associated edge are added. After obtaining the whole reasoning path, we remove Q and reverse all edges to get the final proof P.

The Figure 3 illustrates our IBR model, which can be divided into three modules, (1) **QA and Strategy Prediction**, (2) **Parent Node Prediction**, and (3) **Child Node Prediction**. In order to make the question Q can fully interact with context C (facts and rules) and obtain better representations, IBR uses pretrained RoBERTa (Liu et al., 2019) as the backbone network. The input of RoBERTa is the concatenation of the question Q and the context $C = \{RF_i\}$, separated by special $[SEP]$ token, denoted as $[CLS]$ Q $[SEP]$ $[SEP]$ C $[SEP]$.

IBR only uses the QA prediction and strategy prediction modules once at first to predict the answer A and the strategy of the proof (refer to §3.1, where the latter one will result in different proof

[1] A start node when the negation condition in the next node has no corresponding fact nor rule node, and the negation will be considered as true. E.g., there is no item in C related to "*Anne is big*", its negation "*Anne is not big*" will be considered as true.

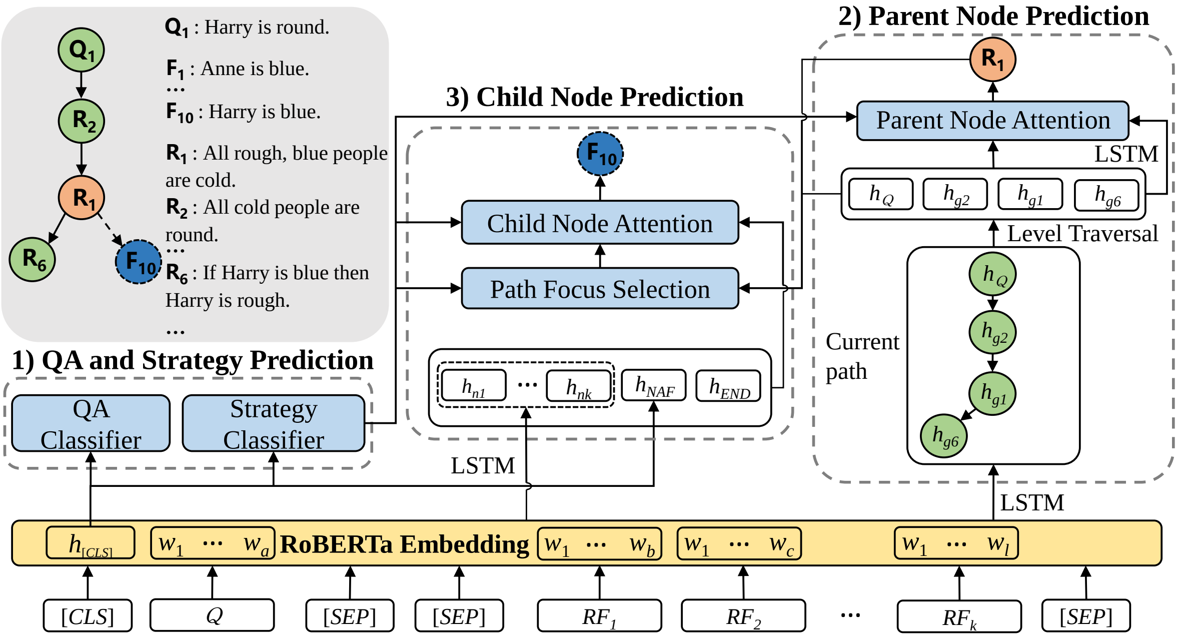


Figure 3: The model architecture of IBR. 1) is only used once at the start, then 2) and 3) are applied iteratively to generate the whole proof. It also illustrates the detailed state when adding F_{10} into the proof (F: facts, R: rules).

generation procedures. In order to improve the reasoning efficiency as well as accuracy, instead of using generated intermediate texts (Liang et al., 2021; Tafjord et al., 2021), all possible nodes (rules and facts) are represented by node embeddings in IBR. The initial state of the proof is only the representation of the question h_Q, then the rest of the reasoning path will be constructed based on it.

Samples with *Fail-Proof* strategy differs from ones with *Proof*, because their proofs are usually short without sub-branches, and only consist of rules due to lacking essential supporting facts. To take the advantage of such a property distinction and extend the applicability compared to former models (Liang et al., 2021) that cannot generate proofs for *Fail-Proof* samples, we apply different actions in modules (2) and (3) depending on the output from strategy prediction.

