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

Kyle Richardson, Ashish Sabharwal

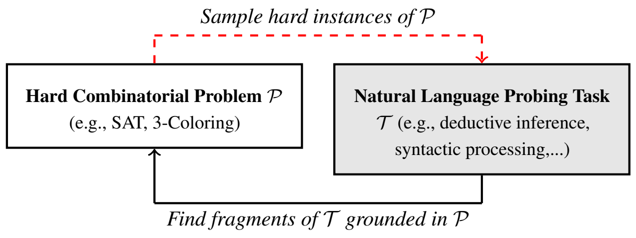
Allen Institute for AI, Seattle, WA, USA
{kyler,ashishs}@allenai.org

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 pretrained transformers (Devlin et al. 2019; Raffel et al. 2020) on a wide range of natural language understanding (NLU) benchmarks (Wang et al. 2019b,a; Xu et al. 2020), there has much 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*).


Sample hard instances of $\mathcal{P}$
Hard Combinatorial Problem $\mathcal{P}$
(e.g., SAT, 3-Coloring)
Natural Language Probing Task
$\mathcal{T}$ (e.g., deductive inference, syntactic processing,...)
Find fragments of $\mathcal{T}$ grounded in $\mathcal{P}$

NL Theory	Γ ={ *Bob is round*. *Alan is blue, rough and young*. If someone is round then they are big. All rough people are green. Big people are not green. }
Conjectures	**1.** Alan is green (**entailment**, $\Gamma \models \mathbf{1}$) **2.** Bob is green (**contradiction**)
Satisfiability	Γ has an interpretation (**sat**) $\Gamma \cup \{\neg\mathbf{1}\}$ (**unsat**), *indirectly proves* $\Gamma \models \mathbf{1}$ $\Gamma \cup \{\mathbf{2}\}$ (**unsat**), *indirectly proves* $\Gamma \models \neg\mathbf{2}$

Figure 1: TOP: An illustration of our general methodology for constructing hard natural language reasoning problems for a task $\mathcal{T}$, by grounding them into a hard combinatorial problem $\mathcal{P}$ and sampling hard instances of $\mathcal{P}$. BOTTOM: An example of a *natural language (NL) theory* (i.e., set of arbitrary *facts* and *rules*) Γ along with two example *conjectures* (i.e., propositions to be proved) and the relationship between entailment and satisfiability.

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 Baroni (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 algorithmic 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 green* is false in Figure 1 requires making 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.[1] 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 $\mathcal{T}$ such as deductive inference over statements expressed in natural language, the key idea is to identify subsets of $\mathcal{T}$ that map to a known hard combinatorial reasoning problem $\mathcal{P}$ such as Boolean satisfiability (SAT), and devise methods to sample hard instances of $\mathcal{P}$ in order to arrive at hard instances of $\mathcal{T}$.

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; Gontier 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 intractably 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 4 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, and Levesque 1996; Cook and Mitchell 1997) to sample hard problems from the critical regions of SAT phase transitions

[1]More technically, such a query can be answered via a linear-time process called *unit propagation* (cf. Zhang and Stickel (1996)), which is often treated as a pre-processing step in many modern theorem provers and SAT solvers.

Algorithm 1: Dataset construction via random SAT

```
Input: Variables set $V = \{v_1, ...v_n\}$ of size $n$, natural language
    templates $\mathcal{R}$ and variables $\mathcal{P}$, 2SAT to 3SAT interpolation parameter $p_{\text{int}}$, negation parameter $p_{\text{neg}}$, clause variable ratio $\alpha$
    range $(\alpha_{\min}, \alpha_{\max})$, STOP condition
Output: NLSat dataset
 1: $\mathbf{D} \leftarrow \{\}$                                        ▷ initialize dataset
 2: repeat
 3:   $P \leftarrow \{\}$                                             ▷ problem/clause set
 4:   $m \sim$ choose $m$, s.t. $\alpha_{\min} \leq \frac{m}{n} \leq \alpha_{\max}$
 5:   for $i := 1$ to $m$ do                                    ▷ generate $m$ clauses
 6:     $s \sim$ choose clause size $k \in (3,2)$ with prob. $(p_{\text{int}}, 1 - p_{\text{int}})$
 7:     $\mathbf{V}' \sim$ choose $s$ unique variables from $V$
 8:     $\mathbf{C} \leftarrow$ negate each $v \in V'$ with $p_{\text{neg}}$                  ▷ new clause
 9:     $t \sim$ choose NL template from $\mathcal{R}$ of size $s$
10:     $d \leftarrow$ instantiate $t$ over $\mathbf{C}$ using variables from $\mathcal{P}$
11:     $\mathbf{P} \leftarrow \mathbf{P} \cup \{d\}$
12:   $\mathbf{D} \leftarrow \mathbf{D} \cup P$
13: until dataset STOP condition is met
```

