Pushing the Limits of Rule Reasoning in Transformers through Natural Language Satisfiability

Kyle Richardson  Ashish Sabharwal
Allen Institute for AI, Seattle, WA, USA {kyler,ashishs}@allenai.org

Abstract

Investigating the reasoning abilities of transformer models, and discovering new challenging tasks for them, has been a topic of much interest. Recent studies have found these models to be surprisingly strong at performing deductive reasoning over formal logical theories expressed in natural language. A shortcoming of these studies, however, is that they do not take into account that logical theories, when sampled uniformly at random, do not necessarily lead to hard instances. We propose a new methodology for creating challenging algorithmic reasoning datasets that focus on natural language satisfiability (NLSat) problems. The key idea is to draw insights from empirical sampling of hard propositional SAT problems and from complexity-theoretic studies of language.

This methodology allows us to distinguish easy from hard instances, and to systematically increase the complexity of existing reasoning benchmarks such as RuleTaker. We find that current transformers, given sufficient training data, are surprisingly robust at solving the resulting NLSat problems of substantially increased difficulty. They also exhibit some degree of scale-invariance—the ability to generalize to problems of larger size and scope. Our results, however, reveal important limitations too: a careful sampling of training data is crucial for building models that generalize to larger problems, and transformer models’ limited scale-invariance suggests they are far from learning robust deductive reasoning algorithms.

Introduction

Motivated by the impressive performance of recent pre-trained transformers [Devlin et al. 2019; Raffel et al. 2020] on a wide range of natural language understanding (NLU) benchmarks [Wang et al. 2019b; Xu et al. 2020], there has been recent interest in investigating the linguistic and reasoning abilities of state-of-the-art neural models [Linzen, Dupoux, and Goldberg 2016; Talmor et al. 2020; Kassner, Kroje, and Schütze 2020; Yanaka et al. 2020; Hupkes et al. 2020; Richardson et al. 2020; inter alia]. One particular thread of work focuses on probing whether transformers can perform logical reasoning over formal theories expressed in natural language [Clark, Tafjord, and Richardson 2020]. Specifically, given a set of systematically constructed natural language theories consisting of a set of explicitly stated rules and facts (e.g., the NL Theory in the bottom part of Figure 1 containing fictional rules about characters Bob and Alan), the goal is to see whether a model can learn to perform deductive reasoning over such theories by correctly answering queries that require making novel inferences (e.g., predicating that Alan is green is true based on knowing that Alan is rough and applying the rule All rough people are green).

While much of this recent work on behavioral probing has centered around small synthetic domains and datasets (see also Weston et al., 2015; Lake and Baron, 2018; Sinha et al., 2019), the appeal of such testing is that it can allow us to uncover the strengths and weaknesses of models in a cost-effective and controlled manner, and ultimately determine whether models are inherently capable of solving certain algo-

Figure 1: TOP: An illustration of our general methodology for constructing hard natural language reasoning problems for a task $T$, by grounding them into a hard combinatorial problem $P$ and sampling hard instances of $P$. BOTTOM: An example of a natural language (NL) theory (i.e., set of arbitrary facts and rules) $\Gamma$ along with two example conjectures (i.e., propositions to be proved) and the relationship between entailment and satisfiability.
rithmetic problems. Given that most behavioral probing studies are performed in a black-box fashion (Ribeiro et al., 2020) and are thus limited to input-output-driven testing, however, the quality and informativeness of a probing study relies on having reliable data that faithfully captures the full target problem space. In particular, to demonstrate that a model can learn a certain algorithmic skill, it must be demonstrated that the model can solve the hardest instances of the target problem. Indeed, recent work (Shin et al., 2019; Wu et al., 2021; Tamari et al., 2021) has revealed various pitfalls associated with synthetic data due to ad-hoc sampling strategies, which can dramatically inflate model performance by undersampling difficult cases in a way that can also harm model generalization.

In evaluating a particular diagnostic dataset for probing logical reasoning, the following question arises: are the problems contained in the target dataset hard in some objective computational sense? For example, while knowing that Bob is round implies that Bob is big, the task of determining whether Bob is round suffice to prove Bob is big also requires multiple inferential steps (i.e., combining the fact Bob is round with the two rules If someone is round then they are big and Big people are not green), the structure of the rules involved is such that there are well-known highly efficient algorithms for computing this inference. A natural question, then, is: can models perform inferences involving more complex reasoning with rules? Answering the hardness question, therefore, involves two additional questions: (Q1) is the formal language used to express the target problem space capable of expressing hard problems (e.g., ones that go beyond simple linear chaining)? (Q2) is the sampling method used to generate target instances able to effectively capture the full problem space?

In this paper, we fix the formal language to be expressive enough such that it can, by design, represent computationally hard problems (thereby addressing Q1). To address Q2, we propose a general methodology, illustrated in the top part of Figure 1. Given a target probing task T such as deductive inference over statements expressed in natural language, the key idea is to identify subsets of T that map to a known hard combinatorial reasoning problem P such as Boolean satisfiability (SAT), and devise methods to sample hard instances of P in order to arrive at hard instances of T.

Specifically, we broaden the scope of (Clark, Tafjord, and Richardson, 2020) to look at natural language satisfiability (NLSat) problems, or types of natural language deductive reasoning problems that formally assume an underlying SAT semantics. Using insights from empirical sampling of hard SAT problems (Selman, Mitchell, and Levesque, 1996) we show how to systematically construct computationally difficult reasoning problems by focusing on such hard rule fragments and by sampling from the critical phase-change regions of SAT. We show that such an approach has twofold utility: 1) distinguishing easy from hard instances that are consequential for training robust models and for reliable evaluation and; 2) for diagnosing and increasing the complexity of existing reasoning benchmarks.

