Teaching Probabilistic Logical Reasoning to Transformers

Anonymous ACL submission

Abstract

In this paper, we evaluate the ability of transformer-based language models in reasoning over uncertain text that includes uncertain rules of reasoning. We cover pre-trained language models (PLMs) and the newer large language models (LLMs). Our evaluation results show that both generations of language models struggle with reasoning over uncertain text. We focus on PLMs and propose a novel Neuro-Symbolic fine-tuning approach, Probabilistic Constraint Training (PCT), incorporating probabilistic logical rules as constraints during fine-tuning. To assess the effectiveness of PCT, we utilize the related corpora and, additionally, create a new and more challenging benchmark that, unlike the previous ones, uses instance-specific rules. Our study demonstrates the potential of PCT, the pioneer method that improves the transformer-based language model's accuracy and explainability of the probabilistic logical reasoning process. Furthermore, PCT equips these models to effectively handle novel situations, including higher reasoning depth, new domains, and complex probabilistic structures.

1 Introduction

PLMs have become popular since they demonstrated high accuracy across a wide range of Natural Language Processing (NLP) tasks (Liu et al., 2019). LLMs are becoming even more popular as they can solve many NLP problems zero-shot (Chen, 2023); however, they are expensive to utilize. Our study focuses on a less explored area of reasoning over uncertain text involving uncertain rules. We will discuss the importance of this area and PLMs' and LLMs' weaknesses in handling this problem. We then propose our solution to improve PLMs, enabling them to surpass the more costly LLMs and transfer their learned reasoning.

Understanding logical and uncertain rules in natural language form has been investigated in recent works (Clark et al., 2020; Saeed et al., 2021). While incorporating hard logical rules is still a research question, in the real world, most of the external knowledge and rules are uncertain. Only a small fraction of the logical rules in DBpedia can be deemed certain (Saeed et al., 2021). Science communication frequently utilizes certainty and uncertainty mainly with the help of hedges (Pei and Jurgens, 2021). Outlining and understanding certainties and uncertainties is required in scientific communications (National Academies of Sciences et al., 2017). This indicates the need for models capable of reasoning over uncertain knowledge.

PLMs and LLMs struggle to reason with numbers and simple mathematical questions expressed in natural language (Mishra et al., 2022), a requirement for inference on probabilistic and uncertain text. PLMs' evaluation of various question-answering (QA) benchmarks show they produce contradictory results (Asai and Hajishirzi, 2020). Such limitations reveal the issues of implicitly using external knowledge by PLMs, making the reasoning process an unexplainable blackbox (Clark et al., 2019). These challenges encountered in PLMs are our motivation to train them to adhere to a simplified probabilistic reasoning process for an explicit integration of logical probabilistic rules.

We utilize two QA datasets: RuleBERT (Saeed et al., 2021) and our newly developed RuleTaker-pro, a probabilistic extension of the RuleTaker dataset (Clark et al., 2020) created to address some of the shortcomings of the RuleBERT dataset. Mainly, we want a dataset with context-specific rules to make the required reasoning more realistic. For example, the probability of two married people being cousins in the context of one culture is high, while it is close to zero in another or, in the medical domain, the prevalence or mortality of a disease can vary depending on the gender or the location (Zirra et al., 2023; Menotti et al., 2023).

The problem involves calculating the probability of a given hypothesis (Query) based on a provided

RuleBERT	RuleTaker-pro
(Fact 1) David is a cousin of Ann. (Fact 2) Mike is a child of Ann. (Rule 1, 0.90) If A is a spouse of B and C is a child of B, then C is a child of A. (Rule 2, 0.15) If A is a cousin of B, then A is a spouse of B.	(Fact 1) Dave is big. (Fact 2) Erin is sad. (Rule 1) Usually, If someone is big then they are green. (Rule 2) Normally, If someone is green then they are round. (Rule 3) Seldom, If someone is sad then they are round.
(Query) Mike is a child of David.	(Query) Dave is round.
Required Steps of Reasoning to Answer	
Fact 1 (1.00) & Rule 2 (0.15) $\implies$ Fact 3: David is a spouse of Ann. (0.15) (Inferred) Fact 3 (0.15) & Fact 2 (1.00) & Rule 1 (0.90) $\implies$ Fact 4: Mike is a child of David. (0.135) (Inferred) **Answer: 0.135**	Fact 1 (1.00) & Rule 1 (0.90) $\implies$ Fact 3: Dave is green. (0.90) (Inferred) Fact 3 (0.90) & Rule 2 (0.80) $\implies$ Fact 4: Dave is round. (0.72) (Inferred) **Answer: 0.72**
Approach: Converting Probabilistic Reasoning Steps to Equality Constraints	
Constraint 1: P(Fact 1) * 0.15 = P(Fact 3) Constraint 2: P(Fact 3) * P(Fact 2) * 0.90 = P(Fact 4)	Constraint 1: P(Fact 1) * 0.90 = P(Fact 3) Constraint 2: P(Fact 3) * 0.80 = P(Fact 4)

Table 1: An example from RuleBERT with two facts and two rules is shown in the left column, and an example from RuleTaker-pro with two facts and three rules is shown in the right column. The steps of reasoning required to infer the Query and the constraints defined on these steps are shown in the bottom rows.

context that includes *textual description* of probabilistic logical rules and facts. Table 1 shows examples of our datasets and their required reasoning steps to answer the Query. We convert the reasoning steps to equality constraints (shown in the Approach section of Table 1) and impose these constraints to ensure consistency of the outputs with the rules during the training of PLMs. In summary, our contributions are as follows:
1) We evaluate both PLMs and LLMs and demonstrate that fine-tuned PLMs outperform more costly LLMs in probabilistic reasoning over text. 2) We propose a new neuro-symbolic approach, Probabilistic Constraint Training (PCT), that explicitly imposes the rules of probabilistic reasoning during PLM fine-tuning. This approach provides a more effective level of abstraction to the models to generalize and transfer reasoning under uncertainty to new domains and the more complex depths of reasoning. 3) We develop a novel evaluation benchmark for probabilistic reasoning over text with context-specific uncertain rules that can not be captured from the training data.[1]

2 Related work

Previous works mostly looked into the integration of certain logic (Saha et al., 2020; Tafjord et al., 2021). The pioneering work on QA with probabilistic rules in text is RuleBERT (Saeed et al., 2021), which serves as the baseline for our comparative study. While RuleBERT pioneers this field and introduces Weighted binary cross-entropy loss to incorporate probabilistic learning in transformers, it lacks a mechanism to follow the probabilistic reasoning steps explicitly. Additionally, our experiments revealed that the rules in textual form in this dataset are not properly utilized by the models (see Section 4.1), which prompted us to introduce RuleTaker-pro with instance-specific rules.

Reasoning Steps. Explicit elucidation of reasoning steps in QA models has been central in recent literature. (Saha et al., 2020) improves PLMs' reasoning by mapping their output to an inference graph, necessitating the model to learn its nodes and edges. While (Tafjord et al., 2021) utilized T5 to create an inference path, this and similar studies have focused on using non-probabilistic logical rules, unlike our approach. (Weber et al., 2019; Rocktäschel and Riedel, 2017) defines an end-to-end differentiable neural network architecture for probabilistic reasoning over entities in text. (Wang and Deng, 2020; Polu and Sutskever, 2020; Tafjord et al., 2021) approach the reasoning for QA by generating an output that follows a predefined formal language for theorem proving given the logical rules, which is a very different approach from ours. (Wu et al., 2023) introduces reasoning in LLMs by generating intermediate reasoning steps as extra output. However, we enable PLMs to incorporate this reasoning in training with no additional output.

