

Transformers as Soft Reasoners over Language

Peter Clark, Oyvind Tafjord and Kyle Richardson

Allen Institute for AI, Seattle, WA

{peterc,oyvindt,kyler}@allenai.org

Abstract

Beginning with McCarthy’s Advice Taker (1959), AI has pursued the goal of providing a system with explicit, general knowledge and having the system reason over that knowledge. However, expressing the knowledge in a formal (logical or probabilistic) representation has been a major obstacle to this research. This paper investigates a modern approach to this problem where the facts and rules are provided as natural language sentences, thus bypassing a formal representation. We train transformers to reason (or emulate reasoning) over these sentences using synthetically generated data. Our models, that we call RuleTakers, provide the first empirical demonstration that this kind of soft reasoning over language is learnable, can achieve high (99%) accuracy, and generalizes to test data requiring substantially deeper chaining than seen during training (95%+ scores). We also demonstrate that the models transfer well to two hand-authored rulebases, and to rulebases paraphrased into more natural language. These findings are significant as it suggests a new role for transformers, namely as limited “soft theorem provers” operating over explicit theories in language. This in turn suggests new possibilities for explainability, correctability, and counterfactual reasoning in question-answering.¹

1 Introduction

AI has long pursued the goal of giving a system explicit *knowledge*, and having it *reason* over that knowledge to reach conclusions, dating back to the earliest years of the field, e.g., McCarthy’s Advice Taker (1959), and Newell and Simon’s Logic Theorist (1956). While this has resulted in impressive applications (e.g., [Metaxiotis *et al.*, 2002]), building and reasoning over the required formal representations has also proved challenging [Musen and Van der Lei, 1988]. In this work, we explore a modern approach to this goal, and ask whether transformers can be trained to reason (or emulate reasoning) using rules expressed in language, thus bypassing a

¹ A live demo and all our datasets are available at <https://allenai.org/data/ruletaker>

(Input Facts:) Alan is blue. Alan is rough. Alan is young.
 Bob is big. Bob is round.
 Charlie is big. Charlie is blue. Charlie is green.
 Dave is green. Dave is rough.

(Input Rules:) Big people are rough.
 If someone is young and round then they are kind.
 If someone is round and big then they are blue.
 All rough people are green.

Q1: Bob is green. True/false? [Answer: T]
 Q2: Bob is kind. True/false? [F]
 Q3: Dave is blue. True/false? [F]

Figure 1: Questions in our datasets involve reasoning with rules. The inputs to the model are the context (facts + rules) and a question. The output is the T/F answer to the question. Here the underlying reasoning for the true fact (Q1) is: Bob is big, therefore rough (rule1) therefore green (rule4). Note that the facts + rules themselves change for different questions in the datasets.

formal representation. If so, new opportunities for question-answering, explainability, correctability, and counterfactual reasoning may become possible.

This goal is quite distinct from question-answering as selecting an answer span in a passage, today’s prevailing paradigm, e.g., [Rajpurkar *et al.*, 2016]. Rather, we want the system to reason over the provided rules to find conclusions that follow. Our goal is also distinct from that of *inducing* rules from examples, e.g., given instances of family relationships, inducing that a parent’s parent is a grandparent [Sinha *et al.*, 2019], something that transformers are already known to do well. Rather, here we provide rules explicitly, and wish transformers to draw appropriate conclusions, as illustrated in Figure 1. Here, rather than inducing rules from examples, our task involves learning to emulate a reasoning *algorithm*.

We provide the first demonstration that this is possible, i.e., that transformers can reason with rules expressed in language. Our approach uses a broadly applicable training regimen: Characterize the desired behavior in a formal way, synthesize formal examples, generate linguistic equivalents, and train a model. The result suggests a new role for transformers, namely as a kind of limited “soft theorem prover” over language (Figure 2). This in turn may allow inspection and control of the knowledge that the model is manipulating, with

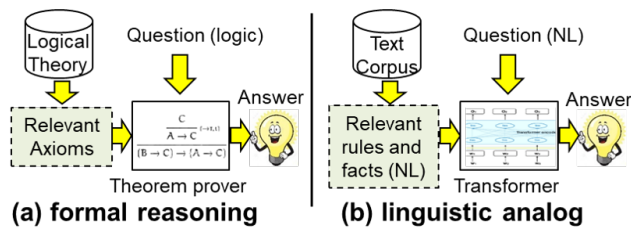


Figure 2: (a) Traditional formal reasoning applies a theorem prover to axioms in order to answer a question. (b) Our work here strives for a linguistic analog, where a transformer serves as a “soft theorem prover” over knowledge expressed linguistically.

potential benefits for explanation, correctability, and counterfactual reasoning.

Our investigations here are in a limited setting: Rules are linguistic expressions of conjunctive implications *condition* $[\wedge \text{condition}]^* \rightarrow \text{conclusion}$, with the semantics of logic programs with negation [Apt *et al.*, 1988]; and reasoning is the deduction of a statement’s truth according to these semantics. However, although there is still a potentially large gap to natural language inference (NLI),² our approach also suggests a path to teaching machines to reason over broader language, with similar potential benefits.

We leave open the question of whether the transformer is actually “reasoning”, and even what that might mean in a neural setting. Rather, we show that transformers can reliably emulate the *i/o* behavior of a formal reasoner, including applied to test data requiring more reasoning than at training time, two hand-authored rulebases, and rulebases rephrased into more natural (crowdsourced) language.