Our results are partly positive: when provided with a sufficient amount of training instances (e.g., >100k examples), recent pretrained transformers can indeed solve non-trivial NLSat problems that far exceed the complexity of existing reasoning benchmarks (e.g., achieving >90% accuracy on quantified rule theories containing up to 70 ground variables). They also exhibit some degree of generalization and scale-invariance, or the ability to generalize to problems of larger scope (e.g., generalizing from propositional theories with 12 variables to ones with 30, while maintaining performance far above random chance).

At the same time, our results also reveal important caveats: 1) the ability of a model to solve hard reasoning problems critically relies on how well its training data is sampled and; 2) the degree to which models are scale-invariant remains limited, suggesting that models trained in the standard paradigm are still far from learning the underlying algorithms needed for robust deductive reasoning.

Related Work

Our work follows the literature on behavioral testing of neural NLU models and builds on work by (Clark, Tafjord, and Richardson, 2020) on probing deductive reasoning, which has spawned a number of subsequent studies (Saha et al., 2020; Contier et al., 2020; Betz, Voigt, and Richardson, 2021; Betz, Richardson, and Voigt, 2021; Tafjord, Mishra, and Clark, 2021; Saparov and Mitchell, 2021; Liang, Bethard, and Surdeanu, 2021). While these studies demonstrate that models are able to solve some deductive reasoning problems, we observe that existing datasets narrowly focus on the simplest deductive reasoning problems when subjected to closer analysis. As we detail in Table 1, the standard RuleTaker dataset, which is based on a fragment of English that is capable of expressing intrinsically hard algorithmic problems, is limited to the easiest types of deductive reasoning problems. As a result, existing models lack robustness when evaluated on harder parts of the problem distribution as we show in Table 2 on a RuleTaker-style dataset sampled using our new sampling strategy.

To find hard reasoning fragments of natural language, we take inspiration from the literature of complexity-theoretic studies of various natural language fragments (Pratt-Hartmann, 2004; Pratt-Hartmann, Third et al., 2006; Pratt-Hartmann and Moss, 2009; Thorne and Calvanese, 2010). Particularly, (Pratt-Hartmann, 2004) looks at the computational properties such as the complexity of satisfiability for various rule fragments of English, which is the motivation behind the grounded relative-clause fragment we describe in the next section. While this work focuses on a worst-case analysis of different linguistic phenomena, we use the results as a guide to find the hard cases for probing the limits of models.

To find hard natural language satisfiability instances, we rely on techniques from the literature on empirical sampling of combinatorial problems, where it has been observed (Cheeseman et al., 1991) that hard instances of different problems lie at various critical thresholds that correlate with the constrainedness of a given problem. We specifically use techniques for generating hard 3SAT problems (Selman, Mitchell, 1991).
Within this space, we follow Shin et al. (2019); Wu et al. we sample hard instances from ordinary SAT problems in developing novel sampling strategies for avoidance, see Davis, Sigal, and Weyuker (1994)[Theorem2.1,p252]. Satisfiability (or its complement) end up being intereducible notions a pre-defined set of English rule languages. Boolean logic and translate them into natural language using possible expressed in language (e.g., those shown in Figure 4) have a reasoning task that involves determining whether a set of rules and Levesque[1996] Cook and Mitchell[1997] to sample hard problems from the critical regions of SAT phase transitions (see Figure 2). To our knowledge, we are first to investigate this work and using SAT-based representations of linguistic problems to empirically find hard natural language reasoning problems (see Hahn, Jurafsky, and Futrell [2021]).

Our study also follows other work on training neural models to solve hard algorithmic problems (Vinyals, Fortunato, and Jaitly 2015; Reed and De Freitas 2015; Cai, Shin, and Song 2017), including SAT (Selsam et al. 2018) and propositional inference (Evans et al. 2018; Traylor, Feiman, and Pavlick 2021); a key difference is our focus on algorithmic problems in natural language and on probing current pre-trained transformers (Devlin et al. 2019; Liu et al. 2019; Raffel et al. 2020). Following studies such as Reed and De Freitas (2015), we also look at the ability of models to be scale-invariant, or scale to problems of larger size and scope. Within this space, we follow Shin et al. (2019); Wu et al. (2021) in developing novel sampling strategies for avoiding the pitfalls of randomly sampled algorithmic datasets, which can give rise to the kinds of biases observed in human-annotated datasets (Gururangan et al. 2018) and limit model generalization.

Dataset Construction and Methodology

Naturallanguage satisfiability (NLSat) is a deductive reasoning task that involves determining whether a set of rules expressed in language (e.g., those shown in Figure 4) have a satisfying assignment or possible interpretation. Following our general methodology shown in the top part of Figure 1, in order to find hard deductive reasoning problems of this kind, we sample hard instances from ordinary SAT problems in Boolean logic and translate them into natural language using a pre-defined set of English rule languages.