3.3 QA and Strategy Prediction Module

This module aims to predict the answer A of the question Q and the corresponding strategy S of proof P. Since the representation of $[CLS]$ token from pretrained models is proven to have the capability of modeling the whole input, we use it as the input feature for both predictions as they condition the global information. The encoded $[CLS]$ by RoBERTa, $h_{[CLS]}$ is passed to a linear layer and the softmax function σ for answer and strategy classification respectively,

$$P_{QA} = \sigma(f_{QA}(h_{[CLS]})),$$

$$P_{Strategy} = \sigma(f_{Strategy}(h_{[CLS]})).$$

Here, f_{QA} and $f_{Strategy}$ indicate the linear layer for QA classification and strategy classification, respectively. P_{QA} and $P_{Strategy}$ are binary-class probability values, the former one is for values of $A \in \{True, False\}$ while the later one is for values of $S \in \{\textit{Proof}, \textit{Fail-proof}\}$.

3.4 Parent Node Prediction Module

This module determines which node in the current reasoning path is going to be the next parent node that a new child node will link to. To better represent the sequential information of each possible node (fact or rule), an LSTM (Hochreiter and Schmidhuber, 1997) is used to further encode the token-level embedding from RoBERTa. The hidden state in the last step is used as the textual representation h_{gi} of a possible parent node RF_i.

In addition, selecting a node from the existing proof path also needs global and structural modeling on the history path. To make this procedure a more convenient representation that involves the order of reasoning, the path is regarded as a tree structure and nodes are reordered by level traversal from top to down. Since Q is always the root node of the tree, e.g., if Q have two children RF_1 and RF_3, and RF_1 has a child RF_2, the reordered representation sequence is $[h_Q, h_{g1}, h_{g3}, h_{g2}]$. We then utilize another LSTM model to encode the reordered representation sequence of the current reasoning path obtained before, extracting the overall state of the path, which is the hidden state h_g at the last time step in this LSTM.

A parent node attention based on the Transformer attention (Vaswani et al., 2017) is used to obtain the weights of all possible parents nodes. It takes h_g and the representation sequence of the current path $\mathbf{H}_p = [h_Q, h_{g1} \dots h_{gt}]$ as input, i.e.

$$\mathrm{Att}(h_g, \mathbf{H}_p) = \sigma(f_Q(h_g)(f_K(\mathbf{H}_p))^T / \sqrt{d}), \quad (1)$$

where f_Q and f_K indicate linear layers, σ is a softmax function, and d is the dimension of h_g. As we discussed in §3.2, different operations are employed for corresponding strategy types of proofs. 1) If the predicted proof strategy is *Proof*, we select the node with the highest weight as the parent node RF_p. 2) If the predicted proof strategy is *Fail-proof*, we use the last node in the current path, i.e. h_{gt} in $\mathbf{H}_P$, as the parent node RF_p, because no sub-branch is included in such proof paths.

3.5 Child Node Prediction Module

This module decides which node will be added to the proof path and linked to the parent node RF_p we have obtained before. To derive the representations of candidate child nodes, similar to §3.4, we apply another LSTM model to the encoded RoBERTa embeddings and get h_{n_i} for RF_i. Since we discussed a special NAF node in §3.1 which may contain information from the whole context, we utilize a linear layer f_{NAF} to transform the $[CLS]$ token embedding $h_{[CLS]}$ into its representation h_{NAF}. Moreover, we initialize a representation h_{END} for the special END node, indicating that the proof generation process will finish here.

During selecting the new child node, we need to consider not only the knowledge of the history path, but also the state of the parent node. To better model such relationships, we propose a **Path Focus Selection** module to generate relevant features before predicting the child node. A 2-layer Transformer model along with a LSTM model is introduced. It first encodes the representations of node sequence $\mathbf{H}_p$ from Parent Node Prediction respectively, then fuses their hidden state via a linear layer f_U,

$$h_F = f_U([\mathrm{Trans}(h_{gp}, \mathbf{H}_p, \mathbf{H}_p); \mathrm{LSTM}(\mathbf{H}_p)]). \quad (2)$$

Here, h_{gp} is the representation of the selected parent node in §3.4, f_U is the linear layer for feature fusing, while $[\cdot;\cdot]$ stands for concatenation. q, k, v in $\mathrm{Trans}(q, k, v)$ indicate the inputs corresponding to Query, Key, and Value in a transformer model, and only the hidden state in the last time step is remained in both Trans and LSTM. It is worth noting that the LSTM used here is a supplementary knowledge source for a better representation according to our empirical study. Such an operation results in a feature h_F that is aware of both the history proof path and the parent node that a child will link to.

This feature h_F will then be used in the Child Node Attention to calculate the attention weights on all possible child nodes. Particularly, an attention model same as Eq. 1 is applied on h_F and a series of child node representations obtained before $\mathbf{H}_c = [h_{n1} \dots h_{nk}, h_{NAF}, h_{END}]$, and the attention weights are defined as $\mathrm{Att}(h_F, \mathbf{H}_c)$. It contains all facts and rules in the context, and the special NAF node as well as END node.