The paper is organized to address the following questions, and contributes the following results:

1. **Can transformers learn to reason with rules?** We train and test on rules expressed in (synthetic) language, and find high (99%) accuracy, including on test questions requiring a greater depth of reasoning than seen during training (scoring up to 95%, Table 1).
2. **Can the trained model solve hand-authored reasoning problems?** We find the trained models are able to solve five of six variants of two independently authored rule-based problems, zero shot (90%+ scores, Table 4).
3. **Do the results transfer to theories expressed in more natural language?** Models also perform well when trained and tested on theories paraphrased into more natural (crowdsourced) language (98% score). The best earlier model can even partially solve these problems zero-shot (66% accuracy, Table 5).
4. **Can the model identify which facts an answer depends on?** We show that the model is largely able to do this (94% F1), including perfect identification for over 70% of the questions. This is a first step towards having a model create an explanation for its conclusions. (Sec-

² NLI is informally defined as making inferences from language that “a person would typically infer” [Dagan *et al.*, 2013], and includes use of many linguistic forms, unstated background knowledge, and sometimes unsound inference steps.

tion 4.5 and Figure 8).

5. **Can other neural architectures learn to reason?** Our experiments show a particular transformer (RoBERTa) is sufficient for our tasks, but is it necessary? We show that two other systems, BERT and ESIM (an LSTM-based model) [Chen *et al.*, 2017], are also able to learn these tasks, albeit with lower scores (95% and 80% respectively, vs. 98%). This suggests that our results are not specific to RoBERTa or transformers, although transformers learn the tasks more easily (Table 6).

2 Related Work

While our work is, to the best of our knowledge, the first systematic study of transformers directly reasoning with rules in language, there are several datasets that make a first step towards this by testing whether neural systems can apply a single rule in a particular situation. Task 15 in the bAbI dataset [Weston *et al.*, 2016] tests whether a rule of the form “Xs are afraid of Ys” can be correctly applied, e.g., “Sheep are afraid of wolves. Gertrude is a sheep. What is Gertrude afraid of? A:wolves”. Similarly, the synthetic, conditional probes in [Richardson *et al.*, 2020] test single rule application. In addition, the datasets QuaRTz [Tafjord *et al.*, 2019] and ROPES [Lin *et al.*, 2019] involve applying general statements to a situation, but also require many other reading comprehension skills, rather than specifically testing reasoning.

Although our core datasets may seem similar to the bAbI dataset [Weston *et al.*, 2016] in using synthetic data, our probes are qualitatively different. Specifically, apart from bAbI Task 15 (above), the underlying rules needed to infer an answer in the bAbI tasks are *implicit*, while our concern here is reasoning with explicit rule sets, potentially different for each example (Figure 1).

Our approach contrasts with prior efforts that attempt to semantically parse language into a formal form, so that a formal reasoner can then be applied [Kamath and Das, 2019]. Despite substantial research, semantic parsing remains challenging, with few examples of systems that can reliably convert multi-sentence text into formal theories. Instead, we explore reasoning with language directly, bypassing the semantic parsing task.

Our work can be seen as evaluating transformers for (a subset of) Natural Logic [MacCartney and Manning, 2014], i.e., formal inference over statements expressed in language. It is also related to textual entailment and Natural Language Inference (NLI) [Manning and MacCartney, 2009], but with the important difference that NLI also allows *unsupported* inferences that “a person would typically infer” [Dagan *et al.*, 2013]. We discuss bridging the gap between our work and NLI in Section 5.3.

Several researchers have developed methods for Neural Theorem Proving (NTP), combining symbolic and neural methods to reason step-wise over language-derived structures, e.g., [Weber *et al.*, 2019]. Similarly, there has been work on SAT solving [Selsam *et al.*, 2019], approximate (DNF) model counting [Abboud *et al.*, 2020], and formula embedding [Abdelaziz *et al.*, 2020] to help solve formal reasoning problems. While our goals are similar, we do not im-

pose any structure on the neural reasoning process, instead wanting to know if the (i/o of the) reasoning process itself is learnable, using knowledge expressed in language.

Our task can perhaps best be viewed as one of *algorithm emulation*, here for systematic reasoning with rules. There have been numerous other demonstrations that transformers either already know [Talmor *et al.*, 2019; Richardson and Sabharwal, 2019] or can learn to emulate other algorithms, including for semantic parsing [He and Choi, 2019], machine translation [Wang *et al.*, 2019], integration [Lample and Charton, 2019], and math [Saxton *et al.*, 2019]. Here we investigate a transformer’s ability to learn rule-based reasoning.

3 Dataset Generation

To investigate a transformer’s ability to emulate rule-based reasoning, we generate five datasets requiring various depths of inference to answer the questions. Each example in a dataset is a triple (*context,statement,answer*), where *context* has the form (*fact*,rule**), *statement* is the question, namely a declarative sentence to prove, and *answer* is either T (true) if *statement* deductively follows from the context, or F if it does not (false under a closed-world assumption, CWA). Facts, rules, and the question statements are expressed in (synthetic) English. Each example is essentially a (linguistic) standalone logical theory with an “Is it true?” question posed against it.

3.1 Overview

To generate each example, we first generate a small theory (facts + rules) in logic, perform forward inference to derive all its implications, then select question statements from those implications (answer=true), and from unproven (positive) facts (answer=false, under the CWA). We generate five datasets, each constrained by the maximum depth of inference required to prove the facts used in its questions (up to depths $D=0, D\leq 1, D\leq 2, D\leq 3$ and $D\leq 5$ respectively). Depth $D=0$ means the true facts can be “proved” by simple lookup in the context (no inference). The fifth dataset, called DMax, contains questions up to depth 5, and is used to test generalization to depths unseen in training on the other four datasets.

3.2 Theory Generation

Theories contain two types of facts:

- attributes $is(e_i, a_j)$ e.g., $is(Alan,Big)$.
- relations $r_k(e_i, e_k)$ e.g., $eats(Dog,Rabbit)$.

The $is()$ predicate assigns attributes to entities, while the $r_k()$ predicates relate two entities. Like people names, the symbols Dog, Rabbit, etc. also denote specific entities, i.e., denote “the dog”, “the rabbit”, etc. Rules are of the form:

$$condition [\wedge condition]^* \rightarrow conclusion.$$

The first *condition* is a predicate whose first argument is a variable,³ and second argument is an attribute or entity. For each subsequent *condition* and the *conclusion*, they are also predicates whose first argument is either the same variable or a previously mentioned entity, and the second argument is a

³ Or with 20% probability, an entity, in order to include some fully grounded rules in the datasets.