Algorithm 1 Dataset construction via random SAT

Input: Variables set $V = \{ v_1, ..., v_n \}$ of size $n$, natural language templates $R$ and variables $P$, 2SAT to 3SAT interpolation parameter $p_{\text{int}}$, negation parameter $p_{\text{neg}}$, clause variable ratio $\alpha$ range ($\alpha_{\text{min}}, \alpha_{\text{max}}$), STOP condition

Output: NLSat dataset

1: $D \leftarrow \{}$  \hspace{1cm} $\triangleright$ initialize dataset
2: repeat
3: $P \leftarrow \{}$  \hspace{1cm} $\triangleright$ problem/clause set
4: $m \sim \text{choose} m, \text{s.t. } \alpha_{\text{min}} \leq \frac{m}{n} \leq \alpha_{\text{max}}$
5: for $i = 1$ to $m$ do  \hspace{1cm} $\triangleright$ generate $m$ clauses
6: $s \sim \text{choose} \text{ clause size } k \in (3,2)$ with prob. $(p_{\text{int}}, 1 - p_{\text{int}})$
7: $V' \sim \text{choose} s \text{ unique variables from } V'$
8: $C \leftarrow \text{negate} \text{ each } v \in V' \text{ with } p_{\text{neg}}$  \hspace{1cm} $\triangleright$ new clause
9: $t \sim \text{choose} \text{ NL template from } R$ of size $s$
10: $d \leftarrow \text{instantiate } t \text{ over } C \text{ using variables from } P$
11: $P \leftarrow P \cup \{d\}$
12: $D \leftarrow D \cup P$
13: until dataset STOP condition is met

Figure 2: Illustration of phase-change and SAT probability for random 2-SAT($p_{\text{int}} = 0$) and 3-SAT($p_{\text{int}} = 1$) problems over a randomly sampled set of examples with varying $\alpha$ and number of variables (5-15).

In this section, we first detail the semantics of SAT and how to identify hard SAT problems (Identifying Hard Problems and Algorithm 1), and then describe the two different fragments of English we use for our experiments (the Grounded Rule Language and Grounded Relative Clause Fragment; see details in Figure 3); both borrow certain grounded and quantified rule constructs from the RuleTaker language and, in the latter fragment, build on some constructions studied in formal linguistics. Finally, we discuss ways of sampling SAT instances of different hardness levels and sizes.

Identifying Hard Problems

The SAT problem is the classic NP-complete problem (Cook 1971). We focus on 3SAT problems where $k = 3$, i.e., each formula is limited to clauses of size three. While 3SAT is computationally hard under a worst-case analysis, this does not mean that all, or even most, 3SAT instances are hard to solve. Indeed, work on empirical sampling of classes of random $k$-SAT problems has revealed that whether a problem is difficult crucially relies on details about the target distribution from which the problems are sampled (Selman, Mitchell, and Levesque 1996; Mitchell and Levesque 1996) as well as the particular parameters employed during sampling.

To obtain hard SAT instances, we rely on a variant of the well-studied random $k$-SAT algorithm first introduced in Selman, Mitchell, and Levesque (1996), which we illustrate in Algorithm 1. In standard $k$-SAT, random formulae of size $m$ containing $n$ variables and clauses of fixed length $k$ are obtained by selecting $m$ clauses (starting line 5) uniformly from the space of $2^k$ ($\binom{n}{k}$) possible clauses (where each clause is constructed by sampling $k$ unique variables (line 7) and negating each with probability $p_{\text{neg}}$ (line 8)). While our primary focus is on 3-SAT, for convenience we include the possibility of sampling mixed 2-SAT/3-SAT problems by introducing an interpolation parameter $p_{\text{int}}$ (shown on line 6, Monasson et al. 1999). Using a suitable fragment of natural language $R$ (see next section), our version additionally includes translating...
each random clause to expressions in natural language (lines 9-10).

A key parameter in random SAT is the clause to variable ratio \( \alpha = \frac{m}{n} \) (computed on line 4 and dictated, in part, by the range \( \alpha_{\text{min}}, \alpha_{\text{max}} \)). This parameter gives rise to phase-change behavior that has implications for problem hardness (Hayes 2005). Such phase changes are illustrated in Figure 2 where \( \alpha \) (x-axis) can be used to determine the probability of a random formula being satisfiable (y-axis). For our purposes, such a curve suggests a principled way to identify hard instances, namely, by selecting formulae from the critical region where problems have roughly 0.5 probability of being satisfiable. The motivation behind sampling in this manner follows much of the work in empirical SAT, where it is found that such problems are constrained in a unique way that makes it difficult to simply guess the correct answer by looking at the superficial patterns, which in our context makes it harder for model to exploit short-cuts (we later provide empirical evidence that narrowly focusing on training instances close to the critical region leads to more robust models that generalize to the overall distribution better than models trained via ad-hoc sampling from the entire space).

**Grounded Rule Language (GRL)**

The **Grounded Rule Language** is a straightforward translation of the clauses in a random Boolean formula into grounded propositional (if-then) rules, similar to some of the rules used by Clark, Tafjord, and Richardson [2020]. For example, as detailed in Figures 3 and 4 a clause with three literals:

\[
\pm v_1 \lor \pm v_2 \lor \pm v_3 \quad (1)
\]

can be translated as \( \text{If (no) } v_1 \text{ and (no) } v_3 \text{ then (no) } v_3 \) (using the standard rules of logical equivalence) where each variable \( v_j \) is subsequently replaced with an English count noun (i.e., any noun that can be made plural and made into a singular form with the determiner *an/an*).

We choose a fixed set of around 50 nouns about food for our main GRL set reported in Table 1 (see examples in Figure 4). Each instance in our main set is characterized by a varying number of SAT variables or nouns, ranging from 5 to 12, which we discuss and motivate below.