Similar to §3.4, we also apply different actions according to our predicted proof strategies before.
(1) If the strategy is *Proof*, we select the child node with the highest attention weight from all candidates as the new node in the proof path.
(2) If the strategy is *Fail-proof*, since RF_p is the last node during reasoning and this procedure is a first-order logical under such a situation, there is no need to make complex modeling on the derived path. Therefore, we directly use its parent node representation h_{gp} rather than encoded state from Transformer in Eq. 2 to get h_F. But LSTM is remained to maintain some basic modeling capability on the path. In child node attention, we mask all fact nodes and select the one with the highest weight among the remaining nodes, because this kind of proof usually only contains rules and such masking can avoid extra errors.

3.6 Training and Inference

The whole model is trained via binary cross-entropy losses from all three above modules jointly,

$$L = L_{QA} + L_{Parent} + L_{Child} + \alpha * L_{Strategy}.$$

L_{QA} and $L_{Strategy}$ correspond to the loss of QA prediction and strategy prediction, respectively. α is a hyperparameter to reweigh the influence of [*CLS*] token. L_{Parent} is the loss for parent node prediction, where the cross-entropy is calculated between the attention weight vector and a one-hot vector indicating the gold parent node. L_{Child} is in a similar way on child node prediction. Note that samples labeled as *Fail-proof* strategy are not involved in the training of parent node prediction. As all their proof paths are chains and the new parent node is always the last node added to the path, so learning about these data may introduce model bias. To determine the gold reasoning order used as the target for training, we set a higher priority of fact nodes than rule nodes, as the clearer subject information is involved in facts. E.g., for a parent node with multiple children, the gold reasoning order of child node prediction is NAF nodes first, then fact nodes, and finally rule nodes. If there are more than one fact or rule nodes, IBR randomly swaps their order within each type at different training epochs.

During inference, IBR first makes predictions on the answer A and strategy S, then generate the parent node and child node iteratively, until the special END node is predicted as the new child node. IBR uses beam search to keep the top-K best proof paths at each proof generation step and select the best one as the final prediction, where the beam size is set as 8.

4 Experiments

Following former studies (Saha et al., 2020; Sun et al., 2021), we evaluate our IBR[2] on three datasets and four settings including fully-supervised training, training using fewer samples, testing on out-of-domain samples, and generalization to more complex proofs or language.

4.1 Setup

Datasets. Experiments are conducted on three datasets raised by Clark et al. (2020)[3], where we use the same test split as previous works for fair comparison:

- **DU0-DU5**: Five synthesized datasets created by translating hand-crafted rules and formal language to natural language. It is divided by the highest depth of proof, where DU stands for "Depth Upto" (DU=0,1,2,3,5). Data in higher DU values also contain samples with lower depth. Note that proofs in DU0 only have one supporting or opposing fact. All related results are reported on DU5 test split.
- **Bird-Electricity**: It is a test-only dataset that contains samples about birds and electric circuits. It is generated in the same way as DU0-DU5, but is in different domains from DU0-DU5.
- **ParaRules**: This dataset consists of 40k questions expressed in paraphrased natural language based on synthetic data, which is created by crowdsourcing. Multiple facts get together in one statement here rather than separated in DU0-DU5.

Baselines. We consider the following baselines[4].

- **RuleTaker (RT)** (Clark et al., 2020): a RoBERTa based model that only predicts answers.
- **PROVER (PV)** (Saha et al., 2020): a method that treats the proof as a graph and predicts all its nodes and edges at once, also using RoBERTa model as the backbone, same as IBR.
- **PROBR (PB)** (Sun et al., 2021): it improves PROVER by introducing the probabilistic graph that jointly considers the answer, nodes and edges.
- **EVR** (Liang et al., 2021): an iterative model that predicts the next proof item by generating textual sub-questions based on logical operator. Note that this model is **not applicable** for samples whose **proof strategy is *Fail-proof*** discussed in §3.1, so we make comparison with it separately.

Metrics. We closely follow previous works to evaluate the performance of models via answer prediction (QA) accuracy and proof generation (PA) accuracy. Since some samples may have multiple gold proofs, a generated proof will be considered correct, as long as its nodes and edges match with the nodes and the edges in any of the gold proofs. Full Accuracy (FA) is also included, where a sample is regarded as correct only both the predicted answer and proof are correct.

4.2 Results under Fully-Supervised Training

We train IBR on the training split of the DU5 dataset and evaluate on the test split of DU5. We compare the performance of IBR with baselines except for EVR in Table 1, while with EVR in Table 2 where only partial test split is included, excluding samples whose proof strategy is *Fail-proof*. Because EVR always fails on these samples (EVR on these excluded samples is given in Appendix A.5).

Obviously, IBR achieves the best proof generation accuracy (PA) as well as full accuracy (FA) among all baseline models, on samples with every depth. Our model also shows a greater advantage on samples with deeper proof path, e.g., 81.7 vs.

[2] Refer to Appendix A.1 for implementation details.

[3] More details are given in Appendix A.2

[4] Results of baselines are obtained from the original papers or by running the released code.