Reasoning QA Datasets. Numerous QA datasets require reasoning including (Weston et al., 2016;

[1]The code and dataset will be available after anonymity.

Tafjord et al., 2019; Tandon et al., 2019), (Yang et al., 2018), ROPES (Lin et al., 2019) and, FOLIO (Han et al., 2022). However, they lack an explicit definition of the logical rules or are not probabilistic. (Frieder et al., 2023) is created to assess the mathematical reasoning of LLMs. However, it has no tasks with probabilistic logical rules. **Constraints.** Our approach's primary contribution is incorporating probabilistic constraints in the loss function. While various studies incorporate logical constraints into the loss function (Nandwani et al., 2019; Li et al., 2019; Asai and Hajishirzi, 2020; Ribeiro et al., 2019; Faghihi et al., 2023; Guo et al., 2020), no work has explored the application of probabilistic constraints in this context to date. Our methodology, PCT, builds on (Nandwani et al., 2019), where logical rules are translated into a soft logic form before inclusion in the loss function. PCT leverages this approach by utilizing the rules of probabilistic reasoning as constraints.

3 Approach

3.1 Problem definition

We focus on the challenge of performing probabilistic logical reasoning within a QA task where a set of facts F, a set of rules R, and a hypothesis h are provided in a **textual** context. While these rules, facts, and hypothesis are provided **only** in their **textual form** as a part of the input to the task, we have their formal information as a part of the data. For example, fact $Big(Dave)$ and the rule $Spouse(A,B)$ & $Child(C,B) \rightarrow Child(C,A)$ would be conveyed to the PLMs as: "Dave is big.", and "If A is a spouse of B and C is a child of B, then C is a child of A.", respectively. The facts and hypothesis consist of factoids that define properties for an entity "Has_Property(Entity)" or relations between two entities "Relation(Entity1, Entity2)". The rules have the form $(p_1, p_2, ..., p_n) \rightarrow q, Pr$, where p_i represents a premise fact, q is a new inferred fact, and Pr is the rule's probability. q's probability is computed as the rule probability multiplied by the premise facts probabilities. If the premise facts are mentioned in the context, they would be certain and have 1.00 probability; otherwise, if they are inferred facts, their probability is derived. The objective is to utilize F and R to infer a probability between 0 and 1 as our task output, which indicates the probability of a given hypothesis h (e.g., h="Sara and John are cousins" obtains a probability of 0.20 by the model).

3.2 Base Model

The backbone of our model is RoBERTa Large, supplemented by two linear layers and a sigmoid activation function applied to its classifier token (CLS). The model takes the context and hypothesis as input, subsequently assigning a probability to the given hypothesis. More precisely, the input sequence is formatted as [CLS] text(R)+text(F) [SEP] text(h) [SEP], where the context comprises the textual representations of rules and facts, denoted by text(R) and text(F) respectively, and text(h) represents the textual form of the hypothesis.

Our LLMs are GPT3.5 and GPT4 (Brown et al., 2020). Due to the high cost of fine-tuning LLMs, we limit our experiments to zero-shot and few-shot. Input comprises a task explanation, text(R)+text(F), and text(h). The explanation instructs the model about the objective and output format, which is either "True","False" (corresponding to a probability greater or less than 0.5), or a number between 0.0 and 1.0 (hypothesis probability).

3.3 Probabilistic Constraints Training

We aim to develop a model capable of following probabilistic reasoning steps to infer a hypothesis probability. These reasoning steps for the examples in Table 1 are outlined in the *Required Steps of Reasoning to Answer* row. In each step, a combination of facts and a rule results in a new *intermediate inferred fact* until the final hypothesis is inferred. These steps are formulated as constraints, and our proposed model is trained to adhere to them by incorporating them into the loss function. The "Approach" row of Table 1 shows examples of the reasoning steps' conversion into constraints in which the probabilities assigned to facts must follow the rule definition. For instance, if Fact 1 and Rule 1 result in a new fact, Fact 1's probability (P(Fact 1) multiplied by Rule 1's probability must be equal to the inferred fact's probability (P(inferred Fact)). In the upcoming subsections, we will explain the process of formulating constraints, the approach of utilizing them, and the inference procedure.

3.3.1 Constraint Conversion

Among the research focused on constraint integration within neural models, we opt for the class of methods that incorporate constraint violation in the loss function during training without altering the model's architecture (Faghihi et al., 2023). In general, to employ the logical and symbolic constraints in deep models, they must be converted into soft logic for the sake of differentiability. Usually, three main approaches are used for this conversion: Product, Gödel, and Łukasiewicz (Li et al., 2019). For instance, the logical rule, $(p_1, p_2, ..., p_n) \rightarrow q$, using the Product surrogate, is written as follows,

$$min(1, P(q)/[P(p_1) * P(p_2) * ... * P(p_n)]), \quad (1)$$

where $P(p_i)$ is the probability of the fact p_i. We can express the enforcement of this implication's truth as follows,

$$|1 - min(1, P(q)/[P(p_1) * P(p_2) * ... * P(p_n)])| = 0, \quad (2)$$

where |.| denotes the absolute value. Adopting this approach, we formulate the probabilistic reasoning as obeying a set of constraints. Our constraints originate from the required computations of probabilistic inference, assuming a particular underlying probabilistic network. We distinguish between *Simple* and *Complex* probabilistic reasoning patterns based on their underlying inference network.

A probabilistic reasoning pattern is *Simple* if any deducible fact can be drawn from it via only a single reasoning path. The examples provided in Table 1 are *Simple* because "Dave is round." can be inferred only from Rule 2 and Fact 3 and Fact 3 can only be inferred from Fact 1 and Rule 1. On the other hand, a *Complex* reasoning encompasses at least one fact that can be deduced from two or more different rules (reasoning paths). By altering the second fact from "Erin is sad" to "Dave is sad", we create a *Complex* example because it enables inference of "Dave is round" from Fact 2 and Rule 3 as well. Only 20% of the examples of our datasets are of the *Complex* type. Hence, our focus lies primarily on formulating the simple version of probabilistic reasoning for defining constraints. The *Complex* examples are still incorporated in our datasets and used during training and testing.

Given a *Simple* network, our model executes probabilistic inference for the rule $(p_1, p_2, ..., p_n) \rightarrow q, Pr$ by multiplying the premise facts' probability with the rule's probability to obtain the inferred fact's probability. Formally, the model should fulfill the constraint,

$$|P(q) - P(p_1) * P(p_2) * ... * P(p_n) * Pr| = 0. \quad (3)$$

Our unique definition of constraint constitutes the key novelty of our approach (see Table 1 for Examples of constraints). To satisfy this constraint, the equation's left side should approach zero. Note that while this constraint guarantees adherence to the rules of probabilistic reasoning, it might not ensure the best results on the end task accuracy, and this remains subject to experimentation.