The bald eagle does not eat the dog. The cat chases the dog.
 The cat eats the bald eagle. The cat is nice. The cat likes the dog.
 The cat likes the rabbit. The dog is furry.
 The rabbit chases the bald eagle. The rabbit eats the bald eagle.
 If someone does not eat the cat then they do not eat the dog.
 If someone likes the bald eagle then they do not like the rabbit.
 If someone eats the bald eagle and they do not eat the rabbit
 then they are furry.
 If someone is furry then they like the cat.
 Q1. The bald eagle likes the cat. True/false? [F]
 Q2. The rabbit likes the cat. True/false? [T]
 Q3. The bald eagle is furry. True/false? [F]

Figure 3: An example of a rulebase and 3 questions using relations with negation. The reasoning for the [T] answer is: The rabbit eats the bald eagle (given), therefore the rabbit is furry (rule3), therefore the rabbit likes the cat (rule4).

new attribute or entity. (In this way, rules are constrained to have at most one variable. Rules are implicitly universally quantified over that variable). For example, the formal form of the first rule in Figure 1 looks:

$$// \text{If someone is young and round then they are kind.} \\ is(?X,Young) \wedge is(?X,Round) \rightarrow is(?X,Kind).$$

Each theory contains 1-16 facts and 1-9 rules generated at random. We generate two types of theory:

1. Type 1 uses only the $is()$ predicate, with 4 entities {Alan,Bob,...} and 7 (non-mutually-exclusive) attributes {Blue,Rough,Young,...}, drawn randomly from pools of 10 names and 14 attributes respectively.
2. Type 2 uses $is()$ and 3 other predicates {likes(), chases(), ...}, 4 entities {Cat,Dog,BaldEagle,...}, and 5 attributes {Big,Furry,...}, drawn randomly from pools of size 6, 10, and 10 respectively.

We also generate a version of each that adds negation (not) in the facts and rule conditions/conclusions (negation-as-failure for conditions, strong negation for conclusions). Figure 1 is an example of Type 1, without negation. Figure 3 is an example of Type 2, with negation. Each dataset contains 100k examples (25k of each Type \times without/with negation). Data is randomly split 70/10/20 into train/dev/test partitions, ensuring no overlap of theories between each partition.

3.3 Forward Inference

Given a randomly generated theory (facts+rules), we perform exhaustive forward inference to find all its implications, noting their proof(s). (As the domains are finite, the number of implications are finite too). For semantics, we treat the rulebase as a logic program, and infer the minimal, supported answer set implied by the program [Apt *et al.*, 1988]. Negations in the rules’ conditions are treated as negation as failure (NAF), and we ensure that the rulebase is stratified to avoid ambiguity and cycles [Bidoit and Froidevaux, 1991]. Inference is performed layerwise to find the minimal supported model, and inconsistent and unstratified rulebases are discarded. We also check that inference proceeds to the depth required, e.g., for the $D\leq 3$ dataset, at least one fact must require depth 3 inference to infer it for all its theories.

3.4 Question Generation and English Synthesis

For each theory, we generate several questions with answer ‘true’ by selecting from the inferred facts, one at each depth of inference from 0 to the dataset’s target depth (e.g., for the $D \leq 2$ dataset, we generate 3 ‘true’ questions at depths $d = 0, 1,$ and 2 for each theory). For each ‘true’ question we also generate a ‘false’ question by negating a conclusion proven at the same depth. We then generate the same number of questions using facts that are unproven (false under a closed-world assumption), drawing equally from unproven, instantiated positive rule conclusions or other unproven positive facts. Half are used as questions labeled as false (via the CWA), and for diversity, half are flipped by negating the fact and changing the label to true (i.e., “ $f?$ False” becomes “Not $f?$ True”). Thus a theory for depth d has (up to) $4(d+1)$ questions, with an equal balance of true and false answers. Each question is also annotated with the inference depth needed to answer it.

Finally the theories and questions are converted into (synthetic) English, using simple natural language templates plus rules to improve fluency (e.g., using pronouns). We use three templates (randomly selected per rule): “If *condition* [and *condition*]* then *conclusion*.”, “All *attribute** people|things are *attribute*.”, and “*attribute** people|things are *attribute*.”, the last two only applicable to rules involving just attributes. Examples are shown in Figures 1 and 3.

4 Experiments

4.1 Models

We conduct all our experiments (bar Section 4.6) using RoBERTa-large, additionally fine-tuned on the RACE dataset [Lai *et al.*, 2017]. We use fixed hyperparameters (learning rate etc), inheriting the settings from RoBERTa on RACE [Liu *et al.*, 2019].

We train RoBERTa to predict true/false (i.e., binary classification) for each question statement. Questions are supplied to RoBERTa as: $[CLS]$ context $[SEP]$ statement $[SEP]$, where *context* is the theory (facts+rules, expressed in language) and *statement* is the fact to try and prove. The $[CLS]$ output token is projected to a single logit. A logit score of >0 is treated as predicting true, otherwise the answer is false. Training is performed using cross-entropy loss. For evaluation, we measure accuracy. (The test data has an equally balance of TRUE/FALSE answers, hence the baseline of random guessing is 50%).

4.2 Can RoBERTa Answer Reasoning Questions?

We train and test RoBERTa models on each of our datasets $D=0, D \leq 1, D \leq 2, D \leq 3,$ and DMax, containing problems requiring reasoning up to depths 0, 1, 2, 3, and 5 respectively. We then test the models on the DMax dataset, that includes problems at depths greater than the other datasets. The results are shown in Table 1. The results suggest the following findings:

1. RoBERTa is able to **master the test data almost perfectly** (99% accuracy, row 1) even though the specific reasoning problems (facts+rules) in each test question are distinct from those in the training set.