We note that while the propositions in these fragments (e.g., *carrot, steak*) deviate slightly from propositions encountered in ordinary language, one interpretation of the resulting theories is that they are akin to cooking recipes: e.g., *(if you have) carrot and not steak then (you need to have) apples*. Figuring out whether the set of sentences is satisfiable is equivalent to deciding whether there is a coherent recipe underlying the rules. The decision to create data in a truncated form (i.e., without verbs) is due to the following considerations: some of the transformer models we probe have strict token limits which are easily exceeded when expressing the target hard computational problems using longer phrases; and leaving out this information does not affect the complexity of the resulting reasoning problem that we are interested in probing.

In the next section, we describe our second fragment that aims to capture more conventional linguistic constructions.

**Grounded Relative Clause Fragment (RCL)**

The **Grounded Relative Clause Fragment** is characterized by the relative clause rule construction *Every X who is (not) a/an Y is (not) a/an Z*, which, via its translation from first-order logic:

\[
\forall x. \ X(x) \land \pm Y(x) \rightarrow \pm Z(y), \quad (2)
\]

corresponds to boolean clauses of the form \( \neg v_1 \lor \pm v_2 \lor \pm v_3 \) containing up to two positive literals (where each variable corresponds to a count noun, or predicates \( X, Y, Z \)). To allow

\[
\text{Figure 3: A syntactic description of the two rule languages used for our experiments.}
\]

\[
\text{Figure 4: Example translations of a satisfiable 3SAT problem (truncated) in boolean logic and two fragments of English (variables in the natural language are highlighted). The bottom shows example interpretations of each expression that demonstrate satisfiability.}
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{Language} & \text{Example Expression} & \text{Satisfying Assignment} \\
\hline
\text{Propositional Logic (3SAT)} & \left( \neg v_1 \lor v_15 \lor v_13 \right) \land \left( \neg v_13 \lor \neg v_12 \lor \neg v_11 \right) \land \left( v_1 \lor v_15 \lor \neg v_13 \right) & v_1=\text{false}, v_{15}=\text{false}, v_{13}=\text{false}, v_{12}=\text{true} \ldots \\
\hline
\text{Grounded Natural Language Fragments} & \text{If carrot and not steak then apples, If apples and grapes then no carrots. If no carrots and no steak then no apples...} & \text{needed: carrots, apples, grapes...} \\
\hline
\text{Grounded Rule Language (GRL)} & \text{Every doctor who is not a philosopher is a baker. No baker who is a gardener is a philosopher. John is a doctor or a philosopher or not a baker...} & \text{John can be a doctor, a baker, not a philosopher and not a gardener...} \\
\hline
\text{Grounded Relative Clause Fragment (RCL)} & \text{Every X who is (not) a/an Y is (not) a/an Z. No X who is (not) a/an Y is (not) a/an Z. Everyone who is (not) a/an X and (not) a/an Y is (not) a/an Z.} & \text{c is (not) a/an X or (not) a/an Y or (not) a/an Z.} \\
\end{array}
\]
for clauses with up to three positive literals, we add the rule template \( \text{Everything that is (not) an } X \text{ and (not) a } Y \text{ is (not) a } Z \), where everything universally quantifiers over the entire domain.

We obtain a mapping to propositional logic by assuming finite domains following some theoretical studies on quantifiers \cite{Westerveltal1984,Szymanetal2016} and work on utilizing propositional logic for various reasoning problems in classical AI \cite{Kautzetal1992,Kautzetal1996}. More specifically, grounding formulas such as Equation \ref{eq:ground} relies on having clause translations \( c \) is (not) \( a/an \) \( X \) or (not) \( a/an \) \( Y \) or (not) \( a/an \) \( Z \) that allow for introducing disjunctive facts that involve constants or proper nouns (denoted as \( e \)); given a set of universally quantified rules and such disjunctive facts, all universals rules are expanded to group propositions over all constants out to arrive at a final grounded formula.

Count and proper nouns are selected from a small inventory of noun types (as above, around 50) about people and their occupations (see again Figure \ref{fig:people}). A particular feature of this fragment is that through such universally quantified rules and their expansion to propositional logic, we can arrive at more complex reasoning problems that significantly increasing the number of \textit{ground variables} and size of the target problems without dramatically increasing the size of the natural language input. The rules in our data are sampled from random 3SAT formulae over a fixed set of (5-8) variables and are coupled with an additional set of random clauses for disjunctive rule instances. While these resulting boolean formulas deviate from strict random 3SAT, the expansion of universal rules over a set of constants preserves the ratio of variables and clauses, which give rise to the same phase-change phenomena illustrated in Figure \ref{fig:complexity} allowing us to find the hard cases in the critical region.