	D	0	1	2	3	4	5	all
	Cnt	6299	4434	2915	2396	2134	2003	20192
QA	RT	100	98.4	98.4	98.8	99.2	99.8	99.2
	PV	100	99.0	98.8	99.1	98.8	99.3	99.3
	PB	100	**99.9**	**99.9**	**100**	**100**	**100**	**99.9**
	IBR	**100**	99.2	99.2	98.9	99.3	99.6	99.4
PA	PV	98.4	93.2	84.8	80.5	72.5	65.1	87.1
	PB	98.4	94.3	86.1	82.0	76.1	72.2	88.8
	IBR	**99.5**	**95.6**	**93.0**	**90.7**	**86.5**	**81.7**	**93.5**
FA	PV	98.4	93.1	84.8	80.5	72.4	65.1	87.1
	PB	98.4	94.3	86.1	82.0	76.1	72.2	88.8
	IBR	**99.5**	**95.6**	**92.9**	**90.7**	**86.5**	**81.6**	**93.5**

Table 1: Results of different models on varying proof depth (**D**) under the fully-supervised setting. Cnt: sample count, RT: RuleTaker, PV: PROVER, PB: PROBR.

	D	0	1	2	3	4	5	all
	Cnt	1934	1934	1934	1934	1934	1934	11604
QA	EVR	99.4	99.3	96.9	93.3	88.9	88.3	94.4
	IBR	**100**	**99.3**	**99.6**	**99.3**	**99.6**	**99.5**	**99.5**
PA	EVR	95.8	92.5	87.7	79.3	77.3	68.8	83.6
	IBR	**98.8**	**96.4**	**94.7**	**92.2**	**88.7**	**83.6**	**92.4**
FA	EVR	95.8	92.5	87.7	79.3	77.3	68.8	83.6
	IBR	**98.8**	**96.3**	**94.6**	**92.2**	**88.7**	**83.5**	**92.3**

Table 2: Results of IBR and EVR on a partial test split of DU5 (exclude *Fail-proof* samples). The models are trained on the train split of DU5.

Data	**QA**			**PA**			**FA**		
	PV	PB	**IBR**	PV	PB	**IBR**	PV	PB	**IBR**
70k	99.3	**99.9**	99.4	87.1	88.8	**93.5**	87.1	88.8	**93.5**
30k	97.8	**99.9**	98.3	72.5	86.8	**89.8**	72.4	86.8	**89.7**
10k	87.1	**99.9**	94.3	44.0	72.4	**75.7**	42.7	72.3	**75.4**

Table 3: Performance comparison using fewer training samples among IBR, PROVER (PV), and PROBR (PB) on the full test split of DU5 after trained on partial DU5 samples.

Data	**QA**		**PA**		**FA**	
	EVR	**IBR**	EVR	**IBR**	EVR	**IBR**
70k	94.4	**99.5**	83.6	**92.4**	83.6	**92.3**
30k	95.7	**99.4**	84.4	**88.2**	84.4	**88.1**
10k	96.2	**97.9**	**82.8**	71.2	**82.8**	70.8

Table 4: Performance comparison using fewer training samples among EVR and IBR on partial test split of DU5 (without *Fail-proof* samples) after trained on partial DU5 samples.

72.2 on PA when depth is 5, illustrating the superiority of iterative models on complex proof paths. Besides, despite not being the best in answer accuracy (QA), there is a very narrow gap between our model and the best one, which proves that IBR is still a comprehensive model covering both subtasks. When compared to EVR, also an iterative model, IBR shows significantly stronger performance on all metrics, benefiting from our elaborate two-fold reasoning process at each step.

4.3 Using Fewer Training Samples

We also explore the performance of IBR when training using fewer data, ranging from 10k to 30k to all the examples (70k) in DU5. The comparison between our model, PROVER (PV), and PROBR (PB) is shown in Table 3, in all three metrics. Our model significantly has the best proof generation performance than the other two baselines in all cases, due to the iterative architecture requiring less global modeling capability and thus fewer training samples. Although PB shows a promising answer prediction accuracy under fewer-data settings, the performance of IBR is close to it while better than PV, e.g., 94.3 vs. 87.1 under 10k. In addition, in Table 4, we also compare with EVR under the same settings but using a different test set that excludes *Fail-proof* samples. EVR outperforms IBR under the 10k setting for proof generation, but IBR is stronger if more training samples are available.

4.4 Evaluation of Out-of-Domain Data

We further test the out-of-domain performance of IBR against baselines on Birds-Electricity dataset to evaluate their robustness, where B1 and B2 are two sets from the birds domain, and E1-E4 are four sets from the electricity domain. Results are shown in Table 5 and Table 6. Note that *Fail-proof* samples are still not involved in the comparison for EVR. Overall, our IBR achieves 2.5% promotion in PA while an equivalent result on QA, compared to PROVER. Despite being the best one on QA, PROBR is also defeated by IBR on both PA and FA. In addition, our model shows more improvement on the hardest E3 and E4 subsets, which further verifies its robustness. When it comes to EVR, we can find its cross-domain capability is relatively weak as it sees a significant drop in PA, and IBR is superior to it without any doubt. Because the cross-domain generation for intermediate texts is much harder, our usage of high-level node features to finished reasoning can alleviate this challenge.

