CAUSALR: Causal Reasoning over Natural Language Rulebases

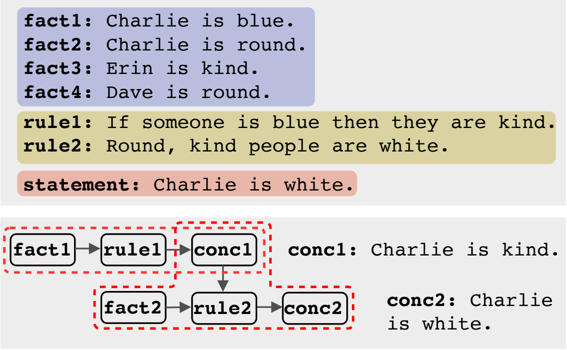
Anonymous ACL submission

Abstract

Transformers have been shown to be able to perform deductive reasoning on a logical rulebase containing rules and statements written in natural language. Recent works show that such models can also produce the reasoning steps (i.e., the *proof graph*) that emulate the model's logical reasoning process. But these models behave as a black-box unit that emulates the reasoning process without any causal constraints in the reasoning steps, thus questioning the faithfulness. In this work, we frame the deductive logical reasoning task as a *causal* process by defining three modular components: rule selection, fact selection, and knowledge composition. The rule and fact selection steps select the candidate rule and facts to be used and then the knowledge composition combines them to generate new inferences. This ensures model faithfulness by assured causal relation from the proof step to the inference reasoning. To test our causal reasoning framework, we propose CAUSALR where the above three components are independently modeled by transformers. We observe that CAUSALR is robust to novel language perturbations, and is competitive with previous works on existing reasoning datasets. Furthermore, the errors made by CAUSALR are more interpretable due to the multi-modular approach compared to black-box generative models. [1]

1 Introduction

The field of AI has long pursued the goal of building systems that can automatically reason over some given *explicit* knowledge to generate conclusions and provide the reasoning steps involved in the process (McCarthy, 1959; Newell and Simon, 1956). However, systems built on formal representation of knowledge has sometimes found this to been challenging (Musen and Van der Lei, 1988).


fact1: Charlie is blue.
fact2: Charlie is round.
fact3: Erin is kind.
fact4: Dave is round.
rule1: If someone is blue then they are kind.
rule2: Round, kind people are white.
statement: Charlie is white.

conc1: Charlie is kind.
conc2: Charlie is white.

Figure 1: **Example of a theory, a statement, and a valid proof graph** - An instance contains multiple facts and rules in blue and yellow respectively, followed by a statement in red. The proof graph describes how the statement can be generated using the theory.

Recently, Clark et al. (2020) proposed a modern version of the problem, where the formal representation of knowledge is replaced by natural language statements in English. Further, they proposed a transformer-based model (Vaswani et al., 2017) RuleTaker, that can predict if a candidate statement is entailed by the natural language statements, by emulating deductive reasoning. While this capability is impressive, it is unclear if such models are also capable of generating the reasoning steps involved in the process (i.e., *proof graph generation*).

Recent works (Saha et al., 2020; Tafjord et al., 2020) have developed systems that can generate the reasoning steps while predicting the statement entailment. However, these systems do not ensure *causality* between generating the proof and predicting the entailment. Since the systems are inherently black-box models, this questions the faithfulness of the model's reasoning process (Lipton, 2018).

In this paper, we look to address these shortcomings by developing a causal framework to solve the deductive reasoning task. While existing methods generate proofs in a single step, in our causal framework we break this process into three steps: rule

[1] Code to reproducte the results have been uploaded and will be published.

selection, fact selection, and knowledge composition. The rule selection step decides the relevant rule to use for an iterative inference step and fact selection *explicitly* uses this rule to select the relevant facts. Then, the knowledge composition step uses the selected rule and facts to reason and generate the next intermediate inference. In Figure 2, we show the model schematics for our system and contrast it with previous methods. Notably, we strictly restrict the information accessible at each step of our framework to make the reasoning process more faithful. For example, in our framework, the fact selection step depends on only the selected rule from the rule selection step, instead of the complete set of rules. Additionally, since we constrain the inputs to each step, this makes each sub-problem easier to learn, leading to an overall more robust reasoning model.

Generative models such as ProofWriter (Tafjord et al., 2020) suffer from another issue that the intermediate proof generation is conditioned on the conclusion (Figure 2 (b)). This is counterintuitive since the model can learn to use the conclusion to generate the corresponding proof, which breaks the causal reasoning chain. This issue is resolved in our framework as we explicitly select the rule and facts required for the inference generation. Thus, proof graphs in our framework are a *by-product* of the selection steps, and the intermediate inference depends directly on the proof. This is a major benefit of using our causal framework for reasoning.

To model these three steps, we develop CAUSALR, in which each component is a transformer-based model learning to perform the modular tasks. Specifically, we use RoBERTa-based models (Liu et al., 2019) for the two selection tasks and a T5-based model (Raffel et al., 2019) for the composition task. Similar to ProofWriter, we use synthetic rulebases to train CAUSALR. Although our model is not end-to-end trainable for proof generation due to explicit selection of rules and facts, this assures a causal relation from proof graph to inference deduction, which is desirable. We experiment with CAUSALR on standard deductive reasoning datasets and multiple robustness datasets. Overall, we find that CAUSALR is more robust to novel language perturbations than baselines, and requires less additional data to generalize to out-of-distribution reasoning tasks. Additionally, our model is up to three times faster at inference due to the constrained input and outputs of different modules. Lastly, we find that the errors made by our model is more interpretable and easier to debug compared to baseline generative models. This further demonstrates the faithfulness of our modularized reasoning framework.

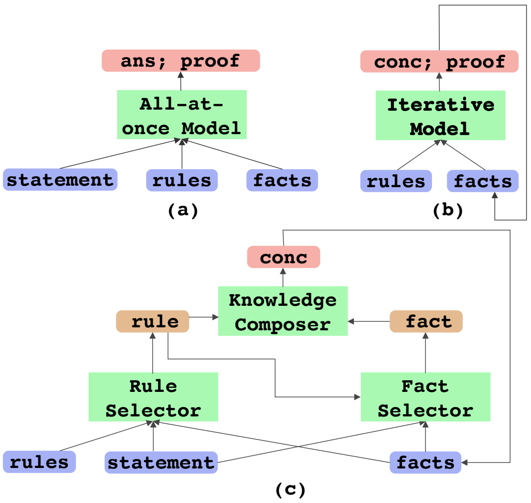


Figure 2: **Reasoning process in different models.** **(a)**: ProofWriter ("All") directly output the entailment prediction and proof graph for given input. **(b)**: ProofWriter ("Iter") iteratively generates the one-step intermediate conclusions and their proofs. **(c)**: CAUSALR selects a rule, then a fact, and finally combines them to generate an intermediate inference. Please refer to Section 3.1 for details.