\section*{Sampling Strategies and Proposed Datasets}

A summary of our datasets is shown in Table \ref{tab:datasets} along with a comparison to the standard RuleTaker dataset converted to SAT \cite{McAllesterSelmanetal1996}. As described above, the NLSat instances that constitute the grounded rule language (GCL\textsubscript{5,12}) and the grounded relative clause fragment (RCL\textsubscript{16,70}) are characterized by the number of variables contained in their underlying Boolean formulae (with \( d_{\#\text{vars}} \) denoting the overall range), which are uniformly represented in each dataset to allow for later inspection of model performance. For each variable amount, the majority of Boolean formulae are sampled from the critical 0.5 (±0.1) probability region by heuristically controlling the \((\alpha_{\min}, \alpha_{\max})\) clause variable ratio range in Algorithm \ref{alg:random} (henceforth, \textbf{hard sampling})\footnote{More details about this conversion and technical details about the RuleTaker language can be found in the appendix.}, which we later show leads to advantages over both \textbf{naive sampling} (i.e., choosing instances randomly within a large range) and \textbf{biased sampling} (i.e., sampling \textit{easy} instances from the extreme ends of the phase change) strategies (see Figure \ref{fig:hard}). Formulae and their translations are then randomly split into train and evaluation sets using a 80\% (train) / 20\% (dev,test) ratio.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
\textbf{Dataset}  & \textbf{Size}  & \textbf{Complexity}  & \textbf{Conflicts}  & \textbf{Decisions}  \\
& (d\_{\#\text{vars}}) & (NP-complete?) & (avg/med.) & (avg/med.) & (avg/med.) \\
\hline
\textbf{RuleTaker} & 130k & yes & 0.0/0.0 & 6.6/0.0 \\
\textbf{GRL}_{5,12} & 187k & yes & 3.4/4.0 & 3.4/4.0 & \\
\textbf{RCL}_{16,70} & 219k & yes & 7.6/6.0 & 29.7/6.0 & \\
\textbf{GRL}_{\text{eval}_{20,50}} & 17k & no & 22.0/13.0 & 29.3/13.0 & \\
\hline
\end{tabular}
\caption{The RuleTaker dataset, while similar in terms of dataset size and formal language complexity as our rule language datasets (GRL and RCL), is substantially simpler in terms of two standard SAT-based empirical complexity metrics: number of \textbf{conflicts} and \textbf{decisions}.}
\end{table}

A particular advantage of having boolean formulae associated with our target data is that we can use automatic reasoning tools to obtain empirical measurements of problem hardness. Using the off-the-shelf theorem prover Z3 \cite{De Mouraetal2008}, we report the average/median \((\text{avg/med.})\) number of \textbf{decisions} (e.g., number of variable assignments after pre-processing) and \textbf{conflicts} (amount of backtracking performed for obtaining more complex proofs) needed by its \texttt{solve} method on each dataset. While such statistics are often tied to the internals of a solver (especially \#\texttt{decisions}), there still are some notable observations.

We see from Table \ref{tab:datasets} that RuleTaker, in spite of its language’s high theoretical complexity, is limited to the simplest forms of deductive inference as evidenced by having very few problems involving any conflicts and decisions at all (the median number for both is 0). In sharp contrast, our new datasets, via our \textbf{hard} sampling strategy, offer a much wider range of problem difficulty. By retrofitting our randomly sampled 3SAT formula to include theories with 2SAT and single propositional rules similar to RuleTaker theories, we are also able to construct a substantially harder RuleTaker dataset (model performance to be discussed in Table \ref{tab:evaluation}).

An important caveat is that while our new problems are of a higher degree of difficulty compared with problems in datasets like RuleTaker, they are still of vastly low complexity (both in terms of number of variables and the statistics shown in Table \ref{tab:datasets}) compared to the much harder SAT instances encountered in the mainstream SAT literature \cite{Jarvisaloeetal2012}. The decision to limit problems to the number of variables we did (e.g., to a maximum of 12 variables for GRL\textsubscript{5,12}), is partly practical, and due to considerations such as token limits in the models we describe next and overall training efficiency. As we will see, these problems, while simple for mainstream SAT solvers, are still quite challenging for state-of-the-art transformer models and thus valuable for advancing research on the latter. Following \cite{Reedetal2015}, the decision also reflects the idea that we should aim for models that can perform \textit{scale-invariant}
reasoning by generalizing from small problems. For this purpose, we create an additional held-out set of considerably larger grounded rule reasoning problems GRL-eval\textsubscript{20,50} to measure scale-invariance.

**Experimental Setup**

**Task Definition.** Formally, a NLSat dataset \( D = \{(p^{(d)}, l^{(d)})\}_{i=1}^{|D|} \) consists of NLSat problems \( p \) (i.e., a set of rules expressed in natural language) paired with a label \( l \in \{\text{sat}, \text{unsat}\} \). The goal is to correctly predict the label (indicating satisfiability or not), thereby reducing to binary classification as in Clark, Tafjord, and Richardson (2020).

**Models.** Following recent studies on rule reasoning (Tafjord, Mishra, and Clark 2021), our investigation centers around the pre-trained text-to-text transformer T5-large model (with around 770M parameters) \(^1\) (Raffel et al. 2020). We also compare against RoBERTa (with around 355M parameters) \(^2\) (Liu et al. 2019). In each case we use the implementation from Wolf et al. (2019). Standardly, models are fine-tuned to generate the target labels by optimizing for the cross-entropy loss over the target sat and unsat tokens or labels. Also standardly, model selection is performed by doing a random search (in the style of Devlin et al. 2019) over target hyper-parameters (focusing especially on learning rate, random seed, and \# training iterations) and selecting models with the highest dev. score. As mentioned above, we also found intermediate pre-training on 60k simpler 2SAT instances (i.e., instances sampled with \( p_{\text{sat}} = 0 \) in Algorithm \(^7\) with simpler natural language rule templates containing only 2 propositions) to be indispensable for stabilizing and improving model training efficiency on our main tasks.