3.3.2 Training

Inspired by (Nandwani et al., 2019), we employ constraints during training without adding architectural overhead. To generate the constraints for each dataset instance, we use the chains of probabilistic reasoning that include the paths of inference for every inferable fact (these are available in the dataset; see section 3.4). The constraints follow the format of Equation 3 and will be used during training in the loss function. Examples of these constraints can be found at the bottom of Table 1. Our training objective centers on minimizing the violation of these constraints. We denote the violation from each constraint as C_i, a scalar value that ranges from 0 to 1, derived from the left-hand side of Equation 3. We initiate the process with warm-up iterations on the original QA task to train the model. Following this, we continue the training while adding the constraint violation losses to the primary loss. As per the methodology outlined in (Nandwani et al., 2019), we apply the dual formulation of the objective as follows,

$$Loss = TaskLoss + \sum_{i=1}^{m} \lambda_j * C_i, \quad (4)$$

where "TaskLoss" denotes the primary task loss aiming to minimize the predicted probability error for the hypothesis. The new additional term is the constraint violation loss used in its dual form with Lagrangian multipliers, λ_j, where j is the index of rule j used in constraint violation i (C_i). m is the number of selected constraints. λ_j is adjusted during training and ultimately indicates a rule's propensity to violation. Consequently, as training progresses, the loss function predominantly impacts the rules with the highest accumulated λ_j. See Appendix A.6 the detailed training algorithm.

3.3.3 Inference

During the inference, the model receives the context that includes textual rules and facts, while the formal rules and constraints that were employed during training are not available to the model. We expect the model to learn to obey the rules that were utilized in the loss function during training. This critical aspect ensures the model's generalizability and transferability across various domains.

3.4 Datasets

RuleBERT. RuleBERT (Saeed et al., 2021) is built using about 100 rules with fixed probabilities that are applied to many examples in the dataset. The fixed probabilities of these rules are extracted from an external source. The number of rules used in a chain to derive the answer determines the depth of reasoning, ranging from 0 to 5. The probability of all possible inferred facts and the depth for each instance is given in the dataset as metadata.

RuleTaker-pro. We developed RuleTaker-pro as a probabilistic variant of RuleTaker (Clark et al., 2020), modifying its crisp logical rules $(p_1, p_2, ..., p_n) \rightarrow q$ (with Pr equal to 1.0) to include probabilities while the rest of the context remains unchanged (examples shown in the right side of Table 1). We leverage a Gaussian random generator to produce probabilities. After assigning probabilities to the rules, we use Problog (De Raedt et al., 2007), a probabilistic logical inference tool that facilitates the encoding of probabilistic facts and rules, to compute the probability of the hypothesis. The resulting rules are similar to RuleBERT rules $(p_1, p_2, ..., p_n) \rightarrow q, Pr$. In the textual form, we include the rule's probability as an adverb of uncertainty like *Usually*, *Normally*, and *Seldom* with associated probabilities of 0.90, 0.80, and 0.15, respectively. As mentioned, a key difference between RuleTaker-pro and RuleBERT is including instance-specific rules. For example, the rule "If A is a cousin of B, then A is a spouse of B." from RuleBERT will always have the probability of 0.15 in all the examples. However, in Ruletaker-pro, the same rule may hold different probabilities depending on the adverb assigned to it in different instances. A rule such as "Usually, if someone is big, then they are green." carries a probability of 0.90 in one context, while "Seldom, if someone is big then they are green." carries a probability of 0.15 in some other context. Given this difference, the model has to extract the rules from each context and can not use the information learned about the rules from the training data. It is notable that ambiguity and cycles are already removed from the RuleTaker dataset for the logical rules and are not an issue in our dataset, as confirmed by our Problog solver. Metadata about the inference of all facts and their depths are in the dataset and will be used to create constraints but not during training or inference. See Appendix A.1 and A.5 for details of data creation and distribution and the adverbs.

4 Experiments

In this section, we address three questions using RuleTaker-pro and RuleBERT datasets: Q1. How do textual rules affect probabilistic reasoning (4.1)? Q2. To what extent does the baseline language model improve with PCT concerning probabilistic reasoning and intermediate inferred facts (4.2)? Q3. Can we transfer the probabilistic reasoning capabilities of the language model when pre-trained with PCT(4.3)? We also present an ablation study to investigate the impact of various losses and datasets on our approach using multiple metrics.

Evaluation Metrics. We use several performance measures following (Saeed et al., 2021). Binary Accuracy (BA) deems predictions correct if ground truth and predicted probability both fall under or over 0.5. The CA25, CA10, and CA1 require the predicted probability to be in a window of ± 0.25, ± 0.10, and ± 0.01 of the ground truth, respectively. (Saeed et al., 2021) applies CA10 and CA1 metrics to dataset splits with isolated rules, while BA is used for all reasoning depths for datasets involving all the rules. For comparison, we use BA for RuleBERT, but we thoroughly evaluate RuleTaker-pro using all relevant criteria. We use an extra metric, CS, to measure soft **Constraint Satisfaction** that deems the constraint (defined in Equation 3) satisfied if the following inequality holds:

$$|P(q) - P(p_1) * P(p_2) * ... * P(p_n) * Pr| < Threshold. \quad (5)$$

This means that the difference between the predicted and calculated probability of an inferred fact, based on premise facts, must be less than a threshold: 0.01 (CS1), 0.10 (CS10), or 0.25 (CS25).

4.1 Effect of Rules in Textual Format

Firstly, we investigate whether RoBERTa utilizes the rule's text of the RuleBERT dataset by keeping and removing them from the context. For example, if we remove the rule's text in Table 1, only Facts 1 and 2 will form the input. We report the results of these two settings in Table 2, where columns indicate the maximum depth of reasoning in training (M1-M5), and rows correspond to the reasoning depth of testing (D1-D5). We omit M0 as depth 0 does not use any rules, making it irrelevant to our investigation of PCT. We observe that the accuracy improves across most models and depths when the rules' text is excluded, suggesting that RoBERTa is *not using* it, and including it may even add unnecessary complexity. Thus, we conjecture that RoBERTa can implicitly learn the probabilities of these rules from the facts and hypothesis in training data alone without using rules' text.

D/M	M1	M2	M3	M4	M5
D1	76.95	79.82	79.92	70.74	64.99
D*	76.84	82.02	82.27	83.60	82.19
D2	77.59	77.88	76.69	70.44	65.41
D*	75.40	78.86	78.22	80.02	78.53
D3	78.47	76.93	76.2	78.80	71.64
D*	77.93	80.65	80.69	82.85	80.63
D4	76.22	73.42	72.40	78.20	73.86
D*	75.06	76.20	77.21	79.62	77.02
D5	77.15	73.06	69.68	77.52	78.16
D*	78.42	75.22	78.76	79.69	76.79

Table 2: BA results of RoBERTa fine-tuned on RuleBERT. * indicates exclusion of rule's texts.