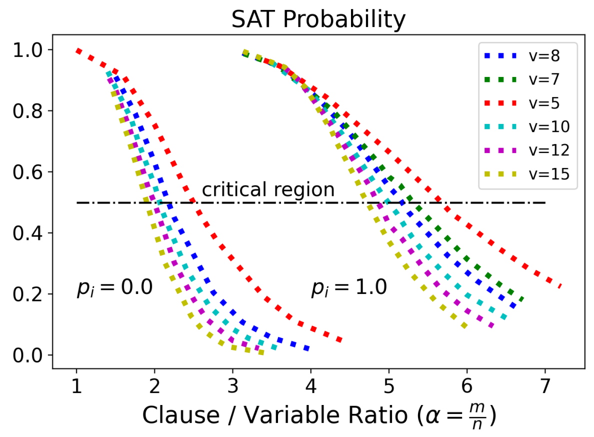


Figure 2: Illustration of phase-change and SAT probability for random 2-SAT($p_{\text{int}} = 0$) and 3-SAT($p_{\text{int}} = 1$) problems over a randomly sampled set of examples with varying α and number of variables (5-15).

(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 pretrained 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

Natural language 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 (Pratt-Hartmann 2010).[2] 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.

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_{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_{int} (shown on **line 6**, (Monasson et al. 1999)). Using a suitable fragment of natural language $\mathcal{R}$ (see next section), our version additionally includes translating each random clause to expressions in natural language (**lines**

[2]As we show in the lower part of Figure 1, logical entailment and satisfiability (or its complement) end up being interedeucible notions for the logics under consideration. For basic results on the connection, see Davis, Sigal, and Weyuker (1994)[Theorem2.1,p252].

Grounded Rule Language (GRL)	If (no) **X** and (no) **Y** then (not) **Z**.
Grounded Relative Clause Fragment (RCL)	Every **X** who is (not) a/an **Y** is (not) a/an **Z**. No **X** who is (not) a/an **Y** is a/an **Z**. Everyone who is (not) a/an **X** and (not) a/an **Y** is (not) a/an **Z**. **c** is (not) a/an **X** or (not) a/an **Y** or (not) a/an **Z**.

Figure 3: A syntactic description of the two rule languages used for our experiments.

Language	**Example Expression**	**Satisfying Assignment**
Propositional Logic (3SAT)	$(\neg \mathbf{v}_1 \vee \mathbf{v}_{15} \vee \mathbf{v}_{13}) \wedge (\neg \mathbf{v}_{13} \vee \neg \mathbf{v}_{12} \vee \neg \mathbf{v}_1) \wedge (\mathbf{v}_1 \vee \mathbf{v}_{15} \vee \neg \mathbf{v}_{13}) \wedge \ldots$	v_1=**false**, v_{15}=**false**, v_{13}=**false**, v_{12}=**true**...
	Natural Language Fragments	
Grounded Rule Language (GRL)	If carrot and not steak then apples, If apples and grapes then no carrots. If no carrots and no steak then not apples...	**needed:** *carrots, apples, grapes,...*
Relative Clause Fragment (RCL)	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...	John can be a **doctor**, a **baker**, not a **philosopher** and not a **gardener**...

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.

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 $\alpha_{\min}, \alpha_{\max}$). This parameter gives rise to phase-change behavior that has implications for problem hardness (Hayes 2003). Such phase changes are illustrated in Figure 2, where α (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 is it 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:

$$\pm \mathbf{v}_1 \vee \pm \mathbf{v}_2 \vee \pm \mathbf{v}_3 \tag{1}$$

can be translated as *If (no)* $\mathbf{v}_1$ *and (no)* $\mathbf{v}_3$ *then (no)* $\mathbf{v}_3$ (using the standard rules of logical equivalence) where each variable $\mathbf{v}_j$ 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 *a*/*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.[3] 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* $\mathbf{X}$ *who is (not) a* $\mathbf{Y}$ *is (not) a* $\mathbf{Z}$, which, via its translation from first-order logic:

$$\forall x.\ \mathbf{X}(x) \wedge \pm \mathbf{Y}(x) \rightarrow \pm \mathbf{Z}(y), \tag{2}$$

corresponds to boolean clauses of the form $\neg \mathbf{v}_1 \vee \pm \mathbf{v}_2 \vee \pm \mathbf{v}_3$ containing up to two positive literals (where each variable corresponds to a count noun, or predicates $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$). To allow

