Identifying the limits of transformers when performing model-checking with natural language

Tharindu Madusanka and **Ian Pratt-hartmann** and **Riza theresa Batista-navarro**
Department of Computer Science
The University of Manchester

Abstract

Can transformers learn to comprehend logical semantics in natural language? Although many strands of work on natural language inference have focussed on transformer models' ability to perform reasoning on text, the above question has not been answered adequately. This is primarily because the logical problems that have been studied in the context of natural language inference have their computational complexity vary with the logical and grammatical constructs within the sentences. As such, it is difficult to access whether the difference in accuracy is due to logical semantics or the difference in computational complexity. A problem that is much suited to address this issue is that of the model-checking problem, whose computational complexity remains constant (for fragments derived from first-order logic). However, the model-checking problem remains untouched in natural language inference research. Thus, we investigated the problem of model-checking with natural language to adequately answer the question of how the logical semantics of natural language affects transformers' performance [1]. Our results imply that the language fragment has a significant impact on the performance of transformer models. Furthermore, we hypothesise that a transformer model can at least partially understand the logical semantics in natural language but can not completely learn the rules governing the model-checking algorithm.

1 Introduction

Recent years have seen a surge of interest in the application of neural networks to the topic of natural language inference (Raffel et al., 2019; Lan et al., 2020; Yang et al., 2019), the central task of which is to recover information entailed by, but not explicitly stated in natural language texts (Lin et al., 2019; Sinha et al., 2019; Geiger et al., 2018; Wang et al., 2021). This problem is of theoretical (as well as practical) interest because the ability to understand the logical consequences of natural language sentences is an essential part of what it is to understand the grammatical constructions and closed-class expressions they contain. More specifically, the ability of neural network models to recognize logical entailments is constitutive of their ability to understand the texts they are processing.

It is important to distinguish two strands of work in this area. The first focuses on *entailment* as defined by human-constructed datasets (Bowman et al., 2015; Williams et al., 2018), where the inferences depend on implicit background knowledge and have a probabilistic character. The second focuses on the recognition of formal logical entailments, for which data sets can be machine-generated using existing symbolic reasoning techniques (Richardson et al., 2020; Richardson and Sabharwal, 2021; Geiger et al., 2018). This latter strand of work is particularly pertinent to the theoretical problem of whether neural network models can learn the logical semantics of natural language. Commonsense knowledge, human judgement and considerations of plausibility are consciously excluded.

A logical problem of great theoretical interest that has not been studied in the context of natural language inference is the model-checking problem: given a formula ϕ and a structure $\mathfrak{A}$, determine whether ϕ is true in $\mathfrak{A}$ ($\mathfrak{A} \models \psi$). The ability to perform model-checking is indicative of a grasp of the logical semantics of the expressions concerned. In the context of natural language inference, we are particularly interested in a variant of the model-checking problem where the structure and the formula are interpreted in natural language. It is noteworthy to emphasise how the model-checking problem differs from other inference problems. In other logical reasoning problems such as satisfiability, computational complexity varies among multiple

[1] dataset and code available at https://github.com/iTharindu/reasoning-withing-a-structure

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 3539–3550

<table>
<tr><td colspan="2">Sturcture (Model) : $\mathcal{D} = \{Alan, Talia, Tony, Hailee, Aria\}$
$\mathcal{P} = \{actors, happy, scholars, love, admire\}$
$actors = \{Aria, Tony, Hailee\}$
$happy = \{Hailee, Talia\}$
$scholars = \{Alan, Tony\}$
$love = \{(Hailee, Talia), (Hailee, Alan), (Haillee, Tony), (Talia, Alan), (Tony, Aria)\}$
$admire = \{(Aria, Tony), (Aria, Hailee), (Talia, Hailee), (Alan, Tony), (Alan, Talia)\}$</td></tr>
<tr><td colspan="2">Structure (Model) in natural language: Alan, Talia, Tony, Hailee and Aria are people in a group. Aria Tony and Hailee are actors. Hailee and Talia are happy. Alan and Tony are scholars. Hailee loves Talia, Alan and Tony. Talia loves Alan. Tony loves Aria. Aria admires Tony and Hailee. Talia admires Hailee, Alan admires Tony and Talia</td></tr>
<tr><td>Formula : $\exists x(actor(x) \wedge \forall y(scholar(y) \Rightarrow love(x, y)))$
Formula in natural language (Sentence) :
Some actors love every scholar
Validity : True</td><td>Formula : $\forall x((actor(x) \wedge happy(x)) \Rightarrow scholar(x))$
Formula in natural language (Sentence) :
All actors who are happy are scholars
Validity : False</td></tr>
</table>

Figure 1: An instance of the model-checking problem, the domain of the structure is represented in $\mathcal{D}$, and predicates are characterised by $\mathcal{P}$. A formula can be valid or not according to the structure *(the formula on the left is valid, while the one on the right is invalid)*. Corresponding natural language representations for both the structure and the formula are also presented.

computational complexity classes (NLOGSPACE to NEXPTIME for fragments considered in this study) in language fragments of the finite variable space (Pratt-Hartmann and Third, 2006; Pratt-Hartmann, 2010). In contrast, in the model checking problem, the computational complexity remains in PTIME for any fragment in the finite variable space (given they are derived from first-order logic). Furthermore, inference in the model-checking problem is fairly straightforward, which has also been evident by low computational complexity. Hence, the model-checking problem provides an ideal problem to analyse how different logically significant words and grammatical constructs (semantics of logic in natural language) affect transformers' ability to reason, as the underlying computational complexity remains in PTIME.