2 Problem Definition

Notations A theory T consists of a set of facts $F = \{f_1, f_2, \dots, f_n\}$ and rules $R = \{r_1, r_2, \dots, r_m\}$ expressed in natural language. An example of a theory is depicted in Figure 1. Here, the sentences in the blue and yellow boxes are facts and rules, respectively. Further, a *proof graph* is a directed graph connecting facts and rules that describe how a specific inference can be obtained from the theory. In Figure 1, the proof graph shows the steps involved in generating the inference "*Charlie is white.*". To generate the proof graph we may need to infer some intermediate conclusions c_i. These inferences are considered as part of the extended facts in the theory. For example, in Fig. 1, "*Charlie is kind*" is an intermediate inference required to generate the correct proof graph.

Deductive Reasoning The task of deductive reasoning is described as follows: given a theory T, and a statement s, predict if the theory supports the statement (*entailment prediction*) and if so, generate the proof graph that supports the statement (*proof generation*). For the example theory and statement in Figure 1, we see that the statement is indeed entailed by the theory and the valid proof graph is shown for the same. The main goal of this task is to evaluate if a model can generate valid reasoning chains in the form of proof graphs to justify its entailment prediction.

Reasoning Robustness We consider an auxiliary task that evaluates the robustness of the reasoning abilities used by the model. Let $\mathcal{P}$ be a perturbation function that modifies a given theory T (statement s) to a theory T' (statement s'), such that (T', s') just has some surface changes in the natural language form but still requires the similar reasoning process as required for (T, s). A function that alters the subjects in the theory to unseen subjects is an example of such perturbation function. We perturb each theory statement pair (T, s) to create an *equivalence set* defined as the set $E_{(T,s)} = \{(T'_1, s'_1) \dots (T'_N, s'_N)\}$, where each (T'_k, s'_k) is derived by perturbing the original theory, and N is the total such perturbations per theory. Note that it is possible to generate different (T'_k, s'_k) pairs by controlling the stochasticity of $\mathcal{P}$. The main goal of of this task is to evaluate the consistency of the model's predictions with minimal variations in the input theory.

Evaluation Protocol We consider three main aspects for evaluating the model performance in our study: (1) ***Entailment accuracy*** measures how accurately the model is able to predict the true statement entailment. (2) ***Proof accuracy*** measures how accurately the model can predict a valid proof for the statement. Following Saha et al. (2020); Tafjord et al. (2020), we use the strict metric for proof evaluation, i.e., for a match to count, both the predicted proof should exactly match a gold proof and the entailment should be correctly predicted. (3) ***Consistency*** measures if the models are consistent in the entailment and proof prediction for different perturbation functions. For a theory statement pair (T, s) and its corresponding equivalence set $E_{(T,s)}$, consistency is defined as $C = \frac{1}{N}\sum_{k=1}^{N} \mathbb{1}[f(T, s) = f(T_k, s_k)]$, where $f(\cdot)$ is the model's prediction. We compute the average consistency for both entailment and proof predictions on an equivalence set and further average across the dataset to report the consistency.

3 The CAUSALR Method

3.1 Approach Overview

As illustrated by the example in Figure 1, to reliably generate a proof graph through deductive reasoning, a model needs to generate multiple one-hop intermediate conclusions. This is the major limitation of models that use the theory to directly predict the proof (Figure 2 (a)), thus questioning the trustworthiness of the reasoning process. Next, it is also intuitive to see that in order to faithfully generate these intermediate inferences, a model should first determine the proof (i.e., know the rules and facts to use) and then use them to infer the conclusion. That is, there is a causal step between determining the proof and then generating the conclusion. We note that ProofWriter ("Iter") lacks in this aspect. As shown in Figure 2 (b), it first generates the conclusion and then the corresponding proof.

Motivated by these points, we propose our causal reasoning framework which breaks the reasoning process into three desirable steps. As shown in Figure 2 (c), in our framework, first a rule r is selected using the rules and facts in the theory. Following that, some relevant facts are selected from the fact list based on the selected rule r. This step does not use the other rules $R \backslash \{r\}$ in the theory. Finally, the selected rule and facts are jointly used to generate a new conclusion c_i. In this framework, the one-step proof is explicitly determined first via the selection steps followed by the inference generation, making the proof a *by-product* of the whole process. In contrast, prior works learned to generate the proof along with intermediate conclusion.

3.2 CAUSALR Modules

At a high level, CAUSALR is an iterative model in which the one-hop intermediate conclusions are generated step-by-step. To model our causal reasoning framework described in Sec. 3.1, we have four components in CAUSALR as follows.

Rule Selector (RS) The rule selector is a RoBERTa-based (Liu et al., 2019) classification model that takes the concatenated statement, facts, and rules as input, and selects a rule that is used to generate an intermediate conclusion in the current iterative step. It takes the input of the form $[CLS]\ s\ [SEP]\ F\ \ [[SEP]\ \ r_i\ ]_m\ [SEP]$, and generates a one-hot output vector by classifying the token embedding from the [CLS] token and [SEP] tokens in front of the rules, via a linear classifier layer. Each classification is a binary classification, but overall only one of the tokens has the positive class. Here s denotes the statement, F is the facts and concatenated with any intermediate conclusions generated in a prior iteration, and $\{r_i\}$ denotes the i^{th} rule in the theory that contains a total of m rules. $[\,]_m$ denotes continued concatenation. An example input and output of the rule selector is shown in Figure 3. If a [SEP] token is selected, we select the rule sentence following the corresponding [SEP] token, otherwise if the [CLS] token is selected, we decide to stop the iteration. That is, the [CLS] selection acts as a stop signal for our iterative model. We note that it is possible to have more than one likely candidate rule since there can be multiple one-hop inferences possible for a given theory. Following Tafjord et al. (2020), we randomly select one of the possible candidate rules at each iteration.

Fact Selector (FS) The fact selector is RoBERTa-based (Liu et al., 2019) token classification model that takes the statement, the rule selected by the rule selector, and facts in the theory, and then predicts a set of candidate facts that can be used with the rule to generate an intermediate conclusion. It takes the input of the form $[CLS]\ s\ [SEP]\ r\ \ [[SEP]\ \ f_i]_n\ \ [SEP]$, where s is the statement, r is the selected rule, and $\{f_i\}$ is the i^{th} fact in the theory containing n total facts. Note that facts also include any previously generated intermediate conclusions. $[\,]_n$ denotes continued concatenation. The output is generated by classifying each [SEP] token embedding in front of a fact using a linear layer, to determine if the corresponding fact is selected or not. An example input and output for the fact selector is depicted in Figure 3. We note that it is possible to have some rules can reason over multiple facts jointly to generate a conclusion. An example of such a rule is "*rule2*" in Figure 1. Hence, this component has the ability to select multiple facts.