[3]Similar arguments are used to justify compositional reasoning probing benchmarks, such as SCAN (Lake and Baroni 2018), which deviate even more sharply from ordinary natural language.

for clauses with up to three positive literals, we add the rule template *Everything that is (not) an* $\mathbf{X}$ *and (not) a* $\mathbf{Y}$ *is (not) a* $\mathbf{Z}$, 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 (Westerståhl et al. 1984; Szymanik et al. 2016) and work on utilizing propositional logic for various reasoning problems in classical AI (Kautz, Selman et al. 1992; Kautz, McAllester, and Selman 1996). More specifically, grounding formulas such as Equation 2 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 **c**); 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 3). 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 **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 2, allowing us to find the hard cases in the critical region.

Sampling Strategies and Proposed Datasets

A summary of our datasets is shown in Table 1, along with a comparison to the standard RuleTaker dataset converted to SAT.[4] As described above, the NLSat instances that constitute the grounded rule language ($\mathbf{GCL}_{5,12}$) and the grounded relative clause fragment ($\mathbf{RCL}_{16,70}$) are characterized by the number of variables contained in their underlying Boolean formulae (with $\mathbf{d}_{\#vars}$ 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 1 (henceforth, **hard** sampling[5]), which we later show leads to advantages over to both **naive sampling** (i.e., choosing instances randomly within a large range) and **biased sampling** (i.e., sampling *easy* instances from the extreme ends of the phase change) strategies (see Figure 5). Formulae and their translations are then randomly split into train and evaluation sets using a 80% (train) / 20% (dev,test) ratio.

		Language complexity and SAT metrics		
Dataset ($\mathbf{d}_{\#vars}$)	**Size**	**Complexity** (NP-complete?)	**Conflicts (avg/med.)**	**Decisions (avg/med.)**
RuleTaker	130k	yes	0.0,/0.0	6.6/0.0
$\mathbf{GRL}_{5,12}$	187k	yes	3.4/4.0	5.4/4.0
$\mathbf{RCL}_{16,70}$	219k	yes	7.6/6.0	29.7/6.0
GRL-eval$_{20,50}$	17k	yes	22.0/13.0	29.3/13.0

Table 1: 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 *conflicts* and *decisions*.

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-shell theorem prover Z3 (De Moura and Bjørner 2008), we report the average/median (**avg/med.**) number of **decisions** (e.g., number of variable assignments after pre-processing) and **conflicts** (amount of backtracking performed for obtaining more complex proofs) needed by its *solve* method on each datasets. While such statistics are often tied to the internals of a solver (especially #*decisions*), there still are some notable observations.

We see from Table 1 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 **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 propositions similar to RuleTaker theories, we are also able to construct a substantially harder RuleTaker dataset (model performance to be discussed in Table 4).[6]

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 1) compared to the much harder SAT instances encountered in the mainstream SAT literature (Järvisalo et al. 2012). The decision to limit problems to the number of variables we did (e.g., to a maximum of 12 variables for $\mathbf{GRL}_{5,12}$), 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 Reed and De Freitas (2015), the decision also reflects the idea that we should

[4]More details about this conversion and technical details about the RuleTaker language can be found in the appendix.

[5]We also added a small number of problems around the critical region in training to encourage diversity.

[6]Such retrofitting involves modifying Algorithm 1 to allow for sampling of clause variables *with replacement* (in **line 7**) which yields clauses with repeated variables that we convert into 2SAT and units. More details are included in the appendix.

aim to train models that can perform *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$_{20,50}$ to measure scale-invariance.