	CE (CA1)				MSE (CA1)			
D/M	M1	M2	M3	Mmax	M1	M2	M3	Mmax
Total	38.21	**38.34**	20.45	33.89	30.39	32.26	26.17	26.04
D1	**56.03**	52.71	29.63	43.77	50.46	49.48	38.15	37.26
D2	36.40	**38.28**	20.31	32.87	26.49	31.13	25.52	28.43
D3	29.30	**31.30**	14.98	28.39	18.81	22.06	18.98	19.90
D4	27.49	**28.53**	14.03	27.11	18.54	21.37	17.79	17.32
D5	24.97	26.78	14.70	**28.29**	19.83	21.14	19.33	15.50
CS1	47.88	35.79	16.22	20.78	25.24	14.88	14.47	12.97
	CE (CA10)				MSE (CA10)			
D/M	M1	M2	M4	Mmax	M1	M2	M3	Mmax
Total	46.45	49.69	49.95	53.25	58.15	62.75	66.67	74.80
D1	61.56	59.55	53.41	56.17	91.76	84.45	80.94	**82.81**
D2	45.76	52.08	52.42	51.59	53.60	69.97	77.25	**77.32**
D3	38.88	44.87	48.37	51.45	42.88	51.20	61.44	**71.60**
D4	37.75	42.45	47.36	51.97	38.04	46.51	51.50	**69.39**
D5	33.63	38.67	43.80	53.07	32.62	37.05	43.30	**63.64**
CS10	52.24	44.97	35.67	38.25	45.13	34.49	32.86	33.34

Table 3: The accuracy of the **baseline** models trained and tested on the RuleTaker-pro dataset. The rows show different test depths (depths 1 to 5). Total indicates the weighted average accuracy of all depths, and CS* shows the constraint satisfaction at the indicated thresholds. The best results for each depth are in bold.

Our baseline differs from (Saeed et al., 2021). This discrepancy arises from our approach of freezing 22 transformer layers for faster training and more fine-tuned hyper-parameters, which yield superior accuracy at higher depths (We use the same loss function, Weighted binary cross-entropy). Moreover, we also train our models with (Saeed et al., 2021) original setting, and again, the text of the rules did not yield any positive impact on the performance (see Appendix A.3 for details).

The baseline results for the RuleTaker-pro dataset are shown in Table 3. The models (in the columns) are trained with maximum depths 1, 2, 3, and 5 (max), as these are the depths provided in the original RuleTaker training data. However, the testing is done on all depths 1 to 5. CS averages over all depths. The table also includes the models trained with different loss functions: Cross-Entropy (CE) and Mean Square Error (MSE). Weighted binary cross-entropy was abandoned due to underperformance on RuleTaker-pro. We use CA1, CA10, and BA metrics as in (Saeed et al., 2021)[2].

Though MSE excels in CA10, CE outperforms MSE in CA1 and BA. The MSE CA10 accuracy drops sharply at higher depths, especially for the models trained at lower depths. Considering MSE's low CS1, we conjecture this sharp decline results from the minor MSE approximation errors at lower depths, magnified at higher depths when multiplied along the chain of probabilities. Given our goal of achieving exact inference probabilities following the path of reasoning, CA1 is a more relevant measure for PCT evaluation. These results indicate transformers can capture probabilistic reasoning patterns to some extent, especially when examples with the same test depth are observed in training.

Unlike RuleBERT, RuleTaker-pro uses example-specific rules, requiring the rule's text to determine the answer. Without rules, our model's predictions are not better than random guesses. In RuleTaker-pro, we initially generated probabilistic rules by including the probability in the text, such as "With the probability of 15%, if someone is green, then they are sad". However, we also considered using adverbs of uncertainty (Farkas et al., 2010) instead of numbers, changing the rule to "Seldom, if someone is green, then they are sad". Adverbs of uncertainty improved the models in Dev BA by 0.5%-2%, thus we followed this approach in RuleTaker-pro creation (see Appendix A.1[3]).

LLM Results To evaluate LLMs, we add instructions and examples (for few-shot settings) to their prompts. The LLM results for RuleTaker-pro are shown in Table 4. We observe that even GPT3.5 with few-shot examples and GPT4 fall short of RoBERTa's accuracy. The gap in accuracy becomes even wider in CA10, where the accuracies remain almost the same as in CA1. This indicates that if the LLM cannot predict the exact probability, its prediction will not be even close to the correct answer. LLMs will be undermined even more after we add PCT and improve RoBERTa's results. The results of LLMs on RuleBERT dataset are as bad as a random baseline (see AppendixA.9 for details of

[2]BA, MSE, and L1 results are in the Appendix A.7 due to the lack of space as we focus mainly on CA1 and CA10.

[3]Dev BA results are shown in Table 10.

prompt instructions and RuleBERT results). While LLMs are capable of probabilistic logical reasoning on RuleTaker-pro, RoBERTa still outperforms them. Given the high cost of utilizing these models, it is still best to fine-tune smaller transformers.

	CA1				CA10			
	RoBERTa	GPT3.5	GPT3.5*	GPT4	RoBERTa	GPT3.5	GPT3.5*	GPT4
D1	44	28	41	41	56	33	45	43
D2	33	20	26	27	52	28	36	37
D3	28	23	25	26	51	25	34	34
D4	27	18	20	17	52	21	33	29
D5	28	18	20	21	53	22	31	29

Table 4: LLM results on RuleTaker-pro for CA1 and CA10 metrics. * indicates using few-shot examples. The chosen RoBERTa model is M5 trained with CE since it performs the best regarding CA1.

4.2 Effectiveness of PCT

Table 6 displays PCT's effect on improving RuleBERT's accuracy over the baseline results in both settings (with and without rules' text), especially at deeper depths. Using PCT, the CS25 accuracy of intermediate inferred facts increases from an average of 50% to over 90%. Increasing the constraint satisfaction of intermediate inferred facts works synergistically with the model's accuracy by compelling the model to reason, thus, enhancing the model, especially at deeper depths. Appendix A.4 includes more details of inferred intermediate facts.

By deploying PCT in RuleTaker-pro, we observe a similar trend to RuleBERT. As illustrated in Table 5, by incorporating exact probabilities into the constraints, PCT improves the accuracy of CA1 for both MSE and CE in most models. MSE without PCT did not learn the exact probabilities (it learned their approximation). Another place where PCT shows improved generalization is when it is used to train the models at lower depths, i.e., 2 and 3, and tested at higher depths. This shows that the reasoning learned with PCT is transferred to higher depths. However, at depth 1, due to the limited number of applicable constraints, the change in accuracy is minor. Overall, the combination of CE and PCT achieves the highest accuracy at CA1 (further analysis in Section 4.3 about this). In CA10, MSE still achieves the best results, and PCT only improves CE for M2 and M3 tested on higher depths. Similar to RuleBERT, we observe a sharp increase of about 50% in the CS in all the models trained with PCT.

Error Analysis. Our findings indicate that improvements in constraint consistency are not always proportionate to improvements in accuracy. This discrepancy is prevalent in nearly all tasks involving constraints, as evidenced by related studies (Ribeiro et al., 2019). Notably, to maintain the consistency of outputs, the model might yield incorrect results. Incorporating PCT encouraged the model to output lower probabilities than the baseline model, thus reducing the magnitude of the constraint loss. For instance, in the model trained at depth 3 with PCT, the average output probabilities for all the test dataset questions declined from a baseline of 52% to 45%. When the model is trained with depth 1 with PCT, the constraint satisfaction decreases, likely due to its reduced ability to accurately process questions with a higher reasoning depth. In short, while the best results are achieved when both CS and CA increase, a high CS does not invariably guarantee a corresponding increase in CA. See Appendix A.8 for detailed examples.

4.3 Transferability Analysis

Experiments in Section 4.2 highlighted the effectiveness of PCT in transferring *reasoning* from a model trained at lower depths to answer questions at higher depths. Here, we evaluate the transferability of PCT from different perspectives.