Knowledge Composer (KC) The knowledge composer is a generative text-to-text transformer T5 (Raffel et al., 2019) (T5-large) that can compose a set of facts and a rule to output a novel conclusion. The input to the model is the selected facts and rule concatenated together, and the output is the intermediate conclusion. An example input and output for knowledge composer is shown in Fig. 3.

Solver The final component is the solver that operates after all iterations have finished (i.e., once the rule selector selects the [CLS] token indicating to stop the iterative inference generation process). Similar to ProofWriter, our solver currently searches for the statement in the generated intermediate inferences (string matching). If found, it predicts that the statement is entailed by the theory. It also search for the negation of the statement [2], and if found, it predicts not entailed. If none of these are present, it predicts "Unknown" since it cannot prove or disprove the statement. The proof graph is constructed by using the one-hop proofs generated by the selected rule and facts at each step. For example, in Figure 1, the red dotted boxes (one-hop proofs) are stitched together to assemble the complete proof. For cases where the entailment is "Unknown", the proof returned is "None", since no proof for the statement exists in the theory. We note that our solver is not a learnable module.

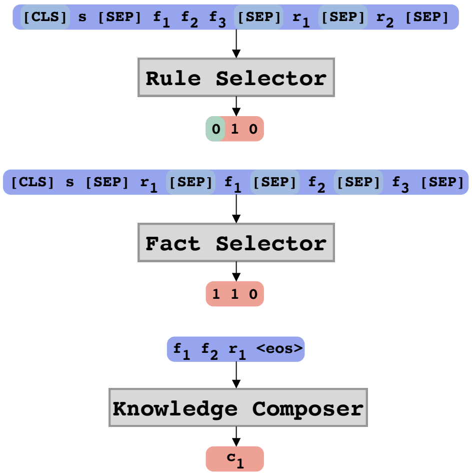

[CLS] s [SEP] f_1 f_2 f_3 [SEP] r_1 [SEP] r_2 [SEP]
Rule Selector
0 1 0
[CLS] s [SEP] r_1 [SEP] f_1 [SEP] f_2 [SEP] f_3 [SEP]
Fact Selector
1 1 0
f_1 f_2 r_1 <eos>
Knowledge Composer
c_1

Figure 3: **Overview of components of CAUSALR** - The rule selector and fact selectors are classification models whereas the knowledge composer is a generation model. The input tokens used for classification by the selectors are highlighted. Rule selector decides to stop based on the output prediction of [CLS] token (highlighted in green). Here, rule r_1, and facts f_1 and f_2 are used to generate the conclusion c_1. Please refer to Section 3.2 for more details.

3.3 Training and Inference

Each component of our model (except the solver, which is deterministic) is trained separately. We use the same dataset as ProofWriter to train these models, but process it such that each model receives only the relevant inputs according to our causal framework. More concretely, suppose for a given theory $T = R + F$, a possible intermediate inference is c obtained by using a rule r and a fact f. Then, a training instance of ProofWriter, which is a T5 (Raffel et al., 2019) model, uses the input $\{R, F\}$ and output $\{c, r, f\}$. We process the same instance to generate three training instances, one for each of rule selector, fact selector, and knowl-

[2] Following ProofWriter, we perform regex to add/remove "not" which suffices for this dataset.

edge composer, respectively, as follows:

$$\begin{aligned} RS\ Input &= \{R, F\};\ \ RS\ Output = \{r\}, \\ FS\ Input &= \{r, F\};\ \ FS\ Output = \{f\}, \\ KC\ Input &= \{r, f\};\ \ KC\ Output = \{c\}. \end{aligned}$$

Our selector models have the statement s as input to the model. Also, the outputs of rule selector and fact selectors are converted to class labels instead of text since our selectors are classification models. We use cross entropy loss to train the rule selector, and binary cross entropy loss to train the fact selector. The knowledge composer is trained on language modeling loss.

At inference time, the rule selector selects a rule to be used for generating one-step conclusions. Then, the fact selector selects some facts based on the selected rule, which is then collectively passed on to the knowledge composer to generate a conclusion. This three-step pipeline is run iteratively until the rule selector predicts a stop signal by selecting the [CLS] token which exits the iteration. Once the iteration finishes, the solver uses the generated intermediate inferences to decide if the statement is entailed or not, and generates a proof accordingly.

Remark on Computational Complexity A practical limitation of ProofWriter is that it performs an exhaustive forward search by enumerating all possible inferences from a given theory. This leads to redundant inferences being generated for proving a particular statement. Additionally, using a text-to-text transformer model adds to the problem since they are usually quite expensive to run at inference time. In CAUSALR, we alleviate this by introducing two changes. First, our causal framework allows only limited information (selected rule and facts) as input to the knowledge composer, thus restricting the input length significantly. Second, augmenting the question to our selector inputs helps reduce the candidate space because these models can learn to prioritize the selection based on the relevance to both the question and the theory. This ensures that CAUSALR does not perform an exhaustive forward search and prioritizes generating relevant inferences over the others. Both these changes lead to an overall improvement in inference speed. We perform more quantitative analysis on this later in Section 5.3.

4 Experimental Setup

Datasets Following (Tafjord et al., 2020; Clark et al., 2020), we use the D* datasets for our experiments. These are a set of multiple datasets - namely D0, D1, D2, D3, D0-D3, and D5. The theory in these datasets are synthetically generated with increasing reasoning depths. For example, D3 dataset contains statements that require *at most* 3-hop reasoning steps. The D0-D3 contains all theories in D3 plus $\sim 20\%$ of the D0-D2 training set theories. We also use the ParaRules dataset (Clark et al., 2020) that contains around 2k theories expressed in paraphrased natural language.

Additionally, we generate three datasets that evaluate the robustness of the reasoning models as follows:

- ***Subject* robustness**: Here, subjects in a theory are perturbed by using some out-of-distribution proper and common names. For example, in Figure 1, "*Charlie*" can be replaced with "*Paul*" which is not used in the D* datasets. We generate five new theories corresponding to each theory of the D3 dataset, by repeatedly perturbing all the proper and common names in the theory.
- ***Attribute* robustness**: Here we sample out-of-distribution attributes. For example, "*blue*" in Figure 1 can be replaced with "*soft*". As above, we generate five new theories for each theory of the D3 dataset.
- ***Subject+Attribute* robustness**: This is a combination of subject and attribute robustness to study model performance when most of the training vocabulary is replaced by out-of-distribution words. Each theory has both novel subject and attribute.

We include more details on the perturbation sets used in our experiments in Appendix A.

Baselines We compare CAUSALR (CAUSALR) with two variants of ProofWriter (Tafjord et al., 2020): All-at-once (PW ("All")) and Iterative (PW ("Iter")), wherever applicable [3]. The PW ("All") model is trained to predict the entailment and generate proof graph directly from the theory and statement in a single step. The PW ("Iter") generates one-step inferences and corresponding proofs iteratively, until all possible inferences are generated, and then stitches the proof graph similar to our method. If not mentioned otherwise, ProofWriter uses a T5-large (Raffel et al., 2019) model.