Experimental Setup

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

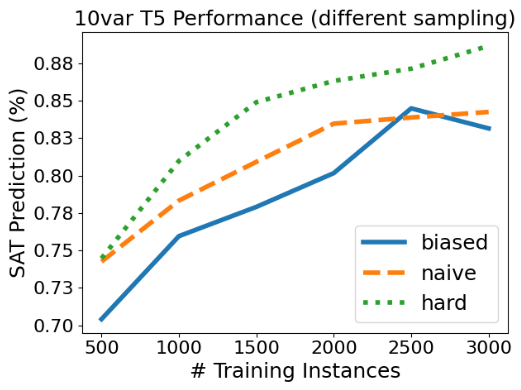
Models. Following recent studies on rule reasoning (Tafjord, Mishra, and Clark 2021), our investigation centers around the pre-trained text-to-text transformer **T5-large** model (with around $770M$ parameters)[7] (Raffel et al. 2020). We also compare against **RoBERTa** (with around $355M$ parameters) (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 by 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 $p_{\text{int}} = 0$ in Algorithm 1 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 **Xvar** column in Tables 2-3). 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 a non-pretrained biLSTM encoder model implemented using AllenNLP (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.[8]

[7]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.

[8]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.



	Accuracy%	
Model (*sampling*)	**easy**$_{5,10}$	**hard**$_{5,10}$
GRL$_{10}$ (T5) (*biased*)	88.4	77.1
GRL$_{10}$ (T5) (*naive*)	89.7	78.7
GRL$_{10}$ (T5) (*hard*)	92.4	86.4

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.

Results and Findings

Given our new set of hard algorithmic datasets, we aim to answer the following general question: *How well can our main transformer models solve these types of* hard *deductive reasoning problems?* As we describe in this section, while transformers perform well on some portion of our tasks and exhibit some degree of generalization, they still seem far from implementing the underlying algorithms needed for robust deductive reasoning. We also emphasize the following subtle point: knowing whether a model effectively solves a particular algorithmic problem or probing task such as SAT critically relies on understanding and specifying the target problem distribution that is being used for model development.[9]

Effective sampling strategies are important for training robustness and reliable evaluation. To assess the effectiveness of our *hard* sampling strategy based on random 3SAT, we performed a smaller-scale experiment that involves training 10var problems in the **GRL** language, as summarized in Figure 5. Here we see that selecting training instances

[9]Such is also a lesson from the literature on hard SAT. Quoting Mitchell and Levesque (1996): *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.*

Dev *Accuracy* % for **easy / hard** (i.i.d and **o.o.d**) instances

	main **GRL** splits (5 to 12 variables problems)				**GRL**-eval (20-50 variables)			
Model$_{num\_var}$	**5var**	**8var**	**10var**	**12var**	**20var**	**30var**	**40var**	**50var**
T5$_{5\_var}$	97.5 / 95.9	89.0 / 83.8	83.3 / 75.4	61.5 / 67.4	65.6 / 60.3	59.7 / 53.5	50.8 / 50.2	50.0 / 50.0
T5$_{8\_var}$	96.2 / 94.0	92.4 / 87.9	87.7 / 81.6	73.6 / 74.8	74.4 / 67.5	67.1 / 58.3	53.5 / 51.2	50.1 / 50.0
T5$_{10\_var}$	93.9 / 89.7	92.7 / 86.3	89.7 / 82.5	79.0 / 76.7	78.6 / 70.0	71.2 / 60.1	54.7 / 51.4	50.1 / 50.0
T5$_{12\_var}$	94.5 / 91.1	91.5 / 84.9	87.7 / 80.7	77.3 / 81.0	77.8 / 70.1	70.7 / 60.3	53.3 / 51.4	50.0 / 50.0
T5$_{5,12}$	98.6 / 98.1	96.0 / 93.6	92.6 / 89.6	85.0 / 88.5	86.5 / **80.7**	84.9 / **72.7**	69.8 / **61.4**	59.1 / **51.8**

Table 2: 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.

Grounded Rule Language **GRL**, Accuracy%

Model	**5var**	**7var**	**8var**	**10var**	**12var**	**Avg.**
Random	50.0	50.0	50.0	50.0	50.0	50.0
T5$_{5,12}$	98.0	95.4	94.3	90.7	88.3	93.4
RoBERTa$_{5,12}$	96.4	92.0	90.2	85.4	83.4	89.5

Grounded Relative Clause Language **RCL**, Accuracy%

Model	**16,21v**	**25,32v**	**35,48v**	**60,70v**	**Avg.**
Random	50.0	50.0	50.0	51.2	50.3
T5$_{16,70}$	95.9	95.3	94.7	92.9	94.7
RoBERTa$_{16,70}$	96.0	95.9	94.9	94.0	95.2

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.

from the critical regions in 3SAT phase-changes (i.e., our default **hard** sampling strategy) allows models to generalize to the overall problem distribution and across other variable sizes (top figure). In contrast, **naive**ly selecting from random parts of the distribution is not much better than selecting only from extreme (easy) ends of the distribution (**biased** sampling), and both lead to models not performing as well (8 points lower) on the hard subset of the evaluation set. Notably, the closeness of results between naive and biased sampling suggests that selecting deductive reasoning data in an *ad hoc* fashion might often give rise to certain biases that harm robustness in a way similar to entirely biased sampling.