Training	Num Q	Mod0 $D = 0$	Mod1 $D \leq 1$	Mod2 $D \leq 2$	Mod3 $D \leq 3$	MMax
Test (own)	~ 20000	100	99.8	99.5	99.3	99.2
Test (DMax)	20192	53.5	63.5	83.9	98.9	99.2
Depth=0	6299	100	100	100	100	100
Depth=1	4434	57.9	99.0	98.8	98.5	98.4
Depth=2	2915	34.3	36.8	98.8	98.8	98.4
Depth=3	2396	20.4	23.1	71.1	98.5	98.8
Depth=4	2134	10.2	11.4	43.4	98.8	99.2
Depth=5	2003	11.2	12.3	37.2	97.6	99.8

Out-of-distribution tests (reasoning depth unseen in training)

Table 1: Accuracy of models (Mod0,...) trained and tested on the five datasets (“Test (own)” row), and tested on all, and different slices, of the DMax test set. The boxed area indicates test problems at depths unseen during training.

2. The Depth=0 model, Mod0, only trained on lookup questions, is (unsurprisingly) **unable to answer questions requiring reasoning** (column Mod0).⁴
3. As we train with increasingly deep inference, the models’ ability to generalize improves. The $D \leq 2$ model (questions involving problems up to depth 2) achieves 71.1% on Depth=3 problems, while **the $D \leq 3$ model generalizes well** right up to the maximum depth tested (e.g. 97.6% for Depth=5 problems).

We additionally test the robustness of the models’ answers by perturbing the original theories. Specifically, for each test fact f that is true, we test whether removing a sentence that is part of the proof of f causes the prediction to (desirably) flip from true to false. We call these sentences in the proof tree *critical sentences*, as the truth of f depends on them. Conversely, removing an *irrelevant* sentence should cause no change to the model’s prediction. As we know the original proof trees for each fact f in the dataset, we can identify the critical and irrelevant sentences by simple inspection of those trees.⁵ Typically, 1-6 sentences of the $\approx 15-20$ sentences are critical for proving each provable fact.

We test this using the no-negation⁶ half of the DMax test set ($\approx 10k$ questions). In this partition, 5904 questions have proofs (are true). (The remaining questions are false under the CWA). For each of these questions, we remove each of the theory sentences s_i in turn, and measure the prediction accuracy on each result. As there are about 19 sentences/theory on average, this results in 113978 “sentence removed” probes (of which 20746 have a critical sentence removed, and 93232 have an irrelevant sentence removed). Ideally, removing a sentence critical to a question f should flip the model’s pre-

⁴ In fact, we see an interesting learning artifact, namely Mod0 scores worse than random (50%) at depths higher than 2. This arises because most questions at these depths are provably true facts, but Mod0 learns to predict all facts are false except those explicitly given (as that is all it has seen at training time), hence systematically gets these wrong.

⁵ If there are multiple, alternative proofs for f , we define a critical sentence as one that is used in *all* the proofs. To support this, we generate and record all possible proofs for each provable fact f

⁶ With negation, the definition of critical sentence becomes more complex because the the theory is non-monotonic (i.e., *removing* a sentence may cause a fact to become *true*). Hence, we omit theories with negation for this analysis.

	Original	Remove Irrelevant	Remove Critical	Remove Any
Accuracy (test)	99.4	99.6	81.2	96.3

Table 2: Accuracy on the DMax (no negation) subset, and all its (113k) perturbed (one context sentence removed) variants. The overall accuracy (Remove Any, last column) is largely unchanged, but with a drop for the subset where a critical sentence was removed.

		Original predictions for true (positive) facts:	
		T	F
New	T	3895 (should have flipped)	10 (incorrectly flips)
Pred.	F	16654 (correct flips)	187 (becomes correct)

Table 3: On the true questions that were originally answered correctly (column 1), the predicted T answer should flip to predicted F when a critical sentence is removed. In practice, we observe this happens 81% of the time (16654/(16654+3895)).

diction from T to F, while removing a noncritical sentence should leave the prediction unchanged as T. We also measure overall performance on the entire dataset of questions with perturbed theories.

The results are shown in Tables 2 and 3. We observe:

1. The **overall accuracy is largely unchanged** on the full collection of questions with perturbed theories, suggesting robustness to these variants (last column, Table 2).
2. For the (20k) questions where the prediction is expected to flip from true to false, we see this flip occurs 81% of the time, Table 3. This suggests **moderate robustness** to this specific type of perturbation, although notably less than for a formal theorem prover (that would make this flip 100% of the time). For the remaining (93k) questions, the prediction (correctly) stays true over 99% of the time (no Table).

4.3 Performance on Hand-Authored Problems

To further test robustness and out-of-distribution performance, we test the trained models on two hand-authored reasoning problems, both including reasoning with negation, written independently of our datasets. Note that these new datasets are used purely as test sets (no training on them, i.e., zero-shot performance); their vocabulary of entities, attributes, and predicates (except for *is()*) are all new to the models at test time. The two test datasets are as follows:

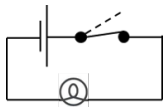
Birds. The “birds” rulebase is a well-known logic problem illustrating the use of “abnormality” predicates [McCarthy, 1984]. We entered Sergot’s formulation of it⁷ verbatim (bar syntax), and generated a series of test questions using the same procedure as earlier. Figure 4 illustrates the problem (in restricted English, exactly as presented to our model) and four example questions. We created two linguistic expressions of the formal theory, Birds1 and Birds2. Birds2 is shown in Figure 4, while Birds1 is identical except “can/cannot fly” is replaced with “is/is not flying” to make the negation (“not”) more explicit (this turns out not to matter). Questions require reasoning up to depth 1.

