REASONING WITH NATURAL LANGUAGE: PROBING TRANSFORMER MODELS' ABILITY TO PERFORM FORMAL REASONING IN NATURAL LANGUAGE

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCHOOL OF ENGINEERING

2025

THARINDU MADUSANKA BATAWALA ACHARIGE
Student id: 10981817

Department of Computer Science

Contents

Word Count: 31328

List of Tables

List of Figures

Abstract

REASONING WITH NATURAL LANGUAGE: PROBING TRANSFORMER MODELS' ABILITY TO PERFORM FORMAL REASONING IN NATURAL LANGUAGE

Tharindu Madusanka Batawala Acharige

A thesis submitted to The University of Manchester
for the degree of Doctor of Philosophy, 2025

Auto-regressive pre-trained transformer models, herein referred to as transformer models, have emerged as the de facto state-of-the-art approach for performing formal reasoning over natural language. However, the extent to which these models can truly acquire the logical semantics of natural language and the rules of inference remains a subject of ongoing debate within the research community. Accordingly, in this thesis, we pose the following central question: Can transformer models learn the logical semantics of natural language and the rules of inference? To answer this question, we identify two key problems that are particularly well-suited to provide insight into this matter. First, we explore the problem of model-checking with natural language, which serves to investigate the first component of our question—namely, whether transformer models can learn the logical semantics of natural language. Second, we explore the problem of natural language satisfiability, through which we investigate whether these models are capable of learning the rules of inference.

In our exploration, we employ various language fragments by systematically varying grammatical constructs with logical significance. In the context of natural language satisfiability, such variations directly influence the computational complexity of the problem space, enabling us to examine the impact of computational complexity on

the reasoning capabilities of transformer models. To ensure a rigorous and faithful evaluation, we take deliberate measures to avoid common pitfalls associated with data synthesis, particularly those arising from random sampling. For instance, in the case of natural language satisfiability, we ensure that the datasets contain a sufficient number of challenging problem instances by sampling from the phase-change region—the region in the clause-variable space where the probability of satisfiability is approximately 0.5. By adopting such a methodologically sound evaluation framework, we derive several key insights. First, linguistic constructs such as generalised quantifiers, binary predicates, and anaphora exert a significant influence on the ability of transformer models to solve model-checking tasks. Second, transformer models can learn the logical semantics of natural language if the underlying problem is simple. Third, the computation complexity of the problem has a significant effect on transformer models' ability to perform reasoning. Fourth, despite their impressive capabilities, these models are still unable to learn the rules of inference.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Publications

First-author publications that are described in this thesis are as follows:

1. **T. Madusanka**, R. Batista-navarro and I. Pratt-Hartmann. Identifying the limits of transformers when performing model-checking with natural language. *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 3539–3550, 2023.

2. **T. Madusanka**, I. Zahid, H. Li, I. Pratt-Hartmann, R. Batista-Navarro. Not all quantifiers are equal: Probing transformer-based language models' understanding of generalised quantifiers. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8680-8692, 2023.

3. **T. Madusanka**, I. Pratt-Hartmann and R. Batista-navarro. Natural language satisfiability: Exploring the problem distribution and evaluating transformer-based language models. *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 15278-15294, 2024.

4. **T. Madusanka**, M. Valentino, I. Zahid, I. Pratt-Hartmann and R. Batista-navarro. Unravelling the Logic: Investigating the Generalisation of Transformers in Numerical Satisfiability Problems. under review at *Association for Computational Linguistics (ACL)*, 2025.

Other publications are as follows,

1. I. Zahid, **T. Madusanka**, R. Batista-Navarro and Youcheng Sun. Probing the Uniquely Identifiable Linguistic Patterns of Conversational AI Agents. under review at *Findings of the Association for Computational Linguistics (ACL)*, pages 4612-4628, 2024.

2. H. Li, Y. Wu, V. Schlegel, R. Batista-Navarro, **T. Madusanka**, I. Zahid, J. Zeng, X. Wang, X, He, Y. Li and G. Nenadic. Which side are you on? A multi-task

dataset for end-to-end argument summarisation and evaluation. under review at *Findings of the Association for Computational Linguistics (ACL)*, pages 133-150, 2024.

3. I. Zahid, Y. Chang, **T. Madusanka**, Y. Sun and R. Batista-Navarro. Multi-Loss Fusion: Angular and Contrastive Integration for Machine-Generated Text Detection. under review at *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7189-7202, 2024.

4. I. Zahid, **T. Madusanka**, Y. Sun and R. Batista-Navarro. Reshaping Linguistic Identity in MultiModal Authorship Attribution. under review at *Transactions of the Association for Computational Linguistics (TACL)*, 2025.

Achievements

1. **Best Paper Award** at ACL 2024 for the paper titled *"Natural language satisfiability: Exploring the problem distribution and evaluating transformer-based language models"* (first-author paper).

2. **Outstanding Paper Award** at EACL 2023 for the paper titled *"Identifying the limits of transformers when performing model-checking with natural language"* (first-author paper).

3. **Runner-up in Carole Globe Award** for Best Student Paper 2024 for the paper titled *"Identifying the limits of transformers when performing model-checking with natural language"*. (first-author paper)

4. Successfully complete an **internship at Amazon**.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my family. To my mother, who has always been by my side and took on the near-impossible task of raising me as a single parent after my father's passing—your strength and love have been my foundation. To my late father, who instilled in me the value of education, I hope I have made you proud. To my brother and sister, who took on my father's role and shielded me from the pain and agony of losing a parent—I am forever grateful. A special shout-out to my fiancée, Nimantha, for lifting my spirits during stressful days—you have been my light in the toughest moments.

I am forever grateful to my supervisors, Dr Ian Pratt-Hartmann and Dr Riza Batista-Navarro, for giving me this incredible opportunity, for their invaluable feedback throughout my PhD journey, and for helping me navigate both research challenges and bureaucratic hurdles. To Iqra, Erick, and Mitchell—thank you for all the fun moments in office 2.122; your companionship made this journey so much more enjoyable.

I would also like to give a special shout-out to my friends from the University of Moratuwa, who, in our own unique way, called ourselves "Pulli Pokura." This wonderful group includes one of my closest friends, Durashi, along with Kalana, Lakmali, Charith, Randika, and many others who have been part of my journey. Finally, I am forever grateful to one of my closest friends, Neranjanae, who pushed me to apply for a PhD and helped me with the applications. You laid the foundation for this journey, and I couldn't have done it without you.

To anyone I might have unintentionally forgotten, please know that your support has not gone unnoticed. I am deeply grateful to everyone who has been a part of this journey.

"Mistakes is the word you're too embarrassed to use. You ought not to be. You're a product of a trillion of them. Evolution forged the entirety of sentient life on this planet using only one tool: the mistake."

— Dr. Robert Ford (Anthony Hopkins)
Westworld

Chapter 1

Introduction: The Problem

"It's not that I'm so smart,
it's just that I stay with problems longer."
– Albert Einstein

The capacity to reason is at the heart of intelligence, underpinning the remarkable cognitive abilities that have positioned the human race as the most dominant species on the planet. Furthermore, it has yielded marvels that rival Mother Nature herself. Reconstructing this potential in a computer system has been a long-standing research interest. Computer systems have shown an impressive ability to reason on structured data, utilising the coded logical apparatus. However, they notoriously struggle with unstructured data in the form of text, audio and images. Reasoning over natural language is particularly compelling as most knowledge in human history has been condensed into books and other textual formats. Indeed, a robust natural language reasoner would have widespread applications in the domains of search and information retrieval, coding agents, conversational agents, etc. Furthermore, it will also lay the foundation when creating human-level conscious intelligence, which is commonly referred to as artificial general intelligence (AGI).

There has been a resurgence in research on natural language reasoning fuelled by two factors. First, the development of large-scale benchmarks, such as the Stanford Natural Language Inference (SNLI) [14] and the Multi-Genre Natural Language Inference (MNLI) [153] datasets, has significantly facilitated the training of neural network-based approaches. Second, the introduction of the transformer architecture [141] has

brought about a paradigm shift in the field. Indeed, the pre-trained transformer models[1], when fine-tuned for specific language objectives have achieved the state-of-the-art in many reasoning benchmarks [108, 30, 160, 76], sometimes even outperforming humans [89, 50]. The scaling laws [62, 52], which suggest that model performance improves linearly as the model size increases exponentially, have driven researchers to develop progressively larger models that frequently surpass their smaller counterparts in performance. Furthermore, the introduction of instruction fine-tuning, frequently employing reinforcement learning through approaches such as reinforcement learning through human feedback (RLHF) [95, 167], has significantly advanced the capabilities of these models. This approach allows models to achieve human-level performance with only a few examples by employing various prompting techniques, such as chain-of-thought [149], self-consistency [145], tree-of-thought [161], and scratchpad [92], without the need for fine-tuning. Indeed, many researchers have argued that large-scale transformer-based language models (LLMs) exhibit emergent reasoning abilities when the number of parameters surpasses a certain threshold [148, 17]. However, subsequent research work has challenged this claim [115], creating an open debate within the research community. Therefore, in this work, we ask a crucial but simple question: *Can transformer models learn the logical semantics of natural language and rules of inference?*

When considering reasoning in natural language text, it is essential to distinguish the two strands of research in this domain. The first, which we shall refer to as *informal reasoning*, encompasses reasoning or entailment processes that rely on implicit background knowledge or common sense, and exhibit a probabilistic character [124, 73, 123]. Although this type of reasoning is advantageous for benchmarking machine-learning approaches, it is inadequate for investigations aimed at determining whether a specific model architecture can indeed learn rules of inference and logical semantics. The second, which we refer to as *formal reasoning*, encompasses reasoning or entailment processes that are defined in a purely logical manner [111, 131]. This type of inference is useful for probing how various linguistic factors of logical significance can affect models' ability to learn rules of inference and logical semantics. Furthermore, in contrast to informal reasoning, where datasets are frequently human-annotated, formal reasoning allows for the generation of datasets through symbolic learning techniques. This symbolic learning-based data generation affords investigators greater control over

[1]For brevity, we use "transformer models" or "transformers" to refer to auto-regressive transformer-based language models

the research process and provides a mechanism for manipulating data configurations to elucidate model behaviour patterns. Consequently, in our study to understand the extent to which transformers learn logical semantics and rules of inference, we consider two fundamental problems of formal reasoning: the problem of model-checking and the problem of determining satisfiability. For both problems, we consider variants in which the formulae and structures involved are translated into natural language.

1.1 Problem Statements

1.1.1 Model-Checking with Natural Language

A structure $\mathfrak{A}$—in the model-theoretic sense—is defined as a tuple consisting of a non-empty set, referred to as the domain of quantification $\mathcal{A}$, along with an interpretation of the signature Σ. The signature is the formal specification of non-logical symbols such as constants, predicates, etc (refer to the section 2.1.3 for a more detailed formal explanation). Then, the problem of model-checking is as follows:

Given: A structure $\mathfrak{A}$ and a formulae ϕ
Return: *True* if ϕ is true according to $\mathfrak{A}$ ($\mathfrak{A} \models \phi$), *False* otherwise.

In this thesis, we are considering a variant of the model-checking problem where $\mathfrak{A}$ and ϕ are translated to natural language, henceforth referred to as *model-checking with natural language*. In this thesis, we are particularly interested in language that is equipped with semantics which translates its sentences into some formal system such as first-order logic, formally such languages are known as *language fragments*. Consequently, the formulae examined in this thesis are restricted to those that can be directly translated into natural language sentences in a manner comprehensible to a competent speaker. Then, the problem of model-checking with natural language is as follows:

Given: Natural language translation of a structure $\mathfrak{A}$ and a formula ϕ denoted by $\mathcal{M}$ and s respectively.
Return: *True* if $\mathfrak{A} \models \phi$, *False* otherwise.

Figure 1.1 depicts an instance of the model-checking problem where the sentence "*Every scientist admires some engineer*" is *True* according to the structure presented. Consider the set of scientists ($\{Peter, Marie\}$); for each of them, we can find a mathematician ($\{Peter, John, Tony\}$) whom they admire. Conversely, the sentence "*Every*

<table>
<tr><td colspan="2">Structure: Domain = {Peter, Tony, Talia, Scarlett, Taylor, Arianna, John, Marie}
Predicates = {actor, musician, scientist, mathematician, happy, love, admire}
actor = {Tony, Scarlett, Arianna}
musician = {Talia, Taylor, Arianna}
engineer = {Tony, Taylor, Scarlett}
scientist= {Peter, Marie}
mathematician = {Peter, John, Tony}
happy = {Tony, Taylor, Scarlett, John}
love = {(Peter, Arianna), (Talia, Scarlett), (Talia, John)}
admire = {(Peter, John), (Peter, Tony), (Peter, Talia), (Scarlett, Arianna), (Scarlett, Taylor), (Marie, John)}</td></tr>
<tr><td colspan="2">Structure in Natural Language: Peter, Tony, Talia, Scarlett, Taylor, Arianna, John and Marie are people in a group. Tony, Scarlett and Arianna are actors. Talia, Taylor and Arianna are musicians. Tony, Taylor and Scarlett are engineers. Peter and Marie are scientists. Peter, John and Tony are mathematicians. Tony, Taylor, Scarlett and John are happy. Peter loves John. Talia loves Scarlett and John. Peter admires John, Tony and Talia. Scarlett admire Arianna and Taylor. Marie admire John.</td></tr>
<tr><td>Sentence: Every scientist admires some engineer</td><td>Sentence: Every actor who is happy is a musician</td></tr>
<tr><td>Formula :
$\forall x(scientist(x) \rightarrow \exists y(mathematician(y) \wedge admire(x,y)))$</td><td>Formula :
$\forall x((actor(x) \wedge happy(x)) \rightarrow musician(x))$</td></tr>
<tr><td>True/False: True</td><td>True/ False: False</td></tr>
</table>

Figure 1.1: An instance of model-checking problem where a formula can be *True* or *False* according to the structure (the formula on the left is *True*, while the one on the right is *False*). Corresponding natural language representations for both the structure and the formula are also presented. When considering the problem of model-checking with natural language, we present only the natural language interpretation of the structure and sentence to the transformer model.

actor who is happy is a musician" is evaluated as *False*, as neither *Tony* nor *Scarlett*, who belong to the set of actors who are happy ($actor \cap happy = \{Tony, Scarlett\}$), are musicians.

The ability to solve model-checking problems in natural language indicates the ability to understand logical semantics from natural language text. Moreover, the problem of model-checking for grammatical constructs we consider in this study is straightforward to implement and is in PTIME. Consequently, this problem serves as an ideal framework for evaluating transformer models' ability to comprehend the logical semantics of natural language.

In this thesis, we consider a variety of grammatical constructs with logical significance. In Chapter 3, we present a preliminary study into the employment of the model-checking problem as a means to assess transformer models' capacity for comprehending the logical semantics of natural language. In Chapter 4, we extend this investigation to encompass a logical construct of longstanding interest at the intersection of logic and linguistic studies: generalised quantifiers. *Generalised quantifiers* define the semantics of sentences that include them in terms of relations between subsets of

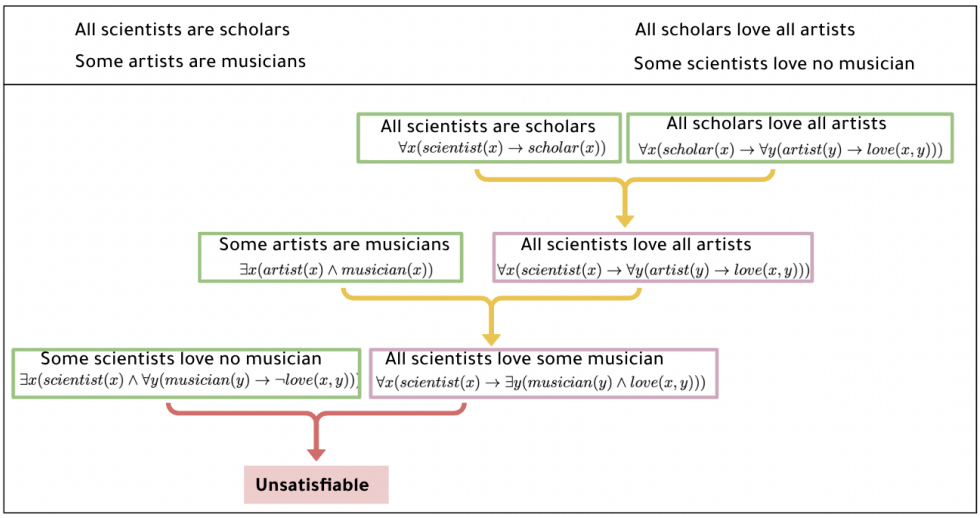

Figure 1.2: An instance of natural language satisfiability problem where the given set of sentences is unsatisfiable. As indicated in the diagram, one can find the inherent contradiction within the given set of sentences by translating them to the equivalent logical formulae.

the structure. Together, these two chapters address the first half of our overarching research question: *"Can transformer models understand the logical semantics of natural language?"*

1.1.2 Natural Language Satisfiability

The problem of determining satisfiability is as follows:

> Given: set of formulae $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$
> Return: *satisfiable* if there exists some structure $\mathfrak{A}$ such that every element of Φ is true in that structure, *unsatisfiable* otherwise

In this thesis, we consider a variant of the satisfiability problem where the set of formulae is translated to a set of natural language sentences, henceforth referred to as *natural language satisfiability*. Therefore, the problem of natural language satisfiability is as follows:

> Given: Natural language translation of a set of formulae $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ denoted by $S = \{s_1, s_2, \ldots, s_m\}$, where s_i is the natural language translation of the formula ϕ_i.
> Return: *True* if Φ is satisfiable, *False* otherwise.

Figure 1.2 depicts an instance of the natural language satisfiability problem where the set of sentences provided is unsatisfiable. From the sentences *"All scientists are scholars"* and *"All scholars love all artists"*, we can infer the sentence *"All scientists love all artists"*. The given sentence *"Some artists are musicians"* and the derived sentence *"All scientists love all artists"* together entail the sentence *"All scientists love some musician"*. However, this sentence, *"All scientists love some musician"*, contradicts the provided sentence *"Some scientists love no musician"*. Therefore, the given set of sentences is unsatisfiable. Alternatively, if the provided set of sentences includes only the first three sentences, no contradictions arise from a purely logical perspective, and the set of sentences is deemed satisfiable.

The ability to solve natural language satisfiability problems is indicative of the ability to learn rules of inference from natural language text. Moreover, unlike the model-checking problem in natural language, the reasoning involved when solving satisfiability problems we consider in this thesis is fairly intricate. Consequently, the problem of natural language satisfiability serves as an ideal framework for assessing transformer models' ability to acquire and apply rules of inference from natural language text. Furthermore, natural language satisfiability problems can belong to various computational complexity classes, depending on the language fragments to which they belong. Computational complexity classes are categories that group decision problems based on the amount of computational resources, such as time or space, required to solve them. Consequently, the study of natural language satisfiability allows for an investigation of the impact of computational complexity on transformer models' capacity to solve reasoning problems.