Figure 1 depicts an instance of the model-checking problem, where the sentence *"Some actors love every scholar"* is *True* according to the structure presented, as the assignment of *"Hailee"* to the variable x makes the corresponding formula $(\exists x(actor(x) \wedge \forall y(scholar(y) \Rightarrow love(x, y))))$ $True$. However, when assessing the sentence *"All actors who are happy are scholars"*, there is no assignment that makes the formula $(\forall x((actor(x) \wedge happy(x)) \Rightarrow scholar(x)))$ $True$ according to the structure (the set of *actors* who are *musicians* is {*Hailee*}, which is not a subset of *scholars'*, namely {*Alan, Tony*}).

In our analysis of transformers' capabilities in the model-checking problem, we ask two fundamental questions, *(1) can transformers perform model-checking with natural language? (2) if so, can transformers understand the logical semantics of natural language: i.e. can transformers comprehend the semantics of distinctively logical words and grammatical concepts such as determiners, relative clauses and anaphora?* To answer the above-mentioned questions, we construct a model-checking dataset ($\mathcal{FO}^2$*-MC* dataset) utilising language fragments. Unlike the work by Richardson and Sabharwal (2021) and Geiger et al. (2018), whose work was limited to only one language fragment, we explore a varied set of fragments. The consideration of linguistic complexity of language fragments led us to ask an additional question: *How does the linguistic complexity of the fragment affect the performance of the transformer model when performing model-checking with natural language?*

The contributions of this paper are as follows: (1) To the best of our knowledge, we are the first to broaden natural language reasoning over formal theories to include model-checking with natural language; (2) We develop a novel algorithm for constructing a dataset for model-checking with natural language; (3) We investigate whether transformers can learn to understand the logical semantics of natural language; (4) In a first-of-its-kind study in rule reasoning, we include complex fragments such as anaphora and relative clauses with transitive verbs; and (5) We provide a systematic analysis of how the linguistic complexity of language fragments affects rule reasoning.

2 Related Work

Our work follows the literature on evaluating neural approaches, especially transformer models on deductive and linguistic reasoning tasks (Richard-

son and Sabharwal, 2021; Richardson et al., 2020; Sinha et al., 2019; Geiger et al., 2018; McCoy et al., 2019; Betz et al., 2021). Moreover, it is also related to other research approaches that have been conducted on data synthesis for rule reasoning problems (Lin et al., 2019; Weston et al., 2016; Tafjord et al., 2019). However, our study is distinct from the above-mentioned work in two ways. Firstly, we focus on an unexplored problem space, model-checking with natural language. Secondly, unlike the above literature, we explore multiple language fragments and provide a deconstruction of how the linguistic complexity of language fragments affects the performance of transformer models in a rule reasoning task.

Our work can also be viewed as broadening the research conducted on training neural networks to perform algorithmic tasks, including learning to solve SAT problems (Selsam et al., 2019; Narodytska et al., 2020), propositional inference (Evans et al., 2018), semantic parsing (He and Choi, 2020; Kamath and Das, 2019), symbolic integration (Lample and Charton, 2020) and natural theorem proving (Weber et al., 2019; Minervini et al., 2020; Saha et al., 2020; Gontier et al., 2020). In our study, we aim to investigate the transformers' ability to emulate the algorithm governing the model-checking problem and comprehend the logical semantics of natural language.

When defining language fragments, we follow the definition set out by Pratt-Hartmann (Pratt-Hartmann, 2003, 2004; Pratt-Hartmann and Third, 2006; Pratt-Hartmann and Moss, 2009), who described it more precisely as a subset of a language equipped with semantics that translates sentences into a formal system such as first-order logic. Moreover we employ their work on fragments of first-order logic as the foundation when constructing the dataset. Notably, Pratt-Hartmann (2004) has investigated the complexity of fragments' first-order logic, and we limit our analysis in this paper to fragments that have been examined in that study. Moreover, we also closely follow the cognitive science literature on model-checking and quantifier verification (McMillan et al., 2005; Szymanik et al., 2013; Szymanik and Zajenkowski, 2010) when defining our experimental evaluation. It provides us with a baseline to compare results from transformer models with the empirical studies that have been conducted with humans.

3 Data Construction

To decide whether transformer models can learn to understand logical semantics of natural language from formulae (sentences) and structures represented in natural language, we developed an algorithm (shown in Algorithm 1) to construct a balanced dataset designed to be free from trivial linguistic patterns that are easily exploitable. This section will outline the data construction methodology in detail.

We sample a set of words (Proper nouns *PrN*, nouns *N*, verbs *Vb*, adjectives *Adj*) from a predefined vocabulary ($\mathcal{V}'$) to form a list of words $\mathcal{V}$. The proper nouns in $\mathcal{V}$ are used to define the domain $\mathcal{D}$ of the structure, while the nouns, verbs and adjectives are used for defining the set of predicates $\mathcal{P}$.