⁷https://www.doc.ic.ac.uk/~mjs/teaching/KnowledgeRep491/ExtendedLP_491-2x1.pdf, p5

If someone is a bird and not abnormal then they can fly.
 If someone is an ostrich then they are a bird.
 If someone is an ostrich then they are abnormal.
 If someone is an ostrich then they cannot fly.
 If someone is a bird and wounded then they are abnormal.
 If someone is wounded then they cannot fly.
 Arthur is a bird. Arthur is not wounded. Bill is an ostrich.
 Colin is a bird. Colin is wounded.
 Dave is not an ostrich. Dave is wounded.
 Q1.Arthur can fly. True/false?[T] Q2.Bill can fly. True/false?[F]
 Q3.Colin can fly. True/false?[F] Q4.Dave can fly. True/false?[F]

Figure 4: Sergot’s “birds” puzzle includes reasoning about abnormality predicates. The dataset contains these and other questions about the single theory.

The circuit has a switch.
 The switch is on.
 The circuit has a light bulb.



If a circuit has a switch and the switch is on then the circuit is complete.
 If a circuit does not have a switch then the circuit is complete.
 If a circuit is complete then a current runs through the circuit.
 If a current runs through a circuit and the circuit has a light bulb then the light bulb is glowing.
 If a current runs through a circuit and the circuit has a bell then the bell is ringing.
 If a current runs through a circuit and the circuit has a radio then the radio is playing.
 Q1. The circuit is not complete. True/false? [F]
 Q2. The light bulb is glowing. True/false? [T]
 Q3. The radio is playing. True/false? [F]

Figure 5: The simple Electricity2 rulebase, an example circuit, and 3 questions about the circuit. (Circuit diagram is for illustration only).

Electricity. We also created a small rulebase about an electrical circuit, describing the conditions for an appliance to function. We created 4 variants of increasing complexity, containing 5, 6, 11, and 12 rules respectively. For each rulebase, we generate different scenarios (the facts) by randomly selecting from possible ground facts. Questions are then generated against each scenario using the same procedure as earlier, resulting in 4 test sets. Figure 5 shows the Electricity2 rulebase with an example scenario plus three questions. Questions against the four rulebases require inference up to depth 2, 3, 3, and 4 respectively.

Results

The results are in Table 4, tested using the earlier trained models. Note that these new problems and vocabularies were unseen during training (i.e., are zero-shot). We observe:

1. The “birds” problems are **solved (almost) perfectly** by all but the non-reasoning (Mod0) model (MMax gets one question wrong on Birds1).
2. The MMax model (trained on DMax) **solves all but one** of these datasets with 90%+ scores.

These are two point demonstrations that the trained models

Test ↓; Train →	Num Q	Mod0 $D = 0$	Mod1 $D \leq 1$	Mod2 $D \leq 2$	Mod3 $D \leq 3$	MMax DMax
Birds1	40	80.0	100	100	100	97.5
Birds2	40	80.0	100	100	100	100
Electricity1	162	77.8	88.9	100	100	96.9
Electricity2	180	70.0	80.0	97.2	100	98.3
Electricity3	624	80.8	93.9	92.8	90.5	91.8
Electricity4	4224	91.9	97.5	93.6	86.0	76.7

All results are zero-shot (these rulebases completely unseen during training)

Table 4: Accuracy of the earlier models tested on hand-crafted rulebases (zero shot, no fine-tuning). Note that the models were *only* trained on the earlier datasets (e.g., Figures 1 and 3), and thus the new rulebases’ entities, attributes, and predicates (bar *is()*) are completely unseen until test time.

can be used to solve novel reasoning problems with high reliability (90%+ in all but one case).

We see one surprising anomaly also: the models trained with deeper reasoning depths do slightly worse on Electricity4 than the depth 1 model, Mod1. From investigation, we find almost all failing questions at higher depths are those where the queried fact f is an unsatisfied rule conclusion (hence should be false), in particular when the first argument of f is not the first argument of one of the rule’s conditions. Because of the way the original dataset was generated, examples similar to this are very rare in the training data, possibly causing this anomaly. More generally this illustrates that even when trained on a diversity of problems, the trained model can have unanticipated blind spots.

4.4 Reasoning with Paraphrased Rules

Our experiments so far have been with synthetic language, but our ultimate goal is to reason over full natural language. To test transfer to more natural linguistic forms, we generated a new dataset of 40k examples, using crowdworkers to paraphrase our theories. Of course, this only tests robustness to paraphrasing, not to arbitrary natural language. Nevertheless, it is a small first step in this direction.

To generate our data, we follow a similar approach to [Sinha *et al.*, 2019]. For this experiment, we used Type 1 theories without negation, i.e., the same form as in Figure 1.

Dataset Generation

To generate the new dataset, called ParaRules, we first generated a novel collection of 10k theories (facts+rules) expressed in synthetic language, as before, then extracted the “fact groups” and rules from each. A “fact group” is all the facts in a theory about a particular person, e.g., (from Figure 1) “Alan is blue. Alan is rough. Alan is young.”, while a rule is just the original “If...then...” sentence. We then asked crowdworkers to creatively re-express the fact-groups and rules, shown to them in English, in their own words. For example, the earlier fact-group might be rewritten as: “Alan is on the young side, but rough. He often feels rather blue.”. Rewritten fact-groups were then turned into templates by variabilizing the person name. Turkers also rephrased each rule (no variabilization needed). Rephrasings were automatically checked to make sure that all the key attributes were mentioned (and no others included), and rejected otherwise.

Alan, who is round, red, kind, and also green, tends to be rather blue. In the snow sits Bob, crying from being cold. Charlie has green teeth and rough skin. People also notice his blue eyes.

A quite nice person who is red and green is also big.
Any big, kind person that turns red is cold to the touch.
Young, kind people have a habit of being nice.
A kind person will certainly be young.

Q1. Dave is nice. True/false? [F]

Q2. Charlie is big. True/false? [F]

Q3. Alan is nice. True/false? [T]

Figure 6: A paraphrased theory in the ParaRules dataset. The reasoning for the true answer here is: Alan is kind (given), therefore young (rule4), therefore nice (rule3).