[3] The code to reproduce numbers of ProofWriter is not publicly available. We either copy results directly from the paper or run our own inference on model checkpoints made available by the authors.

	Entailment Accuracy		**Proof Accuracy**	
d	PW (“Iter”)	CausalR	PW (“Iter”)	CausalR
N/A	99.7	99.6	99.7	99.6
0	100.0	100.0	100.0	100.0
1	99.9	99.7	99.9	99.5
2	99.7	98.9	99.4	97.2
3	99.7	96.6	99.1	95.3
All	99.8	99.2	99.7	98.8

Table 1: Comparison of CausalR with ProofWriter (“Iter”) trained and tested on D0-D3. Baseline results are generated using the checkpoint provided by the authors. For more details please refer to Section 5.1.

Robustness	**PW (“Iter”)**			**CausalR**		
	EA	PA	C	EA	PA	C
Subject	89.6	88.4	87.6	96.8	95.9	96.4
Attribute	89.2	88.8	87.2	88.5	87.3	86.8
Subject+Attribute	86.0	85.7	83.3	88.1	86.9	86.3
Average	88.3	87.6	86.0	91.1	90.0	89.9

Table 2: Comparison of CausalR with ProofWriter (“Iter”) when trained on D0-D3 dataset and tested on different robustness datasets. EA, PA, and C refers to entailment accuracy, proof accuracy, and consistency, respectively. Please refer to Section 5.2 for more details.

5 Experiment Results

We compare CausalR with ProofWriter variants on three settings: generalization on D* datasets, robustness to perturbed theories, efficiency in inference computation. We further conduct qualitative analysis to understand the inference errors.

5.1 Performance on Same Depth Reasoning

In this setting, we train and test both models on D0-D3 dataset. Note, D0-D3 contains statements with reasoning depths up to 3. This compares the ability of the models to generalize to *seen* reasoning depths at train time. The results with increasing depths of reasoning are shown in Table 1. Here, depth “N/A” refers to statements that cannot be proven and hence don’t have an exact proof depth associated with it. We observe that overall both CausalR and ProofWriter (“Iter”) performs comparably (last row with depth ’All’). Further, we find that our model’s performance is lower on $d = 3$, indicating that our models tend to perform weaker with increasing depths. This happens majorly because the rule selector in CausalR tends to incorrectly select the [CLS] token to indicate a stop signal instead of generating more possible intermediate inferences. We discuss more about this in Sections 5.3 and 5.4. Please refer to Appendix B for more results on unseen reasoning depths.

	Entailment Accuracy		**Proof Accuracy**	
d	PW (“Iter”)	CausalR	PW (“Iter”)	CausalR
N/A	98.9	99.3	98.9	99.3
0	99.9	100.0	99.9	100.0
1	79.1	96.0	78.8	95.7
2	76.6	93.4	73.4	91.4
3	72.7	89.8	67.8	85.7
All	89.6	96.8	88.4	95.9

Table 3: Comparison of CausalR with ProofWriter (“Iter”) trained on D0-D3 and tested on subject robustness dataset. Baseline results are generated using the checkpoint provided by the authors. For more details please refer to Section 5.2.

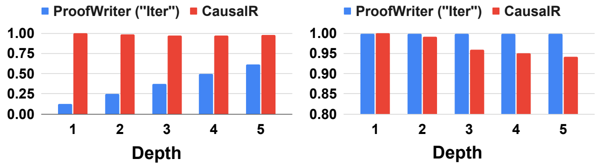
5.2 Robustness to Perturbed Theories

In this section, we test the robustness of ProofWriter (“Iter”) and CausalR on different perturbed theories. As described in Section 4, we test the robustness on three different perturbations: subject, attribute, and subject+attribute. We compare the performance of both models after training on D0-D3 dataset. The consolidated results are shown in Table 2 and depth-wise results for subject robustness are shown in Table 3. We report the entailment accuracy, proof accuracy, and consistency as defined in Section 2. Please refer to appendix C for the depth-wise breakdown of all the datasets. We observe that on subject and subject+attribute robustness, our models are consistently better than ProofWriter whereas on attribute robustness both models perform similarly. Further, we find that on average, CausalR is both more accurate and consistent than the baseline. From this, we conclude that our model relies less on spurious correlations based on the subject while both models likely suffer from similar issues on object perturbations. Since ProofWriter uses the theory to *generate* the intermediate conclusion and proofs, it has the capacity to exploit some spurious patterns that can inflate performance. In contrast, our causal framework restricts this capacity by constraining the inputs to each component as described in Section 3.1. Hence, these robustness evaluations demonstrate one of the prime benefits of our causal and modular approach.

5.3 Study on Inference Efficiency

Here we perform several analysis to evaluate the computational benefits of our method as described in Section 3.3. Inference efficiency is an important aspect of this problem for real-world scenarios where compute can be limited.

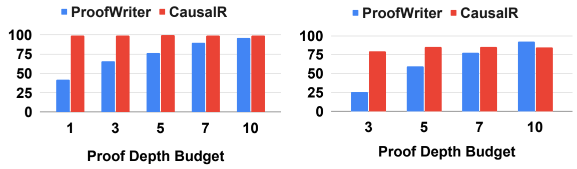
Relevance of generated inferences Here, we study the relevance of the intermediate inferences generated by CausalR and ProofWriter (“Iter”).



(a) Precision

(b) Recall

Figure 4: Comparison of ProofWriter ("Iter") and CAUSALR on precision and recall of generated inferences with increasing reasoning depths.



(a) Depth 1

(b) Depth 3

Figure 5: Depth-wise comparison of ProofWriter ("Iter") and CAUSALR on limited inference budgets. Please refer to Section 5.3 for details.

Let T be the set of intermediate inferences required for generating the proof graph for the statement. Further, let G be the set of intermediate inferences actually generated by a model. Then, the precision and recall are defined as $P = \frac{|T \cap G|}{|G|}$, and $R = \frac{|T \cap G|}{|T|}$ In Figure 4, we plot the precision and recall for both CAUSALR and ProofWriter ("Iter") with increasing reasoning depths. We find that our model has close to 1.0 precision at all depths, whereas ProofWriter has low precision. This demonstrates that our model is able to successfully prune the candidate inference space to generate relevant candidate inferences almost perfectly. In contrast, we see that with increasing depths, our model's recall reduces from close to 1.0 to ≈ 0.95 whereas ProofWriter has a perfect recall at all depths. While the drop is not very drastic, it indicates that our model fails to generate some essential inferences at higher depths. This is mainly because our rule selector decides to stop early and not generate further relevant inferences for some provable statements. Overall, we conclude that CAUSALR always generates inferences that are relevant to solving the instance, although at higher depths it can miss some relevant conclusions.