An essential consideration when synthesising problem instances for logical reasoning tasks—particularly those belonging to higher complexity classes—is that not all problems are hard; in fact, the majority are trivial. Thus, it is of the utmost importance to ensure that the dataset contains a sufficient number of hard instances to facilitate a rigorous and meaningful evaluation. One effective method for achieving this is to sample from the *phase-change region*: the region in the probability of satisfiability versus the clause-to-variable ratio curve where the probability of satisfiability is approximately 0.5. Empirically, this is the region where satisfiability solvers require the longest computational time to determine satisfiability, making it an ideal source of complex instances for evaluation.

In Chapter 5, we explore problems with different linguistic constructs that belong to different computational complexity classes, providing an analysis of the effect of

different linguistic constructs on the ability of transformer models to perform reasoning. In Chapter 6, we extend this work to problem instances with numerical quantifiers such as "At least K" and "At most K" where K is a natural number. Here, we place a particular emphasis on the generalisation aspect of transformer models. Together, these two chapters address the second half of our overarching research question: *"Can transformer models learn the rules of inference from natural language text?"*

1.2 Research Questions and Objectives

Our work is centred around addressing the following overarching question:

> ***Can transformer models learn logical semantics in natural language and rules of inference?***

To adequately answer the aforementioned question, we ask the following subsequent research questions and establish the following research objectives.

RQ1: Can transformer models solve model-checking problems in first-order logic?

For a neural approach to solve model-checking problems in natural language and exhibit rudimentary forms of generalisation, it must learn the logical semantics of the expressions involved. Indeed, as established earlier, this problem serves as an ideal framework for studying transformer models' ability to learn the logical semantics of natural language. However, to the best of our knowledge, this problem remains unexplored. Consequently, we establish the following research objective:

> **RO1**: Conduct a behavioural probe into the ability of transformer models to solve model-checking problem instances with various grammatical constructs and logically significant words.
>
> In our attempt to achieve research objective 1 (RO1), we ask the following research questions.
>
> **RQ1.1** How does the linguistic complexity of the fragment affect the performance of the transformer model when performing model-checking with natural language?
>
> **RQ1.1** Can learning simpler fragments enable transformer models to learn complex ones?

RQ1.1 Do transformer models exhibit scale invariance when solving model-checking problem instances?

RQ2: Can transformer models understand the logical semantics of generalised quantifiers?

The topic of generalised quantifiers has been one of the most extensively studied linguistic constructs in logic for centuries. Thus, we employ the problem of model-checking to explore the ability of transformer models to comprehend them. Consequently, we establish the following research objective:

RO2 Investigate how transformer models comprehend the logical semantics of generalised quantifiers when performing model-checking with natural language.

In our attempt to achieve research objective 2 (RO2), we ask the following research questions.

RQ2.1 How do different quantifiers affect the behaviour of transformer models?

RQ2.2 How do Boolean conjunctions affect the behaviour of transformer models when coupled with different quantifiers?

RQ2.3 Can transformer models exhibit numerical invariance when solving model-checking problem instances?

RQ2.4 What are the factors affecting transformer models' ability to solve model-checking problem instances in zero-shot settings?

RQ3: Can transformer models solve natural language satisfiability problems in first-order logic?

The problem of logical entailment can be reduced to that of determining satisfiability. Consequently, the problem of satisfiability serves as an ideal framework to evaluate the ability of neural approaches to learn rules of inference. For transformer models, particularly pre-trained ones, this necessitates consideration of natural language satisfiability: a variant of satisfiability where logical formulae are expressed as natural language sentences. Furthermore, the computational complexity class of a satisfiability problem depends on the language fragment to which the sentences belong. However, prior research involving natural language satisfiability has been primarily focused only

on propositional logic and its close relatives. Therefore, those research studies have not adequately investigated the effect of computational complexity on transformers' ability to solve natural language satisfiability problems and learn rules of inference. Consequently, we establish the following objective:

RO3 Conduct a behavioural probe into the ability of transformer models to solve natural language satisfiability problems belonging to various computational complexity classes.

In our attempt to achieve research objective 3 (RO3), we ask the following research questions.

RQ3.1 How does computational complexity affect transformer models' ability to perform a logical reasoning task?

RQ3.2 Does acquiring the ability to solve problem instances from computationally simpler language fragments—those for which the satisfiability problem, relatively speaking, belongs to a lower complexity class—facilitate transformers in learning rules of inference for more complex fragments?

RQ3.3 Do transformer models exhibit scale invariance properties when solving natural language satisfiability problems?

RQ3.4 What are the factors affecting transformer models' ability to solve natural language satisfiability problem instances in zero-shot settings?

One of the key factors to notice is that most logical problems are trivial, and worst-case computational complexity is not representative of randomly generated problems. Consequently, we establish the following sub-objective to ensure a sufficient amount of hard problems in train and test sets.

RO3 - sub objecive: Identify the phase change regions for the fragments that are used when generating problems.

RQ4: Can transformer models exhibit different types of generalisation pertaining to the numerical satisfiability problem?

Prior research on employing natural language satisfiability problems to determine whether transformers can learn rules of inference exhibits two key limitations. First, it has not been extended to problems involving numbers, herein referred to as numerical

satisfiability problems. Therefore, it does not provide any insights into the impact of numbers on transformer models' ability to learn rules of inference. Second, it does not adequately explore generalisation, as the focus has been limited to determining scale invariance. Consequently, we establish the following objective:

RO4 Explore transformer models' ability to exhibit different forms of generalisation, such as scale invariance, noise invariance and vocabulary invariance, employing the problem of numerical language satisfiability.

In our attempt to achieve research objective 4, we ask the following research questions.

RQ4.1 Can transformer models solve numerical satisfiability problems in fine-tuned and zero-shot settings?

RQ4.2 Do transformers demonstrate sensitivity to out-of-vocabulary (OOV) terms and numbers?

RQ4.3 Are transformers capable of generalising to problems of larger scale, i.e. exhibit scale invariance?

RQ4.4 How sensitive are transformers to noisy sentences which do not affect the satisfiability of the problem?

As previously discussed, it is crucial to ensure that both the training and test sets include a sufficient number of challenging examples. Therefore, in line with the work on natural language satisfiability for fragments belonging to first-order logic, we establish the following sub-objective:

RO4 - sub objective: Identify the phase change region for the numerical syllogistic fragment that is used when generating problems.

1.3 Thesis Outline

1.3.1 Chapter 2

In this chapter, we establish the background on the model-theoretic notations that appear throughout this work and establish current research work on the ability of transformer models to reason.

1.3.2 Chapter 3

In this chapter, we investigate the ability of auto-regressive transformer models to learn the logical semantics of natural language, employing a previously unexplored problem of model-checking with natural language. This exploration reveals that transformer models can at least partially understand the essence of logical semantics of natural language. Moreover, the study highlights the influence of linguistic complexity on the performance of transformer models when addressing model-checking problem instances.

This chapter is based on a paper titled "Identifying the limits of transformers when performing model-checking with natural language". The paper was accepted and published at the *Proceeding of European Chapter of the Association for Computational Linguistics: EACL 2023 (main conference)*. The publication won the ***Outstanding Paper Award at the EACL 2023***.

1.3.3 Chapter 4

In this chapter, we utilise the problem of model-checking to investigate the ability of transformer models to comprehend generalised quantifiers. The findings indicate that transformer models can understand the logical semantics of generalised quantifiers. Furthermore, the investigation uncovers striking parallels between the behaviour of transformer models and human behaviour, as documented in cognitive studies on quantifier verification.

This chapter is based on a paper titled "Not all quantifiers are equal: Probing Transformer-based language models' understanding of generalised quantifiers". The paper was accepted and published at the *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 (main conference)*.

1.3.4 Chapter 5

In this chapter, we investigate the ability of transformer models to learn the rules of inference by employing the problem of natural language satisfiability. In contrast to prior studies, which predominantly focused on propositional logic or closely related variants, this work examines the ability of transformer models to address natural language satisfiability problems within the domain of first-order logic. By choosing problem instances belonging to different computational complexity classes, we also comment on how computational complexity affects transformer models. This investigation reveals

that both computational and linguistic complexity affect the ability of transformer models to solve natural language satisfiability problems. Furthermore, it has revealed key limitations in transformers' ability to learn rules of inference.

This chapter is based on a paper titled "Natural Language Satisfiability: Exploring the Problem Distribution and Evaluating Transformer-based Language Models". The paper was accepted and published at the *Proceedings of the Annual Meeting of the Association for Computational Linguistics: ACL 2024 (main conference)*. The publication won the ***Best Paper Award at the ACL 2024***.

1.3.5 Chapter 6

In this chapter, we investigate the ability of transformer models to solve numerical satisfiability problems, with a particular focus on their capacity for generalisation. Specifically, we explore transformers' ability to show scale invariance, vocabulary invariance, numerical invariance and noise invariance. The findings reveal that the capacity of transformers to comprehend logical semantics is significantly influenced by the intricacy of the reasoning task. Moreover, the study highlights the limitations of transformer models in achieving both noise invariance and scale invariance.

This chapter is based on a paper titled "Unravelling the Logic: Investigating the Generalisation of Transformers in Numerical Satisfiability Problems". The paper is currently under review at *Association for Computational Linguistics: ACL 2025*.

Chapter 2

Background: Logic, Language and Transformers

"The true meaning of life is to plant trees
under whose shade you do not expect to sit."
– Nelson Henderson

2.1 Logic

2.1.1 Turing Machine

The Turing Machine, a foundational concept in modern computation, has profoundly influenced our understanding of computability and its limits. Proposed by Alan Turing, this abstract model serves as a universal paradigm for computation. Formally, a *Turing Machine* M is defined as a septuple $M =< Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject} >$, where:

- Q is a finite set of states, including the initial state q_0, the accept state q_{accept} and the reject state q_{reject}
- Σ denotes the input alphabet
- Γ represents the tape alphabet $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function

Let Σ^* denote the set of finite strings over Σ, including the empty string ε. A symbol is defined as either an element of Σ or one of the special characters $\sqcup$ (blank) and $\triangleright$ (start-of-tape), which are assumed to be distinct from Σ.

The Turing Machine M operates by manipulating a read/write head over an infinite tape composed of discrete cells. Each cell contains exactly one symbol, and the machine's behaviour is governed by the rules specified in the transition function. The computation process initiates from the start state and continues until either q_{accept} or q_{reject} is reached, at which point the machine halts. In the absence of reaching either of these states, M continues its operation indefinitely. A *configuration* of M is a triple consisting of a state (i.e. an element of Q), a designated tape cell (which we shall think of as the position of the head), and an infinite sequence of symbols (which we take to be written on the tape). As the Turing machine computes, the configuration changes according to the transition function, which defines a set of transitions. A *transition* is a tuple $\tau =< a,s,b,t,\pi >$ where a,b are symbols, s,t are states and $\pi \in \{L,R\}$. The start configuration of M on input w is the configuration $q_0\ w$, which indicates that the machine is in the start state q_0 with its head at the leftmost position on the tape. In the accepting configuration, the state is q_{accept} while in the rejecting configuration, the state is q_{reject}. If the Turing machine can transition from configuration C_i to C_j in a single step, we say C_i *yields* C_j. A Turing machine is *deterministic* if, for a state s and a symbol a, there is at most one transition in τ. Similarly, we can define the *non-deterministic* Turning Machine, where for a given state s and a symbol a, if the machine can proceed according to several possibilities; i.e. the transition function takes the form $\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L,R\})$, where $\mathcal{P}$ denotes the power set. It is important to recognise that every non-deterministic Turing machine has an equivalent deterministic Turing machine. A Turing machine M accepts input w if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where

- C_1 is the start configuration of M on input w,
- each C_i yields C_{i+1}, and
- C_k is the accepting configuration.

The collection of strings that M accepts is the *language of M* or the *language recognised by M*. A language $\mathcal{L}$ is said to be *Turing-recognizable* if some Turing machine recognises it, and such a language is also said to be *recursively enumerable* (or r.e.). When an input is introduced to a Turing Machine, three possible outcomes can occur. The Turing Machine reaches the accept state, reaches the reject state or continues without halting (called looping). Therefore, if a language $\mathcal{L}$ is Turing-recognisable, then a Turing machine will accept all strings of $\mathcal{L}$ and reject or loop for all strings not in $\mathcal{L}$. A language $\mathcal{L}$ is *Turing-decidable* or *decidable* if some Turing machine accepts all strings of $\mathcal{L}$ and rejects all strings that are not in $\mathcal{L}$. It is important to note that every decidable

language is Turing-recognizable, but the converse may not always hold. Under the aforementioned definitions, we can define a *problem* as a classification of some input into positive and negative instances. These inputs are typically represented as strings of symbols from a finite set or alphabet. Under this definition, a problem can be fully described by the set of its positive instances. The *Church-Turing Thesis* states that a problem is computable just in case its set of positive instances (under some natural encoding) is decidable.

In many instances, the determination of whether a problem is computable is achieved through the process of reduction. It involves transforming one problem into another in such a way that a solution to the second problem can be used to solve the first. One way to formally define reduction is through many-one reducibility. To define it, let us first define a computable function. A function $g : \Sigma^* \to \Sigma^*$ is called a *computable* if, some Turing Machine M on every input w, M halts with just $g(w)$ in its tape. A language L_1 with the alphabet Σ_1 is *many-one reducible* to language L_2 with the alphabet Σ_2, written $L_1 \leq_m L_2$, if there is a computable function $g : \Sigma_1^* \to \Sigma_2^*$, where for every $w, w \in L_1 \leftrightarrow g(w) \in L_2$. It is important to note that, under the above notations, if $L_1 \leq_m L_2$ and L_1 is undecidable, then L_2 is undecidable.

2.1.2 Time- and Space-Complexty

While a problem may be theoretically computable (i.e., decidable), its practical solvability can be significantly impeded if the corresponding algorithm requires excessive computational resources in terms of space and time. Consequently, the study of time and space complexity becomes a crucial aspect of computational logic. This analysis allows researchers to evaluate the efficiency and feasibility of algorithms beyond mere computability. Let $f : \mathbb{N} \to \mathbb{N}$ be a function, then we say M is *time-bounded* by f if, for all but almost finitely many inputs x, M terminates after at most $f(|x|)$ steps. We can define being space-bounded in an analogous manner. Let $g : \mathbb{N} \to \mathbb{N}$ be a function, then M is *space-bounded* by g if for all finite inputs x, M terminates and takes at most $g(|x|)$ cells not occupied by x (under the assumption that x is not overwritten). In many cases, rather than being interested in the exact function f that defines the time- and space-bounds, logicians are often interested in complexity classes. This approach allows for a broader categorisation of algorithmic problems based on their resource requirements.

Let F be a set of functions. Then, $\textsc{Time}(F)$ denotes the class of languages recognised by some deterministic Turing machine time-bounded by some $f \in F$. Similarly,

NTIME(F) denotes the class of languages recognised by some non-deterministic Turing machine time-bounded by some $f \in F$. We define SPACE(F) and NSPACE(F) analogously. The above notation provides the basis for defining well-studied complexity theoretic classes. For example, PTIME is the class of languages that are decidable by a Turing Machine M time-bounded by $p \in P$, where P is the set of polynomial functions ($P = \{n \mapsto n^k | k \in \mathbb{N}\}$). Formally, PTIME$= \bigcup_k$TIME$(n^k)$ where $k \in \mathbb{N}$. Similarly, we can define the following time-complexity classes,

$$\text{NPTIME} = \bigcup_k \text{NTIME}(n^k) \qquad \text{EXPTIME} = \bigcup_{p \in P} \text{TIME}(2^{p(n)})$$
$$\text{NEXPTIME} = \bigcup_{p \in P} \text{NTIME}(2^{p(n)})$$

and space-complexity classes,

$$\text{LOGSPACE} = \text{SPACE}(\lceil \tfrac{1}{2} log(n+1) \rceil) \qquad \text{NLOGSPACE} = \text{NSPACE}(\lceil \tfrac{1}{2} log(n+1) \rceil)$$
$$\text{PSPACE} = \bigcup_k \text{SPACE}(n^k) \qquad \text{PSPACE} = \bigcup_k \text{SPACE}(n^k)$$
$$\text{EXPSPACE} = \bigcup_{p \in P} \text{SPACE}(2^{p(n)}) \qquad \text{NEXPSPACE} = \bigcup_{p \in P} \text{NSPACE}(2^{p(n)})$$

To interpret the relationship between the above-mentioned complexity classes, one has to first consider *Savitch's theorem*.

Theorem 2.1 *Savitch's theorem: If* $f(n) \geq log(n)$*, then* NSPACE$(f) \subseteq$ SPACE(f^2)

Savitch's theorem leads to the corollary PSPACE = NPSPACE and EXPSPACE = NEXPSPACE. Based on these and other known relationships, we can establish a partial ordering of complexity classes:

$$\text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{PTIME} \subseteq \text{NPTIME} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}$$
$$\subseteq \text{NEXPTIME} \subseteq \text{EXPSPACE}$$

Identifying a problem within a complexity class sets an upper bound on its complexity, while reductions provide corresponding lower bounds. Indeed, complexity classes are closed under many-one log-space reductions. If language L_1 is mapping reducible to L_2 with some computable function g (every $w, w \in L_1 \leftrightarrow g(w) \in L_2$) that is computable by a deterministic Turing machine using logarithmic memory, then L_1 is *many-one log-space reducible* to L_2, written as $L_1 \leq_{\text{m}}^{\log} L_2$. As mentioned above, complexity classes are closed under many-one log-space reductions, meaning if $L_2 \in C$ and $L_1 \leq_{\text{m}}^{\log} L_2$, then $L_1 \in C$. A language L is C-hard if every language in C reduces to it, and C-complete if it is both C-hard and in C. C-complete problems are the most complex within their class, as all other problems in C reduce to them.

2.1.3 Problem of Satisfiability and Problem of Model-checking

To study the ability of auto-regressive transformer models to understand logical semantics and rules of inference, we delve into two well-studied problems in logic: the problem of model-checking and the problem of determining satisfiability. Specifically, we examine the natural language adaptations of these problems, referred to as model-checking with natural language and natural language satisfiability, respectively. Before formally defining these problems, we first introduce the foundational model-theoretic notation that underpins their formalisation.

A signature Σ is a formal specification of the non-logical symbols available in a logical language, consisting of constants, function symbols, and relation (or predicate) symbols. A structure $\mathfrak{A}$ that interprets a signature $\Sigma = (s_1, \ldots, s_n)$ is defined as a tuple $\mathfrak{A} = (\mathcal{A}, s_1^{\mathfrak{A}}, \ldots, s_n^{\mathfrak{A}})$, where $\mathcal{A}$ is a non-empty set, called the domain of quantification, and $s_k^{\mathfrak{A}}$ denotes the interpretation of symbol s_k in $\mathfrak{A}$. Consider the example structure introduced in Section 1.1.1. This structure can be represented as $\mathfrak{A} = (\mathcal{A}, actor^{\mathfrak{A}}, ..., admire^{\mathfrak{A}})$, where $\mathcal{A} = \{Peter, Tony, ..., Marie\}$, $\Sigma = \{actor, ..., admire\}$, $actor^{\mathfrak{A}} = \{Tony, Scarlett, Arianna\}$, ...