Training	Mod0 $D = 0$	Mod1 $D \leq 1$	Mod2 $D \leq 2$	Mod3 $D \leq 3$	MMax DMax	Mod3+Para $D \leq 3+Para$
Para test	52.9	60.1	61.4	66.1	66.6	98.8
Depth=0	75.5	86.2	83.2	84.5	85.8	99.8
Depth=1	59.9	69.3	73.1	75.7	73.6	99.3
Depth=2	33.2	34.4	40.7	49.3	48.6	98.2
Depth=3	6.9	8.0	8.4	21.7	26.6	96.7
Depth=4	4.2	6.3	7.0	18.3	25.4	90.1

Zero-shot tests (no fine-tuning on the paraphrased rule set)

Table 5: Accuracy with rules paraphrased into more natural language (ParaRules), without fine-tuning (zero shot) and with (last column only). The strongest zero-shot model (MMax) partially solves (66.6%) this problem zero-shot, with strongest performance for depth 0 and 1 inferences.

We use these to assemble the new ParaRules dataset of 40k questions against $\approx 2k$ theories expressed in the paraphrased language. To build each theory, facts were collected by randomly sampling and instantiating fact-group templates with people’s names, and rules were randomly sampled. An example is shown in Figure 6. The train, dev, and test sets were generated using different partitions of the templates, to ensure that no templates were shared between partitions.

As we kept track of the corresponding logic underlying each fact group and rule, we can then generate questions as before: Exhaustively forward-chain on the (logic version of) the theory, discard if a contradiction is hit or reasoning is of insufficient depth (we require at least depth 3 reasoning), and then for each depth select inferred and non-inferred facts as true/false questions as before.

Results

We ran the earlier trained models on the ParaRules test partition (no fine-tuning, i.e., zero shot). The results are shown in Table 5. The strongest model, MMax, partially solves this dataset with a score of 66.6%, higher for questions requiring less inference, and lower for questions requiring more inference. (The below-random scores for $D=0$ reflect the same artifact as earlier, namely predicting everything as false except for facts explicitly given. See Footnote 4).

Note that these results are for zero-shot, with no model exposure to the paraphrased data during training. In contrast, we also trained a model using *both* of the $D \leq 3$ and ParaRules training partitions. The resulting model (last column Table 5) has an accuracy of 98.8% on ParaRules test (even though the

Statement: The lion visits the rabbit. (TRUE). **Depth:** 2
Context: If something visits the lion then it chases the rabbit. **The lion is red.**
 If something sees the squirrel and the squirrel is young then the squirrel chases the rabbit
 **The lion is cold.** The squirrel sees the rabbit. The rabbit chases the squirrel.
 The lion chases the cat. **Red things are young.** The lion sees the rabbit. The cat is young.
If something is cold and young then it visits the rabbit. The squirrel is big.

Figure 7: In this (abbreviated) example, the model has correctly identified the sentences critical to the answer (shown in green). Perfect identification occurs for over 70% of the provable answers (See Figure 8 for a full histogram).

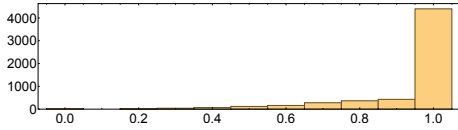


Figure 8: Counts of the F1 scores for predicting which sentences are critical to the proofs of questions in DMax (test, no negation subset). For over 70% of the questions, the model predicts critical sentences perfectly (F1=1.0), with high F1 in the remaining case.

ParaRules test rewordings are distinct from train and dev), showing near-perfect performance is learnable. Although a limited study, this suggests that our findings may extend to rulebases expressed in more natural language.

4.5 Generating Explanations

In Section 4.2, we tested (for the no-negation theories) whether removing a theory sentence s_i caused the prediction for a true fact f to flip to false, and found that sentences causing a flip were very often (98%) part of the original proof of f (i.e., critical sentences), while sentences that did not were not (97%). Using that data about which removed sentences caused a flip, we can build a map of the theory paragraph showing which sentences the model considers critical to a conclusion, a potentially first step to providing an explanation for the model’s answers (see Figure 7).

We can quantify this “explanatory” performance by measuring the per-proof scores of predicted vs. actual critical sentences for each question, measuring the precision, recall, and F1 scores for each question in turn. The (macro)average P/R/F1 scores are P=98.7, R=86.9, and F1=92.4, suggesting a high degree of reliability in predicting sentences critical to a proof. (This is essentially an alternative view on the earlier robustness data, viewed from a per-proof perspective). A histogram of the F1 scores is shown in Figure 8, indicating perfect critical sentence identification for over 70% of the questions, and high F1 for the remaining questions. This suggests the model has some knowledge of the dependencies between the context sentences and a particular conclusion.

4.6 Other Architectures

To what extent are our results specific to RoBERTa? To explore this, we also trained BERT and ESIM (an LSTM-based model for natural language inference) [Chen *et al.*, 2017] on our datasets. As a sanity check we also ran the decomposable attention model (DECOMP) on our data [Parikh *et al.*, 2016]. The results are shown in Table 6.