Performance under inference budget constraints We analyze the performance of CAUSALR and ProofWriter under a fixed inference budget constraint by restricting the total number of conclusions that can be generated. We perform this analysis for different reasoning depths and depict the results in Figure 5. We observe that CAUSALR consistently outperforms ProofWriter on lower budgets. This shows that CAUSALR performs a prioritized generation of conclusions that are relevant to the statement, which can be useful in scenarios with limited inference budgets. See Appendix D for more comparisons.

Inference runtime analysis We next compare the time taken by both the models to solve the complete D5 dev set. Although CAUSALR has three separate modules that run sequentially, it is 3.5 times faster than ProofWriter ("Iter") at inference time on average. We attribute this to the reduced inference candidate search space due to question augmentation, and smaller input size to the T5 component (refer to Section 3.3 for details). Please refer to Appendix E for more details.

5.4 Error Analysis

We further analyze the different errors made by CAUSALR and ProofWriter ("Iter") on 50 randomly sampled errors for each model, from the D0-D3 and the subject robustness dev splits. We manually inspect the proof inferences and compare it with the gold proof to classify the failures. The errors are broadly categorized as follows:

Early stop errors: This is the most frequent error type for both models, accounting for 80% and 50% errors in CAUSALR and ProofWriter, respectively. This occurs when a model incorrectly generates the stop signal and fails to generate all the required inference to prove a statement. We find that our model makes the majority of the mistakes due to early stopping. This can be possibly fixed by improving the rule selector architecture to better model the stop criteria.

Wrong inference: This is the second error type, where the proofs look correct, but the inferred conclusion is incorrect. This accounts for 20% and 30% errors in CAUSALR and ProofWriter, respectively. We observe that our knowledge composer is makes lesser errors on average compared to the ProofWriter generative model.

Other generation errors: ProofWriter makes around 20% errors where the model generated output does not make sense. For example, it can hallucinate facts that are not present in the theory. Such errors are not interpretable and questions the model's inner-working. CAUSALR shows no such error, since the proofs are always interpretable in our model due to the causal framework.

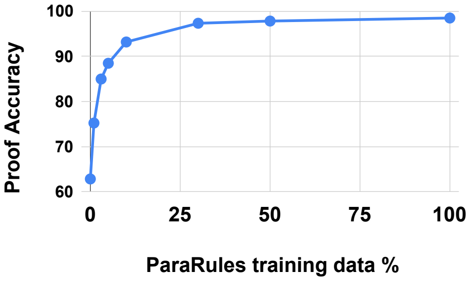


Figure 6: Proof Accuracy of CAUSALR when tested on ParaRules while using limited amount of ParaRules along with D0-D3 for training. See Sec. 5.5 for more details.

Overall, we find that the errors made by CAUSALR are more interpretable than ProofWriter, since we can pin-point which module is at fault. Whereas, in ProofWriter, it is sometimes hard to understand the source of errors due to entangled reasoning and proof generation process. This feature also makes our framework easy to debug to potentially fix some components with techniques like data augmentation. Please refer to Appendix G for more discussion and examples of errors.

5.5 Generalization to paraphrased theories

Next, we test the ability of our model to generalize to unseen language in ParaRules by using limited training supervision. To test this, we first train our model on D0-D3 dataset and test it on the ParaRules dataset. This is a zero-shot evaluation on an unseen language form. In Figure 6 we observe that the performance is significantly worse on this setting as expected. We also evaluated a checkpoint of ProofWriter ("Iter") trained on D0-D3 which achieves a similar performance of 62.13% entailment accuracy [4]. Next, we gradually start adding portions of ParaRules, along with the D0-D3 data, to the training dataset. We find that CAUSALR can quickly achieve reasonable performance using even 10% additional data. This shows that our modularized approach is also efficient in adapting to unseen theories with limited data supervision. For more comparisons with ParaRules trained models, please refer to Appendix F.

6 Related Works

Reasoning in Text Reasoning in text is a well studied problem in NLP. Natural Language Inference (NLI) (MacCartney and Manning, 2014) is one of the most prominent datasets that require reasoning over text to answer if a statement is entailed, contradicted, or neutral, given a hypothesis. More recently, datasets like HotpotQA (Yang et al., 2018), bAbI (Weston et al., 2016), QuaRTz (Tafjord et al., 2019), ROPES (Lin et al., 2019), CLUTRR (Sinha et al., 2019), etc. have studied different aspects of reasoning over textual inputs. These tasks usually require implicit reasoning, where the model needs to internally infer the rules required to solve the task. In contrast, RuleTaker (Clark et al., 2020) deals with explicit reasoning (also known as deductive reasoning).

Proof Generation Recently, some works have been addressing the problem of proof generation from an NL-based theory. Prover (Saha et al., 2020) trains a RoBERTa-based model that predicts nodes and edges of the proof graph. ProofWriter (Tafjord et al., 2020) is a T5-based (Raffel et al., 2019) model, that iteratively generates one-hop conclusions and proofs from a theory. Another work MultiProver (Saha et al., 2021), generates multiple possible proofs for a statement. While we study the same problem of proof generation similar to these works, we focus on the causal reasoning process and design a modular system for proof generation.

Formal Reasoning There are some prior works that try to solve the problem of entailment prediction by first parsing the formal language from text. Neural Theorem Prover (Rocktäschel and Riedel, 2017; Weber et al., 2019) uses neural networks to parse the formal logic from natural language and then reason over them. While this approach is more symbolic, it can lead to many challenges while parsing (Kamath and Das, 2018). The proof generation setting considered here bypasses this step and directly reasons over the given natural language text making it more useful in downstream applications.

7 Conclusion

In this paper, we proposed CAUSALR, a causal reasoning model based on three modular components: rule selection, fact selection, and knowledge composition. CAUSALR ensures causality from proof graph generation to entailment prediction by design. We established the effectiveness of our approach through experiments on testing reasoning robustness to language variations, verifying the ability to quickly generalize to unseen language forms, and analyzing the interpretability of the errors made by our model. We also demonstrate that CAUSALR is faster and more precise at deductive reasoning than existing baselines.

[4] data-augmented training results for ProofWriter are not reported in the figure since the training code is not available

References

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *CoRR*.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *CoRR*.

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.

Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, (3).

Yinhan Liu, Myle Ott, Naman Goyal, and Jingfei Du an. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Bill MacCartney and Christopher D. Manning. 2014. *Natural Logic and Natural Language Inference*, pages 129–147. Springer Netherlands, Dordrecht.

John W. McCarthy. 1959. Programs with common sense. In *Proc. Tedding Conf. on the Mechanization of Thought Processes*, pages 75–91.

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.

Allen Newell and Herbert A. Simon. 1956. The logic theory machine-a complex information processing system. *IRE Trans. Information Theory*, 2:61–79.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. *CoRR*.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. Prover: Proof generation for interpretable reasoning over rules. *CoRR*.