When generating sentences, we follow a template-based approach. A language template is a sentence of natural language with open-clause words replaced by schematic variables; for example, *Some N_1 V is every N_2*. Through substitution of vocabulary items of the appropriate category, we can generate natural language sentences, i.e., *Some artists admire every doctor*. A simple way of defining a fragment of natural language (language fragment) is via a finite set of template sentences. For example, the classical syllogistic fragment can be defined as the sentences confirming the sentence schemata, *All N_1 are N_2, Some N_1 is N_2, No N_1 are N_2* and *Some N_1 are not N_2*. The formula template is a formula of first-order logic with non-logical symbols replaced by schematic variables; for example, $\exists x(N_1(x) \wedge \forall N_2(B(y) \Rightarrow V(x, y)))$. An instance of that formula is the result of the substitution of non-logical symbols of the appropriate type for the schematic variables, i.e., $\exists x(artist(x) \wedge \forall y(doctor(y) \Rightarrow admire(x, y)))$. A language template translates to schematic formulae in a natural way. For example, the classical syllogistic translates to the schematic formulae $\forall x(N_1(x) \Rightarrow \pm N_2(x))$ and $\exists x(N_1(x) \wedge \pm N_2(x))$.

Let $\Phi_{\mathcal{L}}$ denote the set of first-order formula templates that can be translated to natural language templates $\mathcal{L}$. Given a language fragment $\mathcal{L}$ and a vocabulary $\mathcal{V}$, we can obtain a set of formulae $\Phi_{\mathcal{L}}(\mathcal{V})$, such that $\Phi_{\mathcal{L}}(\mathcal{V})$ is a fragment of first-order logic over the vocabulary $\mathcal{V}$, i.e., $\Phi_{\mathcal{L}}(\mathcal{V})$ only contains the vocabulary $\mathcal{V}$. The first-order formula ϕ is selected from the $\Phi_{\mathcal{L}}(\mathcal{V})$, and then the formula ϕ is translated to a natural language sentence s_ϕ using a

template $\mathcal{L}$. A summary of the language fragments we used in our evaluation is provided in the next section.

Algorithm 1 Data Construction - Model-Checking with Natural Language

Input : Language Fragment $\mathcal{L}$ and its corresponding set of first order logic formulae templates $\Phi_{\mathcal{L}}$ along with its equivalent natural language templates $\mathcal{T}_{\mathcal{L}}$. Vocabulary $\mathcal{V}$ that contains Nouns*(N)*, Adjectives*(Adj)*, Verbs*(Vb)* and Proper nouns*(PrN)*. Template $\mathcal{T}$ to convert structure to natural language. Maximum number of domain elements per datapoint n, and maximum number of predicates m

Output : Model-checking dataset $\mathcal{D}$

1: $D \leftarrow \{\}$
2: **repeat**
3: $\mathcal{V} \leftarrow$ randomly select list of words where $\mathcal{V} \subset \mathcal{V}'$ such that $|\{PrN\}| \leq n$ and $|\{N \cup Adj \cup Vb\}| \leq m$
4: $\phi \leftarrow$ randomly generate first order formula using the set of first-order logic formulae $\Phi_{\mathcal{L}}(\mathcal{V})$
5: $s_\phi \leftarrow$ converts ϕ to a natural language sentence using the template $\mathcal{L}$
6: $\ell \leftarrow$ randomly generate ℓ where $\ell \in \{True, False\}$
7: $\mathcal{D} \leftarrow \{PrN\}$, $\mathcal{P} \leftarrow \{N \cup Adj \cup Vb\}$
8: **repeat**
9: $\mathfrak{A} \leftarrow$ generate structure randomly using the signature (vocabulary) $\mathcal{V}$
10: **if** *($\ell = True$ and $\mathfrak{A} \models \phi$) or ($\ell = False$ and $\mathfrak{A} \not\models \phi$)* **then**
11: *correct-structure-found* $\leftarrow True$
12: **end if**
13: **until** *correct-structure-found*
14: $\mathcal{M}_{\mathfrak{A}} \leftarrow$ converts $\mathfrak{A}$ to a natural language using a template $\mathcal{T}$
15: D $\leftarrow$ D $\cup \{\mathcal{M}_{\mathfrak{A}}, s_\phi, \ell\}$
16: **until** *stop condition is met*

The label ℓ is selected randomly from the set $\{True, False\}$. Once ℓ and ϕ are defined, the structure $\mathfrak{A} = (\mathcal{D}, \{\mathcal{P}\}^{\mathfrak{A}})$ is generated, where $\mathcal{D}$ is the domain and $\mathcal{P}$ is the set of predicates and $\{\mathcal{P}\}^{\mathfrak{A}}$ represents an interpretation of $\mathcal{P}$ in $\mathfrak{A}$ and the signature of the structure is $\mathcal{V}$. Assignment of each domain element to the $\mathcal{P}$ in the structure $\mathfrak{A}$ is done randomly, such that for every domain element d_i in $\mathcal{D}$ and predicate $\mathcal{P}_i$, $prob(d_i \textit{ assign to } \mathcal{P}_i) = p_1$ if $\mathcal{P}_i$ is a unary predicate, and for every domain element d_i, d_j in $\mathcal{D}$ and predicate $\mathcal{P}_i$, $prob((d_i, d_j) \textit{ assign to } \mathcal{P}_i) = p_2$ if $\mathcal{P}_i$ is a binary predicate. In our experimentation, we select $p_1 = 0.5$ and $p_2 = 0.75$[2], so that for each predicate $\mathcal{P}_i$, $|\mathcal{P}_i^{\mathfrak{A}}|$ is a normal distribution with a mean of approximately $\frac{|\mathcal{D}|}{2}$, so the loop (in line 8-13) terminates within a reasonable time. We iteratively build structures randomly, and perform model-checking using a model-checker until a structure that meets the criteria defined by the label is found; i.e. if $\ell = True$, then the formula is $True$ according to the structure, $\mathfrak{A} \models \phi$ and vice-versa. Once such structure is identified, it is converted into a paragraph in natural language, $\mathcal{M}_{\mathfrak{A}}$ using a template $\mathcal{T}$.