Transferring Reasoning From Simple to Complex Examples. As highlighted in Section 3.3.1, 20% of the inference questions in RuleTaker-pro are of the *complex* type that are both included in our dataset during training and testing. Tables 7 and 8 present our models' performance on *simple* and *complex* questions separately, with the models predictably faring better on the former. Employing CE+PCT increases accuracy for both question types, making the difference between them negligible. This suggests that the models can do probabilistic reasoning even in *complex* instances. However, for MSE and MSE+PCT models, the performance difference between question types remains substantial. The CE+PCT model's enhanced ability to learn probabilistic reasoning results from PCT teaching exact probabilities. MSE model does not see the same benefit due to cascading errors in the approximated probabilities, as discussed in Section 4.2. However, in the case of MSE, adding PCT still improves accuracy.

Domain Transfer. We evaluated the transferability of the probabilistic reasoning and constraint satisfaction capabilities to another domain by training our model on RuleTaker-pro with CE+PCT and

	CE+PCT (CA1)				MSE+PCT (CA1)			
D/M	M1	M2	M3	Mmax	M1	M2	M3	Mmax
Total	.0(-0.21)	39.5(+1.2)	**41.1(+20.7)**	37.6(+3.7)	**37.4(+6.9)**	34.7(+2.5)	36.4(+10.3)	34.3(+8.3)
D1	**53.37(-2.66)**	50.8(-1.9)	50.5(+20.9)	46.9(+3.1)	**56.50(+6.04)**	49.8(+0.3)	52.6(+14.5)	37.6(+0.3)
D2	37.44(+1.04)	40.4(+2.1)	**42.2(+21.8)**	37.0(+4.2)	35.99+(9.05)	34.2(+3.1)	**38.1(+12.6)**	33.8(+5.4)
D3	26.47(-2.83)	32.9(+1.6)	**36.0(+21.1)**	32.4(+4.0)	25.97(+7.16)	25.8(+3.8)	26.5(+7.5)	**32.6(+12.7)**
D4	26.55(-0.94)	31.9(+3.4)	**33.9(+19.9)**	31.8(+4.7)	24.10(+5.56)	25.5(+4.1)	24.9(+7.1)	**31.6(+14.3)**
D5	23.37(-1.6)	30.4(+3.6)	**33.4(+18.7)**	31.4(+3.1)	22.05(+2.22)	24.0(+2.8)	24.0(+4.6)	**33.1(+17.6)**
CS1	44.94(-2.94)	42.69(+6.9)	34.55(+18.33)	35.25(+14.47)	20.59(-4.65)	19.34(+4.46)	15.42(+0.95)	13.06(+0.09)
	CE+PCT (CA10)				MSE+PCT (CA10)			
D/M	M1	M2	M3	Mmax	M1	M2	M3	Mmax
Total	38 46.6(+0.15)	50.8(+1.1)	52.5(+2.5)	**52.9(-0.4)**	58.9(+0.75)	63.3(+0.6)	67.2(+0.6)	68.7(-6.1)
D1	59.73(-1.83)	57.8(-1.8)	50.5(-2.9)	**57.9(+1.7)**	**92.41(+0.65)**	82.7(-1.8)	83.5(+2.5)	70.9(-11.9)
D2	47.78(-2.02)	51.4(-0.7)	42.2(-10.3)	**51.8(+0.2)**	57.76(+4.16)	73.2(+3.3)	**76.1(-1.1)**	68.2(-9.2)
D3	39.21(+0.33)	47.4(+2.5)	**50.5(+2.2)**	50.0(-1.4)	41.54(-1.34)	52.2(+1.0)	60.4(-1.0)	**68.6(-3.0)**
D4	36.25(-1.5)	47.0(+4.5)	50.1(+2.7)	**50.4(-1.6)**	36.15(-1.89)	47.3(+0.8)	51.4(-0.1)	**70.0(+0.6)**
D5	35.14(-1.51)	47.0(+8.3)	48.6(+4.8)	**49.8(-3.2)**	34.03(+1.41)	38.1(+1.0)	44.6(+1.3)	**63.8(+0.2)**
CS10	49.79(-2.45)	47.30(+2.33)	45.64(+9.97)	46.89(+8.64)	49.63(+4.5)	36.05(+1.56)	34.90(+2.04)	33.79(+0.40)

Table 5: RuleTaker-pro results trained with PCT. In parenthesis, the change caused by PCT is compared to Table 3.

D/M	M1	M2	M3	M4	M5
D1	78.3(+1.3)	83.1(**+3.2)**	77.5(-2.4)	77.9(**+7.2)**	67.7(**+2.8)**
D*	79.1(+2.3)	81.7(-0.3)	82.4(**+0.1)**	84.1(**+0.5)**	81.1(-1.1)
D2	78.93(+1.4)	79.7(**+1.8)**	76.6(-0.1)	78.0(**+7.5)**	68.9(**+3.5)**
D*	78.5(+3.1)	79.7(**+0.8)**	77.3(-0.9)	80.9(**+0.9)**	77.7(-0.8)
D3	79.1(+0.7)	80.8(**+3.9)**	81.3(**+5.1)**	81.3(**+2.5)**	78.9(**+7.3)**
D*	79.8(+1.9)	83.4(**+2.8)**	81.9(**+1.2)**	86.2(**+3.3)**	82.2(**+1.6)**
D4	77.7(+0.5)	77.0(**+3.6)**	79.0(**+6.6)**	80.8(**+2.6)**	82.6(**+8.7)**
D*	77.4(+2.4)	81.4(**+5.2)**	80.2(**+3.0)**	85.1(**+5.4)**	81.3(**+4.3)**
D5	77.8(-0.7)	74.1(**+1.6)**	84.1(**+14)**	82.7(**+5.2)**	88.6(**+10)**
D*	80.1(+1.6)	84.3(**+9.1)**	84.3(**+5.5)**	86.1(**+6.5)**	83.6(**+6.8)**

Table 6: PCT accuracy of RuleBERT with the change caused by PCT shown inside the parenthesis. * indicates exclusion of rule's texts.

CA1	CE			CE+PCT		
Model	M2	M3	Mmax	M2	M3	Mmax
Simple	39.16	20.87	34.21	41.17	40.13	37.86
Complex	34.11	18.33	32.30	36.66	38.02	36.47

Table 7: RuleTaker-pro results on Simple and Complex examples trained with Cross Entropy.

CA1	MSE			MSE+PCT		
Model	M2	M3	Mmax	M2	M3	Mmax
Simple	33.77	27.45	27.17	36.14	37.99	35.01
Complex	24.50	19.60	20.23	27.49	28.40	30.76

Table 8: RuleTaker-pro results on Simple and Complex examples trained with Regression.

Mode	CE+PCT			CE		
	M2	M3	M5	M2	M3	M5
D2	+7	+8	+7	-1	+4	+18
D3	+8	+6	+1	-1	+7	+10
D4	+11	+5	0	-3	+4	+1
D5	+13	-3	+3	-5	+3	-11
CS25	+3	+14	+7	0	+2	-1

Table 9: Improvements in the binary accuracy (BA) and constraints satisfaction of RuleBERT models in Table 2 after transfer learning from RuleTaker-pro. Transfer learning results are shown for a model trained on RuleTaker-pro with CE+PCT on the left and CE on the right of the table.

fine-tuning it on RuleBERT. This transfer direction is selected due to the RuleTaker-pro model's superior constraint satisfaction. Results are presented in Table 9, with the left side detailing the improvements in both BA and CS25, confirming the usefulness of RuleTaker-pro as a base model. The right side of the table shows the effects of excluding PCT to ensure the improvements are not the result of increased data alone. In this scenario, only lower-depth results showed improvement, while higher depths and CS remained unaffected.