4.5 Generalization Ability

Generalize to higher depths. Following the previous work (Sun et al., 2021), we test the general-

	Test	B1	B2	E1	E2	E3	E4	all
	Cnt	40	40	162	180	624	4224	5270
QA	RT	97.5	100.0	96.9	98.3	91.8	76.7	80.1
	PV	95.0	95.0	100.0	100.0	89.7	84.8	86.5
	PB	100.0	**100.0**	100.0	100.0	**98.2**	**95.6**	**96.3**
	IBR	**100.0**	97.5	**100.0**	**100.0**	89.2	84.1	86.0
PA	PV	92.5	95.0	95.1	91.7	72.3	80.6	80.7
	PB	100.0	100.0	**97.5**	93.3	79.3	77.7	79.3
	IBR	**100.0**	**100.0**	95.6	**94.4**	**80.2**	**82.4**	**83.2**
FA	PV	92.5	95.0	95.1	91.7	71.8	80.6	80.5
	PB	100.0	**100.0**	**97.5**	93.3	**79.3**	77.7	79.3
	IBR	**100.0**	97.5	95.6	**94.4**	78.2	**82.4**	**82.9**

Table 5: Out-of-domain performance comparison among RuleTakers (RT), PROVER (PV), and PROBR (PB) on Birds-Electricity dataset after training on DU5.

	Test	B1	B2	E1	E2	E3	E4	all
	Cnt	28	28	72	90	312	1206	1736
QA	EVR	67.8	64.2	83.3	80.0	76.2	83.8	81.6
	IBR	**100.0**	**96.4**	**100.0**	**100.0**	**92.9**	**100.0**	**98.6**
PA	EVR	32.1	35.7	58.3	50.0	45.5	70.3	63.1
	IBR	**100.0**	**100.0**	**91.6**	**91.1**	**91.3**	**95.2**	**94.3**
FA	EVR	32.1	32.1	58.3	50.0	45.5	70.3	63.1
	IBR	**100.0**	**96.4**	**91.6**	**91.1**	**87.1**	**95.2**	**93.5**

Table 6: Out-of-domain performance comparison among EVR and IBR on partial Birds-Electricity dataset (exclude *Fail-proof* samples) after training on DU5.

Data	QA			PA			FA		
	PV	PB	**IBR**	PV	PB	**IBR**	PV	PB	**IBR**
DU0	**68.7**	56.9	53.5	44.4	**50.7**	47.0	42.8	41.3	**47.0**
DU1	73.7	**97.7**	73.1	63.8	63.9	**64.6**	61.9	63.9	**64.5**
DU2	89.6	**99.9**	89.6	72.6	74.5	**76.3**	72.3	74.4	**76.2**
DU3	98.6	**99.9**	98.6	79.1	83.2	**87.4**	79.1	83.2	**87.4**
DU5	99.3	**99.9**	99.4	87.1	88.8	**93.5**	87.1	88.8	**93.4**

Table 7: Performance of generalization ability between PROVER (PV), PROBR (PB), and IBR when testing on the test split of DU5, after trained on DU0, DU1, DU2, DU3, and DU5, respectively.

	D	0	1	2	3	4	all
	Cnt	2968	2406	1443	1036	142	8008
QA	PV	99.7	98.6	98.2	96.5	88.0	98.4
	PB	99.8	**99.7**	**99.9**	**99.8**	**100**	**99.8**
	IBR	**99.9**	98.8	97.5	96.3	88.7	98.4
PA	PV	99.5	98.0	88.9	90.0	76.1	95.4
	PB	99.5	98.0	88.9	**90.1**	**82.4**	95.6
	IBR	**99.8**	**98.8**	**91.1**	89.0	75.3	**95.9**
FA	PV	99.4	97.3	88.7	89.9	76.1	95.1
	PB	99.4	98.0	88.9	**90.1**	**82.4**	95.5
	IBR	**99.7**	**98.1**	**90.9**	89.0	75.3	**95.7**

Table 8: Performance on ParaRules test set, after trained on combined D3+ParaRules training partitions, including PROVER (PV), PROBR (PB), and IBR.

ization ability of IBR by first training the model on the training splits of DU0, DU1, DU2, and DU3, then test them on the test split of DU5 with deeper proof paths respectively[5]. Results are shown in Table 7. We notice that all models suffer performance degeneration especially when the proof depth of the training set is lower, because it is hard for the model to learn complex reasoning based on simple proof paths. However, IBR still realizes the best performance in terms of PA and FA, especially on **DU3**, where it gets 4.2% PA/FA promotion to PROBR and even outperforms PROVER trained on the whole **DU5** data. These observations again prove that iterative approaches can better learn the detailed reasoning step by step, obtaining a better generalization capability than at-once models.