Another way to create a data point would be to generate $\mathfrak{A}$ and ϕ and perform the validity check using a model-checker to acquire the label as opposed to pre-defining the label and iteratively constructing $\mathfrak{A}$ to match the label. However, such an approach can introduce easily exploitable linguistic patterns such as having the label $False$ for most sentences containing determiners *all*, *every* or *no*, or having label $True$ for sentences containing determiners *some* or *a*.

When constructing sentences, we make sure each predicate only appears once within a sentence. So sentences like *every artist is an artist* would not be generated. Furthermore, we also remove cases where no elements are assigned to a predicate *P*$_i$ and perform re-balancing, since they also introduced easily exploitable patterns. For example, the sentence *"every musician who is a actor is happy"* is trivially $True$ if there are no *musicians* or *actors*.

3.1 Language fragments

A language fragment is defined as a language that is equipped with semantics that translates its sentences to a formal system, such as first-order logic. We defined our language fragments based on the work of Pratt-Hartmann (2004). Table 1 shows the language fragments used, along with an example for each fragment and the corresponding first-order formula. As indicated in the data construction algorithm, we employ a template-based approach when implementing both language fragments and their formal method representations. We limit our evaluations to fragments of first-order logic and bound

Language Fragment	Example
Syllogistic	Every musician is a artist $\forall x(musician(x) \Rightarrow artist(x))$
Relational syllogistic (Re-Syl)	All teachers remember some engineer $\forall x(artist(x) \Rightarrow \exists y(engineer(y) \wedge remember(x, y)))$
Relative clauses without transitive verbs (Rel)	All economists who are not happy are cynics $\forall x((economist(x) \wedge \neg happy(x) \Rightarrow cynic(x))$
Relative clauses with transitive verbs (Rel-TV)	No cynic like any scholar who is a expert $\forall x(cynic(x) \Rightarrow \forall y((scholar(y) \wedge cynic(y)) \Rightarrow \neg like(x, y)))$
Anaphora	Some judge warns no juror who hate him $\exists x(judge(x) \wedge \forall y((juror(y) \wedge hate(y, x)) \Rightarrow \neg warn(x, y)))$

Table 1: Language fragments we utilised along with an example for each of them and its corresponding first-order logic formula.

the number of functions within a formula to have a maximum of four. We also limit the maximum number of quantifiers per formula to two, producing only unary or binary formulae. The rationale is to have natural sounding sentences. As outlined in the description of the template structure provided in **Appendix A: Templates of Language Fragments**, to address the ambiguity that can arise with anaphora or relative clauses, we bind the anaphora or relative clauses to the same element. For example, anaphora always refer to the first noun in the sentence. Even though we limit our data construction and evaluation to only these fragments, we emphasise that the data construction methodology and experimental evaluation we have conducted can be executed with any arbitrary fragment of natural language.

3.2 Boolean Coordinators

One interesting experiment is to evaluate how transformer models perform when Boolean coordinators are introduced to the sentences. To that end, we used Boolean coordinators ($\bigwedge(and), \bigvee(or)$) to combine sentences and create more difficult problem instances. The resultant first-order formula Ψ of such sentences can be formed by combining individual first-order formulae using either $\bigwedge$ or $\bigvee$. Model-checking is then performed on Ψ (is $\mathfrak{A} \models \Psi$ or $\mathfrak{A} \not\models \Psi$?), and the condition in Algorithm 1 *(line 11)* is modified accordingly.

The natural language sentences are combined accordingly using the coordinating conjunctions *and* or *or*. We did not consider the case where *and* as well as *or* are present in the final sentence, since the order of operations cannot be enforced in natural language settings and hence would be ambiguous. We evaluated transformer models varying the number of coordinators *k*, where $k = \{0, 1, 2\}$, to investigate how incorporating Boolean coordinators affect the accuracy.

4 Experimental Setup

In this section, we describe the experiments we conducted in order to address our research questions.

4.1 Problem definition

Formally the $\mathcal{FO}^2$*-MC* dataset can be defined as $\{(p^{(d)}, \ell^{(d)})\}_d^{|D|}$ where $p^{(d)}$ is an instance of the model-checking problem (concatenation of the structure $\mathcal{M}_{\mathfrak{A}}$ and sentence s_ϕ delimited by a separator *SEP* token), and $\ell \in \{True, False\}$ is the label. The task is to correctly predict the label ℓ, thereby reducing it to a binary classification problem.

4.2 Transformer models

To investigate the capabilities of transformers in model-checking with natural language, we performed experiments on the $\mathcal{FO}^2$*-MC* dataset using three prominent transformer architectures: BERT, RoBERTa and T5.

BERT. Bidirectional Encoder Representations from Transformers or BERT (Devlin et al., 2018) use bi-directional conditioning in all of its network's layers to consider both the left and right context. BERT has become the standard transformer architecture and has been evaluated against many NLI datasets (Richardson et al., 2020; McCoy et al., 2019), hence we believe it provides a baseline for assessing the complexity of the task and difficulty of the $\mathcal{FO}^2$*-MC* dataset. We used the `BERT-base`

(uncased) model with around 110M parameters.

RoBERTa. Robustly Optimized BERT Pretraining Approach or RoBERTa (Liu et al., 2019) is based on the BERT architecture but trained in a more optimised manner. It has been used for rule reasoning tasks such as RuleTaker (Clark et al., 2021), and is considered as another baseline model in our experiments. We made use of the RoBERTa-base model which has around 125M parameters.