If ϕ is a formula with free variables $x = (x_1, \ldots, x_n)$, $\mathfrak{A}$ is a structure, and $a = (a_1, \ldots, a_n)$ tuple of elements of $\mathcal{A}$, then we say $\mathfrak{A} \models \phi[a]$ to indicate that the assignment $a \mapsto x$ satisfies $\phi(x)$ in $\mathfrak{A}$. We say ϕ is *True* with respect to a structure $\mathfrak{A}$ with the domain of quantification $\mathcal{A}$, if there exists $a \in \mathcal{A}^n$ such that $\mathfrak{A} \models \phi[a]$. A formula may or may not contain free variables. For example, consider the formula $\forall x \forall y(p(x,y) \wedge q(y,x))$ for some binary predicates p and q; here, both variables x and y are bound by quantifiers. In this study, we focus exclusively on formulae that do not contain any free variables, which we refer to as *closed formulae*. For a closed formula ϕ and a structure $\mathfrak{A}$, we write $\mathfrak{A} \models \phi$ to indicate that ϕ is *True* in $\mathfrak{A}$. As an example, consider the closed formula $\phi_1 = \forall x(scientist(x) \rightarrow \forall y(mathematician(y) \rightarrow admire(x,y)))$ and the aforementioned structure $\mathfrak{A} = (\mathcal{A}, actor^{\mathfrak{A}}, ..., admire^{\mathfrak{A}})$. As established in Section 1.1.1, the formula ϕ_1 is True in $\mathfrak{A}$. Consequently, we write this as $\mathfrak{A} \models \phi_1$. In contrast, for another closed formula $\phi_2 = \forall x((actor(x) \wedge happy(x)) \rightarrow musician(x))$, we established in Section 1.1.1 that it is False in $\mathfrak{A}$. Consequently, we denote this as $\mathfrak{A} \not\models \phi_2$.

Then, we can define the problem of model-checking (for a closed formula) as follows:

Given: A closed formula ϕ and a structure $\mathfrak{A}$
Return: *True* if $\mathfrak{A} \models \phi$, *False* otherwise.

We say a closed formula ϕ is *satisfiable* if there exists some structure $\mathfrak{A}$ such that $\mathfrak{A} \models \phi$. We can define satisfiability for a set of close formulae in a similar manner; a set of close formulae Φ is satisfiable if there exists some structure $\mathfrak{A}$ such that $\mathfrak{A} \models \phi$ for every $\phi \in \Phi$. Then we can define the problem of determining satisfiability as follows:

Given: A set of close formulae $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$
Return: *True* if Φ is satisfiable, *False* otherwise.

2.2 Logic and Language

The primary objective of this work is to investigate the reasoning capabilities of transformer models when applied to natural language text. To this end, we introduce two variants of the well-established model-checking and satisfiability problems, wherein the formal constructs—such as structures and formulae—are translated into natural language representations. We designate these variants as model-checking with natural language and natural language satisfiability problems.

2.2.1 Language Fragments

The interplay between language and logic has been a subject of scholarly interest for millennia, tracing back to the philosophical inquiries of Aristotle. Of particular interest to logicians is the subset of natural language that can be directly formalised into logical systems, commonly referred to as language fragments. Formally, we define a *language fragment* as a set of sentences in some natural language equipped with truth-conditional semantics in terms of which the dual notions of validity and satisfiability can be defined. Consider, for example, the set of sentences of the forms

Every p is a q	Some p is a q
No p is a q	Some p is not a q,

where p and q range over common (count) nouns such as *artist*, *beekeeper*, *carpenter* .... Their semantics can be given by the familiar translation into first-order logic over a non-logical vocabulary (of unary predicates) corresponding to the schematic variables in question:

$$\begin{array}{ll} \forall x(p(x) \rightarrow q(x)) & \exists x(p(x) \wedge q(x)) \\ \forall x(p(x) \rightarrow \neg q(x)) & \exists x(p(x) \wedge \neg q(x)). \end{array}$$

These semantics assume that universal sentences have no existential import (i.e. *Every p is a q* does not entail that there *are* any *p*'s); otherwise, however, they are uncontentious, yielding a faithful reconstruction of the notions of validity (of an argument) and satisfiability (of a set of sentences) via the usual model-theoretic definitions. In this way, we have defined a fragment of English—one corresponding, modulo translation, to the classical syllogistic of Aristotle's *Prior Analytics*, Book A [7].

Generalising, say that a *sentence template* is a sentence of some natural language in which certain open-class words have been replaced by schematic variables; and say that a *formula template* of some logic is a formula (with no free variables) in which the same schematic variables are treated as elements of the non-logical signature, appropriately typed. Then a simple way to define a natural language fragment is by means of a finite list of sentence templates paired with corresponding formula templates giving their semantics in a way judged appropriate by competent speakers. This study introduces a variety of sentence templates for sentence generation by systematically varying grammatical constructs.

For model-checking tasks, we employ distinct sentence templates that correspond to different language fragments, enabling an examination of how various grammatically and logically significant constructs influence transformer models in a reasoning task. If $\mathcal{L}$ is a language fragment, we use $MC(\mathcal{L})$ to denote the following problem:

> Given: A sentence s in $\mathcal{L}$ and a natural language translation of a structure $\mathcal{M}$,
> Return: $True$ if s is True in $\mathcal{M}$; $False$ otherwise.

The exact complexity bounds for solving model-checking problems for language fragments we consider in this study remain unexplored. Nevertheless, under the assumption of finite structures and formulae restricted to unary and binary predicates, we can establish that all language fragments $\mathcal{L}$ we consider in this study, $\mathcal{MC}(\mathcal{L})$ is in PTIME. Furthermore, as illustrated by the example in Section 1, the reasoning processes involved are relatively straightforward. This makes the model-checking problem with natural language an ideal context for investigating the extent to which transformer models capture the logical semantics of natural language. However, to the best of our knowledge, this specific problem has not yet been studied within the framework of behaviour-probing methodologies for deep learning models, such as transformers.

For the satisfiability task, we employ a variety of sentence templates belonging to several different language fragments when generating sentences. If $\mathcal{L}$ is a language fragment, we denote $Sat(\mathcal{L})$ as the following problem:

Given: a finite set Φ of sentences in $\mathcal{L}$,
Return: *True* if Φ is satisfiable; *False* otherwise.

Natural language satisfiability problems, unlike model-checking with natural language problems, have gained attention from logicians. Notably, the exact complexity bounds for several language fragments analysed in this study have already been established. For example, determining satisfiability for the classical syllogistic fragment, herein referred to as $\mathcal{S}^-$, is NLOGSPACE-complete. If we consider the extended-syllogism $\mathcal{S}$ which includes $\mathcal{S}^-$ and extends by the following templates,

Every non-p is a q	Some non-p is a q
No non-p is a q	Some non-p is not a q,

$Sat(\mathcal{S})$ is also NLOGSPACE-complete. On the other if we consider extended relational-syllogism $\mathcal{V}$ which extends $\mathcal{S}$ by the following templates,

Some/every (non-) p r's some/every (non-) q
No (non-) p r's any/every (non-) q
Some (non-) p does not r any/every (non-) q,

$Sat(\mathcal{V})$ is EXPTIME-complete. The complexity class of $Sat(\mathcal{L})$ depends on the language fragment $\mathcal{L}$.Therefore, the problem of natural language satisfiability is particularly well-suited for investigating the impact of computational complexity on the performance of transformer models. Moreover, unlike the model-checking problem, the reasoning aspect is fairly intricate, and any algorithm that effectively solves satisfiability problems requires an understanding of underlying rules of inference. Consequently, the natural language satisfiability problem represents the ideal framework to determine the extent to which transformer models can learn rules of inference from language text.

2.3 Logic, Language and Transformers

2.3.1 Rise of Transformers

Transformer models, introduced by Vaswani et al [141], represent a foundational improvement in deep learning architectures, characterised by the self-attention mechanism that facilitates the modelling of complex dependencies within sequential data. A core component of this is the multi-headed attention mechanism that mitigates the challenges of modelling long-term dependencies by allowing the model to attend to different positions in the input sequence from multiple representation subspaces simultaneously.

Specifically, each head learns a distinct attention distribution, enabling the model to capture a variety of semantic and syntactic relationships at different scales. For example, one head might focus on immediate context, while another attends to distant dependencies, with the aggregate representation preserving both local and global context. This design enables the model to capture diverse patterns across different aspects of the input. Unlike recurrent architectures, such as LSTMs [53] and GRUs [23], which process input sequentially, transformers employ a fully parallelised design, thereby significantly reducing training times and addressing challenges associated with long-range dependencies. Their emergence as state-of-the-art models is particularly evident in natural language processing (NLP), where models like BERT [30], GPT [106, 107, 16], and T5 [108] have redefined benchmarks through pretraining on massive corpora and fine-tuning for downstream tasks. This pretraining-fine-tuning paradigm has demonstrated exceptional utility in transfer learning, allowing models to generalise effectively to downstream tasks with minimal labelled data [13]. Indeed, these pre-trained models fine-tuned for downstream tasks have achieved state-of-the-art results in almost all language processing benchmarks, even in some cases outperforming the human baseline. Notably, models such as RoBERTa [76], T5 [108], and DeBERTa [50] have surpassed the human baseline in the GLUE benchmark [143], while DeBERTa has further achieved superior performance in the more challenging SuperGLUE benchmark [142].

Building upon the success of large-scale pretraining, subsequent research demonstrated that increasing model size, dataset size, and compute resources leads to predictable improvements in performance, a phenomenon formally described as scaling laws [62, 52]. These empirical findings revealed that loss functions and downstream task performance follow power-law relationships with respect to parameter count, dataset size, and training compute, highlighting the importance of training ever-larger models on high-quality data. This insight motivated the development of large language models (LLMs)[1] such as GPT-3 [16], PaLM [24], OPT [164], and Bloom [155], which exhibited phenomenal performance in few-shot and zero-shot learning, significantly reducing reliance on supervised fine-tuning. However, as model size increased, conventional supervised fine-tuning proved insufficient for aligning models with human intent, leading to the development of instruction fine-tuning approaches, where models

[1]The term Large Language Model is generally used to define large-scale transformer-based language models. Although there is no well-defined exact number of parameters required for a language model to be “large”, the general consensus is that any language model with billions of parameters is considered to be an LLM.

are explicitly trained to follow human instructions across diverse tasks [25]. A pivotal improvement in this direction was Reinforcement Learning from Human Feedback (RLHF), introduced in OpenAI's InstructGPT [95], which refines model outputs by optimising against human preference rankings, thereby enhancing alignment.

Alongside efforts to improve transformer models, researchers explored prompting techniques to leverage pre-trained models without extensive fine-tuning, specifically utilising zero-shot and few-shot prompting. Several prompting strategies, including chain-of-thought [149], self-consistency [145], scratchpad [92], and tree-of-thought [161], have demonstrated the ability to improve the accuracy of model outputs by structuring intermediate reasoning steps more effectively. Notably, these approaches achieve enhanced performance without altering the model's gradient values. These approaches, in combination with instruction fine-tuning, have drastically improved model usability and generalisation across diverse NLP applications.

The development of large-scale transformer models coupled with various prompting approaches led to the observation of emergent abilities: qualitative changes in model behaviour that arise at certain scale thresholds, such as in-context learning, compositional reasoning, and multi-step problem-solving [148, 17]. These capabilities, absent in smaller models, suggest that sufficiently large networks develop implicit representations of complex structures without explicit supervision. However, alternative explanations have been proposed, including the possibility that emergent abilities are a mirage formed by the non-linear and discontinuous evaluation metric chosen by the evaluation scheme [115]. Nonetheless, researchers and the broader public continue to speculate whether the paradigm shift introduced by transformer-based language models, achieved through scaling and novel training mechanisms, can ultimately lead to human-level conscious intelligence. A fundamental component of intelligence is reasoning, arguably its most critical aspect. Consequently, assessing these models' capacity for reasoning is essential in determining their potential to attain human-level intelligence and in guiding the development of model architectures that may achieve this capability. In our work, building upon prior work on attempting to understand the reasoning prowess of transformer models, we ask a simple question: "Can transformer models understand logical semantics and learn rules of inference?"

2.3.2 Two strands of Research into Reasoning

When considering reasoning with transformer models, it is important to recognise the two stands of research in this domain. The first strand integrates reasoning with

concepts such as common sense and background knowledge and has a probabilistic character, which we refer to as *informal reasoning* [124, 73, 123, 12, 77]. In this approach, datasets are predominantly human-annotated, and such reasoning is frequently employed to benchmark transformer models, assessing their performance relative to other models or human baselines. The second strand defines reasoning strictly based on logical or mathematical rules, deliberately excluding factors such as common sense and background knowledge, which we refer to as *formal reasoning*. Datasets here are often synthesised through symbolic learning techniques [110, 26, 131, 111].

Adversarial attacks on transformer models have demonstrated that these models frequently exploit superficial cues embedded within the datasets [43, 81]. Furthermore, new breeds of large-scale transformer models—which are typically instruction fine-tuned—have exhibited superior performance compared to human baselines [25, 1, 134], even in cases where such cues were absent from both the training and evaluation datasets. This has fuelled the research interest in the second strand of reasoning. Furthermore, this form of reasoning is ideal in behaviour probes aimed at determining transformer models' strengths and weaknesses when performing reasoning [5, 56]. Although formal reasoning presents some of the most challenging benchmarks for evaluating these models, its role in behavioural probing arguably remains the most significant use case for studying transformer-based architectures.

2.3.3 Formal Reasoning ability of Transformers

The capacity for formal reasoning remains one of the most formidable challenges for contemporary transformer-based language models. This limitation has led researchers to design increasingly rigorous benchmarks, particularly in coding and mathematical problem-solving tasks, which necessitate a high degree of structured logical inference [51, 42, 120, 57]. Notably, state-of-the-art performance on FrontierMath currently stands at 9.2%[2]. The difficulty of these tasks is underscored by the performance of GPT-4, which, despite achieving a top 10% score on a simulated bar examination [94], managed only a 1% success rate on FrontierMath [42]. Although there is a

[2]The OpenAI o3-mini model has achieved the highest reported performance on the FrontierMath benchmark. However, concerns have been raised regarding potential conflicts of interest, as OpenAI has provided financial support for the FrontierMath initiative and had access to certain problem instances within the dataset. A key factor contributing to the dataset's difficulty is its novelty; the problem instances were independently contributed by mathematicians rather than being extracted from pre-existing tests or corpora. As a result, in principle, no portion of the dataset should have been available during pre-training, except for OpenAI models. Notably, the highest score attained by a non-OpenAI model is 2%, recorded by Gemini Flash 2.0

pre-requisite of having a mathematics or coding background requirement for solving these tasks, memorisation is not particularly a problem for state-of-the-art large-scale transformer-based language models [136]. These models are trained on extensive corpora, encompassing vast amounts of publicly available web data, including textbooks, research papers, and programming repositories. Given this extensive exposure, one can reasonably infer that the requisite mathematical and coding knowledge is already embedded within the latent space of these models. However, the true difficulty arises from the models' ability to effectively retrieve, manipulate, and apply this knowledge in a structured manner to perform multi-step reasoning. This hypothesis is further supported by empirical findings demonstrating that the performance of these models significantly improves when augmented with external tools designed for stepwise reasoning. For instance, integrating language models with symbolic solvers, automated theorem provers, or code execution tools has been shown to enhance their performance on complex reasoning tasks [166, 162, 10]. Such results highlight the fundamental limitation of contemporary transformer architectures in performing abstract logical reasoning autonomously.

It is worth emphasising that the aforementioned neuro-symbolic approaches, wherein large-scale transformer models are integrated with symbolic solvers, have garnered significant interest in recent years [166, 138, 10, 20, 22, 159]. In this framework, the transformer model primarily functions as a hypothesis generator, proposing a series of logical steps that may lead to the correct solution. The symbolic solver subsequently processes these steps to derive a formally verifiable answer. This hybrid approach has demonstrated notable success in tackling complex mathematical and logical reasoning tasks, where purely neural methods struggle to perform multi-step deductions effectively. A prominent example of the effectiveness of this paradigm is DeepMind's AlphaGeometry, which achieved state-of-the-art performance in Olympiad-level geometry problem-solving. AlphaGeometry manages to solve 25 out of 30 problems, surpassing the average silver medalist score of 20/30 and closely approaching the average gold medalist performance of 25.9/30 [138]. However, not all transformer-based solvers rely on external symbolic tools. Researchers have explored the possibility of leveraging transformers independently to construct SAT solvers, theorem provers, and other automated reasoning systems [131, 4, 157]. In many cases, transformer models are fine-tuned, often using synthetic data, to perform the required reasoning steps. While these approaches have demonstrated promise, they often struggle with problems

requiring deep, stepwise reasoning, due to the intrinsic limitations of neural architectures in handling long-range dependencies and symbolic manipulation. While our study benchmarks transformer models in formal reasoning tasks under both zero-shot and fine-tuned settings, our primary objective is to systematically evaluate their strengths and limitations in performing formal reasoning.

Extensive research has focused on the behaviour-probing transformer models with a particular emphasis on generalisation. These research studies encompass various facets of generalisation, such as compositional generalisation [32, 97, 93], length generalisation [5, 166], and easy-hard generalisation [111, 70]. Compositional generalisation pertains to the model's ability to understand and generate novel combinations of known components. Research work has explored how various factors, such as model depth [99], recursive re-tuning [97], and positional encoding [93] affect transformer models' at exhibiting compositional generalisation. Although they have found configurations that lead to better compositional generalisation, Dziri et al. [32] have identified inherent limitations within transformer architectures that contribute to poor compositional generalisation. They hypothesise that transformers often tackle compositional tasks by simplifying multi-step reasoning into linearised path matching. Furthermore, due to error propagation, transformers may inherently struggle with high-complexity compositional tasks that exhibit novel patterns. Length generalisation pertains to the model's ability to handle sequences longer than those encountered during training. Researchers have explored various modifications aimed at improving the length generalisation aspect of transformer models, specifically focused on arithmetic tasks. These modifications to positional embeddings, such as Abacus embeddings [82], NoPE [64], FIRE [71], and randomised positional encodings [166]. Other methods focus on architectural changes, such as introducing looping mechanisms [36] or incorporating hand-crafted bias corrections in attention score matrices [31]. Additionally, fine-tuning approaches with various prompting approaches, such as scratchpad, have been explored to enhance length generalisation [5]. Although this has led to non-trivial length generalisation, they have found factors, such as data order and random weight initialisation, to have a significant effect on transformer models' ability to generalise. Easy-to-hard generalisation pertains to the model's ability to solve more challenging tasks than the ones encountered during training. Many instances of easy-to-hard generalisation are captured by other forms of generalisation. For instance, research on length generalisation examines how models handle increasing problem lengths, which often correlate with increased difficulty. Conversely, some studies specifically analyse easy-to-hard

generalisation from a data distribution perspective. Shin et al. [121] identify overlap data points—instances containing both easy and hard patterns—as a key mechanism for easy-to-hard generalisation. In contrast, Richardson and Sabharwal [111] found that having a sufficient amount of hard samples is crucial. In our work, we place particular emphasis on length generalisation and the easy-to-hard generalisation aspect. We modify the scale of problems during testing, affecting both their length and hardness, to assess how transformer models adapt to these changes.