We observe that the strongest BERT model trained up to depth 3 (Mod3) masters the dataset that includes higher inference depths (DMax) with 95%+ accuracy, while ESIM’s

	Mod0	Mod1	Mod2	Mod3	MMax
Training	$D = 0$	$D \leq 1$	$D \leq 2$	$D \leq 3$	DMax
Test (own):					
RoBERTa	100	99.8	99.5	99.3	99.2
BERT	100	99.3	98.2	97.0	96.9
ESIM	100	90.3	87.8	84.2	80.0
DECOMP	72.5	68.2	58.6	57.8	64.1
Test (DMax):					
RoBERTa	53.5	63.5	83.9	98.9	
BERT	53.5	64.1	90.6	95.3	
ESIM	53.5	66.4	73.2	79.6	
DECOMP	56.5	58.1	56.4	57.4	

(Includes questions at depths unseen during training)

Table 6: Transformers (RoBERTa,BERT) are sufficient but not strictly necessary for this task, although other architectures (ESIM) do not score as well.

scores are lower ($\approx 80\%$). Note that unlike RoBERTa and BERT, ESIM was not pre-trained on large amounts of text, perhaps contributing to its lower scores. This suggests that our results are not specific to RoBERTa or transformers, although transformers seem to learn the tasks more easily. As expected, DECOMP does not do well (random score is 50%), suggesting the datasets are not trivially solvable.

Finally, to explore the role of pretraining, we generated a version of the $D \leq 3$ dataset in which every word was (systematically) replaced by a random word, so that there was no grammaticality in the theories. After training, RoBERTa scores 83.3% on the test partition, substantially below the original 99.3%, suggests that pretrained knowledge is playing an important role.

5 Discussion and Future Work

Although our demonstrations have been in a limited setting, the implications of being able to predictably reason with language are significant. With further advances, we may potentially be able to:

- author theories in English (e.g., Figure 5), thus sidestepping the intricacies of formal languages and offering new opportunities for easy creation and maintenance of knowledge.
- have the machine apply *general* knowledge, e.g., from Wikipedia, to explainably solve novel problems
- teach our AI when it makes a mistake, by providing the missing facts and/or correcting the erroneous ones it used (“instructable systems”).
- reason about counterfactual situations. For example, we might describe a world in which plastic is a type of metal, and see how the conductivity of objects change. This useful capability has previously been out of scope for transformers.

Our RuleTaker models demonstrate these capabilities in a narrow setting. We now discuss additional steps needed to achieve these goals more broadly.

5.1 Extending The Theory Language

While we have shown that transformers can emulate a form of deductive reasoning, our demonstrations have been with

small theory sizes (< 20 facts, < 10 rules), small domains (< 100 possible ground facts), and with a limited rule language (at most one variable that is universally quantified over). Expanding the expressiveness of the rule language would enhance the model’s utility. For example, we have not yet explored using multi-variable rules such as “If a person’s father is a second person, and the second person’s father is a third person, then the first person’s grandfather is the third person,” limiting what can be stated (e.g., rules of transitivity). Similarly there are other forms of reasoning we would like to train the model to handle, e.g., taxonomic inheritance, reasoning with disjunctive conclusions, and handling functional relations (“A country has exactly one capital”). This again requires characterizing the semantics of such statements, and generating training data showing the valid conclusions.

More generally, there are many natural language statements whose formal meaning is less clear (e.g., “Most birds fly”, “It often rains in Seattle in winter.”). To apply our methodology to statements with more complex semantics would require new training data, either synthesized from a richer formal representation and model of inference,⁸ or collected from people.

5.2 Generating Training Data

We assume that our synthetic training data is sufficiently representative of the real problems that the model will eventually be used for. However, it is possible that the generation procedure under-represents or misses some important types of theory, potentially giving the model a “blind spot” on novel problems if it is unable to fully generalize. (A minor example of this was the MMax results on Electricity4, last paragraph of Section 4.3). It would be valuable to find ways to characterize the different *types* of inference problems in the space, and design training curricula to ensure they are systematically covered and/or the model is able to generalize to them. Adversarial approaches to generation, where the generator learns to create theories that are hard for a partially trained model, may be useful in this context, e.g., [Kalyan *et al.*, 2019].

5.3 Natural Language Inference (NLI)

We have shown that transformers can perform deductive inference over English statements. However, human reasoning over language - natural language inference (NLI) - is not always deductive. In particular, NLI allows for *unsupported* inferences that “a person would typically infer” [Dagan *et al.*, 2013], while we have used a precise model of inference in which *all* of a rule’s conditions need to be proven true in order for the conclusion to follow. Our model may still be quite far from that required for fully natural reasoning over language. For example, we would like our model to still proceed if there are gaps in the explicitly provided knowledge, providing the missing knowledge is “obvious” (and not contradicted by the explicitly provided facts), perhaps by leveraging its pretrained knowledge. Similarly, our model’s treatment of negation as failure (NAF) sometimes clashes with intuitions about NLI, for example given (just) “If my car does not have

gas then it is not working.” our model will conclude (given nothing else) that “My car is not working.” as it cannot *prove* that “My car has gas.”.

This raises a fundamental tension about the nature of the reasoning we ultimately desire: We want reasoning to be rigorous (conclusions justified by the information provided), but also “soft” (tolerant of phrasing differences and common-sense knowledge gaps), and strictly speaking these two goals are in conflict. Our experiments with Turk-authored language illustrates tolerance of phrasing differences, which we view as desirable, although in a strict deductive sense it is unjustified to conclude (say) “A person is green” from “Charlie has green teeth” (Figure 6). Similarly we would like the model to tolerate minor, unstated taxonomic gaps, for example given “Buildings have roofs” conclude “My house has a roof”, even if “Houses are buildings” is not explicitly stated (but *not* conclude that result if it is explicitly stated that “Houses are *not* buildings”). Characterizing which inferences should be deductive vs. which can be assumed in NLI, and training a model to combine explicitly stated knowledge with implicit (pretrained) knowledge, remain significant open challenges.

6 Conclusion

Just as McCarthy advocated 60 years ago for machines reasoning (“taking advice”) in logic, we have shown (in a restricted setting) that machines can be trained to reason over language. While we have assumed a particular semantics of inference, the methodology we have used is general: Characterize the desired behavior in a formal way, synthesize examples, generate linguistic equivalents, and train a model. The result, at least within our experiments, appears to be both natural and robust, in a way distinct from working with the original formalization.