Swarnadeep Saha, Prateek Yadav, and Mohit Bansal. 2021. multiprover: Generating multiple proofs for improved interpretability in rule reasoning. *CoRR*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. 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 4506–4515, Hong Kong, China. 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.

Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. 2020. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *CoRR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 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. *CoRR*.

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*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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.

A Robustness dataset details

The robustness dataset is created by replacing all subjects (attributes) in the D3 dataset with unseen subjects (attributes) to create the subject (attribute) robustness set. For this, we first curate new sets of subjects and attributes to be used as a global pool to sample from while replacing existing subjects and attributes from the theory. These sets are detailed below:

Subject proper name pool: {‘George’, ‘Paul’, ‘Ronald’, ‘Emma’, ‘Magnus’, ‘Timothy’, ‘Chris’, ‘Molly’, ‘Diana’, ‘Joseph’, ‘Becky’, ‘Kurt’, ‘Ivan’, ‘Steve’, ‘Laura’, ‘Oliver’, ‘Adam’, ‘Larry’}

Subject common name pool: {‘mother’, ‘father’, ‘baby’, ‘child’, ‘toddler’, ‘teenager’, ‘grandmother’, ‘student’, ‘teacher’, ‘alligator’, ‘cricket’, ‘bird’, ‘wolf’, ‘giraffe’, ‘dinosaur’, ‘thief’, ‘soldier’, ‘officer’, ‘artist’, ‘shopkeeper’, ‘caretaker’, ‘janitor’, ‘minister’, ‘salesman’, ‘saleswoman’, ‘runner’, ‘racer’, ‘painter’, ‘dresser’, ‘shoplifter’}

Attribute pool: {‘maroon’, ‘brown’, ‘black’, ‘orange’, ‘cordial’, ‘friendly’, ‘adorable’, ‘old’, ‘soft’, ‘violent’, ‘intelligent’, ‘square’, ‘warm’, ‘large’, ‘cylindrical’, ‘spherical’, ‘tiny’, ‘microscopic’, ‘brilliant’, ‘noisy’, ‘playful’, ‘tender’, ‘gracious’, ‘patient’, ‘funny’, ‘hilarious’, ‘thorny’, ‘sensitive’, ‘diplomatic’, ‘thoughtful’}

Then, for each theory in the D3 dataset, we replace *all* the subjects in the theory with randomly sampled subjects (without replacement) from the candidate set to create a perturbed theory. We perform this replacement operation to generate five different perturbed theories. These perturbed theories are called equivalence set. Note that the only change in each theory in an equivalence set is the subjects being replaced by some randomly sampled subjects. For example, “cat” in the original theory might be replaced by “child” in one perturbation, and with “teacher” in yet another perturbation. We follow the same procedure to create attribute and subject+attribute robustness sets.

B Generalization to reasoning depths

In this section, we experiment with a setting where models are trained on depths less than or equal to 3 (i.e., $d \leq 3$) and tested on D5 dataset that contains statements that require reasoning up to depth 5 (i.e., $d \leq 5$). Here, we test the generalization of the models to reasoning depths that are *unseen* at training time. These results are shown in Table 4. From this table, we observe that overall our model performs significantly better than ProofWriter (“All”) on proof accuracy ($+7.5\%$), but has a lower performance compared to ProofWriter (“Iter”) (-3%). This shows that compared to ProofWriter (“Iter”), our models are weaker at generalizing to unseen reasoning depths. This happens majorly because our rule selector tends to stop the inference iterations earlier, which means some essential inferences are not generated by the model. Thus, this leads to lower performance with increasing reasoning depths.

	Entailment Accuracy			Proof Accuracy		
d	PW (“All”)	PW (“Iter”)	CAUSALR	PW (“All”)	PW (“Iter”)	CAUSALR
N/A	97.4	99.2	99.4	97.4	99.2	99.4
0	100.0	100.0	100.0	100.0	100.0	100.0
1	99.9	99.1	99.5	99.3	97.5	99.2
2	99.7	98.9	98.5	97.6	96.4	96.1
3	99.7	98.4	93.4	91.2	95.5	85.5
4	99.5	97.5	88.8	46.9	93.4	77.4
5	98.9	96.5	79.2	24.4	82.3	68.1
All	98.7	98.8	95.9	85.6	96.4	92.7

Table 4: D5 dataset depth-wise performance comparison of CAUSALR trained on D0-D3 with ProofWriter (“All”) and ProofWriter (“Iter”) trained on D3 and D0-D3 respectively. Baseline results are copied from Tafjord et al. (2020). Refer to Section 5.1 and Appendix B for more details.

But, we make another interesting observation here. The drops in entailment and proof accuracy with increasing depths are similar for CAUSALR. For instance, considering the performance drops between $d = 4$ to $d = 5$, CAUSALR has $\sim 9.5\%$ drop in both entailment and proof accuracy. In contrast, ProofWriter (“All”) and ProofWriter (“Iter”) drops approximately 22% and 11%, respectively in proof accuracy for a mere 1% drop in entailment accuracy. This raises some concern on the causality of the proof generation process used for entailment prediction in these models, since it seems like the answer prediction and proof generation are not dependent via the same reasoning paths. In contrast, our causal framework grounds the entailment prediction to the proofs and this leads to more consistent performance variations in CAUSALR.

C Robustness to Perturbed Theories

Here, we show the detailed depth-wise performance of CAUSALR and ProofWriter (“Iter”) trained on D0-D3 dataset and evaluated on different robustness datasets as described in Section 4. The

	Entailment Accuracy		Proof Accuracy	
d	PW (“Iter”)	CausalR	PW (“Iter”)	CausalR
N/A	98.9	99.3	98.9	99.3
0	99.9	100.0	99.9	100.0
1	79.1	96.0	78.8	95.7
2	76.6	93.4	73.4	91.4
3	72.7	89.8	67.8	85.7
All	89.6	96.8	88.4	95.9

Table 5: Comparison of CausalR with ProofWriter (“Iter”) trained on D0-D3 and tested on subject robustness dataset. Baseline results are generated using the checkpoint provided by the authors. For more details, please refer to Appendix C.

	Entailment Accuracy		Proof Accuracy	
d	PW (“Iter”)	CausalR	PW (“Iter”)	CausalR
N/A	99.7	99.2	99.7	99.2
0	100.0	100.0	100.0	100.0
1	83.7	83.8	83.4	83.1
2	73.2	72.3	71.9	69.7
3	66.4	63.2	64.4	57.9
All	89.2	88.5	88.8	87.3

Table 6: Comparison of CausalR with ProofWriter (“Iter”) trained on D0-D3 and tested on attribute robustness dataset. Baseline results are generated using the checkpoint provided by the authors. For more details, please refer to Appendix C.