We aim to study the generalisation aspect of transformer models to probe their ability to learn logical semantics and rules of inference. This has been a common theme where researchers exploit the behavioural traits of these models by their ability to exhibit various forms of generalisations. This includes these models' inability to form mental models [47], learn shortcuts [75] and prefer case-based learning over rule-based learning [56]. In our behavioural probe to understand the extent to which transformers can learn logical semantics and rules of inference, we employ two problems: the problem of model-checking with natural language and the problem of natural language satisfiability. To the best of our knowledge, the former remains unexplored in existing literature, offering a novel avenue to assess how transformer models handle various logical constructs. Although this remains a better-suited problem to study the effect of various logical constructs, prior studies have utilised problems grounded in linguistic theory to examine the impact of logical constructs—such as negation [63, 54, 91] and generalised quantifiers [29]. For instance, Ettinger [34] explored the effect of negation on transformer models employing a suite of psycholinguistic benchmarks. In our work, we aim to study the effect of different logical constructs on transformer models, thereby studying the extent to which transformer models understand logical semantics, employing a problem purely based on logic: model-checking with natural language. Unlike model-checking with natural language, some work has explored the extent to which transformer models can solve the problem of satisfiability [158, 19, 96], including natural language satisfiability [111]. Our work on natural language satisfiability closely follows the work by Richardson and Sabharwal [111] in their attempt to identify the limitations of transformer models employing natural language satisfiability problems. In contrast to prior work that primarily focuses on propositional logic or its variants, our study examines a diverse set of problems spanning various language fragments. This approach allows us to assess how different factors influence transformer models' ability to learn rules of inference.

Chapter 3

Unravelling the logical semantics: Identifying the limits of transformers when model-checking with natural language

"If you change the way you look at things,
the things you look at change."
– Wayne Dyer

This chapter is based on a paper titled *"Identifying the limits of transformers when performing model-checking with natural language"*. The paper was accepted and published at the Proceeding of European Chapter of the Association for Computational Linguistics: `EACL 2023` (main conference). Notably, the paper received the prestigious **Outstanding Paper Award**.

In this chapter, we probe the ability of transformer models to comprehend the logical semantics of natural language by employing problem model-checking with natural language.

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. Perhaps the most suitable problem for answering the aforementioned question is that of model-checking with natural language. The reasoning involved in model-checking is straightforward, and for finite structures and formulae with unary and binary predicates, it remains in PTime. 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 affect transformers' performance. 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.

3.1 Introduction

Recent years have seen a surge of interest in the application of neural networks to the topic of natural language inference [108, 160], the central task of which is to recover information entailed by, but not explicitly stated in natural language texts [73, 124, 40]. 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 [14, 153], 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 [110, 111, 40]. 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

Sturcture (Model) : $\mathcal{D} = \{Alan, Talia, Tony, Hailee, Aria\}$ $\mathcal{P} = \{actors, happy, scholars, love, admire\}$ $actors = \{Aria, Tony, Hailee\}$ $happy = \{Hailee, Talia\}$ $scholars = \{Alan, Tony\}$ $love = \{(Hailee, Talia), (Hailee, Alan), (Haillee, Tony), (Talia, Alan), (Tony, Aria)\}$ $admire = \{(Aria, Tony), (Aria, Hailee), (Talia, Hailee), (Alan, Tony), (Alan, Talia)\}$	
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*	
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**	**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**

Figure 3.1: 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.

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 $\mathfrak{A}$ ($\mathfrak{A} \models \phi$). 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 computational complexity classes (NLOGSPACE to NEXPTIME for fragments considered in this study) in language fragments of the finite variable space [104, 101]. 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 3.1 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 [111] and Geiger et al. [40], whose work was limited to only one language fragment, we explore a varied set of fragments. The consideration of the 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—in a linguistic sense—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.

3.2 Methodology

3.2.1 Data Constructions

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 $\mathcal{P}$.

As delineated in Section 2.2.1, our methodology for sentence generation employs a template-based framework. These sentence templates are derived from a diverse set of language fragments, aiming to capture the impact of various syntactic structures on the reasoning capabilities of transformer models. Let $\Phi_{\mathcal{L}}$ denote the set of first-order formula templates that can be translated to sentence 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.

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

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 sentence 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 $\mathcal{D}$

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

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*.

3.2.2 Language fragments

A language fragment is defined as a language that is equipped with semantics that translate its sentences to a formal system, such as first-order logic. We defined our

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 likes 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 hates him $\exists x(judge(x) \wedge \forall y((juror(y) \wedge hate(y,x)) \Rightarrow \neg warn(x,y)))$

Table 3.1: Language fragments we utilised along with an example for each of them and its corresponding first-order logic formula.

language fragments based on the work of Pratt-Hartmann [101]. Table 3.1 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 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 Section 3.8.1, 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.

3.2.3 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 affects the accuracy.

3.3 Experimental Setup

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

3.3.1 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.

3.3.2 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 [30] 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 [110, 81], 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 [76] is based on the BERT architecture but trained in a more optimised manner. It has been used for rule reasoning tasks such as RuleTaker [26], 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. [131] and Richardson and Sabharwal [111] on rule reasoning, we primarily centre our experiments around Text-to-Text Transfer Transformer or T5 models [108]. 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 [154]. 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 [65]. 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.

3.3.3 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 [111], 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 3.2.2, i.e., syllogistic, relational syllogistic (Re-Syl), relative clauses

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 3.2: 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 3.3: 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.

(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 3.2). 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 3.3). 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 3.2, 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 3.4) or more predicates (see Table 3.5) 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.

Language Fragment	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{D}\rvert$+1	$\lvert\mathcal{D}\rvert$+2	$\lvert\mathcal{D}\rvert$+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 3.4: 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.

3.4 Results and discussion

Transformer models can solve model-checking with natural language problems with satisfactory accuracy, given adequate training instances, as depicted in Table 3.2. 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 illustrated in Tables 3.2 and 3.3. 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 3.6, 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 [129, 128], 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

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 3.5: 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

transitive verbs. According to the results in Table 3.6, only the number of quantifiers seems to affect the performance of the transformers, and not the exact quantifier used. Cognitive studies [128] 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 3.7. 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 3.3. 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 hypothesise that transformers at least partially learn to understand the essence of logical semantics of natural language. Table 3.3 also indicates a substantial difference in performance between the `T5-base` model and the

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 3.6: 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.

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

Table 3.7: 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.

`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 [26] and NLSat [111], 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 3.4 and 3.5. Even if the number of predicates increases, the accuracy of the transformer model remains relatively unchanged (see Table 3.5). However, if the number of domain elements increases, the model performance drastically decreases (see Table 3.4). 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.

3.5 Related Work

Our work follows the literature on evaluating neural approaches, especially transformer models on deductive and linguistic reasoning tasks [111, 110, 124, 40, 81]. Moreover, it is also related to other research approaches that have been conducted on data synthesis for rule reasoning problems [73, 152, 132]. 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 [119, 90], propositional inference [35], semantic parsing [49, 61], symbolic integration [69] and natural theorem proving [146, 85, 114, 44]. 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 [100, 101, 104, 103], 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 [101] 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 [83, 128, 129] 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.

3.6 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.

3.7 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 generalisation 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.

3.8 Supplementary materials

3.8.1 Templates of Language Fragments

Tables 3.8, 3.9, 3.10, 3.11 and 3.12 contain templates for the Syllogistic, Re-Syl, Rel, Rel-TV, and Anaphora fragments respectively. Each table contains natural language

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 3.8: 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 noun as the object	*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 noun as the subject	*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 3.9: Templates for the Re-Syl 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.

templates that are employed to construct sentences and their corresponding first-order formulae. 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*.

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 3.10: Templates for the Rel 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 3.11: Templates for the Rel-TV 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 3.12: Templates for the Anaphora 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.

Chapter 4

Not all quantifiers are equal: Probing Transformers' understanding of generalised quantifiers

"The meaning of life is to find your gift.
The purpose of life is to give it away."
– Pablo Picasso

This chapter is based on a paper titled *"Not all quantifiers are equal: Probing Transformer-based language models' understanding of generalised quantifiers"*. The paper was accepted and published at the Proceeding of Conference on Empirical Methods in Natural Language Processing: `EMNLP 2023` (main conference).

In the previous chapter, we delve into transformer models' ability to understand logical semantics by employing the problem of model-checking with natural language. However, the language fragments we considered were limited to first-order logic. In this chapter, we extend our investigation to incorporate generalised quantifiers—a logical construct that has garnered significant interest from logicians for centuries.

Abstract

How do different generalised quantifiers affect the behaviour of transformer-based language models? The recent popularity of transformer models and the central role generalised quantifiers have traditionally played in linguistics and logic bring this question into particular focus. The current research investigating this subject has not utilised a task defined purely in a logical sense, and thus, has not captured the underlying logical significance of generalised quantifiers. Consequently, they have not answered the aforementioned question faithfully or adequately. Therefore, we investigate how different generalised quantifiers affect transformers by employing a textual entailment problem defined in a purely logical sense, namely, model-checking with natural language. Our approach permits the automatic construction of datasets with respect to which we can assess the ability of transformers to learn the meanings of generalised quantifiers. Our investigation reveals that transformers generally can comprehend the logical semantics of the most common generalised quantifiers, but that distinct quantifiers influence transformers in varying ways.

4.1 Introduction

Generalised quantifiers have been a topic of much interest for more than a century in logic and linguistics [37, 151, 39, 87]. By capturing the interplay between quantity and cardinality, they provide a useful lens through which to understand human language and cognition [139, 129]. Since transformer-based language models strive to stimulate human-like language understanding [141, 30, 108, 95, 24], it is essential to determine the extent to which they can comprehend generalised quantifiers. Assessing the depth of understanding that transformers possess for any given concept is best achieved by evaluating their proficiency in applying it. In the case of generalised quantifiers, the most suitable evaluation task is textual entailment. This is particularly relevant because altering quantifiers can fundamentally change the logical inferences derived from a given text, reinforcing the integral role that quantifiers occupy within the scope of the textual entailment task.

When discussing entailment, it is vital to acknowledge two distinct strands of research in the literature. The first strand incorporates background knowledge and common sense into entailment, imbuing it with a probabilistic character [14, 153, 142]. The second strand examines textual entailment in a purely logical sense, eliminating the influence of background knowledge and common sense [111, 116, 78]. While the

Structure in Natural Language: *Talia, Hailee, Ava, Aria, Tony, Roger, Peter and Solomon are members of some group. Talia, Ava, and Solomon are bee-keepers. Hailee, Ava and Tony are scientists. Tony, Roger, Solomon, Ava, Aria and Hailee are Musicians. Roger, Solomon, Ava and Aria are guitarists. There are no designers in the group. Peter, Tony and Roger are artists. Only Solomon is an engineer.*	
Sentence: At least 3 musicians are guitarists **validity:** $True$	**Sentence:** All bee-keepers are scientists **validity:** $False$

Figure 4.1: An instance of the model-checking with natural language problem, the sentence "At least 3 musicians are guitarists" is $True$ according to the structure since the set musicians $X = \{Roger, Solomon, Ava, Aria\}$ are also guitarists and $|X| \geq 3$. However, the sentence " All bee-keepers are scientists" are $False$ as the set of bee-keepers $\{Talia, Solomon\}$ are not scientists

first form of entailment proves beneficial for a multitude of practical applications, it is not ideal in an investigation centred on identifying the impact of linguistic properties with logical significance, such as generalised quantifiers and negation. The empirical evaluation of linguistic constructs under this kind of entailment gets compromised due to its intricate association with other concepts. Consequently, it is challenging to differentiate the performance variation due to linguistic properties from those attributable to concepts like common sense and background knowledge. However, prior literature has only investigated generalised quantifiers in the context of entailment that incorporates background knowledge and common sense [29, 6] and naturally suffers from the same predicament. The second strand of textual entailment by defining entailment in a purely logical sense circumvents the aforementioned shortcoming. Consequently, it offers a conducive environment for conducting evaluations centred around linguistic constructs.

The logical problem that is most suited to study the influence of language constructs of logical significance is that of model-checking: given a formula ϕ and a structure $\mathfrak{A}$, determine whether ϕ is true in $\mathfrak{A}$ ($\mathfrak{A} \models \phi$). In the context of natural language, we are interested in a variant of the model-checking problem where the structure and the formula are translated into natural language. An instance of model-checking with natural language problems is depicted in Figure 4.1. From a complexity-theoretic point-of-view, model-checking in most formal languages is, comparatively speaking, straightforward. Indeed the model-checking problem with a fixed number of free variables and a finite structure is in PTIME. This is in contrast to other logical problems, such as satisfiability, whose problems for various fragments of logic can belong to different computational complexity classes [101, 104]. Yet, solving instances of the model-checking problem with natural language requires a comprehensive understanding of the logical semantics

of the expressions involved. Thus, it provides an ideal test environment to faithfully evaluate the extent to which generalised quantifiers affect transformer-based language models.

In this study, we embark on an in-depth investigation into transformer models' understanding of generalised quantifiers utilising the model checking problem, juxtaposing this with cognitive science research on quantifier verification tasks [129, 130, 83]. A critical part of our exploration involves the evaluation of pre-trained models prior to any fine-tuning. Thus, allowing us to discern whether any differences identified are intrinsic to the models themselves or introduced through the process of fine-tuning. Additionally, we consider the complexities arising from the integration of Boolean conjunctions and negation with generalised quantifiers. This aspect of our study sheds light on the intricate dynamics that exist between these linguistic elements and the challenges they pose to transformers. This comprehensive analysis paves the way for a more nuanced understanding of how transformers handle intricate linguistic constructs such as generalised quantifiers.

The key contributions of the present research can be summarised as follows: (1) To the best of our knowledge, this study represents the first exploration into the effects of generalised quantifiers within a logical entailment context; (2) We analyse the effect on transformers when quantifiers are paired with diverse logical constructs like negation and Boolean conjunctions; (3) We compare and contrast the behaviour of transformers with quantifiers with that of quantifier verification experiments done with human beings; and (4) We delve into how well transformers comprehend generalised quantifiers in a zero-shot context employing prompt engineering approaches such as chain-of-thought-prompting [149] and provide comparisons between pre-trained and fine-tuned models.

4.2 Methodology

4.2.1 Language Fragments and Generalised Quantifiers

We refer to the definition of language fragments and sentence templates introduced in Chapter 2.2.1. As depicted in Chapter 2.2.1, the Aristotelian syllogistic fragment[7] can be characterised by the following set of sentence templates.

All As are Bs	*Some As are As*
No A is a B	*Some B are not Bs*

In this work of literature, we employ a slightly extended version of the Aristotelian

GQ	Logical Denotation
All	$\{(X,Y) \mid X \subseteq Y \subseteq \mathfrak{A}\}$
Some	$\{(X,Y) \mid X \cap Y \neq \emptyset \, and \, X,Y \subseteq \mathfrak{A}\}$
At least K	$\{(X,Y) \mid \lvert X \cap Y \rvert \geq K \, and \, X,Y \subseteq \mathfrak{A}\}$
At most K	$\{(X,Y) \mid \lvert X \cap Y \rvert \leq K \, and \, X,Y \subseteq \mathfrak{A}\}$
Less than K	$\{(X,Y) \mid \lvert X \cap Y \rvert < K \, and \, X,Y \subseteq \mathfrak{A}\}$
More than K	$\{(X,Y) \mid \lvert X \cap Y \rvert > K \, and \, X,Y \subseteq \mathfrak{A}\}$
K	$\{(X,Y) \mid \lvert X \cap Y \rvert = K \, and \, X,Y \subseteq \mathfrak{A}\}$
Most	$\{(X,Y) \mid \lvert X \cap Y \rvert > \lvert X - Y \rvert \, and \, X,Y \subseteq \mathfrak{A}\}$
Few	$\{(X,Y) \mid \lvert X \cap Y \rvert < \lvert X - Y \rvert \, and \, X,Y \subseteq \mathfrak{A}\}$

Table 4.1: The generalised quantifiers (GQ) we used in our experimental setup, along with their logical denotation defined on some structure $\mathfrak{A}$.

syllogistic to allow negations at the subject, (e.g., Some non-musicians are beekeepers) and generalised quantifiers when generating sentences.

Generalised quantifiers define the semantics of sentences that include them in terms of relations between subsets of the structure [128]. Consider for example "All musicians are artists". The determiner phrase "All" in this sentence specifies a relation between the set of musicians and the set of artists, namely that the former is a subset of the latter. More generally, "All" in a structure $\mathfrak{A}$ expresses the binary quantifier:

$$\{(X,Y) \mid X \subseteq Y \subseteq \mathfrak{A}\}$$

This idea can be generalised to accommodate other quantifiers. Consider the sentence "At least K musicians are artists" where $K \in \mathbb{N}$. The phrase "At least K" likewise expresses a relation between the set of musicians and the set of artists, namely the cardinality of their intersection is at least K, that is, "At least K"in $\mathfrak{A}$ expresses the binary quantifier:

$$\{(X,Y) \mid \lvert X \cap Y \rvert \geq K \, and \, X,Y \subseteq \mathfrak{A}\}$$

In our scholarly inquiry, we examine logical quantifiers such as "All", numerical quantifiers such as "At least K" and propositional quantifiers such as "Most". The quantifiers employed, and their logical denotation on structure $\mathfrak{A}$ are depicted in Table 4.1. We utilise these generalised quantifiers when defining language fragments for sentence generation. Let $\mathcal{T}_Q$ be the sentence template which defines the language

Algorithm 2 Data Construction - Model checking with Generalised Quantifiers

Input : The Quantifier Q and corresponding sentence template $\mathcal{T}_Q$, a natural language template $\mathcal{M}$ to convert the structure to natural language, the vocabulary of proper nouns $\bar{D}$ and ordinary nouns $\bar{P}$, minimum and maximum number of domain elements d^{min} and d^{max}, minimum and maximum number of predicates p^{min} and p^{max}
Output : model checking dataset $\mathcal{D}$

$\mathcal{D} \leftarrow \{\}$
repeat
  $D, P \leftarrow$ sample from vocabularies $\bar{D}$ and $\bar{P}$ such that $d^{min} \leq |D| \leq d^{max}$, $p^{min} \leq |P| \leq p^{max}$
  $A, B \leftarrow$ sample two predicates from P
  $\bar{A}, \bar{B} \leftarrow$ negate A, B with p_{neg}
  $s \leftarrow$ substitute predicates $\bar{A}$ and $\bar{B}$ for schematic variables in the template $\mathcal{T}_Q$
  $\ell \leftarrow$ sample from $\{True, False\}$
  repeat
    $\mathfrak{A} \leftarrow$ generate structure randomly using (D, P)
    $\bar{\ell} \leftarrow$ MODELCHECKER$(\mathfrak{A}, (Q, \bar{A}, \bar{B}))$
  until $\ell = \bar{\ell}$ ‘
  $M \leftarrow$ translates $\mathfrak{A}$ to natural language using the template $\mathcal{M}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{M, s, \ell\}$
until *stop condition is met*

fragment corresponding to the quantifier Q. For example, consider the quantifier “All”, the corresponding template $\mathcal{T}_{All}$ takes the form of “All (non-)As are (not) Bs” where A and B are replaced by ordinary nouns. Section 4.8.1 depicts the sentence templates used to define language fragments for each of the quantifiers.