**Evaluation.** We train models separately on our two languages (GRL and RCL) and their respective datasets (see again Table \(1\) in the manner described above. We report accuracy across sub-samples of evaluation data characterized by varying numbers of variables (i.e., the \#var column in Tables \(2\)). To better understand model generalization, we also experiment with training on small samples of data with a different number of variables for GRL as well as evaluation on a larger held-out GRL-eval set and easy and hard instances, as shown in Table \(2\). To better understand how different sampling strategies affect model performance, we perform experiments that measure the effect of different sampling strategies as shown in Figure \(5\).

Lastly, to verify the difficulty of our tasks, we also experimented on a non-pretrained biLSTM encoder model implemented using AllenNLP \(^3\) (Gardner et al. 2018). While not shown in the tables, we found, consistent with the results of Clark, Tafjord, and Richardson (2020), that such models perform near random chance. \(^4\)

\(^1\)At an earlier iteration we also performed experiments T5-11b and found comparable performance. We note that a particular appeal of T5 is its use of relative positional embeddings which allows us to evaluate on larger problems such as those in our GRL-eval set that exceed the 512 token limit from pre-training.

\(^2\)As a check, we also verified that the same models obtained comparable results to the biLSTM baselines reported by Clark, Tafjord, and Richardson (2020) on the original RuleTaker dataset.

\(^3\)Such is also a lesson from the literature on hard SAT. Quoting Mitchell and Levesque (1995): “Random formulas have been used by many researchers to empirically evaluate the performance of SAT testing programs. The value of such studies depends upon careful selection of formula distribution... When using random formulas, an extensive enough study of the distribution’s parameter space must be carried out ... if the results are to be meaningful.

---

Figure 5: Training on hard problems (our proposal) is much more effective than training on problems sampled in a naive or biased manner. TOP: Comparison of 10 variable model trained using different sampling strategies and tested across the full distribution of hard (problems in critical region) and easy (problems at extreme of distribution) 5 to 10 variable problems (dev). BOTTOM: performance of the same 10 variable models on these different categories of problems.
models exhibit limited generalization. Performance (dev) of models trained on GRL problems containing differing numbers of a certain size and evaluated on easy and hard cases also of varying size. LEFT: Generalization across different combinations of problem sizes for training and evaluation. RIGHT: Generalization to larger instances never seen during training.

| Model_name_var | 5var | 7var | 8var | 10var | 12var | Avg. |
|----------------|------|------|------|-------|-------|------|
| T5_{12}var     | 97.5 | 97.9 | 98.0 | 98.3  | 98.5  | 98.4 |
| T5_{70}var     | 93.9 | 98.0 | 97.7 | 89.7  | 82.5  | 90.0 |
| T5_{22}var     | 94.5 | 91.1 | 91.5 | 84.9  | 87.7  | 80.7 |
| T5_{12}var     | 98.6 | 98.1 | 96.0 | 93.6  | 92.6  | 89.6 |

Table 3: Models trained on hard sets are surprisingly good at some hard tasks in the i.i.d. setting. Performance (test) of models on the GRL and RCL fragments, split into performance on problems with differing number of variables.

The importance of sampling for having reliable evaluations is further revealed in our experiments on sampling hard RuleTaker evaluation data from hard SAT, as shown in Table 4. While it is unclear what the exact distribution of problems defined in the RuleTaker domain is, the results in this table clearly demonstrate the efficacy of our general sampling framework in identifying hard datasets. They also show that even small changes in problem difficulty (namely, the relatively modest increase in standard empirical hardness measures, #conflicts and #decisions, as seen earlier in Table 1) can lead to dramatic differences in performance on existing benchmarks (e.g., a 59 point drop in performance for RoBERTa). This is reinforced by the differences in results between the easy/hard problems shown in Table 2 (e.g., 80% vs. 71% (avg) performance difference between easy/hard 20-40 variable problems for the full T5_{5,12} model). Thus, without a proper understanding of the full distribution of target problems, it is often easy to draw inaccurate general conclusions about model capability by inadvertently focusing on easy instances.

Models trained on hard sets can solve some hard tasks. When looking at results on the hard instances (Table 3), we see that models trained on large collections of various types (i.e., on 150k-160k instances, see again Table 1) far outpace our baselines and achieve high performance on problems with not too many variables (e.g., 5-7 variables problems for GRL and 16-48 variables problems for RCL). A particularly intriguing result is the higher performance of models on the RCL language (with around 93-94% accuracy on problems with 60-70 ground variables) which was designed to be more complex by having quantified rules and constants that expand out to a much larger set of boolean variables. Given that the underlying rules were constructed from random 3SAT formula with a relatively smaller set of variables (5-8), this suggests that the model is able to learn some form of symmetry between the underlying rules and the instantiated rule propositions related to constants.

Models exhibit limited generalization. Less impressive results are shown in the Table 1 where we see that models trained on small variable problems and fewer data fail to generalize to larger problems (e.g., generalizing from 5 variables to 10 or 12 variables). More strikingly, we see that even our best models fail to solve the GPL-eval evaluation task; while this is not altogether surprising, it suggests that state-of-the-art transformer models are still far from learning the underlying algorithms associated with deductive inference.
Closing Remarks

With the advent of increasingly larger pre-trained models, including those that now allow for processing of tens of thousands of tokens (Beltagy, Peters, and Cohan 2020), understanding the limits of how much aggregation of information over text models are capable of is an important area of study. Given that the type of algorithmic tasks we study in this paper are concerned with the most complex forms of information aggregation, we believe that our results can bear on these bigger issues about model design. When optimized for problem hardness, we see that models on our datasets still exhibit little ability to generalize in a scale-invariant fashion that is required for effectively generalizing their reasoning abilities to larger problems. Moving forward, we believe that new modeling approaches and architectures (e.g., ones that focus on problem decomposition (Andreas et al. 2016; Khot et al. 2021)) might be a fruitful avenue, which we believe our new algorithmic tasks and sampling strategies for finding hard datasets can assist in exploring.