5 Conclusion and Future Work

Addressing the problem of reasoning over uncertain rules in textual format, we create a new dataset, RuleTaker-pro, extending the limited resources for studying this issue. We investigate how uncertain rules can be represented in the text and used by the models. We propose a novel approach that explicitly uses the rules of probabilistic reasoning as constraints in the loss. This approach improves the performance and reasoning of the backbone language models. Our experiments on LLMs have revealed that they struggle to perform probabilistic reasoning in zero-shot and few-shot scenarios, despite their impressive capabilities. Our future objective is to develop models that utilize the text of the rules more effectively and transfer their reasoning abilities to more realistic QA domains featuring uncertainty and more advanced structures of probabilistic reasoning.

6 Limitations

One limitation of our work is the fixed structure of the rules in our datasets, which limits the model's transferability to other domains with more open forms of explaining probabilistic rules. Another limitation is that we take a small step to formalize probabilistic reasoning over text. However, this does not mean the outcome language models are fully capable of language understanding and reasoning. Finally, running our models, based on RoBERTa large while possible, is computationally expensive, limiting their usage with all our different settings. This is exacerbated when it comes to utilizing Large Language Models that, in their current state, are very expensive to use even in zero-shot and few-shot settings.

References

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Wenhu Chen. 2023. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *CoRR*, abs/2002.05867.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, page 2468–2473, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hossein Rajaby Faghihi, Aliakbar Nafar, Chen Zheng, Roshanak Mirzaee, Yue Zhang, Andrzej Uszok, Alexander Wan, Tanawan Premsri, Dan Roth, and Parisa Kordjamshidi. 2023. Gluecons: A generic benchmark for learning under constraints.

Richárd Farkas, Veronika Vincze, György Móra, János Csirik, and György Szarvas. 2010. The CoNLL-2010 shared task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning – Shared Task*, pages 1–12, Uppsala, Sweden. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, and Julius Berner. 2023. Mathematical capabilities of chatgpt.

Quan Guo, Hossein Rajaby Faghihi, Yue Zhang, Andrzej Uszok, and Parisa Kordjamshidi. 2020. Inference-masked loss for deep structured output learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2754–2761. International Joint Conferences on Artificial Intelligence Organization. Main track.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. 2022. Folio: Natural language reasoning with first-order logic.

Tao Li, Vivek Gupta, Maitrey Mehta, and Vivek Srikumar. 2019. A logic-driven framework for consistency of neural models. 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 3924–3935, Hong Kong, China. 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.

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. *CoRR*, abs/1907.11692.

Alessandro Menotti, Paolo Emilio Puddu, Hanna Tolonen, and Anthony Kafatos. 2023. Cardiovascular mortality in northern and southern european cohorts of the seven countries study at 60-year follow-up. *Journal of Cardiovascular Medicine*, 24(2):96–104.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523, Dublin, Ireland. Association for Computational Linguistics.

Yatin Nandwani, Abhishek Pathak, Mausam, and Parag Singla. 2019. A primal dual formulation for deep learning with constraints. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Engineering National Academies of Sciences, Medicine, Division of Behavioral, Social Sciences, Education, and Committee on the Science of Science Communication: A Research Agenda. 2017. *Communicating Science Effectively: A Research Agenda*. National Academies Press (US), Washington (DC). Copyright 2017 by the National Academy of Sciences. All rights reserved.

Jiaxin Pei and David Jurgens. 2021. Measuring sentence-level and aspect-level (un)certainty in science communications. *CoRR*, abs/2109.14776.

Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *CoRR*, abs/2009.03393.

Marco Tulio Ribeiro, Carlos Guestrin, and Sameer Singh. 2019. Are red roses red? evaluating consistency of question-answering models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6174–6184, Florence, Italy. Association for Computational Linguistics.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. *CoRR*, abs/1705.11040.

Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. 2021. RuleBERT: Teaching soft rules to pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1460–1476, Online and Punta Cana, Dominican Republic. 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.

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.

Niket Tandon, Bhavana Dalvi, Keisuke Sakaguchi, Peter Clark, and Antoine Bosselut. 2019. WIQA: A dataset for "what if..." reasoning over procedural 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 6076–6085, Hong Kong, China. Association for Computational Linguistics.

Mingzhe Wang and Jia Deng. 2020. Learning to prove theorems by learning to generate theorems. In *Advances in Neural Information Processing Systems*, volume 33, pages 18146–18157. Curran Associates, Inc.

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

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv: Artificial Intelligence*.

Dingjun Wu, Jing Zhang, and Xinmei Huang. 2023. Chain of thought prompting elicits knowledge augmentation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6519–6534, Toronto, Canada. 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. Zirra, S. C. Rao, J. Bestwick, R. Rajalingam, C. Marras, C. Blauwendraat, I. F. Mata, and A. J. Noyce. 2023. Gender differences in the prevalence of parkinson's disease. *Movement Disorders Clinical Practice*, 10(1):86–93.

A Appendix

A.1 RuleTaker-pro

In creation of RuleTaker-pro, we utilize 8 different adverbs of frequency shown in the Table 11. Using adverbs of frequency improved the Dev binary accuracy consistently in all depths. The results are shown in Table 10.

CE	M1	M2	M3	Mmax
With adverbs	96.24	94.97	93.12	89.71
With explicit probabilities	95.78	93.71	92.52	88.01

Table 10: Dev BA for models M1 to Mmax trained with CE loss.

In order to make a balanced dataset with an equal number of labels, we generate a random probability for each rule based on a Gaussian random generator. Then the adverb with the closest probability to the generated probability is chosen. The rule probability generations are generated so that half of the answers are above and half are below 0.50.

The algorithm to change a logical context to a probabilistic one is shown in Algorithm 1. "FIND_ADVERB" function gets a random probability from 0 to 100 as input and returns an adverb to it based on the closest probability of an adverb in Table 11. In the procedure "ADD_PROBABLITIES", a logical context and question are given as input. Then, in line 6, it is randomly decided whether or not the final answer to this instance should be above or below 0.50 to ensure balance in the final results of the dataset. In the rest of the algorithm, until the pre-selected above or below 0.50 probability for the answer is achieved, random probabilities would be assigned to the rules in the context. The random function that assigns these probabilities is a Gaussian function with a mean of 40 and std of 60. the random probabilities are added with the value h, initially set to $depth * 10$, and it increases or decreases slightly to help achieve the desired answer after reaching failure. h is created based on the depth of the dataset group to create a balanced average of answer probabilities. See section A.5 for the final statistics about the created dataset and their mean final answer. Also, a realistic example of the created dataset is shown and analyzed in section A.8.

Algorithm 1 Assigning Gaussian-based probabilities to logical rules to crate a probabilistic dataset while ensuring that the resulting dataset is balanced with heuristics.