	Model accuracy (%)		
evaluation	Majority	**RT**-T5	**RT**-RoBERTa
RuleTaker (RT) (standard i.i.d.)	43.0	97.5	98.7
Hard RT (our methodology)	50.0	57.7	59.6

Table 4: While RT models excel on standard RuleTaker evaluation, they perform close to random on a Hard RT challenge set constructed using our methodology.

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 39 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 2, 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.

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 propositional semantics). To this end, we observe that there are 4 distinct atomic forms of a 3SAT clause corresponding to 0, 1, 2, or 3 positive literals (after accounting for logical equivalences obtained via the commutativity of disjunction). All of these atomic forms, denoted $\pm\mathbf{X} \vee \pm\mathbf{Y} \vee \pm\mathbf{Z}$ (where $\pm$ indicates the literal may be positive *or* negated), can be represented by one of the aforementioned grounded rules, with appropriately placed *not* modifiers:

The original RuleTaker dataset includes instantiations of the above single rule that cover the 4 distinct atomic forms.[10] A similar argument can be made for proving the NP-completeness of our other languages (for background on NL complexity, see Pratt-Hartmann (2004, 2010)).

Retrofitting 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 theories (see again the example in Figure 1) includes both *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: $\neg A \vee B$) 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., $A \vee B \vee C$) 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: $A \vee A \vee A$ 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 We use the open world assumption (OWA) version of RuleTaker from Tafjord, Mishra, and Clark (2021)[11]. In contrast to the initial version of the dataset from Clark, Tafjord, and Richardson (2020), 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.

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 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*.

[10] Some corresponding rules from the original dataset: *If the tiger is not big and the tiger is not blue then the tiger is cold, 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.*

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

Step 1: Find random 3SAT instances with modification to Algorithm 1 that allows for clauses with **repeat**ed variables (i.e., removing the *uniqueness* constraint on line **7** to sample *with replacement*)	$(\mathbf{v}_1 \vee \neg\mathbf{v}_2 \vee \neg\mathbf{v}_5) \wedge (\neg\mathbf{v}_2 \vee \neg\mathbf{v}_3 \vee \neg\mathbf{v}_4) \wedge \underbrace{(\neg\mathbf{v}_5 \vee \neg\mathbf{v}_5 \vee \neg\mathbf{v}_5)}_{\textbf{repeat (unit)}} \wedge \underbrace{(\mathbf{v}_1 \vee \mathbf{v}_1 \vee \mathbf{v}_1)}_{\textbf{repeat (unit)}} \wedge (\mathbf{v}_3 \vee \mathbf{v}_4 \vee \neg\mathbf{v}_2) \wedge (\neg\mathbf{v}_2 \vee \neg\mathbf{v}_3 \vee \neg\mathbf{v}_1) \wedge ... \wedge \underbrace{(\neg\mathbf{v}_1 \vee \neg\mathbf{v}_1 \vee \mathbf{v}_3)}_{\textbf{repeat (2SAT)}}$
Step 2 Remove repeats, split formula into **rules** (i.e., 2/3 SAT clauses) and **facts** (i.e., units); find problems whose **rules** are satisfiable.	$\underbrace{(\mathbf{v}_1 \vee \neg\mathbf{v}_2 \vee \neg\mathbf{v}_5) \wedge (\neg\mathbf{v}_2 \vee \neg\mathbf{v}_3 \vee \neg\mathbf{v}_4) \wedge (\mathbf{v}_3 \vee \mathbf{v}_4 \vee \neg\mathbf{v}_2)...}_{\textbf{rules}(sat.)} \wedge \underbrace{\neg\mathbf{v}_5 \wedge \mathbf{v}_1}_{\textbf{facts}}$
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**.	**Rules:** *If* $\underbrace{\textit{the lion is not red}}_{\neg\mathbf{v}_1}$ *and* $\underbrace{\textit{the lion is round}}_{\mathbf{v}_2}$ *then* $\underbrace{\textit{the lion is not green}}_{\neg\mathbf{v}_5}$. *If the* $\underbrace{\textit{lion is round}}_{\mathbf{v}_2}$ *and* $\underbrace{\textit{the lion is young}}_{\mathbf{v}_3}$ *then* $\underbrace{\textit{the lion is not rough}}_{\neg\mathbf{v}_4}$*...* **Facts:** $\underbrace{\text{The lion is not green}}_{\neg\mathbf{v}_5}$... **Conjecture:** $\underbrace{\text{The lion is red}}_{\mathbf{v}_1}$.