T5. Following the work done by Tafjord et al. (2021) and Richardson and Sabharwal (2021) on rule reasoning, we primarily centre our experiments around Text-to-Text Transfer Transformer or T5 models (Raffel et al., 2019). T5 frames all NLP tasks (e.g., classification, translation, semantic textual similarity) into a unified text-to-text format where both input and output are always strings; this is slightly different from BERT and RoBERTa which, when fine-tuned on classification tasks, output a class label. In our experiments, we employed two T5 models: T5-base with approximately 220M parameters and T5-large with approximately 700M parameters.

In experimenting with each of the three types of models above, we utilised the Huggingface library (Wolf et al., 2019). The transformer models are fine-tuned to predict the target label ($True$ or $False$) by optimising for the binary cross-entropy loss over the targets using the Adam optimiser (Kingma and Ba, 2015). Since the dataset is balanced (i.e., both training and test data have approximately an equal number of samples labelled as $True$ and $False$), we made use of accuracy as our evaluation metric.

4.3 Proposed Dataset and Evaluation

To answer the question of whether transformers can perform model-checking with natural language, we trained transformer models, namely, T5, BERT and RoBERTa, using the $\mathcal{FO}^2$-*MC* dataset in the manner mentioned above. During data construction, we incorporated the same vocabulary as Richardson and Sabharwal (2021), with the addition of transitive verbs where the number of verbs is equivalent to the number of adjectives. The vocabulary contains approximately 2000 names (proper nouns), 156 nouns, 64 adjectives and 65 verbs. The names are used as the domain elements while nouns, adjectives and verbs form predicates, whereas verbs constitute binary predicates while the nouns and adjectives form unary predicates. Furthermore, when generating the model, we limited the number of domain elements to be less than 4 ($|\mathcal{D}| \leq 4$) and the number of predicates to be less than 8 ($|\mathcal{P}| \leq 8$). We trained transformer models using training instances that include sentences that belong to language fragments we introduced in Section 3.1, i.e., syllogistic, relational syllogistic (Re-Syl), relative clauses (Rel), relative clauses with transitive verbs (Rel-TV) and anaphora. In each case (for each language fragment), models were trained with 500K unique data points and evaluated against a held-out 100K test set (see Table 2). Moreover, we experimented with training the models (T5-base and T5-large) using a dataset that contains sentences belonging to all the fragments, so that we could investigate how simpler fragments help transformers understand the logical semantics of natural language of complex ones (see Table 3). The training set in this experiment comprises 500K unique data points with approximately 100K data points belonging to each language fragment. The results of this experiment, along with the results depicted in Table 2, also provide the answer to the question of how the linguistic complexity of the language fragment affects the performance of transformers in model-checking with natural language. To better understand model generalisation and scale invariance, we evaluated the transformer model (T5-large) on a held-out evaluation set whose structure contains more domain elements (see Table 4) or more predicates (see Table 5) than that of the training set. To comprehend how Boolean coordinators affect the accuracy of transformer models across different language fragments, we also trained and evaluated with data points whose sentences have Boolean coordinators in them.

4.4 Results and discussion

Transformer models can solve model-checking with natural language problems with satisfactory accuracy, given adequate training instances, as depicted in Table 2. For all language fragments, transformers manage to yield an accuracy of over 70%. It is also evident from Table 2 that there is no significant difference in performance between the considered transformer models.

The linguistic complexity of the language fragments that generate the sentence has a significant impact on the overall performance, as il-

Model	Syllogistic	Re-Syl	Rel	Rel-TV	anaphora
T5-base	99.9	76.6	95.0	73.6	70.3
BERT-base	99.0	78.2	90.7	75.6	73.9
RoBERTa-base	99.6	79.2	90.1	72.1	71.0

Table 2: Accuracy of transformer models (BERT, T5 and RoBERTa) across different language fragments.

Model	All	Syllogistic	Re-Syl	Rel	Rel-TV	anaphora
T5-base	75.9	80.0	76.7	74.8	74.2	73.6
T5-large	88.2	99.8	81.8	99.3	82.3	77.7

Table 3: The transformers are trained using a dataset that contains sentences belonging to all language fragments. The results are broken down into respective language fragments, and *All* indicates the overall (average) accuracy across the language fragments.

Language Fragment	$\|\mathcal{D}\|$	$\|\mathcal{D}\|$+1	$\|\mathcal{D}\|$+2	$\|\mathcal{D}\|$+4
Syllogistic	99.8	92.2	87.5	76.1
Re-Syl	81.8	67.6	63.2	55.6
Rel	99.3	90.4	84.2	73.3
Rel-TV	82.3	67.3	62.1	56.4
anaphora	77.7	65.0	61.1	49.9

Table 4: The accuracy of the `T5-large` model evaluated on out-of-scope data; the training instances have a maximum of 4 domain elements $|\mathcal{D}| \leq 4$ while the evaluation set contains 5 ($|\mathcal{D}| + 1$), 6 ($|\mathcal{D}| + 2$), 8 ($|\mathcal{D}| + 4$) domain elements, the number of predicates remains the same between train and evaluation sets.