1: **function** FIND_ADVERB(x)
2:     Determine adverb and its associated probability based on the range of x in Table 11
3:     **return** adverb
4: **end function**

5: **procedure** ADD_PROBABLITIES(c, q, d) $\triangleright$ c is context, q is question and d is the depth of the dataset group (not the instance)
6:     $Above0.50 \leftarrow$ RANDOM$(False, True)$
7:     $h \leftarrow 10 * depth$
8:     **while** not Answer is $Above0.50$ **do**
9:         $new\_c = c$
10:         **for** each rule in $context$ **do**
11:             p_i = RANDOMGAUSS(40,60)+h
12:             $adverb =$ FIND_ADVERB(p_i)
13:             add $adverb$ to new_c
14:         $Answer \leftarrow$ PROBLOG(new_c, q)
15:         **if** $Above0.50$ **then**
16:             $h = h + 5$
17:         **else**
18:             $h = h - 5$
19: **end procedure**

Adverbs	always	usually	normally	often	sometimes	occasionally	seldom	never
Probability	1.00	0.90	0.80	0.65	0.50	0.30	0.15	0.0

Table 11: The adverb of uncertainty and their respective probabilities that we link to them.

A.2 ProbLog

ProbLog is a tool that allows us to encode probabilistic facts and rules. Then it will calculate any queries in the context of the defined facts and rules, which is exactly what we need for RuleTaker-pro. For example, Table 1's right column would be shown in Problog pseudo code in the Figure 1a.

```
Input:

Dave_is_big .
Erin_is_sad .
0.90:: Green :- Big .
0.80:: Round :- Green .
0.15:: Round :- Sad .

query(Dave_is_round).
query(Erin_is_round).

Output:

[(Dave_is_round, 0.72),
(Erin_is_round, 0.15)]
```

(a) Encoding of the Table 1's right column example in ProbLog pseudo code.

```
Input:

Dave_is_big .
Dave_is_sad .
0.90:: Green :- Big .
0.80:: Round :- Green .
0.15:: Round :- Sad .

query(Dave_is_round).

Output:

[(Dave_is_round, 0.762)]
```

(b) Encoding of the Table 1's right column example in ProbLog pseudo code if the second fact is replaced with "David is sad."

A more complicated example would occur when there is more than one way to reach an inferred intermediate fact. Imagine that the second fact in the example of Table 1's right column is "David is sad.". In that case, the probability that "David is round" would be 0.762 as shown in Figure 1b.

A.3 Training RoBERTa with RuleBERT's Original Setting

The original RuleBERT baseline from (Saeed et al., 2021) is shown in Table 12. We also train our models with their settings, both with and without including the text of the rules. These new results are shown in Table 13. The text of the rules is still not useful for the models.

A.4 CA25 Accuracy of Intermediate Inferred Facts

CA25 Intermediate Inferred Facts for M5 is depicted in Figure 2. The model is trained for 6 epochs to show the accuracy over time. PCT accuracy remains consistently over 0.90 while the baseline models accuracy fluctuates and remains below 0.60.

	M1	M2	M3	M4	M5
D1	86.0	88.4	88.7	88.9	88.9
D2	65.5	73.0	75.1	75.0	72.0
D3	58.1	63.6	68.4	69.0	65.6
D4	46.8	54.7	62.6	66.6	62.7
D5	35.6	49.6	70.3	78.5	74.4

Table 12: RuleBERT baseline results trained and tested on different depths (Saeed et al., 2021).

D/M	M1	M2	M3	M4	M5
D1	76	91	87	91	93
D1*	88	90	88	92	89
D2	76	87	79	83	83
D2*	87	88	77	78	74
D3	67	85	76	76	73
D3*	84	85	73	72	67
D4	66	82	69	63	51
D4*	82	80	65	60	51
D5	53	75	54	34	28
D5*	80	68	44	29	21

Table 13: M shows the maximum depth of the training data, and D shows the depth of the test data. Rows with * indicate the exclusion of the text of the rules.

A.5 RuleTaker-pro depth and data distribution

Statistics about the splits, their unique context and questions, and their balanced average answer produced by our algorithm are shown in Table 14.

RuleTaker-pro depth distribution for all depths and the number of True and False labels are shown in Table 15.

A.6 Training Parameres

The PCT algorithm pseudo-code is shown in Algorithm 2. Lines 2-4 apply the taskloss, and lines 5-13 apply constraints loss and update the λ_j. The rate at which λ_j is updated depends on *PCT variable* (α) decayed at each iteration's end.

To train RuleTaker-pro, we use RoBERTa Large for four epochs with a learning rate of $1e-5$. When

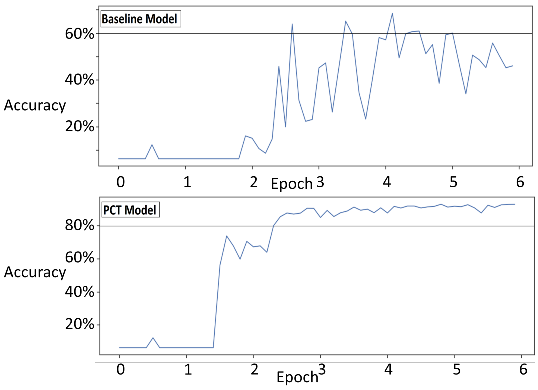

Figure 2: The CS25 of intermediate inferred facts over 6 Epochs of training for M5.

Split	Depth	Total Row	Unique Query	mean Answer
Train	1	13549	807	0.49
Train	2	16145	810	0.48
Train	3	19960	812	0.48
Train	5	23805	812	0.50
Dev	1	1946	551	0.50
Dev	2	2290	586	0.48
Dev	3	2837	629	0.48
Dev	5	3412	694	0.50
Test	1	3930	690	0.49
Test	2	4592	718	0.48
Test	3	5687	765	0.48
Test	5	6829	789	0.50

Table 14: RuleTaker-pro Dataset Statistics

we use PCT, the alpha (PCT variable) varies from 1.0 to 0.001 depending on the depth of the training dataset with higher depths training with smaller alphas.

To train RuleBERT, we also use RoBERTa Large for four epochs, but we freeze the first 22 layers of the transformer. The learning rate varies between numbers $1e-6$ for higher depth datasets with more examples and $2e-6$ for lower depth datasets. When using PCT, the alpha is 0.01 for lower depths (1-3) and 0.001 for higher depths (4-5). In Table 16, the effect of alpha on the PCT Dev BA is shown. As shown, a higher alpha will help the model reach higher accuracy earlier. However, the best result is achieved with an alpha of 0.01.

A.7 Additional RuleTaker-pro Results

In Table 17, The binary results for RuleTaker-pro trained with MSE and CE is shown.

A.8 Error Analysis Examples

We analyze an example shown in Figure 3 that benefited from PCT. Initially, the base model predicted 0.50 for the final answer, which was incorrect, as

Algorithm 2 PCT algorithm

for each batch in data **do**
  Apply model on batch to get the logits
  Calculate Taskloss (CE/MSE/L1loss)
  Backward propagate the loss
  if Not warm-up iteration **then**
    Get the next constraints batch
    Apply model on constraints batch
    $cl \leftarrow 0$ ▷ initialize constraints loss
    for each constraint **do**
      $l \leftarrow abs(q - p_1 \times p_2 ... \times p_n \times Pr)$
      $cl \leftarrow cl + l \times \lambda_j$
      $\lambda_j \leftarrow \alpha \times l$
    Backward propagate the cl
  Take optimizer step,and Reset gradients
  decay α

the answer should have been 0.85. After training the model using PCT, the model correctly predicted 0.85. This demonstrates the potential of the PCT model for incorporating additional constraints in the inference process. However, it should be noted that this is an ideal case that may not always be reproduced in practice. The PCT model can be adapted to alter the probability of the depth2 fact to satisfy the constraint if needed. In other scenarios, the model may keep the 0.50 prediction for depth 3 and change the prediction for depth 2. In this case, the model satisfies the constraint, yet the final prediction is incorrect. In the worst case, the model may predict 0.0 for all elements and still satisfy the constraint.