Acknowledgments

We thank the members of the Aristo team at AI2 for their feedback at various stages of this work, in particular Peter Clark and Oyvind Tafjord, as well as the Beaker team (https://beaker.org/) for their assistance and support with experiments. Special thanks also to Gregor Betz and Christian Voigt for helpful discussions.

References

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016. Learning to compose neural networks for question answering. In NAACL.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.

Betz, G.; Richardson, K.; and Voigt, C. 2021. Thinking Aloud: Dynamic Context Generation Improves Zero-Shot Reasoning Performance of GPT-2. arXiv preprint arXiv:2103.13033.

Betz, G.; Voigt, C.; and Richardson, K. 2021. Critical Thinking for Language Models. Proceedings of IWCS.

Cai, J.; Shin, R.; and Song, D. 2017. Making neural programming architectures generalize via recursion. In ICLR.

Cheeseman, P. C.; Kanefsky, B.; Taylor, W. M.; et al. 1991. Where the really hard problems are. In IJCAI, volume 91, 331–337.

Clark, P.; Tafjord, O.; and Richardson, K. 2020. Transformers as soft reasoners over language. In IJCAI.

Cook, S. A. 1971. The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing, 151–158.

Cook, S. A.; and Mitchell, D. G. 1997. Finding hard instances of the satisfiability problem: A survey. Satisfiability Problem: Theory and Applications, 35: 1–17.

Davis, M.; Sigal, R.; and Weyuker, E. J. 1994. Computability, complexity, and languages: fundamentals of theoretical computer science. Elsevier.
Mitchell, D. G.; and Levesque, H. J. 1996. Some pitfalls for experimenters with random SAT. Artificial Intelligence, 81(1-2): 111–125.

Monason, R.; Zecchina, R.; Kirkpatrick, S.; Selman, B.; and Troyansky, L. 1999. Determining computational complexity from characteristic ‘phase transitions’. Nature, 400(6740): 133–137.

Pratt-Hartmann, I. 2004. Fragments of language. Journal of Logic, Language and Information, 13(2): 207–223.

Pratt-Hartmann, I., and Moss, L. S. 2009. Logics for the relational syllogistic. The Review of Symbolic Logic, 2(4): 647–683.

Pratt-Hartmann, I.; Third, A.; et al. 2006. More fragments of language. Notre Dame Journal of Formal Logic, 47(2): 151–177.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.

Reed, S.; and De Freitas, N. 2015. Neural programmer-interpreters. arXiv preprint arXiv:1511.06279.

Ribeiro, M. T.; Wu, T.; Guestrin, C.; and Singh, S. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In ACL.

Richardson, K.; Hu, H.; Moss, L.; and Sabharwal, A. 2020. Probing natural language inference models through semantic fragments. In AAAI-2020, 8713–8721.

Saha, S.; Ghosh, S.; Srivastava, S.; and Bansal, M. 2020. PRover: Proof generation for interpretable reasoning over rules. In EMNLP.

Saparov, A.; and Mitchell, T. M. 2021. A Generative Symbolic Model for More General Natural Language Understanding and Reasoning. arXiv preprint arXiv:2105.02486.

Selman, B.; Mitchell, D. G.; and Levesque, H. J. 1996. Generating hard satisfiability problems. Artificial intelligence, 81(1-2): 17–29.

Selsam, D.; Lamm, M.; Bünz, B.; Liang, P.; de Moura, L.; and Dill, D. L. 2018. Learning a SAT solver from single-bit supervision. In ICLR.

Shin, R.; Kant, N.; Gupta, K.; Bender, C.; Trabucco, B.; Singh, R.; and Song, D. 2019. Synthetic datasets for neural program synthesis. Proceedings of ICLR.

Sinha, K.; Sodhani, S.; Dong, J.; Pineau, J.; and Hamilton, W. L. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In EMNLP.

Szymanki, J.; et al. 2016. Quantifiers and cognition: Logical and computational perspectives, volume 96. Springer.

Tafjord, O.; Mishra, B. D.; and Clark, P. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. ACL Findings.

Talmor, A.; Elazar, Y.; Goldberg, Y.; and Berant, J. 2020. oLMpics–On what Language Model Pre-training Captures. TACL.

Tamari, R.; Richardson, K.; Sar-Shalom, A.; Kahlon, N.; Liu, N.; Tsarfaty, R.; and Shahaf, D. 2021. Dyna-bAbI: unlocking bAbI’s potential with dynamic synthetic benchmarking. arXiv preprint arXiv:2112.00086.

Thorne, C.; and Calvanese, D. 2010. The data complexity of the syllogistic fragments of English. In Logic, Language and Meaning, 114–123. Springer.

Traylor, A.; Feiman, R.; and Pavlick, E. 2021. AND does not mean OR: Using Formal Languages to Study Language Models’ Representations. In Proceedings of ACL.

Vinyls, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In NeurIPS.

Wang, A.; Prukachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019a. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In NeurIPS.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In ICLR.