lustrated in Tables 2 and 3. Transformers achieve near-perfect performance for fragments such as syllogistic and Rel. However, they only achieve a moderate level of accuracy for fragments such as Re-Syl, Rel-TV, and anaphora. The later-mentioned fragments have transitive verbs, which results in the respective structures containing binary relationships. It is harder to learn binary relationships as opposed to unary ones. Furthermore, the sentences in the fragments Re-Syl, Rel-TV, and anaphora can have two quantifiers, while the sentences in fragments syllogistic and Rel are restricted to only one. As depicted in Table 6, the number of quantifiers in the sentence influences the performance of the transformer models in solving model-checking problems. Sentences with two quantifiers are more difficult to decode than sentences with only one quantifier, as evidenced by cognitive studies on quantifier verification (Szymanik and Zajenkowski, 2010; Szymanik et al., 2013), which is also unsurprising. There is a difference between the accuracy of single quantifier sentences and the average accuracy of the syllogistic fragment and Rel fragment. This difference is due to single quantifier sentences belonging to other fragments, whose sentences include transitive verbs. According to the results in Table 6, only the number of quantifiers seems to affect the performance of the transformers, and not the exact quantifier used. Cognitive studies (Szymanik et al., 2013) suggest quantifiers themselves affect human performance on model-checking problems, which is not evident here, implying human reasoning on language is somewhat different to what is occurring in transformer models.

Another linguistic property that seems to affect the performance of transformers is Boolean coordinators. **The accuracy of transformer models decreases when Boolean coordinators are introduced to the sentences**, as illustrated in Table 7. The difference in performance when the number of coordinators changes from 0 to 1 is higher than that of when it is increased from 1 to 2. However, the number of Boolean coordinators has a lower effect on accuracy compared to other linguistic properties such as the number of quantifiers.

Learning the simple fragments enables transformers to learn complex ones, as depicted in Table 3. When training data contains sentences from all the language fragments, the performance of complex fragments such as anaphora is higher than if it only includes sentences of that respective fragment. Transformer models can learn to understand logically significant words such as determiners and grammatical constructs such as relative clauses more easily from simpler fragments than complex ones. So when learning the logical semantics of complex fragments, transformers can employ this knowledge. Hence, we can hy-

Language Fragment	$\|\mathcal{P}\|$	$\|\mathcal{P}\|$+1	$\|\mathcal{P}\|$+2	$\|\mathcal{P}\|$+4
Syllogistic	99.8	96.6	96.3	95.9
Re-Syl	81.8	80.4	80.4	80.3
Rel	99.3	99.1	99.1	98.9
Rel-TV	82.3	81.4	81.2	80.6
anaphora	77.7	76.9	76.6	76.6

Table 5: The accuracy of the `T5-large` model evaluated on out-of-scope data; the training instances have a maximum of 8 predicates $|\mathcal{P}| \leq 8$ while the evaluation set contains 9 ($|\mathcal{P}| + 1$), 10 ($|\mathcal{P}| + 2$), 12 ($|\mathcal{P}| + 4$) predicates, the number of domain elements remains unchanged

number of quantifiers	quantifier (s)	Accuracy
one	overall	95.6
	$\forall$ (all)	95.8
	$\exists$ (some)	95.4
two	overall	80.8
	$\forall \circ \forall$ (all - all)	80.4
	$\forall \circ \exists$ (all - some)	81.8
	$\exists \circ \forall$ (some - all)	81.2
	$\exists \circ \exists$ some - some)	80.2

Table 6: The change in accuracy of the `T5-large` model across different quantifiers. The syllogistic and Rel fragments contain only one quantifier, while Re-Syl, Rel-TV, and anaphora fragments can have two quantifiers.

pothesise that transformers at least partially learn to understand the essence of logical semantics of natural language. Table 3 also indicates a substantial difference in performance between the `T5-base` model and the `T5-large` model. The `T5-large` model achieves an overall accuracy of 88.2% but only manages to achieve an accuracy of around 80% (Re-Syl:81.8%, Rel-TV: 82.3%, anaphora: 77.7%) for language fragments with transitive verbs. This accuracy level is lower than the accuracy that transformer models yielded in other rule reasoning benchmarks such as RuleTaker (Clark et al., 2021) and NLSat (Richardson and Sabharwal, 2021), which suggests that the $\mathcal{FO}^2$-*MC* dataset is a formidable linguistic reasoning benchmark.

Transformer models exhibit limited generalisation and scale-invariance, as illustrated in Tables 4 and 5. Even if the number of predicates increases, the accuracy of the transformer model remains relatively unchanged (see Table 5). However, if the number of domain elements increases, the model performance drastically decreases (see Table 4). The reason could be that the attention mechanism in the transformer correctly identifies which areas in the structure to examine for a given sentence, but the transformer model still cannot emulate the model-checking algorithm properly. Moreover, the degradation in performance is relatively equivalent for all language fragments, suggesting that decrement is not correlated to the grammatical structure of the sentence. Hence, we can conjecture that transformers can learn to understand the logical semantics of natural language but still cannot learn to emulate the underlying model-checking algorithm.

number of Boolean coordinators	Accuracy
$k = 0$	75.9
$k = 1$	70.7
$k = 2$	67.6

Table 7: The accuracy of the T5-base model when trained and evaluated against problem instances that have Boolean coordinators. k denotes the number of Boolean coordinators in a sentence. Each sentence contains only one type of coordinator (either *and* or *or*), if any.

5 Conclusion

We investigate the limits of transformers in an unexplored problem space of model-checking with natural language employing language fragments. We use five different language fragments and explore how linguistic complexity and other linguistic properties such as Boolean coordinators affect rule reasoning in transformer models. In a broader sense, our study is to determine whether transformer models can learn to understand the logical semantics of natural language and emulate the model-checking algorithm. We posit that transformers can learn logically significant words and grammatical constructs but fall short when learning the underlying algorithm. Moreover, different linguistic properties such as the language fragment, Boolean coordinators and the number of quantifiers have a notable impact on the learning ability of the transformers. Thus, an interesting future direction would be to investigate how these linguistic properties affect more complex reasoning tasks like natural language satisfiability.