	Entailment Accuracy		Proof Accuracy	
d	PW (“Iter”)	CausalR	PW (“Iter”)	CausalR
N/A	99.8	99.2	99.8	99.2
0	99.9	100.0	99.9	100.0
1	74.2	83.1	73.9	82.5
2	65.3	71.1	64.3	68.3
3	59.9	62.1	58.7	56.7
All	86.0	88.1	85.7	86.9

Table 7: Comparison of CausalR with ProofWriter (“Iter”) trained on D0-D3 and tested on subject+attribute robustness dataset. Baseline results are generated using the checkpoint provided by the authors. For more details, please refer to Appendix C.

results for subject, attribute, and subject+attribute robustness evaluations are shown in Tables 5, 6, and 7, respectively. We observe that ProofWriter (“Iter”) performs significantly worse compared to CausalR on subject robustness. The results on subject+attribute robustness are mostly comparable, while in attribute robustness our model performs worse. The drop in performance show that both the models are sensitive to attributes in the theory to varying degree. But the strong sensitivity of ProofWriter (“Iter”) to the subject perturbations is questionable, since the causality of the model’s reasoning process seems to be compromised because the model learns some spurious correlations using the subjects.

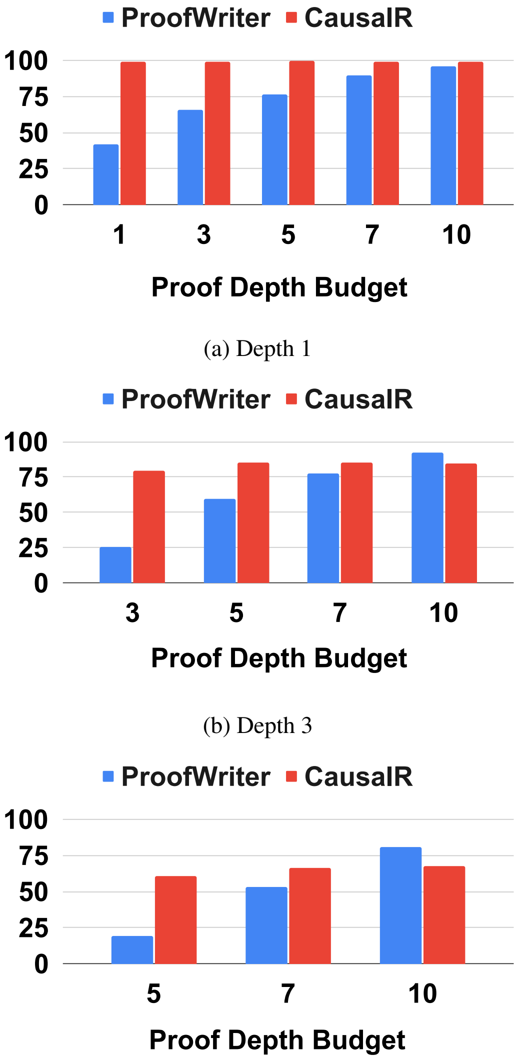
In another setting, we train different components of our model on the robustness data and check if that leads to some performance gains. These results are reported in Table 8. We find that it is indeed possible to improve the performance of individual components of our model by robust data augmentation. This also indicates that our individual components are flexible to intervention by data augmentation. Such abilities are lacking in ProofWriter.

D Inference Budget Analysis

In the inference budget analysis, we compare the performance of CausalR and ProofWriter under an inference budget constraint, i.e., we restrict the total number of intermediate conclusions that can be produced by both models. We perform this analysis on three different depth datasets ($d = \{1, 3, 5\}$) and upper bound the number of inferences by $B = \{1, 3, 5, 7, 10\}$. We ensure that the budget is at least equal to the depth of the statements under consideration since proving a statement requires a model to generate inferences equal to at least the depth. From Figure 7 we observe that for all depths CausalR consistently outperforms ProofWriter on lower budgets. Only when the budget increases to 10, ProofWriter compares with or sometimes outperforms our model. This analysis demonstrates that CausalR performs a prioritized generation of conclusions that are relevant to the statement, which can be useful in scenarios with limited inference budgets.

E Runtime Analysis

For inference runtime analysis, we time the evaluation of both CausalR and ProofWriter (“Iter”) on D5 dev set. Note that D5 dataset contains statements that require *at most* five reasoning steps to generate an answer. The runtime for both methods are shown in Table 9. These results were obtained by running the inference algorithm on NVIDIA GeForce RTX 2080 Ti GPUs for both models. We observe that ProofWriter (“Iter”) has an almost constant runtime since it always generates all possible inferences for a theory. In contrast, our runtime



(a) Depth 1

(b) Depth 3

(c) Depth 5

Figure 7: Depth-wise comparison of ProofWriter ("Iter") and CausalR (both trained on D0-3 dataset) on limited inference budgets. Please refer to Appendix D for details.

Trained module	Subj Perturbation		Attr Perturbation	
	EA	PA	EA	PA
Base (trained on D0-D3)	96.8	95.9	88.5	87.3
Rule selector (RS)	97.3	96.7	89.6	86.8
Fact selector (FS)	96.7	95.8	88.6	87.1
Knowledge composer (KC)	98.5	97.6	89.4	88.1
RS + FS + KC	99.1	98.5	97.1	91.0

Table 8: Comparison of variants of CausalR where different components (RS, FS, KC) and their combinations are trained and tested on robustness datasets. Please refer to Appendix C for more details.

d	ProofWriter ("Iter")	CausalR
0	1.01	0.12
1	1.01	0.20
2	1.00	0.28
3	1.01	0.36
4	1.01	0.46
Avg	1.01	0.28

Table 9: Evaluation runtime (in hours) of CausalR and ProofWriter ("Iter"). Please refer to Appendix E for more details.

increases almost linearly with increasing depth. On average, CausalR is 3.5 times faster at inference than ProofWriter ("Iter").

F Results on ParaRules training

Following (Tafjord et al., 2020), we compare the performance of ProofWriter ("All") and CausalR on the ParaRules dataset, when trained on a combined partition of D3 and ParaRules train set. The ParaRules dataset contains complex linguistic expressions in the theories that are more realistic than D* dataset theories, making it a more challenging dataset. These results are shown in Table 10, with a reasoning depth breakdown as before. We note that numbers for ProofWriter ("Iter") are not reported in the paper, and no trained checkpoint is available either, so we omit it from our comparisons. Also, the reported results for ProofWriter ("All") are from evaluating a T5-11B model while ours is a T5-large model. Here, we see that our model performs better at higher depths compared to the baseline which demonstrates that CausalR is better at handling paraphrases.