4.2.2 Data Construction

We develop a data construction algorithm (Algorithm 2) to construct a balanced dataset free from easily exploitable trivial linguistic patterns. The algorithm constructs a set of triplets (M, s, ℓ), where M is the natural language translation of the structure, s is a sentence of the relevant fragment $\mathcal{T}_Q$ and ℓ is a label ($True/False$) specifying whether s is true in M. To construct (M, s, ℓ), apart from $\mathcal{T}_Q$, the algorithm takes the vocabularies $\bar{D}$, $\bar{P}$ also as inputs. The vocabulary $\bar{D}$ comprises proper nouns employed to characterise domain elements, while the vocabulary $\bar{P}$ comprises ordinary nouns that characterise predicates. We draw a random sample of elements D and P, from vocabularies $\bar{D}$ and $\bar{P}$ to construct the structure. Two random nouns are sampled from P, each is then negated

with probability p_{neg}, and these are finally substituted for the two schematic variables in the template $\mathcal{T}_Q$ to form the sentence s.

Given (probably negated) words $\bar{A}$, $\bar{B}$, a generalised quantifier Q and a structure $\mathfrak{A}$, the model-checker determines $\mathfrak{A} \models s$, where s is the sentence formed by substituting $\bar{A}$, $\bar{B}$ for schematic variables in the templates $\mathcal{T}_Q$. This involves first determining the extensions of $\bar{A}$ and $\bar{B}$ in $\mathfrak{A}$ and then applying the meaning of generalised quantifier Q to these sets. Consider the example put forth in the section 4.1, $(Q, \bar{A}, \bar{B})$ corresponding to the sentence "All bee-keepers are scientists" is $(All, beekeepers, scientist)$. The model-checker determines the extensions of *beekeepers* and *scientists* in the corresponding structure $\mathfrak{A}$ to be $\{Talia, Ava, Solomon\}$ and $\{Hailee, Ava, Tony\}$, respectively. The quantifier *All* dictates that in order for the sentence to be True, the former needs to be a subset of the latter. However, the set $\{Talia, Ava, Solomon\} \not\subseteq \{Hailee, Ava, Tony\}$, thus, the model-checker assigns *False* as the validity label $\bar{\ell}$.

This setup with relative ease can be extended when Boolean conjunctions are introduced to the sentences. Consider, for instance, a sentence pair s_1 and s_2, formed using the predicates $(\bar{A}_1, \bar{B}_1)$ and $(\bar{A}_2, \bar{B}_2)$, respectively, for some quantifier Q, merged using Boolean conjunction $\odot \in \{\wedge, \vee\}$. To adapt to this scenario, the algorithm can be augmented by effecting a simple modification to step 10, transforming it into $\bar{\ell} \leftarrow$ MODELCHECKER$(\mathfrak{A}, (Q, \bar{A}_1, \bar{B}_1)) \odot$ MODELCHECKER$(\mathfrak{A}, (Q, \bar{A}_2, \bar{B}_2))$.

4.2.3 Prompts for Zero-shot Evaluation

Given that transformer-based language models undergo pre-training through a certain form of language modelling objective, the most common approach to evaluate these models in the zero-shot setting is by employing prompt engineering [16]. Consequently, we formulate prompts following a template-based strategy, utilising the constructed tuples (M, s, ℓ). We adopt two distinct types of templates. The first adheres to a more traditional form of prompting, which we refer to as standard prompting. The second type of template is based on the concept of chain-of-thought prompting [149]. Chain-of-thought prompting is a technique in which an example problem instance, accompanied by an explanation of the underlying thought process, is used to guide the model towards generating more precise responses. We depict the exact templates in Section 4.8.2.

4.3 Experimental Setup

Formally, the problem of model-checking with natural language can be defined as a binary classification problem where the training data takes the form $\{M^{(d)}, s^{(d)}, \ell^{(d)}\}_{(d)}^{|D|}$ where M is the natural language translation of the structure, s is the sentence and ℓ = True, False is the label.

4.3.1 Transformer-based language models

To explore the transformer-based language models' ability to comprehend different generalised quantifiers, we employ a set of transformer models that have a proven track record in textual entailment problems, namely, T5, Flan-T5, DeBERTa, LLaMA and GPT. We fine-tune T5, Flan-T5, and DeBERTa models while employing Flan-T5, LLaMA and GPT in a zero-shot setting. We provide a more detailed description of the fine-tuning is provided in Section 4.8.4

T5 Following the prior work on textual entailment defined purely in a logical sense [111, 131, 78], we utilise the T5 model in our experimental setup as one of the baseline models. The T5 model [108] employs a unified text-to-text format where all inputs and outputs are textual strings. We fine-tune the `T5-large` model with 770M parameters to perform the model-checking task.

Flan-T5 Fine-tuned Language Net [25], also known as Flan, is based on instruction fine-tuning [147] with the objective of making the transformer model generalise better to unseen tasks. The Flan-T5 model, considered to be an improvement to T5, applies instruction fine-tuning on the T5-model family. Thus, we primarily centred our experimental setup around the Flan-T5 model. We fine-tune the `Flan-T5-large` model with 770M parameters and utilise `Flan-T5-base` with 220M parameters, `Flan-T5-large`, `Flan-T5-xl` with 3B parameters and `Flan-T5-xxl` with 11B parameters in the zero-shot setting.

DeBERTa-v3 Due to the recent success of the DeBERTa-v3 model [50] in solving natural language inference tasks, we utilise it as a baseline model. The DeBERTa architecture improves upon the BERT and Roberta models using a disentangled attention mechanism and an enhanced mask decoder. DeBerta-v3 further improves the architecture by utilising an ELECTRA-style pre-training with Gradient Disentangled

Embedding Sharing. We fine-tune the `DeBERTa-v3-large` model with around 304M parameters.

ChatGPT Due to the recent success of ChatGPT in solving many natural language tasks in a zero-shot setting [8], we employ it in a similar context. Similar to InstructGPT [95], ChatGPT is trained to follow human instructions but follows a slightly different data collection approach.

LLaMA Considering that Flan-T5 and ChatGPT are trained to follow instructions, we decided to use a transformer which has not been explicitly trained to follow instructions as one of our baselines. Thus, we employ `LLaMA-30B` model in zero-shot settings. The LLaMA is said to outperform GPT-3 in most baselines and achieve comparable performance with respect to state-of-the-art transformer models [137].

4.3.2 Dataset and Evaluation

To fine-tune and evaluate transformer models, we construct train and test sets with 72K and 36K unique problem instances with 8K and 4K data points for each generalised quantifier. We arbitrarily select $[d^{min}, d^{max}] = [8, 14]$ and $[p^{min}, p^{max}] = [5, 10]$ when constructing problem instances. We construct a balanced dataset, and thus, we use accuracy as the main metric but chose to depict the overall precision, recall and f1-score to provide a more detailed analysis. We deem this setup answers the question, *"How do different quantifiers affect the behaviour of transformer models?"*. As the problem instances contain negations, our experiment will also provide insight on the effect of negation when intertwined with quantifiers on transformer models' understanding of language. We construct separate train and test sets with 72K and 36K instances with sentences containing Boolean conjunctions to answer the question, *"How do Boolean conjunctions affect the behaviour of transformer models when coupled with different quantifiers?"*. By evaluating these fine-tuned models against problem instances with higher K values in the numerical quantifiers than that of the train set, we ask the question *"Do transformer models learn to comprehend the logical semantics of generalised quantifiers?"*. To answer the questions, *"How do pre-trained transformer models comprehend different quantifiers?"* and *"Do they have any biases when performing a simple entailment task?"*, we evaluate transformer models in a zero-shot setting. We found that the problem instances with $[d^{min}, d^{max}] = [8, 14]$ and $[p^{min}, p^{max}] = [5, 10]$ are challenging for transformer models in zero-shot settings. Consequently, the use of the

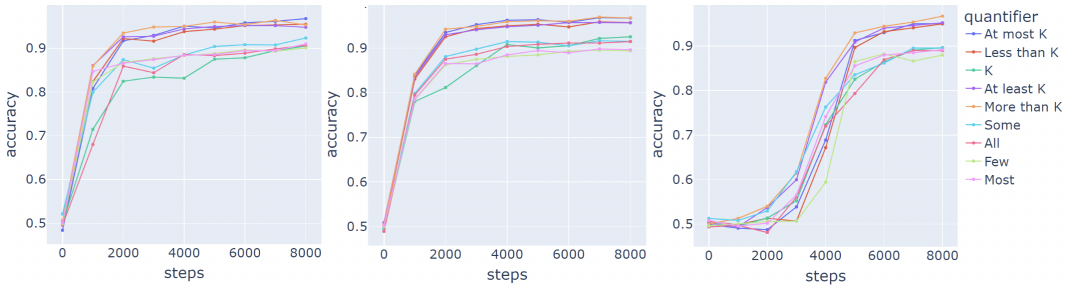

Figure 4.2: The rate of convergence of (a) `Flan-T5-large` and (b) `DeBERTa-v3-large` and (c) `T5-large` models break down based on different quantifiers

same test set did not yield any meaningful insights. Thus, we formulate a much simple problem instance with $[d^{min}, d^{max}] = [3,6]$ and $[p^{min}, p^{max}] = [2,4]$. A more detailed description of the dataset is provided in Section 4.8.3.

4.4 Results and Discussion

The ability of transformer-based language models to solve instances of the model-checking problem is differentially influenced by various generalised quantifiers. As demonstrated in Table 4.2, transformer models appear to encounter the most difficulty with proportional quantifiers such as "Most" and "Few". Interestingly, this empirical observation aligns with cognitive science research, which also highlights the complexities faced by humans in interpreting proportional quantifiers [129, 83, 139]. In addition, the performance related to the quantifier "K" is notably lower in comparison to other numerical quantifiers. A sentence incorporating the quantifier "K" is probabilistically more likely to be *False* given a random structure. Therefore, in a balanced dataset, the determination of the truth value of a sentence containing the quantifier "K" necessitates a more detailed examination compared to other numerical quantifiers considered in this study. However, as illustrated in Figure 4.2, given an adequate number of training steps, all transformers attain satisfactory performance levels across all quantifiers. Moreover, the newer transformer models, such as DeBERTa-v3 and Flan-T5, exhibit a faster convergence rate compared to T5.

As expressed by their precision and recall values, transformers often predict *True* for quantifiers such as "All" and "Less than K", but often predict *False* with respect to quantifiers like "Some" and "More than K". We attribute this to be an overcorrection introduced during fine-tuning. Consider the sentence with the quantifier "Less than K": "Less than K artists are engineers". This sentence is more likely to be *False* in

GQ	*ac*	*pr*	*re*	*f1*
All	91.4	88.6	95.3	91.8
Some	92.2	96.8	88.0	92.2
At least K	94.2	97.7	90.4	93.9
At most K	96.6	96.5	96.6	96.5
Less than K	95.1	93.4	97.2	95.3
More than K	94.9	96.9	93.0	94.9
K	90.8	86.7	96.2	91.2
Most	90.4	92.9	87.3	90.0
Few	91.1	93.1	89.2	91.1

Table 4.2: The test scores for the `Flan-T5-large` model across various generalised quantifiers. The abbreviations *ac*, *pr*, *re* and *f1* denote accuracy, precision, recall and F1 score values.

the context of the real world for K values we consider in this study $(8 \leq K \leq 14)$ since there are more than 14 artists who are engineers in the world. This proposition remains true even when negations are introduced to the sentences. Thus, we speculate that transformers overcorrect during fine-tuning and predict $True$ or $False$ accordingly.

Transformers show evidence of learning the logical semantics of generalised quantifiers. As illustrated in Figure 4.3, when tested with a dataset containing higher K values than that of the train set, the accuracy of transformers only decreases slightly for all numerical quantifiers. Therefore, we posit that transformer models possess the capacity to learn the logical semantics associated with generalised quantifiers. Our conclusions regarding generalisation bear resemblances to prior work conducted on model-checking with natural language [78]. Their research also supports the premise that transformer models are capable of comprehending the logical semantics of natural language. Additionally, we highlight the contrast between the demonstrated ability of transformers to generalise in the context of model-checking problems, and their apparent lack of such generalisation when solving satisfiability problems [116, 111]. We hypothesise that this distinction is due to the different complexity levels associated with these two types of problems and the necessity to understand complex inference rules when solving satisfiability problems.

The Boolean conjunctions have a significant effect, while negation has much less

	ac		*pr*		*re*		*f1*	
	AND	*OR*	*AND*	*OR*	*AND*	OR	*AND*	*OR*
All	91.0	83.7	89.0	80.9	93.8	89.5	91.3	84.9
Some	83.4	92.3	89.0	96.4	75.9	88.2	81.9	92.1
At least K	90.9	86.9	93.3	84.7	88.8	89.4	91.0	87.0
At most K	86.4	93.2	95.4	95.1	76.5	91.0	84.9	93.0
K	87.5	74.2	88.9	69.7	85.7	86.3	87.3	77.1
Less than K	88.5	90.6	92.3	90.3	84.0	91.0	87.9	90.7
More than K	92.5	88.5	93.2	85.4	91.6	92.9	92.4	89.0
Most	82.2	81.3	82.6	80.7	83.0	83.0	82.8	81.8
Few	82.2	81.0	83.4	81.8	80.1	79.0	81.7	80.4

Table 4.3: The test accuracy values for the `Flan-T5-large` model across various generalised quantifiers, broken down based on the Boolean conjunction (*AND*, *OR*) in the sentence. The abbreviations *ac*, *pr*, *re* and *f1* denote accuracy, precision, recall and F1 score values.

effect on fine-tuned transformers when coupled with generalised quantifiers. As demonstrated in Table 4.3, it is apparent that fine-tuned transformers possess the capacity to comprehend negations. In contrast, as depicted in Figure 4.4 (c), prior to the fine-tuning process, negations exert a considerable influence. Consequently, we propose that fine-tuning plays a significant role in enhancing the ability of transformers to understand negations. The inclusion of Boolean conjunctions significantly reduces the accuracy for all quantifiers. Noticeably, quantifiers for whom transformers tend to predict *True* often tend to have higher accuracy in the context of the *OR* operation compared to *AND* and vice-versa. In our findings, we discovered that quantifiers for which transformers frequently predict the label as *True* also display elevated recall values for *OR* operations and diminished recall for *AND* operations. Conversely, quantifiers that transformers often predict as *False* exhibit higher precision for *AND* operations and lower precision for *OR* operations.

The number of parameters, training process and type of prompting can influence the transformer models' performance when solving model-checking problem instances in zero-shot settings. The performance for Flan-T5 models exhibits the power law relationship with the number of parameters, as illustrated in Figure 4.4 (a). This empirical finding is consistent with prior research analysing language models'

Figure 4.3: The accuracy value when tested against problem instances with sentences containing higher K values than that of the train set. The results are broken down based on the numerical quantifier.

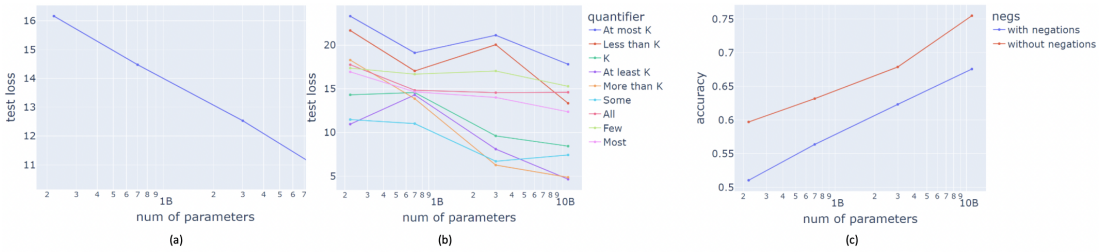

Figure 4.4: The (a) overall test loss and (b) test loss break down based on quantifier and (c) accuracy values breakdown based on the availability of negations for the Flan-T5 model with a different number of parameters in zero-shot settings, the number of parameters variates from 220M to 11B.

performance variation with factors such as the number of parameters, dataset size and computational resources [62]. However, upon breaking down the performance metrics based on the quantifiers, the resulting graph (Figure 4.4 (b)) is observed to be less uniform compared to the representation of overall performance. We attribute this behaviour to the inherent probabilistic aspect of the predictions formulated by transformers since language models are trained to find the most probable next word given a set of words. This probabilistic nature of language models can lead to inaccurate predictions, especially in a logical context.

The Flan-T5 model with fewer parameters outperformed the ChatGPT and LLaMA models in a zero-shot setting, as depicted in Table 4.5. This phenomenon is unsurprising since Flan-based models are very effective in tasks naturally verbalised as instructions due to their employment of instruction fine-tuning [147]. Upon contrasting the efficiency of ChatGPT and Flan-T5 models in the context of standard and chain-of-thought

GQ	s^0n^0	s^0n^1	s^1n^0	s^1n^1
All	91.6	94.7	89.0	89.6
Some	95.3	91.8	91.9	89.7
At least K	94.7	94.6	93.9	94.6
At most K	96.2	97.0	96.5	96.6
Less than K	96.1	95.5	95.7	95.0
More than K	95.6	95.2	95.4	94.9
K	90.9	91.8	91.5	89.7
Most	93.6	92.4	88.1	88.6
Few	91.7	92.5	87.4	89.3

Table 4.4: The test accuracy values for the `Flan-T5-large` model across various generalised quantifiers breakdown based on the negations in the sentence. The s, n denote subject and predicate nominative, $1, 0$ denotes having or not having a negation at s, n. For example s^0n^1 denote no negation at subject and negation at predicate nominative.

prompting techniques, it is observed that the discrepancy in accuracy metrics across these two distinct prompting methodologies is not substantial. However, the LLaMA model generated both *True* and *False* when generating the label when standard prompting is used, failing to follow the instruction properly. We attribute this failure in the LLaMA model to its training process, which, unlike the other two models, is not trained to follow instructions. When subjected to the chain-of-thought prompting approach, the LLaMA model displayed more consistency, generating either a *True* or *False* label. Thus, we infer that the inclusion of examples assists the LLaMA model in generating more concise outputs.

Accuracy values for transformers in zero-shot settings vary drastically with different quantifiers. As depicted in Table 4.5, transformers struggle with numerical quantifiers whose cardinality of intersection has an upper bound, such as "At most K" and "Less than K". This diminished performance can be attributed primarily to the transformer' tendency to predict the label *False* more frequently in sentences incorporating these specific quantifiers. We hypothesise this phenomenon is due to two factors. First, the background knowledge is already embedded in transformers from pretraining. As indicated previously, the sentences containing the above quantifiers coupled with a low K value are often false in a real-world scenario. Second, the prior

	ChatGPT		Flan-T5		LLaMA
	st	*ch*	*st*	*ch*	*ch*
All	57.8	59.3	59.5	65.6	50.5
Some	77.7	80.0	79.4	79.1	58.0
At least K	84.1	80.0	87.1	87.1	67.3
At most K	42.9	42.2	50.6	46.9	45.3
Less than K	48.5	44.4	63.0	64.6	49.2
More than K	82.6	75.9	86.5	88.2	53.2
K	68.1	70.1	76.6	70.5	53.4
Most	54.6	65.4	65.7	69.1	57.9
Few	44.4	43.2	57.6	55.9	46.4
Overall	62.3	62.3	69.6	69.9	53.5

Table 4.5: The test accuracy values for the ChatGPT, Flan-T5-xxl and LLaMA-30B model in zero shot settings, *st* denotes standard-prompting approach while *ch* denotes the chain-of-thought prompting approach.

cognitive research on quantifiers has demonstrated that quantifiers with a downward monotone, such as "Few" or "Less than K", present more processing challenges for humans compared to those with an upward monotone, such as "Most" and "More than K" [41, 163, 2]. Since transformers are trained on human-generated data, it is highly likely that these models have incorporated this cognitive trait into their understanding of language, which, in turn, affects their responses. Moreover, a deeper analysis of the answers generated through chain-of-thought prompting revealed that even when the predicted label is correct, the overall answer is often incoherent. This coherence deficit in transformers, coupled with their difficulties in handling certain quantifiers, suggests that these models are yet to achieve proficiency in learning even the simplest inferential rules.