The ability to reason (or emulate reasoning) over rules expressed in language has potentially far-reaching implications. For example, rules might be easily authored by a person, sidestepping some of the intricacies of a formal language (a simple kind of “programming in English”); or they could be retrieved from natural sources (e.g., science texts, Wikipedia). Similarly, if the answer is wrong, the user may be able to directly teach the system by providing general missing knowledge (or correcting erroneous knowledge) that can then also be used for new problems - a step towards instructable algorithms. Finally, the mechanism opens the door to neural counterfactual reasoning. For example, we can modify the earlier “birds” rulebase to describe a world in which birds typically don’t fly, but where ostriches can fly, and see the consequences. To encourage further progress, an interactive demo and all our datasets are available at <https://allenai.org/data/rulemaker>

Acknowledgements

Thanks to Chitta Baral, Jonathan Berant, Oren Etzioni, Matt Gardner, Ashish Sabharwal, and Alon Talmor for comments on earlier drafts.

⁸ If one even exists - formal reasoning is still far from modeling all of natural language inference.

References

- [Abboud *et al.*, 2020] R. Abboud, I. Ceylan, and T. Lukasiewicz. Learning to reason: Leveraging neural networks for approximate dnf counting. In *AAAI*, 2020.
- [Abdelaziz *et al.*, 2020] Ibrahim Abdelaziz, Veronika Thost, Maxwell Crouse, and Achille Fokoue. An experimental study of formula embeddings for automated theorem proving in first-order logic. *arXiv*, 2002.00423, 2020.
- [Apt *et al.*, 1988] K. Apt, H. Blair, and A. Walker. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming.*, 1988.
- [Bidoit and Froidevaux, 1991] N. Bidoit and C. Froidevaux. General logical databases and programs: Default logic semantics and stratification. *Inf. Comput.*, 91:15–54, 1991.
- [Chen *et al.*, 2017] Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced lstm for natural language inference. In *ACL*, 2017.
- [Dagan *et al.*, 2013] Ido Dagan, Dan Roth, Mark Sammons, and Fabio Zanzotto. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool, 2013.
- [He and Choi, 2019] Han He and Jinho D. Choi. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert. *ArXiv*, abs/1908.04943, 2019.
- [Kalyan *et al.*, 2019] Ashwin Kalyan, Oleksandr Polozov, and Adam Kalai. Adaptive generation of programming puzzles. Technical report, Georgia Tech, 2019. (<https://openreview.net/forum?id=HJeRveHKDH>).
- [Kamath and Das, 2019] Aishwarya Kamath and Rajarshi Das. A survey on semantic parsing. In *AKBC'19*, 2019.
- [Lai *et al.*, 2017] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy. RACE: Large-scale reading comprehension dataset from examinations. In *EMNLP*, 2017.
- [Lample and Charton, 2019] G. Lample and F. Charton. Deep learning for symbolic mathematics. In *ICLR*, 2019.
- [Lin *et al.*, 2019] Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. Reasoning over paragraph effects in situations. In *Proc. MRQA Workshop (EMNLP'19)*, 2019. also [arXiv:1908.05852](https://arxiv.org/abs/1908.05852).
- [Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: a robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [MacCartney and Manning, 2014] Bill MacCartney and Chris Manning. Natural logic and natural language inference. *Computing Meaning*, 47:129–147, 2014.
- [Manning and MacCartney, 2009] Christopher D. Manning and Bill MacCartney. *Natural language inference*. Stanford University, 2009.
- [McCarthy, 1959] John W. McCarthy. Programs with common sense. In *Proc. Tedding Conf. on the Mechanization of Thought Processes*, pages 75–91, 1959.
- [McCarthy, 1984] J. McCarthy. Applications of circumscription to formalizing commonsense. In *NMR*, 1984.
- [Metaxiotis *et al.*, 2002] Kostas S Metaxiotis, Dimitris Askounis, and John Psarras. Expert systems in production planning and scheduling: A state-of-the-art survey. *Journal of Intelligent Manufacturing*, 13(4):253–260, 2002.
- [Musen and Van der Lei, 1988] Mark A Musen and Johan Van der Lei. 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, 1988.
- [Newell and Simon, 1956] A. Newell and H. Simon. The logic theory machine—a complex information processing system. *IRE Trans. Information Theory*, 2:61–79, 1956.
- [Parikh *et al.*, 2016] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *EMNLP*, 2016.
- [Rajpurkar *et al.*, 2016] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016.
- [Richardson and Sabharwal, 2019] Kyle Richardson and Ashish Sabharwal. What does my qa model know? devising controlled probes using expert knowledge. *ArXiv*, abs/1912.13337, 2019.
- [Richardson *et al.*, 2020] Kyle Richardson, Hai Hu, Lawrence S Moss, and Ashish Sabharwal. Probing natural language inference models through semantic fragments. In *AAAI'20*, 2020.
- [Saxton *et al.*, 2019] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *ICLR*, 2019.
- [Selsam *et al.*, 2019] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *ICLR*, 2019.
- [Sinha *et al.*, 2019] K. Sinha, S. Sodhani, J. Dong, J. Pineau, and W. Hamilton. CLUTRR: a diagnostic benchmark for inductive reasoning from text. In *EMNLP*, 2019.
- [Tafjord *et al.*, 2019] Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. Quartz: An open-domain dataset of qualitative relationship questions. In *EMNLP*, 2019.
- [Talmor *et al.*, 2019] A. Talmor, Y. Elazar, Y. Goldberg, and J. Berant. oLMpics - on what language model pre-training captures. *ArXiv*, abs/1912.13283, 2019.
- [Wang *et al.*, 2019] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. Wong, and L. Chao. Learning deep transformer models for machine translation. In *ACL*, 2019.
- [Weber *et al.*, 2019] Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. Nlprolog: Reasoning with weak unification for question answering in natural language. In *ACL*, 2019.
- [Weston *et al.*, 2016] J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards AI-Complete question answering: A set of prerequisite toy tasks. In *ICLR*, 2016.