6 Limitations

The results in our work closely follow the trends reported by prior work in the domain of identifying the limits of transformers in logical reasoning. Specifically, the transformers exhibit limited generalization beyond the underlying distribution in training data. However, due to the empirical nature of the study, it is not guaranteed that all other transformer-based models or other neural networks would exhibit the same pattern.

Moreover, the study focuses on several language fragments with varying linguistic complexity such that one would be able to quantify the influence of linguistic properties on a logical reasoning problem. However, the fragments considered in this study are not the only language fragments in existence and, as such, would limit the comprehensiveness of the discussion, and there could be other fragments of language which behave differently when evaluated against transformer models.

References

Gregor Betz, Kyle Richardson, and Christian Voigt. 2021. Thinking aloud: Dynamic context generation improves zero-shot reasoning performance of gpt-2. *arXiv preprint arXiv:2103.13033*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. 2018. Can neural networks understand logical entailment? In *International Conference on Learning Representations*.

Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. 2018. Stress-testing neural models of natural language inference with multiply-quantified sentences. *arXiv preprint arXiv:1810.13033*.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Chris Pal. 2020. Measuring systematic generalization in neural proof generation with transformers. *Advances in Neural Information Processing Systems*, 33:22231–22242.

Han He and Jinho D. Choi. 2020. Establishing Strong Baselines for the New Decade: Sequence Tagging, Syntactic and Semantic Parsing with BERT. In *Proceedings of the 33rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS'20, pages 228–233.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing. In *Automated Knowledge Base Construction (AKBC)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Corey T. McMillan, Robin Clark, Peachie Moore, Christian Devita, and Murray Grossman. 2005. Neural basis for generalized quantifier comprehension. *Neuropsychologia*, 43:1729–1737.

Pasquale Minervini, Matko Bosnjak, Tim Rocktäschel, Sebastian Riedel, and Edward Grefenstette. 2020. Differentiable reasoning on large knowledge bases and natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI February 7-12, 2020*, pages 5182–5190. AAAI Press.

Nina Narodytska, Hongce Zhang, Aarti Gupta, and Toby Walsh. 2020. In search for a sat-friendly binarized neural network architecture. In *International Conference on Learning Representations*.

Ian Pratt-Hartmann. 2003. A two-variable fragment of english. *Journal of Logic, Language and Information*, 12(1):13–45.

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

Ian Pratt-Hartmann. 2010. Computational complexity in natural language. *The handbook of computational linguistics and natural language processing*, page 43.

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

Ian Pratt-Hartmann and Allan Third. 2006. More fragments of language. *Notre Dame Journal of Formal Logic*, 47(2):151–177.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Kyle Richardson, Hai Hu, Lawrence Moss, and Ashish Sabharwal. 2020. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721.

Kyle Richardson and Ashish Sabharwal. 2021. Pushing the limits of rule reasoning in transformers through natural language satisfiability. *arXiv preprint arXiv:2112.09054*.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. 2019. Learning a sat solver from single-bit supervision. In *International Conference on Learning Representations*.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. 2019. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4506–4515, Hong Kong, China. Association for Computational Linguistics.

Jakub Szymanik, Shane Steinert-Threlkeld, Marcin Zajenkowski, and Thomas F Icard III. 2013. Automata and complexity in multiple-quantifier sentence verification. *Logic and Interactive RAtionality Yearbook 2012*, page 133.

Jakub Szymanik and Marcin Zajenkowski. 2010. Comprehension of simple quantifiers: Empirical evaluation of a computational model. *Cognitive Science*, 34(3):521–532.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. Quartz: An open-domain dataset of qualitative relationship questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5941–5946, Hong Kong, China. Association for Computational Linguistics.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.

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

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomás Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Sub-fragment	Natural Language Template	First order logic formula
with a quantifier	*D (non-)N_1 is/are (not) (a) N_2*	$\forall x(\pm N_1(x) \Rightarrow \pm N_2(x))$ *or* $\exists x(\pm N_1(x) \wedge \pm N_2(x))$

Table 8: Templates for the syllogistic fragment, D denotes the determiner while N_1 and N_2 symbolise nouns.

Sub-fragment	Natural Language Template	First order logic formula
dual quantifiers	*D_1 (non-)N_1 (does not/do not) V D_2 (non-)N_2*	$\forall x(\pm N_1(x) \Rightarrow \forall y(\pm N_2(x) \Rightarrow \pm V(x,y)))$ *or* $\forall x(\pm N_1(x) \Rightarrow \exists y(\pm N_2(y) \wedge \pm V(x,y)))$ *or* $\exists x(\pm N_1(x) \wedge \forall y(\pm N_2(x) \Rightarrow \pm V(x,y)))$ *or* $\exists x(\pm N_1(x) \wedge \exists y(\pm N_2(x) \wedge \pm V(x,y)))$
With Proper nouns quantifier in the subject	*D N (does not/do not) V P*	$\forall x(\pm N(x) \Rightarrow \pm V(x,P))$ *or* $\exists x(\pm N(x) \wedge \pm V(x,P))$
With Proper nouns quantifier in the object	*P (does not/do not) V D N*	$\forall x(\pm N(x) \Rightarrow \pm V(P,x))$ *or* $\exists x(\pm N(x) \wedge \pm V(P,x))$

Table 9: Templates for the relational syllogistic fragment, D_1 and D_2 denote determiners, P denotes Proper nouns and V represents the verb while N, N_1 and N_2 symbolise nouns.