4.5 Related Work

Our work follows the literature on probing how different linguistic properties affect the behaviour of neural approaches such as transformer-based language models [78, 26, 18, 58, 34]. Specifically, our investigation is closely related to the literature whose linguistic

properties of interest are generalised quantifiers [29, 6]. Our exploration differentiates from prior research in two key ways. First, we explore generalised quantifiers employing a task that is defined purely in a logical sense. Thus, we provide a more faithful investigation into how transformers comprehend generalised quantifiers. Second, our research also integrates a comprehensive analysis of how the interaction of negations and Boolean conjunctions with quantifiers influences transformers' performance in a simple entailment task. We follow the logical denotations introduced in logical studies to formalised generalised quantifiers when formulating our task [151, 87, 39, 38, 98] and draw parallels with cognitive science work on quantifier verification in our experimental setup [130, 83, 127].

Our evaluation scheme for evaluating transformer models in a zero-shot setting builds upon prior literature on prompt engineering [16, 66, 149]. However, ours is the first literature evaluating transformer models on the model-checking problem in zero-shot settings.

4.6 Conclusion

We investigated how generalised quantifiers affect the behaviour of transformer-based language models by employing the problem of model-checking. We found that different generalised quantifiers affect transformer models in varying ways when solving model-checking problems in both fine-tuned and zero-shot settings. Based on empirical findings on generalisation, we posited transformers can learn to understand the logical semantics of generalised quantifiers. Moreover, our experimental setup in the zero-shot setting demonstrated that a multitude of factors, such as the training process, size of the models and type of prompts, can affect the ability of transformers to solve a simple entailment task. Thus, a compelling avenue for future research is to probe how varying factors affect transformer-based language models when solving a more complex entailment task, like determining satisfiability.

4.7 Limitations

Due to the empirical nature of this study, it suffers from an inductive dilemma on three fronts. One, in the front of transformers, the second related to generalised quantifiers and the third in relation to prompts we explored in zero-shot settings. We explored several transformer-based language models that are in line with prior literature and probe

how different generalised quantifiers affect their behaviour. Nonetheless, due to the empirical nature of this investigation, it is plausible that some transformer architectures could deviate from the behavioural norms discussed in this paper when interacting with generalised quantifiers. A similar limitation applies to the range of generalised quantifiers examined, as the ones employed in our study do not represent the entire spectrum of generalised quantifiers. In zero-shot settings, this limitation further extends to the prompt templates we employed. We consider two types of prompt templates, but there is a multitude of alternative ways prompts can be formulated by using the (M, s, ℓ) triplets.

4.8 Supplementary materials

4.8.1 Sentence Templates

When constructing sentences, as mentioned in the methodology section, we employ sentence templates. Let Q be a generalised quantifier, and $\mathcal{T}_Q$ be the sentence template for the corresponding quantifier Q. Then $\mathcal{T}_Q$ take the general form,

$$Q \text{ (non-) } A\text{s are (not) } B\text{s}$$

The inclusion of "non/not" is determined by the availability of the negations and A, B are ordinary nouns. Consider the quantifier "At most K" for some natural number K. Then the corresponding sentence template takes the form "At most K (non-) As are (not) Bs". Table 4.6 depicts sentence templates corresponding to quantifiers considered in this study.

We employ the tuples (M, s, ℓ) to delineate prompts for the language modelling objective, providing a framework for evaluating the effectiveness of transformer models in zero-shot settings. As mentioned in the methodology section, we explored two types of prompts. One, we informally called standard prompts and the other is based on chain-of-thought-prompting. The standard prompting is conceptualised by the following template,

> Q: Given the following scenario, M. Is the sentence s True
> or Falseaccording to the scenario?
> A:

The chain-of-thought prompting employs an example problem instance with an explanation of the thought process, thereby facilitating a more precise response from

GQ	Sentece Template
All	All (non-) *A*s are (not) *B*s
Some	Some (non-) *A*s are (not) *B*s
At least K	At least *K* (non-) *A*s are (not) *B*s
At most K	At most *K* (non-) *A*s are (not) *B*s
Less than K	Less than *K* (non-) *A*s are (not) *B*s
More than K	More than *K* (non-) *A*s are (not) *B*s
K	*K* (non-) *A*s are (not) *B*s
Most	Most (non-) *A*s are (not) *B*s
Few	Few (non-) *A*s are (not) *B*s

Table 4.6: The generalised quantifiers (GQ) we used in our experimental setup, along with their sentence templates

transformers. If we let (M_e, s_e, ℓ_e) represent this example problem instance and E elucidate the thought process, the chain of thought prompting can then be defined using the template,

> Q: Given the following scenario, M_e. Is the sentence s_e True or False according to the scenario?
> A: ℓ_e. E
> Q: Given the following scenario, M. Is the sentence s True or False according to the scenario?
> A:

4.8.2 Prompt templates for Zero-shot settings

4.8.3 Dataset details

We utilised nine quantifiers when constructing sentences. We construct train and test sets with 72K and 36K unique data points with 8K and 4K data points for each quantifier for fine-tuning and evaluating transformers. As the vocabulary, we employed the Richardson and Sabharwal [111] vocabulary of nouns, which contains a list of professions (e.g., "artist", "doctor"), that we extended by adding more professions. The dataset contains an equal number of *True* and *False* problem instances for each generalised quantifier. When constructing the dataset for fine-tuning transformers, we select $[d^{min}, d^{max}] = [8, 14]$ and $[p^{min}, p^{max}] = [5, 10]$. For sentences with numerical quantifiers, we select a K values randomly from the range $[1, |D|]$, where $|D|$ denotes the

GQ	maximum	minimum	mean
All	161	47	90.1
Some	165	48	89.7
At least K	160	47	92.4
At most K	162	48	92.2
Less than K	163	49	92.4
More than K	168	49	92.1
K	164	46	89.9
Most	162	45	89.8
Few	158	47	89.6

Table 4.7: The minimum, maximum and mean number of words (tokens) in problem instances ($M + s$) when separated by SPACE

number of domain elements selected when formulating the problem instance. The minimum, maximum and mean number of tokens for problem instances of each quantifier is depicted in Table 4.7. To evaluate transformers' behaviour with Boolean conjunctions, we also constructed separate train and test sets with 72K and 36K data points. The dataset contains an equal number of problem instances for each conjunction, and quantifier pair. Moreover, since the intention was to compare the effects of generalised quantifiers on transformer-based language models, so we decided to use the simplest form of language templates, i.e. syllogistic.

We also emphasise the rationality behind the iterative approach we used in constructing the data. An alternative way of constructing problem instances is to derive the label ℓ using the model-checker instead of iteratively creating structures to match a pre-defined label and a sentence. However, this alternative approach can induce easily exploitable patterns. Consider the quantifier "K", "All" and "Some". For a random structure, quantifiers "K" and "All" are more likely False, while the quantifier "Some" is probabilistically $True$.

4.8.4 Fine-tuning Details

Formally, we define the task as a binary classification problem where the objective of the transformer-based language model is to predict the label ℓ ($True$ or $False$) given the natural language interpretation of the structure M and the sentence s as the inputs. We

select and fine-tune three transformers, namely T5, Flan-T5, and DeBERTa-v3, all of which have previously demonstrated their efficiency and reliability in resolving textual entailment tasks. According to prior literature, the performance of transformers mostly depends on the pre-trained data, and size of the models rather than the architectural choice [108, 62]. Moreover, the accuracy values yielded for all transformers are similar. Thus, we expect a similar behaviour for other transformer architectures as well. Since the transformers achieve satisfactory accuracy and since the central research interest is to analyse the behaviour of transformers rather than identifying the best-performing transformer, we do not perform any hyperparameter tuning. Moreover, exploring several different transformers and performing hyperparameter tuning leaves a higher carbon footprint [126]. Due to the nature of the research question, we consider such an exploration unnecessary.

Loss function and optimizer We fine-tune each transformer model to predict the label ℓ given the (M, s) by reducing the binary cross entropy loss over the target using the ADAM [65] optimizer.

Batch size Utilising gradient checkpointing for memory-efficient fine-tuning, we set the batch size to 36.

Number of epochs We fine-tune each transformer model for 4 epochs, resulting in 8000 steps.

maximum token length We set the maximum token length to 512 since the maximum problem length is much lower than that, thus, we do not truncate the inputs.

learning rate We set the learninig rate of 1×10^{-5}

We utilise Huggingface [154] implementation when experimenting with the transformer model models we consider in this study.

Chapter 5

Natural Language Satisfiability: Exploring the Problem Distribution and Evaluating Transformers

"The mind, once stretched by a new idea,
never returns to its original dimensions."
– Oliver Wendell Holmes

This chapter is based on a paper titled *"Natural Language Satisfiability: Exploring the Problem Distribution and Evaluating Transformer-based Language Models"*. The paper was accepted and published at the Proceeding of the Association for Computational Linguistics: ACL 2024 (main conference). Notably, the paper received the prestigious **Best Paper Award**.

In the previous two chapters, we explored the ability of transformer models to solve model-checking problems, demonstrating that these models capture the essence of logical semantics in natural language. In this chapter, we focus on the other half of the research question, "Can the transformer models learn rules of inference?" employing the problem of natural language satisfiability.

Abstract

Efforts to apply transformer-based language models to the problem of reasoning in natural language have enjoyed ever-increasing success in recent years. The most fundamental task in this area to which nearly all others can be reduced is that of determining satisfiability. However, from a logical point of view, satisfiability problems vary along various dimensions, which may affect transformer models' ability to learn how to solve them. The problem instances of satisfiability in natural language can belong to different computational complexity classes depending on the language fragment in which they are expressed. Although prior research has explored the problem of natural language satisfiability, the above-mentioned point has not been discussed adequately. Hence, we investigate how problem instances from varying computational complexity classes and having different grammatical constructs impact transformer models' ability to learn rules of inference. Furthermore, to faithfully evaluate transformers, we conduct an empirical study to explore the distribution of satisfiability problems.

5.1 Introduction

The impressive performance of transformer-based language models in natural language inference tasks [30, 160, 108, 76] has created a surge of interest in the development of linguistic and deductive reasoning benchmarks to evaluate these models [110, 40, 131, 159]. One such area of interest is the ability to recognise valid entailments, understood in a strictly logical sense [26, 111], where the inference does not depend on background knowledge and commonsense. The ability of transformers to understand this type of entailment is indicative of their ability to learn rules of inference, understand the logical semantics of natural language and emulate complex algorithms.

Since the inferences we shall be concerned with do not depend on background knowledge or intuitive plausibility, they can be captured using the apparatus of formal logic. Logicians usually find it convenient to reconstruct entailment in terms of satisfiability: a set of formulae Φ is *satisfiable* if there is a structure $\mathfrak{A}$—in the model-theoretic sense—such that every formula of Φ is true in $\mathfrak{A}$. A set of formulae Φ *entails* a formula ψ just in case $\Phi \cup \{\neg\psi\}$ is not satisfiable. The same duality holds for natural language (which incorporates a concept of negation) just as it does for formal logic. Thus, we consider the following problem: given a collection of sentences expressed in

Sentences	All scholars love all artists Some artists are musicians Some scholars love no musician	All scholars love all artists all artists are musicians Some scholars love all musicians
Formulae	$\forall x(scholar(x) \Rightarrow \forall y(artist(y) \Rightarrow love(x,y)))$ $\exists x((artist(x) \wedge musician(x))$ $\exists x(scholar(x) \wedge \forall y(musician(y) \Rightarrow \neg love(x,y)))$	$\forall x(scholar(x) \Rightarrow \forall y(artist(y) \Rightarrow love(x,y)))$ $\forall x((artist(x) \Rightarrow musician(x))$ $\exists x(scholar(x) \wedge \forall y(musician(y) \Rightarrow love(x,y)))$
Satisfiability	unsatisfiable	satisfiable

Figure 5.1: The table depicts two instances of a satisfiability problem; one is unsatisfiable while the other is satisfiable. In the first example, the first two formulae imply $\forall x(scholar(x) \rightarrow \exists y(musicians(y) \wedge love(x,y)))$, while the last formula is $\exists x(scholar(x) \wedge \forall y(musician(y) \rightarrow \neg love(x,y)))$—a direct contradiction, hence unsatisfiable. In the The second example (satisfiable), a structure $\mathfrak{A}$ can be easily found that makes all formulae *True*: imagine, for instance, a world where scholars exist, all musicians are artists, and everyone loves everyone.

a natural language such as English, determine whether it is—in a strictly logical sense—satisfiable. Figure 5.1 illustrates two instances of the satisfiability problem in English, one positive and one negative. Note that any method for determining satisfiability yields a method for recognising entailments and *vice versa*.

We approach this problem in a controlled way. The sentences in Figure 5.1 feature only rudimentary grammatical resources: the determiners *some*, *no* and *all*, transitive verbs with unqualified noun-phrases as subjects and objects, and the copula *is*. It is well-known from formal logic that the computational complexity of the satisfiability problem for a logic depends on its expressive power. For example, the satisfiability problem for propositional logic (known as SAT) is NPTIME-complete, but the corresponding problem for the two-variable fragment of first-order logic is NEXPTIME-complete, while for the whole of first-order logic is r.e.-complete (i.e. undecidable); for a survey, see Pratt-Hartmann [102]. The same is true when it comes to fragments of natural languages [100, 101, 104], as we explain presently. The question therefore arises as to how transformers' ability to learn rules of inference is affected by the fragment of language within which they operate.

In recent years, researchers have analysed the limitations of various neural approaches when solving satisfiability problems [119, 35, 19], including natural language satisfiability problems [111]. However, in all cases, the problems are in propositional logic or its close relatives, and as such, the expressive power of the expressions or sentences is limited. Moreover, it restricts the problem space to be in a single computational complexity class, NP-complete. We overcome these limitations by generating sentences from various *fragments* of English that are related to more varied fragments of first-order logic. By utilising language fragments of varying expressive power, we

also provide an analysis of how computational complexity affects transformers' ability to grasp the rules governing logical entailment.

When investigating transformers' ability to solve instances of natural language satisfiability problems, it is important to ensure that the training and test datasets include a sufficient number of challenging problem instances. The difficulty is that randomly constructed sets of formulae (or sentences) are, depending on the sampling parameters, in many cases easily seen either to be satisfiable or to be unsatisfiable, a situation dramatically illustrated in the case of propositional logic satisfiability [118, 27, 86], where challenging problems occur only when the ratio of clauses to propositional variables is close to a critical threshold, i.e., the so-called *phase-change region*. In this region, where the probability of satisfiability for a randomly generated problem instance is close to 0.5, algorithms for determining satisfiability typically exhibit long running times. Therefore, to demonstrate that transformer models can learn rules of inference, we must ensure that they work on the hardest region of the target problem space. Indeed, recent work has shown some pitfalls associated with synthetic data due to under-sampling challenging problems [122, 156]. Consequently, we conduct an empirical study to determine, for each of the language fragments investigated, the relevant "phase-change" region where the challenging problem instances are to be found.

The contributions of this paper are as follows. (1) We identify a number of fragments of English featuring a range of logico-syntactic constructions involving determiners, the copula, transitive verbs, relative clauses and bound-variable anaphora. (2) We empirically determine the phase-change region for these fragments and construct datasets containing instances of the satisfiability problems sampled from these regions. (3) We investigate the ability of transformer-based language models to solve the satisfiability problem for the investigated fragments and carry out a systematic analysis of how the underlying computational complexity of the satisfiability problem correlates with transformer models' ability to grasp the relevant inferential principles. (4) Furthermore, we explore the proficiency of transformer models in solving instances of the satisfiability problem within a zero-shot context. To the best of our knowledge, this investigation represents the first attempt to probe the behaviour of transformer models in zero-shot scenarios with respect to their efficacy in solving satisfiability problems.

5.2 Methodology

5.2.1 Language Fragments

We reference the definitions of language fragments and sentence templates introduced in Chapter 2.2.1, along with the definition of the corresponding formula templates. We consider five English fragments in the sequel, all defined in this way: (i) the fragment $\mathcal{S}$, a version of the syllogistic in which—for the sake of logical uniformity—we additionally allow negation at the subject (e.g. *Some non-artist is not a beekeeper*); (ii) the fragment $\mathcal{W}$, which adds relative clauses to the subjects of sentences in $\mathcal{S}$ (e.g. *Every artist who is not a musician is a writer*); (iii) the fragment $\mathcal{V}$, which extends $\mathcal{S}$ with main clauses featuring transitive verbs and a quantifying determiner (e.g. *Every carpenter admires some writer*); (iv) the fragment $\mathcal{Z}$, which extends $\mathcal{W}$ with relative clauses featuring transitive transitive verbs (e.g. *Every carpenter who admires some writer is an electrician*); and (v) the fragment $\mathcal{A}$, which extends $\mathcal{Z}$ with certain sentence templates involving bound-variable anaphora (e.g. *Some artist hates no beekeeper who admires him*). The precise sets of sentence templates defining these fragments are presented at Section 5.8.1. Since the relevant sentence templates correspond to sets of formula templates of first-order logic, our five language fragments may, to all intents and purposes, be identified with the corresponding fragments of first-order logic; hence we alternate freely between sentences and their logical translations, as the context requires.

If $\mathcal{L}$ is a natural language fragment, we denote by $Sat(\mathcal{L})$ the problem:

> Given: a finite set Φ of sentences in $\mathcal{L}$,
> Return: Y if Φ is satisfiable; N otherwise.