It has been observed that the predicted probabilities of the PCT models are lower on average than those of the baseline models. This is due to the fact that lower predicted probabilities make it easier to satisfy the constraints, and thus, even models that improve overall accuracy tend to have lower average predicted probabilities.

A.9 LLM prompt instructions and additional results

To effectively evaluate LLMs like, we adjust our approach with our datasets to make them suitable for zero-shot and in-context settings for generative models. These adaptations involved adding a text explaining the task before the context. For RuleBERT, we use the following explanation, *"Answer the following logical probabilistic question with only one word, True or False."* and add the proba-

	D0 True	D0 False	D1 True	D1 False	D2 True	D2 False	D3 True	D3 False	D4 True	D4 False	D5 True	D5 False
M1 Training Dataset	10626	10719	6422	6452	0	0	0	0	0	0	0	0
M2 Training Dataset	9590	9485	4613	4465	3441	3469	0	0	0	0	0	0
M3 Training Dataset	7441	7650	4438	4272	2930	2949	2597	2642	0	0	0	0
Mmax Training Dataset	2616	2720	3802	3692	2442	2520	2118	2026	1852	1858	1761	1734

Table 15: RuleTaker-pro depth distribution for all depths and the number of True and False labels.

Depth3	Epoch1	Epoch2	Epoch3	Epoch4	Epoch5	Epoch6
Baseline	49	70	77.95	75.85	70.925	72.62
PCT with $\alpha = 0.1$	49	**79.15**	78.42	76.9	77	64.51
PCT with $\alpha = 0.01$	49	79.32	**80.87**	79.32	78.17	78.57
PCT with $\alpha = 0.001$	49	70.90	78.55	**80.85**	78.55	78.75

Table 16: Accuracy obtained using PCT during training with different hyper-parameter (α) for depth 3 of reasoning for 6 epochs on RuleBERT dataset. Normally we train our models for 4 epochs, but here we use 6 epochs to observe the learning process better.

	CE Loss				MSE Loss			
BA	M1	M2	M3	Mmax	M1	M2	M3	Mmax
Total	76.93	82.65	88.74	91.05	76.19	84.84	87.73	91.39
D1	97.19	94.85	92.18	93.39	97.28	95.92	92.64	94.33
D2	75.58	89.11	91.26	91.26	74.41	90.91	91.88	91.74
D3	68.19	77.35	89.42	91.00	42.88	81.93	88.59	90.34
D4	65.16	71.35	84.93	88.70	38.04	74.38	81.82	89.17
D5	58.61	65.05	80.96	88.31	57.70	66.96	76.43	88.21
MSE	M1	M2	M3	Mmax	M1	M2	M3	Mmax
Total	0.1574	0.1278	0.0965	0.0716	0.4693	0.6585	0.6298	0.0716
D1	0.0866	0.0996	0.1076	0.0983	0.1992	0.0173	0.0190	0.0983
D2	0.1939	0.1261	0.1065	0.0876	0.1902	0.0257	0.0247	0.0876
D3	0.2352	0.1826	0.1149	0.0818	0.1915	0.0698	0.0313	0.0818
D4	0.2511	0.2003	0.1281	0.0710	0.1910	0.0982	0.0423	0.0797
D5	0.3082	0.2436	0.1428	0.710.	0.1963	0.1237	0.0618	0.0710
L1	M1	M2	M3	Mmax	M1	M2	M3	Mmax
Total	0.2505	0.2216	0.1903	0.1664	0.3628	0.1055	0.0798	0.1664
D1	0.2004	0.2138	0.2236	0.2175	0.3693	0.0525	0.0581	0.2175
D2	0.3118	0.2434	0.2243	0.2076	0.3638	0.0770	0.0786	0.2076
D3	0.3528	0.3010	0.2316	0.1972	0.3642	0.1570	0.1032	0.1972
D4	0.3672	0.3182	0.2443	0.1960	0.3659	0.2090	0.1326	0.1960
D5	0.4136	0.3519	0.2495	0.1761	0.3737	0.2480	0.1627	0.1761

Table 17: The Binary accuracy, MSE and L1 of the baseline model trained and tested on the RuleTaker-pro dataset at different depths.

bility of the rules to their text. For RuleTaker-pro, we use *"Answer the following logical probabilistic question in the format .##, which is the probability of the question asked rounded to 2 decimals, for example, .13%"*. After this text, we provide the context and pose the hypothesis as a question.

To test RuleBERT in LLMs, we included the probability of the rules in the text; Otherwise, the model has no way of extracting them. The results are shown in Table 18.

Model	GPT3.5	GPT3.5*	GPT4
Depth1	19%	43%	29%
Depth2	58%	53%	46%
Depth3	58%	58%	60%
Depth4	51%	56%	46%
Depth5	56%	43%	58%

Table 18: RuleBERT BA results are show for GPT3.5 and GPT4. * indicates few-shot setting.

Context: The cow is round. Always, If something is nice and round then it does not visit the lion. The mouse visits the cow. The rabbit does not see the cow. The lion is round. The rabbit is big. The cow likes the rabbit. The lion likes the rabbit. Always, If something is big and it does not see the rabbit then it visits the mouse. The mouse is green**. Usually, If something visits the lion then it visits the mouse**. Always, if something is green, then it visits the lion. Always, if the rabbit is big, then the rabbit is green. Hypothesis: The rabbit visits the mouse. (Depth 3), P3 = 85% Required Intermediate Facts: The rabbit visits the lion. (Depth 2), P2 = 100% The rabbit is green. (Depth 1), P1 = 100% The rabbit is big. (Depth 0), P0 = 100%
Base model: The rabbit visits the mouse. (Depth 3) , P3 = 50% The rabbit visits the lion. (Depth 2), P2 = 100% Constraint: P2*85% ≠ P3 (violated Constraint and Incorrect Answer)
PCT Model The Ideal Case: The rabbit visits the mouse. (Depth 3), P3 = 85% The rabbit visits the lion. (Depth 2), P2 = 100% Constraint: P2*85% = P3 (Satisfied Constraint and Correct Answer)
PCT Model The Problem Case: The rabbit visits the mouse. (Depth 3), P = 50 % The rabbit visits the lion. (Depth 2), P = 59 % Constraint: P2*85% = P3 (Satisfied Constraint and Incorrect Answer)
PCT Model The Worst Case: The rabbit visits the mouse. (Depth 3), P = 0 % The rabbit visits the lion. (Depth 2), P = 0 % Constraint: P2*85% = P3 = 0 (Satisfied Constraint and Incorrect Answer)

Figure 3: In the given example, the fact "The rabbit visits the lion." can be inferred from the context with a probability of 1.00 at depth 2. Both the base model and the PCT model accurately predicted the probability of this fact. However, only the PCT model took into account the additional bold rule in the text, which led to an 0.85 probability for the hypothesis.