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

Ground Rules	If the **c** is (not) $\mathbf{X}$ then the **c** is (not) $\mathbf{Y}$. If the **c** is (not) $\mathbf{X}$ and the **c** is (not) $\mathbf{Y}$ then the **c** is (not) $\mathbf{Z}$.
Quantified Rules	If something is $\mathbf{X}$ and (not) $\mathbf{Y}$ then it is (not) $\mathbf{Z}$. If something is (not) $\mathbf{X}$ then it is (not) $\mathbf{Y}$. All $\mathbf{X}$, $\mathbf{Y}$ things are (not) $\mathbf{Z}$

Figure 7: A subset of the rule templates encountered in the original RuleTaker language from Clark, Tafjord, and Richardson (2020).

clausal form	**rule translation**
$\pm\mathbf{X} \vee \pm\mathbf{Y} \vee \pm\mathbf{Z}$	*If the c is (not)* $\mathbf{X}$ *and the c is (not)* $\mathbf{Y}$ *then the c is (not)* $\mathbf{Z}$.

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.

De Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, 337–340. Springer.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Evans, R.; Saxton, D.; Amos, D.; Kohli, P.; and Grefenstette, E. 2018. Can Neural Networks Understand Logical Entailment? *Proceedings of ICLR*.

Gardner, M.; Grus, J.; Neumann, M.; Tafjord, O.; Dasigi, P.; Liu, N.; Peters, M.; Schmitz, M.; and Zettlemoyer, L. 2018. AllenNLP: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Gontier, N.; Sinha, K.; Reddy, S.; and Pal, C. 2020. Measuring systematic generalization in neural proof generation with transformers. In *NeurIPS*.

Gururangan, S.; Swayamdipta, S.; Levy, O.; Schwartz, R.; Bowman, S. R.; and Smith, N. A. 2018. Annotation artifacts in natural language inference data. In *NAACL*.

Hahn, M.; Jurafsky, D.; and Futrell, R. 2021. Sensitivity as a Complexity Measure for Sequence Classification Tasks. *TACL*.

Hayes, B. 2003. Computing Science: On the Threshold. *American Scientist*, 91(1): 12–17.

Hupkes, D.; Dankers, V.; Mul, M.; and Bruni, E. 2020. Compositionality decomposed: how do neural networks generalise? *JAIR*, 67: 757–795.

Järvisalo, M.; Le Berre, D.; Roussel, O.; and Simon, L. 2012. The international SAT solver competitions. *AI Magazine*, 33(1): 89–92.

Kassner, N.; Kroje, B.; and Schütze, H. 2020. Are Pre-trained Language Models as Symbolic Reasoners over Knowledge? In *CoNLL*.

Kautz, H.; McAllester, D.; and Selman, B. 1996. Encoding plans in propositional logic. *KR*, 96: 374–384.

Kautz, H. A.; Selman, B.; et al. 1992. Planning as Satisfiability. In *ECAI*, volume 92, 359–363.

Khot, T.; Khashabi, D.; Richardson, K.; Clark, P.; and Sabharwal, A. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. *Proceedings of NAACL*.

Lake, B.; and Baroni, M. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *ICML*, 2873–2882. PMLR.

Liang, Z.; Bethard, S.; and Surdeanu, M. 2021. Explainable Multi-hop Verbal Reasoning Through Internal Monologue. In *NAACL*, 1225–1250.

Linzen, T.; Dupoux, E.; and Goldberg, Y. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *TACL*, 4: 521–535.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mitchell, D. G.; and Levesque, H. J. 1996. Some pitfalls for experimenters with random SAT. *Artificial Intelligence*, 81(1-2): 111–125.

Monasson, 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. 2010. Computational complexity in natural language. *The handbook of computational linguistics and natural language processing*, 57.

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*.

Szymanik, 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*.

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

Wang, A.; Pruksachatkun, 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.