A Templates of Language Fragments

Tables 8, 9, 10, 11 and 12 contain templates for the syllogistic fragment, relational syllogistic fragment, relative clauses fragment (without transitive verbs), relative clauses (with transitive verbs), and anaphora respectively. Each table contains natural language templates that are employed to construct sentences and their corresponding first-order formulae. As mentioned in the methodology section, we build upon the vocabulary from Richardson and Sabharwal (2021). The set of determiners includes all, every, some, a and no, where every sentence type is converted to the most natural-sounding sentences; i.e. sentences such as *every artist does not like every beekeeper* would be translated into *no artists like any beekeeper*. Each sentence that is rendered using templates of the syllogistic fragment and relative clause (without transitive verbs) fragment would include exactly one quantifier, which would determine the determiner of the sentence. The templates of the relational syllogistic, relative clause (with transitive verbs) and anaphora could comprise either two quantifiers or one (if the sentence contains proper nouns, then it would have only one quantifier).

Sub-fragment	Natural Language Template	First order logic formula
with a quantifier	*D (non-)N_1 who is/are (not) (a) N_2/A_1 is/are (not) (a) N_3/A_2*	$\forall x.(\pm N_1(x) \wedge \pm N_2/A_1(x) \Rightarrow \pm N_3/A_2(x))$ *or* $\exists x.(\pm N_1(x) \wedge \pm N_2/A_1(x) \wedge \pm N_3/A_2(x))$

Table 10: Templates for the relative clauses (without transitive verbs) fragment, D denotes the determiner and N_1, N_2 and N_3 symbolise nouns while A_1 and A_2 represent adjectives.

Sub-fragment	Natural Language Template	First order logic formula
dual quantifiers, relative clause in the subject	*D_1 (non-)N_1 who (does not/ do not) V D_2 (non-)N_2 is/are (not) (a) N_3*	$\forall x(\pm N_1(x) \wedge \forall y(\pm N_2(y) \Rightarrow \pm V(x,y)) \Rightarrow \pm N_3(x))$ *or* $\forall x(\pm N_1(x) \wedge \exists y(\pm N_2(y) \wedge \pm V(x,y)) \Rightarrow \pm N_3(x))$ *or* $\forall x(\pm N_1(x) \wedge \forall y(\pm N_2(y) \Rightarrow \pm V(x,y)) \Rightarrow \pm N_3(x))$ *or* $\forall x(\pm N_1(x) \wedge \forall y(\pm N_2(y) \Rightarrow \pm V(x,y)) \Rightarrow \pm N_3(x))$
dual quantifiers, relative clause in the object	*D_1 (non-)N_1 (does not/do not) V D_2 (non-)N_2 who is/are (not) (a) N_3*	$\forall x(\pm N_1(x) \Rightarrow \forall y((\pm N_2(y) \wedge \pm N_3(y)) \Rightarrow \pm V(x,y)))$ *or* $\forall x(\pm N_1(x) \Rightarrow \exists y(\pm N_2(y) \wedge \pm N_3(y) \wedge \pm V(x,y)))$ *or* $\exists x(\pm N_1(x) \wedge \forall y((\pm N_2(y) \wedge \pm N_3(y)) \Rightarrow \pm V(x,y)))$ *or* $\exists x(\pm N_1(x) \wedge \exists y(\pm N_2(y) \wedge \pm N_3(y) \wedge \pm V(x,y)))$
with Proper nouns	*D (non-)N_1 who (does not/ do not) V P is/are (not) (a) N_2*	$\forall x(\pm N_1(x) \wedge \pm V(x,P) \Rightarrow \pm N_2(x)$ *or* $\exists x(\pm N_1(x) \wedge \pm V(x,P) \wedge \pm N_2(x))$

Table 11: Templates for the relative clauses (with transitive verbs) fragment, D_1 and D_2 denote determiners, P denotes Proper nouns and V represents the verb while N_1 N_2 and N_3 symbolise nouns.

Sub-fragment	Natural Language Template	First order logic formula
dual quantifiers	*D_1 (non-)N_1 (does not/do not) V_1 D_2 (non-)N_2 who (does not /do not) V_2 him/her/them*	$\forall x(\pm N_1(x) \Rightarrow \forall y(\pm N_2(y) \wedge \pm V_2(y,x) \Rightarrow \pm V_1(x,y)))$ *or* $\forall x(\pm N_1(x) \Rightarrow \exists y(\pm N_2(y) \wedge \pm V_2(y,x) \wedge \pm V_1(x,y)))$ *or* $\exists x(\pm N_1(x) \wedge \forall y(\pm N_2(y) \wedge \pm V_2(y,x) \Rightarrow \pm V_1(x,y)))$ *or* $\exists x(\pm N_1(x) \wedge \exists y(\pm N_2(y) \wedge \pm V_2(y,x) \wedge \pm V_1(x,y)))$
With Proper nouns	*P (does not/do not) V_1 D (non-)N who (does not/do not) V_2 him/her*	$\forall x(\pm N(x) \wedge \pm V_2(x,P) \Rightarrow \pm V_1(P,x))$ *or* $\exists x(\pm N(x) \wedge \pm V_2(x,P) \wedge \pm V_1(P,x)$

Table 12: Templates for the relative clauses (with transitive verbs) fragment, D_1 and D_2 denote determiners, P denotes Proper nouns and V_1 and V_2 represent verbs while N, N_1 and N_2 symbolise nouns.