Generalize to complex language. We also evaluate whether IBR can be applied to samples where questions and statements are expressed in more human-like natural language. Following Clark et al. (2020), we train models on the combined training partitions of DU3 and ParaRules then test them on the ParaRules test set. To our best knowledge, it is the dataset that is closest to real-world applications. Table 8 demonstrates that our model sees a slight promotion in PA/FA while a similar accuracy as PROVER in QA, indicating that IBR still has good applicability when doing reasoning on more complicated and natural texts.

5 Analysis

5.1 Ablation Study

To explore the effects between different components in our model, we consider the following ablations: 1) IBR +Gold-Parent: given the gold parent nodes during inference to explore the accuracy of child node prediction; 2) IBR +Gold-Child: given the gold child nodes to verify the accuracy of parent node prediction; 3) *w/o* QA: removing QA task in loss to check its impact on proof generation; 4) *w/o* node LSTM: using mean pooling rather than LSTM encoding to get the representations of nodes; 5) *w/o* focus LSTM: Removing the supplementary LSTM in path focus selection.

Results on the whole DU5 test split are given in Table 9. As the numeric performance shows,

[5] We remove the position embedding in path focus selection to proceed to this test, see Appedix A.1 for details

Models	QA	PA	FA
IBR	99.4	93.5	93.5
IBR +Gold-Parent	99.4	95.6	95.3
IBR +Gold-Child	99.4	99.6	99.3
w/o QA	-	93.7	-
w/o node LSTM	99.5	93.2	93.2
w/o focus LSTM	99.6	92.6	92.4

Table 9: Results of ablation studies on DU5 dataset. We use IBR as the backbone.

giving either gold parent nodes or gold child nodes can benefit the performance especially the later one. This signifies that our parent node prediction achieves promising accuracy while the prediction of child nodes can be further improved. Moreover, IBR can still learn to generate proofs without supervision from answers. And LSTM encoders attribute to a better representation of both the nodes and the path that has been derived.

5.2 Latency Analysis

To demonstrate the computational efficiency of IBR, we compare the per sample inference time of IBR with EVR, also an iterative proof generation model, on the test split of DU5. Additionally, we also compare the per sample inference time of IBR with PROVER and PROBR, both at-once models. All models are tested on one `NVIDIA Tesla-V100` GPU with the same batch size and the beam size of IBR sets to 1 for a fair comparison. As shown in Figure 4, our IBR could achieve up to ×119.5 speedup compared with EVR, benefiting from our reasoning based on node and path features rather than intermediate texts. It is also noticeable that the runtime of EVR grows linearly with depth, while such an effect is slight on our model. Because EVR needs to infer on all contexts at every step, but IBR uses a simplified parent node prediction based on the derived path. Figure 5 illustrates that IBR is also faster than PROVER because PROVER has some constraints during post-processing in inference, like ensuring proof connectivity, which takes extra time.

6 Conclusion

This paper presents IBR, a proof generation model via iterative backward reasoning for rule-based QA tasks. We equip the reasoning procedure with detailed hidden state tracking by predicting nodes and edges in the proof path iteratively backward from the question, and allow the model to reason on the elaborate representations of nodes and history paths. Our model is more interpretable than previous at-once models, and is also more effective and efficient than former iterative models. Experiments also demonstrate the superiority of IBR to various baselines on proof generation under various settings.

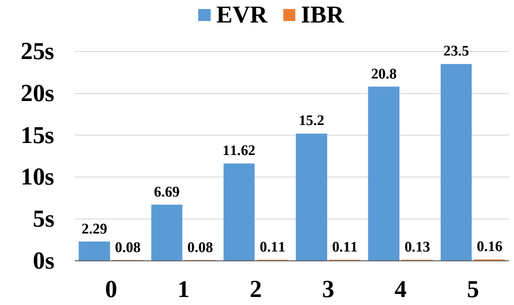


Figure 4: Per-sample inference runtime (in second) of EVR and IBR on DU5 dataset with varying depths.

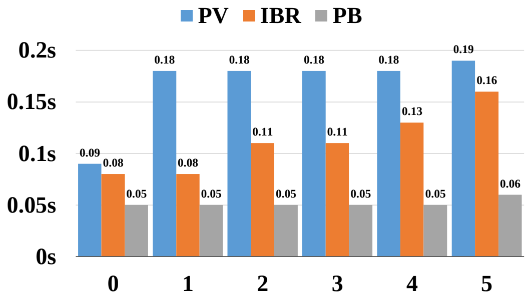


Figure 5: Per-sample inference runtime (in second) of PROVER (PV), IBR, and PROBR (PB) on DU5 dataset with varying depths.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (61876053, 62006062, 62176076), the Shenzhen Foundational Research Funding JCYJ20210324115614039, Shenzhen Science and Technology Program JSGG20210802154400001, and the Joint Lab of HITSZ and China Merchants Securities.