G Error Analysis

This is a follow-up of Section 5.4, where we delve deeper into the error analysis by discussing different error examples and their potential reasons. First, the stop errors are easy to understand. These

	Entailment Accuracy		Proof Accuracy	
d	PW (“All”) [T5-11B]	CausalR	PW (“All”) [T5-11B]	CausalR
0	99.9	100.0	99.9	100.0
1	99.3	99.6	99.3	99.6
2	98.3	97.6	97.7	97.4
3	98.2	95.4	96.5	95.1
4	91.5	91.6	83.1	91.6
All	99.1	98.7	98.5	98.6

Table 10: Comparison of CausalR with ProofWriter (“All”) [T5-11B] when trained on D3+ParaRules and tested on ParaRules. Results for ProofWriter (“All”) [T5-11B] are copied from the paper. Please refer to Appendix F for more details.

are cases where the model just decides to stop instead of generating any further conclusions. For our model, this can happen if the rule selector is under confident while selecting rules and it learns that a safer fallback is to stop generating rules. This aspect can probably be improved by a better modeling of the rule selector. We plan to explore this in future works.

Next we look at some of the wrong inferences generated by both models in Tables 11 and 12. We observe that errors made by CausalR are rather naive with small mistakes in the final conclusion (shown in red in Table 12). In contrast, ProofWriter tends to generate an invalid conclusion with no relation to the generated proof (rows 1 and 2 in Table 11). It also makes many non-interpretable generation errors where the model's output format is completely violated or the model seems to hallucinate some facts (rows 3-6 in Table 11). Thus, we observe the benefit of our causal framework as the errors are interpretable and more believable. In contrast, errors made by ProofWriter clearly show that its inference reasoning process can often not rely on the proof at or, or even the generated proof sometimes doesn't make sense.

H Hyperparameters

We use RoBERTa-large models (Liu et al., 2019) to model the rule selector and fact selector in CausalR. For selecting the best hyperparameters for both these components, we selected the max training epochs in: $\{10, 15, 20\}$, warmup updates in: $\{0.05, 0.1\}$, weight decay in: $\{0.1, 0.01, 0.001\}$, learning rate in: $\{3e\text{-}6, 5e\text{-}6, 1e\text{-}6\}$, and batch size in $\{16, 32\}$.

We use T5 (Raffel et al., 2019) (T5-large) to model the knowledge composer in CausalR and train it using the default hyperparameters available in the Hugging Face transformers library (Wolf et al., 2020).

Error Type	**Input** (*format: complete theory*)	**Output** (*format: $answer$ = conclusion ; $proof$ = # rule & facts*)
Wrong inference	$answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: Cold people are red. sent2: If someone is quiet then they are cold. sent3: Smart, blue people are quiet. sent4: If Paul is blue then Paul is furry. **sent5**: If someone is blue then they are quiet. sent6: If Steve is furry and Steve is smart then Steve is white. sent7: If Chris is red and Chris is furry then Chris is blue. sent8: Quiet people are cold. **sent9**: Chris is blue. sent10: Chris is white. sent11: Ronald is red. sent12: Paul is quiet. sent13: Paul is smart. sent14: Steve is blue. sent15: Steve is furry. sent16: Steve is red. sent17: Steve is smart. sent18: Steve is white.	$answer$ = Dave is quiet. ; $proof$ = # sent5 sent9
Wrong inference	$answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: Cold people are red. **sent2**: If someone is quiet then they are cold. sent3: Smart, blue people are quiet. sent4: If Paul is blue then Paul is furry. sent5: If someone is blue then they are quiet. sent6: If Steve is furry and Steve is smart then Steve is white. sent7: If Chris is red and Chris is furry then Chris is blue. sent8: Quiet people are cold. sent9: Chris is blue. sent10: Chris is white. sent11: Ronald is red. sent12: Paul is quiet. sent13: Paul is smart. sent14: Steve is blue. sent15: Steve is furry. sent16: Steve is red. sent17: Steve is smart. sent18: Steve is white. **sent19**: Dave is quiet.	$answer$ = Dave is not a cat. ; $proof$ = # sent2 sent19
Generation error - output format violated	$answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: If someone likes the grandmother then the grandmother eats the artist. sent2: If someone eats the artist then the artist is young. sent3: If the artist eats the grandmother then the artist is cold. sent4: If someone eats the artist and the artist eats the grandmother then the grandmother eats the artist. sent5: If someone is red and big then they like the grandmother. **sent6**: If someone visits the grandmother then they are red. sent7: If someone is rough then they eat the grandmother. sent8: If someone likes the artist then they visit the grandmother. sent9: The artist likes the grandmother. sent10: The grandmother eats the artist. sent11: The grandmother is big. sent12: The grandmother is cold. sent13: The grandmother is young. sent14: The grandmother likes the artist. **sent15**: The grandmother visits the artist. sent16: The artist is young.	$answer$ = The red sent6 sent15
Generation error - hallucinating facts	$answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: All young things are smart. **sent2**: All smart, young things are nice. sent3: If something is young and nice then it is quiet. sent4: Becky is big. sent5: Becky is green. sent6: Becky is young. sent7: Larry is big. sent8: Larry is quiet. sent9: Larry is red. sent10: Larry is young. **sent11**: Bob is smart.	$answer$ = Bob is nice. ; $proof$ = # sent2 & sent11 sent16
Generation error - repeated facts	$answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: If Adam is green then Adam is big. sent2: Green things are rough. sent3: If Joseph is cold then Joseph is big. sent4: All blue things are cold. sent5: All green things are rough. sent6: All big things are kind. sent7: If something is rough and big then it is green. sent8: All kind, blue things are cold. sent9: Joseph is blue. sent10: Adam is blue. sent11: Molly is cold. sent12: Joseph is cold. **sent13**: Charlie is cold.	$answer$ = Charlie is sent13 sent13
Generation error - output format violated	$answer$; $proof$; $question$ = What is one single-hop inference? ; $context$ = sent1: Nice things are green. sent2: If Ivan is smart then Ivan is blue. sent3: Green things are smart. sent4: All cold things are nice. sent5: Big things are blue. sent6: If Ronald is nice then Ronald is big. sent7: If something is green and big then it is not kind. **sent8**: If something is smart and nice then it is kind. **sent9**: Ronald is smart. sent10: Larry is cold. sent11: Laura is nice. sent12: Ivan is kind. sent13: Larry is nice. sent14: Larry is green. sent15: Bob is smart.	$answer$ = Bob is kind. ; $proof$ = # sent8 & sent9 sentnice

Table 11: Examples of inference errors made by ProofWriter (“Iter”).

Error Type	**Input (*format: [facts], rule*)**	**Inference**
Wrong inference	[the racer needs the janitor.], if someone needs the janitor then the janitor likes the racer.	the janitor likes the race.
Wrong inference	[oliver is big.], big people are young.	the oliver is young.
Wrong inference	[the shoplifter needs the shoplifter., the shoplifter needs the dinosaur.], if something needs the dinosaur and it needs the shoplifter then it is round.	the shop is round.

Table 12: Examples of inference errors made by CAUSALR while composing rules and facts using the knowledge composer.