Westerståhl, D.; et al. 1984. Some results on quantifiers. Notre Dame Journal of Formal Logic, 25(2): 152–170.

Weston, J.; Bordes, A.; Chopra, S.; Rush, A. M.; van Merriënboer, B.; Joulin, A.; and Mikolov, T. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. arXiv preprint arXiv:1502.05698.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771.

Wu, Z.; Kreiss, E.; Ong, D. C.; and Potts, C. 2021. ReaSCAN: Compositional Reasoning in Language Grounding. In NeurIPS 2021 Datasets and Benchmarks Track.

Xu, L.; Hu, H.; Zhang, X.; Li, L.; Cao, C.; Li, Y.; Xu, Y.; Sun, K.; Yu, D.; Yu, C.; Tian, Y.; Dong, Q.; Liu, W.; Shi, B.; Cui, Y.; Li, J.; Zeng, J.; Wang, R.; Xie, W.; Li, Y.; Patterson, Y.; Tian, Z.; Zhang, Y.; Zhou, H.; Liu, S.; Zhao, Z.; Zhao, Q.; Yue, C.; Zhang, X.; Yang, Z.; Richardson, K.; and Lan, Z. 2020. CLUE: A Chinese Language Understanding Evaluation Benchmark. In COLING.

Yanaka, H.; Mineshima, K.; Bekki, D.; and Inui, K. 2020. Do Neural Models Learn Systematicity of Monotonicity Inference in Natural Language? In ACL.

Zhang, H.; and Stickel, M. E. 1996. An Efficient Algorithm for Unit Propagation. Proc. of AI-MATH, 96.

Appendix

**RuleTaker Details**

**Complexity of RuleTaker Language** The rules in the original RuleTaker language (Clark, Tafjord, and Richardson 2020) take two general forms: grounded rules and quantified rules, a subset of which is shown in Figure 7. To demonstrate the NP-completeness of the RuleTaker language, it suffices to show that an arbitrary 3SAT formula $F$ can be expressed in this rule language such that $F$ is satisfiable if and only if the resulting RuleTaker theory is satisfiable (under...
We note that a similar argument can be made for proving the theories (see again the example in Figure 1) includes both retrofitted random 3SAT to RuleTaker Theories

An example of how we retrofit random 3SAT to create hard RuleTaker instances is shown in Figure 6. Given that RuleTaker

taken from the RuleTaker templates from Figure 7. Treat

Step 3 Translate rules and facts to English using the RuleTaker templates from Figure 7. Treat some facts as conjectures, or the queries to be proven given the Rules and Facts.

Figure 6: An illustration of the retrofitting algorithm used to find hard RuleTaker theories (rule and facts) from random 3SAT using a contrived example with grounded rules over 5 variables.

Ground Rules
If the c is (not) X then the c is (not) Y.
If the e is (not) X and the c is (not) Y then the c is (not) Z.

Quantified Rules
If something is X and (not) Y then it is (not) Z. If something is (not) X then it is (not) Y. All X, Y things are (not) Z

The original RuleTaker dataset includes instantiations of the above single rule that cover the 4 distinct atomic forms.11

11Some corresponding rules from the original dataset: If the tiger is not big and the tiger is not blue then the tiger is not green. If the mouse is kind and the mouse is green then the mouse is blue. If the tiger is young and the tiger is big then the tiger is not blue. If the tiger is not blue and the tiger is not young then the tiger is not green.

2SAT clauses (i.e., rules corresponding to clauses with two propositions, e.g., If the lion is red then it is rough, in clausal form: ¬A ∨ B) and units (i.e., clauses with single propositions, e.g., The lion is red or A), a particular difficulty is converting random 3SAT formula (where each clause contains exactly 3 propositions, e.g., A ∨ B ∨ C) to such forms.

Our idea is to modify Algorithm 1 to allow for repeated clause variables that we can subsequently convert to 2SAT and units; technically this amounts to altering line 7 to allow for sampling with replacement such that we can produce clauses of the following form: A ∨ A ∨ A that we can convert to facts (e.g., The lion is red). As in the ordinary application of Algorithm 1 such a procedure can be performed to produce boolean formulae containing a differing number of variables. To keep the problems of comparable size to the original RuleTaker, we created problems using a mixture of 5, 6, 7 boolean variables. As a consequence, these problems are still of relatively low complexity comparing to the types of reasoning problems we pursue in our new datasets.

RuleTaker version For our experiments, we use the open world assumption (OWA) version of RuleTaker from Tafjord, Mishra, and Clark. In contrast to the initial version of the dataset from Clark, Tafjord, and Richardson, which makes a closed-world assumption (CWA) and is limited to two-way entailment classification, the OWA include three classes: Yes (entailment), No (contradiction), Unknown.

To verify the correctness of the semantics, we compared against a manual SAT-based and SMT-based implementation of the RuleTaker language, which is available at https://github.com/allenai/language_fragments. We found around 1% mismatched labels between the official dataset due to apparent errors in the translation from the CWA dataset and performed experiments on the corrected version of the dataset.

12Publicly available at: https://allenai.org/data/proofwriter. We trained models on the depth 3-text, which was empirically shown to have high generalization across the different depth reasoning tasks in Clark, Tafjord, and Richardson (2020).

The lion is red
Rules: The lion is not red and the lion is round then the lion is not green. If the lion is red or the lion is young then the lion is not rough...
Facts: The lion is not green... Conjecture: The lion is red.