References

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.

Deng Cai and Wai Lam. 2019. Core semantic first: A top-down approach for AMR parsing. 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 3799–3809, Hong Kong, China. Association for Computational Linguistics.

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.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. 2021. Explainable multi-hop verbal reasoning through internal monologue. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1225–1250, Online. Association for Computational Linguistics.

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.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3622–3628. ijcai.org.

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*, abs/1907.11692.

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*, 7:335–352.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

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.

Nan Shao, Yiming Cui, Ting Liu, Shijin Wang, and Guoping Hu. 2020. Is Graph Structure Necessary for Multi-hop Question Answering? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7187–7192, Online. Association for Computational Linguistics.

Changzhi Sun, Xinbo Zhang, Jiangjie Chen, Chun Gan, Yuanbin Wu, Jiaze Chen, Hao Zhou, and Lei Li. 2021. Probabilistic graph reasoning for natural proof generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages

3140–3151, Online. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

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.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Luke S. Zettlemoyer and Michael Collins. 2012. Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. *ArXiv preprint*, abs/1207.1420.

A Appendix

A.1 Implementation Details

Parameter	Value
Training Epochs	8
Optimizer	AdamW
Batch Size	16
RoBERTa Learning rate	1e-5
QA and Strategy Pre Learning rate	1e-5
Parent Node Pre Learning rate	2e-4
Child Node Pre Learning rate	5e-4
All LSTM Learning rate	1e-3
Dropout Rate	0.1
LSTM hidden state for parent node and child node encoding	1024
LSTM hidden state for path encoding in parent node prediction	1024
Transformer hidden state in path focus selection	1024
LSTM hidden state in path focus selection	256
Seed	42

Table 10: Implementation details of IBR.

We implement our model based on PyTorch along with Huggingface-Transformers toolkit[6]. We use RoBERTa$_{\text{Large}}$ model[7] as our backbone encoder to generate token-level representations. Table 10 shows the implementation details of IBR, including learning rates for different modules. All linear layers used in our model have one layer. The model trained after 8 epochs will be used in the evaluation. We remove functional words without lexical meaning like "a" and "the" from facts, rules, and questions to shorten the input length, so each training epoch takes about 2 hours. We select these hyper-parameters according to tuning them empirically based on the performance. All experiments are run on NVIDIA Tesla-V100 GPUs. The main experiment performance of IBR fluctuates by one point.

A.2 Dataset Details

We next introduce the details of the three datasets used in our experiment. All of them are firstly applied in rule-based QA and proof generation tasks in Clark et al., 2020.

Split	D	Num	*Fail-proof* Num	*Proof* Num	Avg. Node
Train	0	21,359	14,597	6,762	0.62
	1	15,380	8,618	6,762	1.82
	2	10,112	3,350	6,762	3.37
	3	8,389	1,627	6,762	4.98
	4	7,456	694	6,762	6.90
	5	6,987	225	6,762	9.26
	all	69,683	29,111	40,572	3.35
Test	0	6,299	4,365	1,934	0.59
	1	4,434	2,500	1,934	1.77
	2	2,915	981	1,934	3.36
	3	2,396	462	1,934	4.99
	4	2,134	200	1,934	6.98
	5	2,003	69	1,934	9.47
	all	20,181	8,577	11,604	3.33

Table 11: The statistics of train and test split in DU5 dataset. *Fail-proof* and *Proof* indicate different proof strategies we discussed in §3.1. Avg. Node indicates the average node number in a proof path.

DU0-DU5: A series of synthesized datasets where rules and facts are all generated via manually designed logical programming, while questions are generated by combining random logical operations among them. Data are divided into 5 subsets according to their maximum reasoning depth (D) in the proof path, D = 0, 1, 2, 3, 5. There are 100k questions in each subset, where 70k / 10k / 20k samples in the train / validation / test partition respectively. D = 0 means that the question can be proven directly using a fact in contexts. In our experiment in §4, we only use the data from DU5 for testing because it covers all possible depths, while the train set is the train split in DU5 except §4.5, where we use train split from DU0, DU1, DU2 and DU3 for training. We provide some statistics of DU5 in Table 11.

Birds-Electricity: It is a set of data that only contains 5k test samples for the evaluation of robustness and out-of-domain performance of models. The Birds data only require reasoning up to depth 1 and 2 (B1 and B2), while Electricity data have reasoning depths ranging from 1 to 4. Both of them include new vocabulary that is not included in DU0-DU5.

ParaRules: A more challenging dataset contains paraphrased samples on the synthesized ones via crowdsourcing. It has 40k questions against about 2k theories. The statements are expressed in a more natural way, posing a discrepancy between DU0-DU5. It has 28k / 4k / 8k samples in the train / validation / test split respectively. In §4.5, we combine it with the extensive DU3 for training,

[6] https://github.com/huggingface/transformers

[7] https://huggingface.co/roberta-large

resulting in a train set containing 119k samples.