It can be shown that Sat($\mathcal{S}$) is NLOGSPACE-complete and Sat($\mathcal{V}$) is EXPTIME-complete [103, Theorems 4.11 and 6.3]. It is completely routine to show that Sat($\mathcal{W}$) is NPTIME-complete. With a little effort, one can show that Sat($\mathcal{Z}$) is also EXPTIME-complete, using essentially the same techniques as for Sat($\mathcal{V}$). Finally, Sat($\mathcal{A}$) is NEXPTIME-complete: membership in NEXPTIME follows from the fact that all the corresponding formula templates are in the two-variable fragment of first-order logic [46]; NEXPTIME-hardness follows by an almost identical argument to that given for the rather larger English fragment E2V in [100]. The proof, along with a detailed complexity-theoretic analysis of the language fragments discussed, are depicted in Section 5.8.2. Given these results, it is natural to ask whether language models have greater success in learning to recognise valid entailments (or, dually: satisfiability of sets of sentences) for

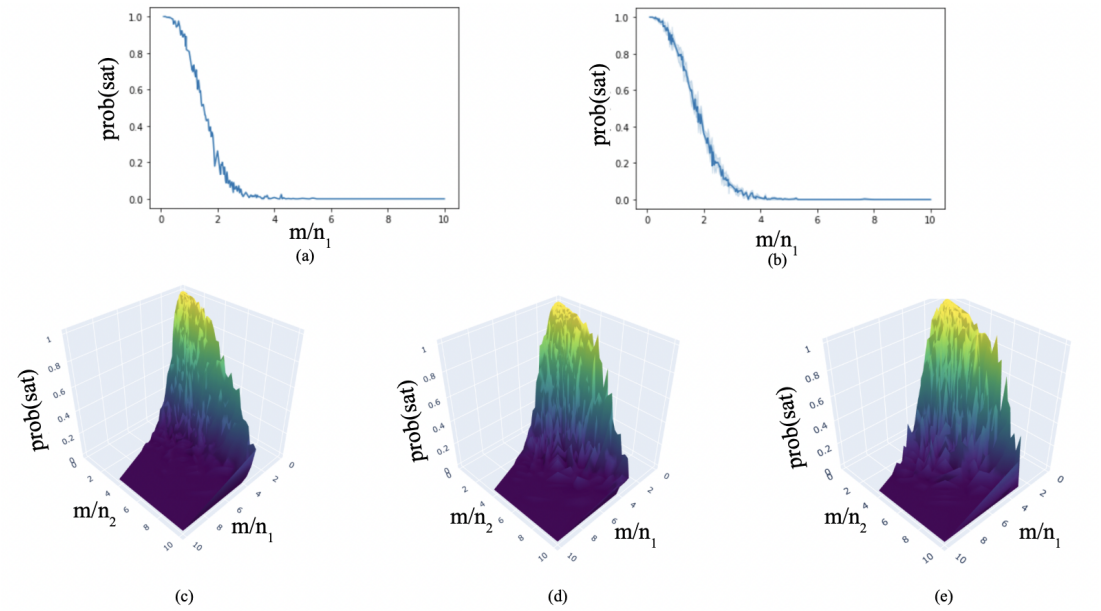

Figure 5.2: The probability of satisfiability for sets of sentences in the language fragments: (a) $\mathcal{S}$, (b) $\mathcal{W}$, (c) $\mathcal{V}$, (d) $\mathcal{Z}$ and (e) $\mathcal{A}$. Here, m denotes the number of clauses, and n_1, and n_2 the number of common nouns and transitive verbs, respectively, in the sampled vocabulary.

the computationally easier fragments, or whether training on easier fragments aids in recognising entailments in harder ones.

5.2.2 Identifying the phase change region

It is important to realise that the worst-case complexity bounds for satisfiability problems are not necessarily representative of randomly generated problem instances, a phenomenon that is well understood in the case of propositional logic. Suppose that k is a positive integer and that a *k-clause* is a disjunction of k or fewer proposition letters or negated proposition letters. The k-SAT problem is the following: given a finite set of clauses Γ, return 1 if Γ is satisfiable (i.e. if there is a truth-value assignment making all the clauses in Γ *true*), and 0 otherwise. This problem is NPTIME-complete for all $k \geq 3$. However, under certain conditions, it is trivial to decide random instances of these problems with high probability. Fixing $k = 3$, consider a randomly generated instance Γ of 3-SAT consisting of m clauses featuring n proposition letters $p_1, \ldots, p_n$ (and their negations). If the ratio $\frac{m}{n}$ is large, we have a highly constrained problem with few degrees of freedom, so the probability of satisfiability is close to zero; if, on the other hand, the ratio $\frac{m}{n}$ is small, the probability of satisfiability is close to unity. In either

case, it is trivial for any algorithm to achieve high performance when predicting the correct answer. Only for values in a relatively narrow range of $\frac{m}{n}$, commonly referred to as the *phase-change region*, is the problem challenging, with this range (centred on a value close to 4.17) narrowing as n increases [118, 86].

A similar phenomenon is observed for the satisfiability problems for natural language fragments studied here: the satisfiability of a given set of sentences may in many cases be easily determined with high probability by measuring the number of degrees of freedom in the given instance. Consider, for example, the problems Sat($\mathcal{S}$) and Sat($\mathcal{W}$). Any instance of these problems is characterised by the number n_1 of common nouns featured, and the number m of sentences involved. We find that the probability of satisfiability for randomly generated instances is close to 0.5 only when the ratio $\alpha = \frac{m}{n_1}$ is in a relatively narrow band. For the other fragments considered in this study, which feature transitive verbs as well as common nouns, we must consider not only the ratio α but also the ratio $\beta = \frac{m}{n_2}$, where n_2 is the number of transitive verbs in the given problem instance. Again, we find that the probability of satisfiability for a randomly generated problem is close to 0.5 only for pairs of values (α, β) lying in a relatively constrained region. We continue to refer to this region, informally, as the *phase-change region*: notice that this is a region in 2-dimensional space. It is important to remember that the shape and location of the phase-change region depend on the fragment considered; however, in all cases, the gradients involved are observed to become sharper with increasing n. In assessing the ability of transformers to learn to solve the satisfiability problem for the fragments considered in this paper, we must make sure to construct datasets involving only challenging instances, namely, those selected from the phase-change region: prowess at solving cases chosen uniformly over parameter space is a poor test of a system's grasp of logical principles. For definiteness, we take the phase-change region for a fragment $\mathcal{L}$, denoted $\lambda_{\mathcal{L}}$, to be the set of input parameters for which the probability of satisfiability is [0.35, 0.65]. This set can be determined empirically by random sampling, much as in the original studies of 3-SAT. The variation of probability of satisfiability with other factors is outlined in Appendix 5.8.6.

5.2.3 Data Construction

For each of the language fragments $\mathcal{L}$ considered above, we constructed a dataset with which to fine-tune transformers. Each dataset is a set of *data points*. Each data point is a pair consisting of a set of sentences Φ from $\mathcal{L}$, and a label (True or False) indicating whether Φ is satisfiable. To generate a single sentence of Φ, we select at random one of

the sentence templates defining $\mathcal{L}$ and instantiate its schematic variables by uniform sampling from a collection of n_1 common nouns and (for the applicable fragments) n_2 transitive verbs; repeating this whole process m times then yields a set Φ of cardinality m. The label determining the satisfiability of Φ is determined by applying a theorem prover (in our case, `Z3`) to the corresponding set of formulae of first-order logic as given by the formula templates. We construct data points for various values of m, n_1 and (where appropriate) n_2, taking care to employ only those combinations within the phase change region, $\lambda_{\mathcal{L}}$. As explained above, $\lambda_{\mathcal{L}}$ has been determined empirically.

Algorithm 3 Data Construction - Natural language satisfiability

Input : Language Fragment $\mathcal{L}$; phase-change region $\lambda_{\mathcal{L}}$ (for $\mathcal{L}$); vocabulary of unary predicates $\mathcal{U}$; vocabulary of binary predicates $\mathcal{V}$; range for number of unary predicates $[n_1^{min}, n_1^{max}]$; range for number of binary predicates $[n_2^{min}, n_2^{max}]$.
Output : natural language satisfiability dataset $\mathcal{D}$.

$D \leftarrow \{\}$
repeat
  $m, n_1, n_2 \leftarrow$ sample m, n_1, n_2 such that $n_1^{min} \leq n_1 \leq n_1^{max}$, $n_2^{min} \leq n_2 \leq n_2^{max}$ and $(\frac{m}{n_1}, \frac{m}{n_2}) \in \lambda_{\mathcal{L}}$
  $\mathcal{U}^*, \mathcal{V}^* \leftarrow$ sample n_1 unary predicates and n_2 binary predicates from $\mathcal{U}$ and $\mathcal{V}$ respectively, $|\mathcal{U}^*| = n_1$ and $|\mathcal{V}^*| = n_2$
  for $i = 1$ to m **do**
    $t_i \leftarrow$ randomly sample template from language fragment $\mathcal{L}$
    $s_i \leftarrow$ substitute predicates from $\mathcal{U}^*$ and $\mathcal{V}^*$ for schematic variables in t_i
    $\phi_i \leftarrow$ translate s_i to a first-order logic formula
  end for
  $\ell \leftarrow$ `SAT-solver`$(\{\phi_1, ..., \phi_m\})$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\langle \ell, \{s_1, ..., s_m\} \rangle\}$
until *stop condition is met*

For the fragments $\mathcal{S}$ and $\mathcal{W}$, we sample values of n_1 from the range $[n_1^{min}, n_1^{max}]$, where $n_1^{min} = 6$, and $n_1^{max} = 16$. For the fragments $\mathcal{V}$, $\mathcal{Z}$, and $\mathcal{A}$, we sample values of n_1 from the range $[n_1^{min}, n_1^{max}]$, where $n_1^{min} = 3$, $n_1^{max} = 8$, and we sample values of n_2 from the range $[n_2^{min}, n_2^{max}]$, where $n_2^{min} = 3$ and $n_2^{max} = 8$. In each case, we generate a set of values m for which $\alpha = m/n_1$ and (for the appropriate fragments) $\beta = m/n_2$ lie in $\lambda_{\mathcal{L}}$. The entire protocol for generating datasets is given in Algorithm 3. We use the theorem prover `Z3` to determine the satisfiability of the generated set of formulae, $\Phi = \{\phi_1, \ldots, \phi_m\}$. For each language fragment, we construct a training set with 120K data points, an evaluation set with 10K data points and a test set with 10K data points to

fine-tune and evaluate transformer models. We note that the above experimental setup can be directly adapted to any language fragment.

For evaluating transformers in zero-shot settings we set $n_1^{min} = 5$, and $n_1^{max} = 10$ for fragments $\mathcal{S}$ and $\mathcal{W}$ and $n_1^{min} = 3$, $n_1^{max} = 5$, $n_2^{min} = 2$ and $n_2^{max} = 5$ for fragments $\mathcal{V}$, $\mathcal{Z}$, and $\mathcal{A}$. We then construct an additional 1200 problem instances for each language fragment and formulate the prompt using the constructed problem instances employing a template-based approach. The exact template is depicted in Section 5.8.3. Further details regarding the datasets are given in Section 5.8.4.

5.3 Experimental Setup

The objective of the transformer model is to approximate Ω given training instances $\mathcal{D}_{tr} = \{(P^{(d)}, \Omega(P))^{(d)}\}_{(d)}^{|D_{tr}|}$. Since the $\Omega(P)$ is a binary label, the problem can be modelled as a binary classification problem, where the label is set to be 1 or 0 based on satisfiability.

5.3.1 Transformer-based language models

To examine transformers' ability to solve hard instances of natural language satisfiability problems, and investigate their capability to learn rules of inference, we fine-tuned two well-known transformer models which have a proven track record of solving textual entailment problems.

T5. Following the work done by Richardson and Sabharwal [111] and Tafjord et al. [131] on logical reasoning in natural language, we primarily centred our experiments around the text-to-text transformer or T5 [108]. The T5 model employs a unified text-to-text format where all inputs and outputs are textual strings, in contrast to BERT-styled models. We utilised the `T5-large` model, which has around 700M parameters.

DeBERTa-v3. Due to the recent success of the DeBERTa-v3 model [50] in solving natural language inference tasks, we decided to use it as a baseline model. The DeBERTa architecture improves upon the BERT and RoBERTa models using a disentangled attention mechanism and enhanced mask decoder, and version 3 further improves the architecture by utilising an ELECTRA-style pre-training with Gradient Disentangled Embedding Sharing. We employ the `DeBERTa-v3-large` model with around 304M parameters.

Each of the transformer models is fined-tuned by reducing the binary cross entropy loss over the target using Adam optimiser [65] and we used the HuggingFace [154] implementation when experimenting with the above-mentioned transformer model. A detailed description of the fine-tuning process is described in Section 5.8.5.

To investigate transformer models' ability to solve satisfiability problems in *zero-shot* settings, we employed three well-known models.

GPT. Due to the recent success of ChatGPT and GPT-4 solving many natural language processing tasks in zero-shot settings, we employed them in a similar context [8, 94]. Both ChatGPT and GPT-4 models are trained to follow human instructions utilising Reinforcement Learning from Human Feedback (RLHF).

LLaMa. To better compare the effect of model size and architecture, we also utilise the `LLaMA-2-chat-70B` model in zero-shot settings. LLaMa-2 achieves comparable performance with state-of-the-art language models such as ChatGPT and PALM [137]. Similar to ChatGPT and GPT-4, the LLaMa-2-chat model also employs RLHF.

5.3.2 Proposed Dataset and Evaluation

To evaluate the ability of transformer models to solve natural language satisfiability problem instances, we designed several experiments. Firstly, to answer the questions, "Can transformer models solve hard natural language satisfiability problems?" and "How does computational complexity affect transformer models' ability to perform a logical reasoning task?", we trained and evaluated the transformer models mentioned in Sec. 5.3.1 against each of the language fragments introduced in Sec. 5.2.1 (see Table 5.1). Secondly, to answer the question "Do the computationally simpler language fragments help transformer models learn rules inference in complex ones?", we trained the same transformer models using a dataset that comprises problem instances from all language fragments (see Table 5.2). We employed two variants of this joint dataset, one with 600K data points with 120K data points from each fragment and the other with 120K data points with 24K data points from each fragment. Thirdly, to answer the question "Do transformer models show the ability to generalise and show scale-invariance properties?", we evaluated the same transformers against a dataset containing more predicates (variables) than that of the training set (see Table 5.3). This provided clarity into transformers' ability to learn rules of inference from natural language satisfiability problem instances. Finally, to answer the question, "What factors affect

Fragment	T5-large	DeBERTa-v3-large
$\mathcal{S}$	88.3	89.0
$\mathcal{W}$	80.7	78.6
$\mathcal{V}$	79.5	77.2
$\mathcal{Z}$	75.1	75.2
$\mathcal{A}$	70.1	70.9

Table 5.1: Accuracy of transformer models (T5-large and DeBERTa-V3) when fine-tuned and evaluated across the fragments $\mathcal{S}$, $\mathcal{W}$, $\mathcal{V}$,$\mathcal{Z}$, and $\mathcal{A}$.

transformer models' ability to solve satisfiability problem instances?", we evaluated large-scale transformer-based language models in zero-shot settings. We employ a much smaller dataset in this evaluation and the dataset description is detailed in Section 5.8.4. The datasets we constructed were balanced with an equivalent number of satisfiable and unsatisfiable instances, and, consequently, we used accuracy as the evaluation metric.

5.4 Results and Discussion

The ability of transformer models to solve natural language satisfiability problems is affected by the underlying computational complexity of the problem, as determined by the language fragment in question. As shown in Table 5.1, the performance of transformers considered declines as the computational complexity class increases. It also can be observed that descent is steeper from $\mathcal{S}$ to $\mathcal{W}$ and more gradual henceforward. We posit two reasons for this phenomenon. Firstly, the problems that are in NLOGSPACE can be solved by the detection of relatively simple configurations (forbidden configuration of relatively low length in case the sentences are unsatisfiable) that transformers can easily learn to recognise, while those that are NPTIME-hard (or harder) are characterised by more intricate structures. Secondly, the problems in the simpler fragments have fewer tokens in their input sentences compared to those in the more complex language fragments; and transformers (like other neural approaches) find it more challenging to process long dependencies than short ones [21, 11]. However, computational complexity is not the only factor that influences the behaviour of the transformer models when performing logical inference tasks, as evidenced by the difference in performance when predicting the satisfiability of problem instances in $\mathcal{V}$ and $\mathcal{Z}$, both of which are EXPTIME-complete. The two fragments contain the same

Fragment	T5-large$_{600k}$	T5-LARGE$_{120k}$
$\mathcal{S}$	86.7	74.1
$\mathcal{W}$	85.0	73.7
$\mathcal{V}$	83.7	69.5
$\mathcal{Z}$	83.3	68.6
$\mathcal{A}$	82.2	68.0

Table 5.2: Accuracy of transformer models (T5-large) when trained on a dataset containing problem instances of all fragments; the test accuracies are broken down based on the language fragment, ($\mathcal{S}$, $\mathcal{W}$, $\mathcal{V}$,$\mathcal{Z}$, and $\mathcal{A}$). T5-large$_{600k}$, and T5-large$_{120k}$ indicate that the training set contains 600K and 120K data points respectively.

language properties, such as relative clauses and transitive verbs, but $\mathcal{Z}$ has sentences that retain both of those properties together while $\mathcal{V}$ does not. Thus, some sentences in $\mathcal{Z}$ are intrinsically more linguistically complicated than those in $\mathcal{V}$. As the neural approaches we employed are pre-trained language models that were trained on large language corpora, the linguistic complexity of the input has a noticeable effect on their performance.

Provided adequate data, learning to solve satisfiability problem instances of simpler fragments can help transformers learn to solve that of complex ones. When the dataset contains problem instances from all fragments, as shown in Table 5.2, the accuracy value for complex fragments increases when trained with an adequate number of data points (accuracy values for 600K). We hypothesise two reasons for this; first, fragments whose computational complexity for solving satisfiability problems is high have sentences from the fragments such as $\mathcal{S}$, which are much simpler; second, even in complex language fragments such as $\mathcal{A}$, the inference is heavily governed by the sentences belonging to simpler fragments (such as involved in proofs when the set of sentences is unsatisfiable). Thus, having problem instances belonging only to simpler fragments helps transformers decode reasoning patterns caused by sentences belonging to them. However, this raised a concern as to the true difficulty of the underlying dataset, even when sampled from the phase-change region. When the number of data points is low, say 120K data points, the performance of the transformers decreases for all fragments. We consider 24K data points from a single language fragment inadequate to learn any meaningful representation required for determining satisfiability. This is somewhat dissimilar to the results yielded by the computationally simpler rule reasoning

Fragment	$n_1 + n_2 \leq 20$	$n_1 + n_2 \geq 20$
$\mathcal{S}$	75.0	71.3
$\mathcal{W}$	75.3	71.2
$\mathcal{V}$	69.0	68.9
$\mathcal{Z}$	68.9	68.5
$\mathcal{A}$	64.3	58.0

Table 5.3: Transformer's (T5-large) ability to generalise to harder problems. The models were trained on problems with $6 \leq n_1 + n_2 \leq 16$ and evaluated against problems with $n_1 + n_2 \geq 16$.

problem of model-checking with natural language [78].

Transformer models failed to generalise and learn rules of inference required to solve satisfiability problem instances. We investigated whether transformer models learn to apply rules of inference and comprehend the underlying algorithm associated with the satisfiability-solving task by evaluating them against a dataset containing more predicates (thus more clauses) than that of the training test. As depicted in Table 5.3, transformers failed to generalise, and accuracy across all fragments decreased regardless of the computational complexity, suggesting transformers do not systematically generalise. This hypothesis is reinforced by the evident failure of transformers to generalise even for fragment $\mathcal{S}$, for which they exhibited a good accuracy level. Prior literature on the systematic generalisation of neural approaches [68, 45] also corroborates our finding that neural networks generally do not generalise well outside the training distribution. Moreover, experiments conducted by Richardson and Sabharwal [111] on scale invariance yielded equivalent results.