A.3 Possible Limitations of Our Model

Since our strategy prediction module and operations corresponding to different strategies in node prediction modules are specially designed for the current datasets, we may need to redesign some specific operations to reach the best performance, if some novel proof types are included in new datasets. But we believe our architecture will still take effect without modification. Besides, the interpretability of IBR is not so strong as former works like EVR that make use of intermediate texts.

A.4 Strategy Accuracy of IBR

D	Cnt	Strategy Accuracy
0	6299	99.9
1	4434	99.1
2	2915	99.3
3	2396	99.0
4	2134	99.2
5	2003	99.7
All	20192	99.4

Table 12: Strategy accuracy of IBR on test split of DU5 after training on training split of DU5.

We provide the strategy prediction accuracy on DU5 in Table 12. It proves that IBR is also well able to make predictions on the proof strategies. This is partly due to RoBERTa's powerful representation capability. On the other hand, there is a certain connection between the answer to the question and the strategy, and there are some common elements at the semantic representation level that can be learned together.

A.5 Performance of EVR and IBR on *Fail-proof* Samples

As we have discussed in §4.2, EVR (Liang et al., 2021) is not applicable for samples containing *Fail-proof* proofs, because it cannot obtain proper intermediate questions to proceed correct following reasoning. Here, we compare our model with EVR on these samples in DU0-DU5, as illustrated in Table 13. Although EVR can achieve promising performance on answer prediction (QA) for these samples, it cannot generate any correct proof path in such cases, which have already been discussed in its original paper.

	D	**0**	**1**	**2**	**3**	**4**	**5**	**all**
	Cnt	4365	2500	981	462	200	69	8577
QA	EVR	99.7	99.1	**98.9**	**99.1**	**98.5**	100	**99.4**
	IBR	**100**	**99.1**	98.3	97.6	96.5	**100**	99.3
PA	EVR	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	IBR	**99.8**	**95.0**	**89.5**	**84.4**	**65.5**	**28.9**	**95.0**
FA	EVR	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	IBR	**99.8**	**95.0**	**89.5**	**84.4**	**65.5**	**28.9**	**95.0**

Table 13: The performance of EVR and IBR on the partial test split of DU5 that only contains samples whose proofs strategies are *Fail-proof*.

A.6 Proof Generation samples

We provide some proof generation samples in Figure 6 for a better understanding of this task, where questions, all contexts, and the proof path generated by our IBR are given (all consistent with the given labels).

Rules:

R_1: If someone is nice and kind then they like the bear.

R_2: If someone sees the dog and they eat the bear then the bear is cold.

R_3: If someone is big then they eat the cat.

R_4: If someone is big then they do not see the rabbit.

R_5: If someone is not big and they do not eat the dog then the dog is cold.

R_6: If someone is cold then they like the rabbit.

R_7: If someone likes the rabbit then they see the dog.

R_8: If the dog eats the cat then the dog is kind.

R_9: If someone likes the dog and they do not eat the cat then the dog eats the bear.

Facts:

F_1: The bear eats the cat.

F_2: The bear eats the rabbit.

F_3: The cat eats the dog.

F_4: The cat eats the rabbit.

F_5: The cat likes the bear.

F_6: The cat sees the rabbit.

F_7: The dog is round.

F_8: The dog likes the bear.

F_9: The dog likes the cat.

F_{10}: The dog sees the bear.

F_{11}: The rabbit eats the bear.

F_{12}: The rabbit is big.

F_{13}: The rabbit is cold.

F_{14}: The rabbit is not kind.

F_{15}: The rabbit does not like the cat.

F_{16}: The rabbit sees the bear.

Q_1: The bear is cold.
A_1 : True
Proof generated by IBR:
Proof Depth = 3 , Strategy: Proof

Q_2: The dog does not see the dog.
A_2: False
Proof generated by IBR:
Proof Depth = 3, Strategy: Proof

Q_3: The dog eats the bear.
A_3: False
Proof generated by IBR:
Proof Depth = 1, Strategy: Fail-proof

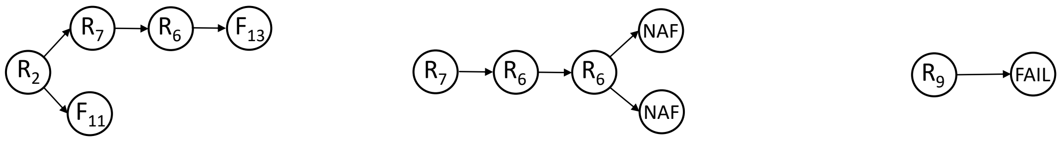


Figure 6: Some proof cases generated by IBR, along with all contexts and questions, including two proof strategies, *Proof* and *Fail-proof*.