Pre-trained transformer-based language models do not achieve adequate performance when solving even the simplest satisfiability problems in zero-shot settings, and factors such as model architecture, number of variables and exact prompt can affect models' ability to follow instructions. The GPT-4 model achieved the best performance among the models we consider and is the only model whose results are reasonably higher than random guessing. As depicted in Figure 5.3, notably, the GPT-4 model outperformed ChatGPT by a considerable margin for all language fragments, which is in line with prior evaluations on algorithmic tasks such as coding and answering Mathematics questions [94]. Another notable observation is that the effect of the computational complexity of the language fragments when solving natural language

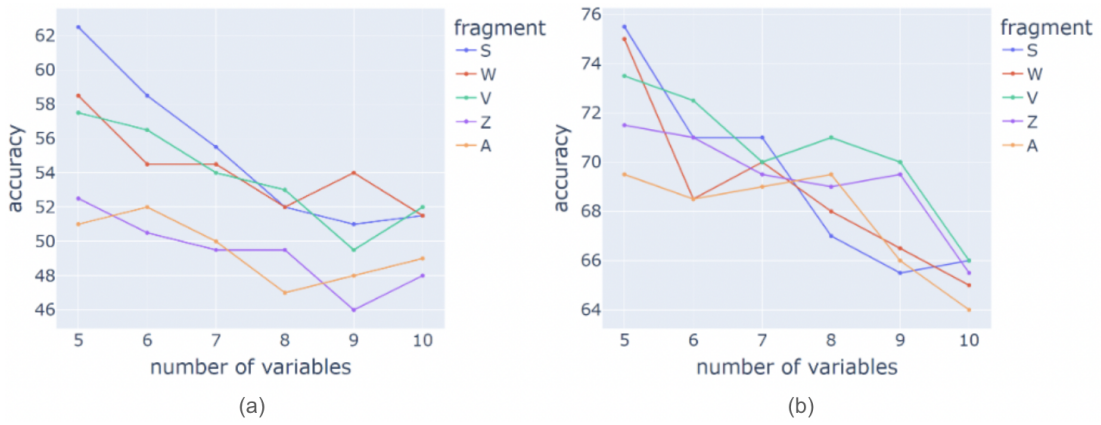

Figure 5.3: Variation of accuracy for (a) ChatGPT and (b) GPT-4 for language fragments, $\mathcal{S}$, $\mathcal{W}$, $\mathcal{V}$,$\mathcal{Z}$, and $\mathcal{A}$ for different number of variables

satisfiability problem instances is considerably lower compared to fine-tuned models.

A deeper analysis of the answers generated also shows that GPT-4 is better at explicitly following the instructions. For example, if the prompt indicates only to generate "satisfiable" or "unsatisfiable", and asks to generate nothing else, the GPT-4 model follows that instruction explicitly more often than the other models. Conversely, our experimentation with LLaMa-2 reveals a contrary trend; for most problem instances, the models do not follow the instructions and often generate something other than "satisfiable" or "unsatisfiable".

When we introduced a modification to the prompt by incorporating an illustrative example and requesting the generation of "True" or "False" instead of "satisfiable" or "unsatisfiable", the model consistently produced responses of True or False. We hypothesise the model's failure to adhere to the instructions of the original prompt can be attributed to two factors. Firstly, the terms "satisfiable" and "unsatisfiable" are relatively infrequent in the text data the language model has been trained on. Consequently, from a probabilistic standpoint, the likelihood of these specific terms being the most probable next tokens is low. Therefore, the model often generates something other than "satisfiable" or "unsatisfiable". Secondly, the inclusion of an illustrative example within the modified prompt provides the model with an unambiguous context regarding the desired output. However, even in this scenario, the model's performance is equivalent to random guessing.

Unsurprisingly, when the number of variables increases the accuracy of GPT models decreases. We posit two reasons for this observed phenomenon, Firstly, from a logical perspective, the increment in the number of variables increases the difficulty of solving

the satisfiability problem. As the number of variables grows, the search space expands, rendering the task of finding a valid assignment more challenging. Secondly, the increment of the number of variables increases the total number of tokens within a given problem instance. This increase in token count compels the model to grapple with longer-term dependencies in the input. This gradual decrement along with the recurring failure of fine-tuned transformers to generalise, leads to the hypothesis that, notwithstanding their impressive performance, state-of-the-art transformer models are still far from learning the rules of inference underlying logical reasoning tasks, and the algorithms required to apply them.

5.5 Related Work

Our work closely follows the literature on identifying strengths and weaknesses of neural approaches, including transformer-based language models on deductive reasoning tasks [81, 43, 73]. Neural networks, particularly graph neural networks (GNN), have been utilised to solve satisfiability problem instances [158, 19]. As mentioned in the introduction, various studies have extended that work to natural language satisfiability, utilising transformers instead of GNNs [111]. However, the sentences considered fail to exercise the full range of inference patterns licenced by commonly encountered grammatical constructions. Moreover, in each case, the underlying satisfiability problems were derived from that for propositional logic (i.e. SAT), thus, confining attention to a single problem type. Indeed in the case of natural language satisfiability, the sentences are sometimes hardly natural-sounding. For example, the Grounded Rule Language (GRL) fragment introduced in Richardson and Sabharwal [111] includes sentences such as *"If carrot and not steak then apples"*.

Our work can also be viewed as an attempt to identify various limitations that affect transformers' ability to emulate algorithms. There have been numerous studies of transformers' ability to solve algorithmic tasks, including SAT-solving [119, 90], semantic parsing [49, 61], model-checking [78, 80], theorem proving [146, 85, 114, 150] etc. We extend this work to a family of related problems spanning multiple computational complexity classes, thus, providing a breakdown of the impact of computational complexity on transformers' ability to solve algorithmically challenging problems.

5.6 Conclusion

We have investigated transformer-based language models' ability to solve instances of natural language problems belonging to different language fragments with varying computational complexity. Our investigation demonstrated that the computational complexity of the fragment has a noticeable effect on transformers' ability to perform a logical inferential task. Even in a simpler language fragment for which the transformer models achieved a reasonable performance, the model failed to adjust to distribution shift and generalise beyond its training distribution. Thus, we posit that transformer models do not reliably disentangle the patterns that exist within the dataset and the rules of inference needed when determining satisfiability. Therefore, it is imperative to acknowledge that a considerable body of research remains to be conducted in order to enhance the capacity of these models to comprehend rules of inference. We also acknowledge that the generalisation aspect of transformer models may need a deeper analysis, which we leave for future work.

5.7 Limitations

The empirical study presented in this paper exhibits two principal limitations. We explored several transformer models with different architectures that are in line with prior work and studied how computational complexity and other linguistic factors affect those models when learning an inferential task. However, due to the empirical nature of the research, there could exist a language model architecture whose behaviour deviates from the claims made in this paper. Similarly, the language fragments employed in this literature are not the only fragments in existence and, as such, would affect the comprehensiveness of the discussion.

Moreover, when constructing the dataset, we sampled problem instances from the phase-change region, and it was observed that the theorem prover took significantly more time to determine the satisfiability when data points were sampled from the phase-change region as opposed to random sampling. However, it does not provide a guarantee as to whether all instances are non-trivial, and it is probable that some problem instances are still trivial to solve.

5.8 Supplementary materials

5.8.1 Definition of language fragments

The sentence templates defining $\mathcal{S}$, together with the corresponding formula templates, are as follows:

Every (non-) p is a q No (non-) p is a q
Some (non-) p is (not) a q.

We can define the first-order formulae for giving semantics for the above sentence templates as follows:

$$\forall x(\pm p(x) \rightarrow \pm q(x))$$
$$\exists x(\pm p(x) \wedge \pm q(x)),$$

where $\pm\psi$ is either ψ or $\neg\psi$.

The sentence templates for $\mathcal{W}$ are those of $\mathcal{S}$ together with

Every (non-) o who is (not) a p is a q
No (non-) o who is (not) a p is a q
Some (non-) o who is (not) a p is (not) a q.

We can define the first-order formulae for giving semantics for the above sentence templates as follows:

$$\forall x((\pm o(x) \wedge \pm p(x)) \rightarrow \pm q(x))$$
$$\exists x(\pm o(x) \pm p(x) \wedge \pm q(x)).$$

The sentence templates for $\mathcal{V}$ are those of $\mathcal{W}$ together with

Some/every (non-) p r's some/every (non-) q
No (non-) p r's any/every (non-) q
Some (non-) p does not r any/every (non-) q.

We can define the first-order formulae for giving semantics for the above sentence templates as follows,

$$\forall x(\pm p(x) \rightarrow \forall y(\pm q(y) \rightarrow \pm r(x,y)))$$
$$\forall x(\pm p(x) \rightarrow \exists y(\pm q(y) \wedge \pm r(x,y)))$$
$$\exists x(\pm p(x) \wedge \forall y(\pm q(y) \rightarrow \pm r(x,y)))$$
$$\exists x(\pm p(x) \wedge \exists y(\pm q(y) \wedge \pm r(x,y))).$$

The sentence templates for $\mathcal{Z}$ are those of $\mathcal{V}$ together with

Some (non-) o who r's/does not r
some/every (non-) p is (not) a q
Every (non-) o who r's/does not r
some/every (non-) p is a q
No (non-) o who r's/does not r
any/every (non-) p is a q.

We can define the first-order formulae for giving semantics for the above sentence templates as follows:

$$\forall x(\pm o(x) \wedge \forall y(\pm p(y) \rightarrow \pm r(x,y)) \rightarrow \pm q(x))$$
$$\forall x(\pm o(x) \wedge \exists y(\pm p(y) \wedge \pm r(x,y)) \rightarrow \pm q(x))$$
$$\exists x(\pm o(x) \wedge \forall y(\pm p(y) \rightarrow \pm r(x,y)) \wedge \pm q(x))$$
$$\exists x(\pm o(x) \wedge \exists y(\pm p(y) \wedge \pm r(x,y)) \wedge \pm q(x)).$$

The sentence templates for $\mathcal{A}$ are those of $\mathcal{Z}$ together with

Some/every (non-) o r's some/every (non-)
p who s's/does not s him
No (non-) o r's any/every (non-) p who
s's/does not s him
Some (non-) o does not r any/every (non-) p
who s's/does not s him.

We can define the first-order formulae for giving semantics for the above sentence templates as follows:

$$\forall x(\pm o(x) \rightarrow \forall y(\pm p(y) \wedge \pm s(y,x) \rightarrow \pm r(x,y)))$$
$$\forall x(\pm o(x) \rightarrow \exists y(\pm p(y) \wedge \pm s(y,x) \wedge \pm r(x,y)))$$
$$\exists x(\pm o(x) \wedge \forall y(\pm p(y) \wedge \pm s(y,x) \rightarrow \pm r(x,y)))$$
$$\exists x(\pm o(x) \wedge \exists y(\pm p(y) \wedge \pm s(y,x) \wedge \pm r(x,y))).$$

5.8.2 Computational Complexity

In this section, we justify the complexity-theoretic information provided in Section 5.2.1. The case of $\mathcal{S}$ requires no work: this is the fragment denoted $\mathcal{S}^{\dagger}$ in [103], where the complexity of satisfiability is established in Theorem 4.10, p. 668. For $\mathcal{W}$, the upper bound of NPTIME follows easily from the fact that all the relevant formulae are in the 1-variable fragment of first-order logic; the lower bound follows by a straightforward reduction from 3-SAT, as given, for example, for the smaller fragment denoted $\mathcal{E}_1$

in [101, Theorem 2, p. 213]. The fragment $\mathcal{V}$ is slightly larger than the fragment denoted $\mathcal{E}_2$ in [101], whose satisfiability problem is shown to be EXPTIME-complete in Theorem 3, p.215. This settles the lower bound for Sat($\mathcal{V}$). For the upper bound, we note that the proof depends essentially on the fact that all of the first-order formulae involved contain just one occurrence of any binary predicate; therefore, the same proof applies to the fragment $\mathcal{V}$ almost without change of wording. Furthermore, the same argumentation applies to $\mathcal{Z}$. The only remaining case to consider is $\mathcal{A}$, which is dealt with by the following theorem.

Theorem 1 The computational complexity of the problem of determining satisfiability in a set of sentences in $\mathcal{A}$ is NEXPTIME complete

Proof: A moment's thought shows that every sentence of $\mathcal{A}$ is expressed by a formula of the two-variable fragment of first-order logic, $\mathcal{FO}^2$, whose satisfiability problem is NEXPTIME-complete [46]. Thus, Sat($\mathcal{A}$) is in NEXPTIME. To establish a matching lower bound, we take an existing proof of the NEXPTIME-hardness of Sat($\mathcal{FO}^2$) and show that it can be reproduced using only the resources of $\mathcal{A}$. The proof in question proceeds by reduction from a known NEXPTIME-hard problem $\mathcal{P}$ (in this case, $\mathcal{P}$ is the problem of determining whether an exponential-sized grid can be tiled in a certain tiling system), and encoding any instance of $\mathcal{P}$ as a set Φ of $\mathcal{FO}^2$-formulae such that Φ is satisfiable if and only if the given instance of $\mathcal{P}$ is positive. First, let us define the short-forms $\lambda_X := \bigvee_{i=0}^{n-1} \neg X_i(x)$, $\varepsilon_X := \bigwedge_{i=0}^{n-1}(X_i(x) \leftrightarrow X_i(y))$, $\iota_X(x,y) := \bigwedge_{i=0}^{n-1}((X_i(x) \leftrightarrow X_i(y)) \leftrightarrow \neg \bigwedge_{j=0}^{i-1} X_j(x))$ and $\eta(x,y) := \iota_X(x,y) \wedge \varepsilon_Y(x,y)$. Then, it can be shown that such encodings (see for example [102] pp. 90 ff) require only that we can write $\mathcal{FO}^2$-formulae of the following forms:

$$\forall x \bigvee_{k=1}^{m} c_k(x) \wedge \forall x \bigwedge_{i,j=1, i\neq j}^{m} \neg(c_i(x) \wedge c_j(x)) \tag{5.1}$$

$$\exists x[(\neg X_0(x) \wedge \ldots \wedge \neg X_n(x)) \wedge (\neg Y_0(x) \wedge \ldots \wedge \neg Y_n(x))] \tag{5.2}$$

$$\forall x[(\pm X_0(x) \wedge \ldots \wedge \pm X_n(x)) \wedge (\pm Y_0(x) \wedge \ldots \wedge \pm Y_n(x)) \rightarrow c_i(x)] \tag{5.3}$$

$$\forall x(\lambda_X(x) \rightarrow \exists y \eta(x,y)) \tag{5.4}$$

$$\forall x \forall y((\eta(x,y) \wedge c(x)) \rightarrow \neg c'(y)). \tag{5.5}$$

Thus, if we can reproduce the effect of such formulae in $\mathcal{A}$, then we will have established that Sat($\mathcal{A}$) is NEXPTIME-hard. For identifying a set of equisatisfiable formulae for form (5.1), consider the forms on either side of the conjunction separately. Form $\forall x \bigvee_{k=1}^{m} c_k(x)$ is equisatisfiable with the conjunction of $\forall x(c_1(x) \vee d_1(x))$,

$\forall x(d_1(x) \to (c_2(x) \vee d_2(x)), \forall x(d_2(x) \to (c_3(x) \vee d_3(x)), \ldots, \forall x(d_{m-2}(x) \to (c_{m-1}(x) \vee c_m(x))$ where $d_1, d_2, \ldots, d_{m_2}$ are fresh unary predicates. Both the form $\forall x(c_1(x) \vee d_1(x))$ and forms of the type $\forall x(d_i(x) \to (c_k(x) \vee d_j(x)) \equiv \forall x((\neg c_k(x) \wedge \neg d_k(x)) \to \neg d_i(x))$ are directly in $\mathcal{A}$. Form $\forall x \neg(c_i(x) \wedge c_j(x))$ is directly in $\mathcal{A}$. Therefore, we can find a set of equisatisfiable formulae for form (5.1) in $\mathcal{A}$. Form (5.2) is treated similarly. Let us consider form (5.3). It is equisatisfiable with the conjunction of the following forms,

$$\forall x(P_i(x) \wedge Q_i(x) \to c_i(x)) \tag{5.6}$$

$$\forall x((\pm X_0(x) \wedge \ldots \wedge \pm X_n(x)) \to P_i(x)) \tag{5.7}$$

$$\forall x((\pm Y_0(x) \wedge \ldots \wedge \pm Y_n(x)) \to Q_i(x)). \tag{5.8}$$

We can represent form (5.3) by an equisatisfiable set of formulae in $\mathcal{A}$, since (5.6) is directly in $\mathcal{A}$ and (5.7) and(5.8) can be represented similarly to $\forall x \bigvee_{i=1}^{m} c_i(x)$. Now, let us direct our attention to form (5.4). Let r be a fresh binary predicate and consider the forms,

$$\forall x \forall y(r(x,y) \to \eta(x,y)) \tag{5.9}$$

$$\forall x(\lambda_X(x) \to \exists y r(x,y)). \tag{5.10}$$

Evidently, the form (5.4) is equisatisfiable with the conjunction of (5.9) and (5.10). Writing out $\lambda_X := \bigvee_{i=0}^{n-1} \neg X_i(x)$, form (5.10) becomes

$$\forall x(\bigvee_{i=0}^{n-1} \neg X_i(x) \to \exists y r(x,y)) \equiv \bigwedge_{i=0}^{n-1} \forall x(\neg X_i(x) \to \exists y r(x,y)). \tag{5.11}$$

Form (5.11) is in $\mathcal{A}$. Now, let us consider form (5.9), we can re-write it as follows:

$$\forall x \forall y(r(x,y) \to \eta(x,y)) \equiv \forall x \forall y(r(x,y) \to \bigwedge_h \gamma_h(x,y)) \tag{5.12}$$

$$\equiv \bigwedge_h \forall x \forall y(r(x,y) \to \gamma_h(x,y)) \tag{5.13}$$

where $\gamma_h(x,y))$ is either $X_i(x) \leftrightarrow X_i(y)$ or $(X_i(x) \leftrightarrow X_i(y)) \leftrightarrow \neg \bigwedge_{j=0}^{i-1} X_j(x)$. Let us consider each of the formulae separately. Consider form $\forall x \forall y(r(x,y) \to (X_i(x) \leftrightarrow$

$X_i(y)))$. We can write it as a conjunction of the following two forms,

$$\forall x \forall y(r(x,y) \rightarrow (X_i(x) \rightarrow X_i(y))) \tag{5.14}$$

$$\forall x \forall y(r(x,y) \rightarrow (X_i(y) \rightarrow X_i(x))). \tag{5.15}$$

We can write form (5.14) equivalently as $\forall x \forall y((X_i(x) \wedge \neg X_i(y)) \rightarrow \neg r(x,y))$, which in in $\mathcal{A}$ and form (5.15) can be represented in a similar manner. Next consider $\forall x \forall y(r(x,y) \rightarrow ((X_i(x) \leftrightarrow X_i(y)) \leftrightarrow \neg \bigwedge_{j=0}^{i-1} X_j(x)))$, this is equisatisfiable with the conjunction of the following forms,

$$\forall x \forall y(r(x,y) \rightarrow (s(x,y) \leftrightarrow p(x))) \tag{5.16}$$

$$\forall x \forall y(s(x,y) \leftrightarrow (X_i(x) \leftrightarrow X_i(y))) \tag{5.17}$$

$$\forall x(p(x) \leftrightarrow \neg \bigwedge_{j=0}^{i-1} X_j(x)) \tag{5.18}$$

where p is a fresh unary predicate while s is a fresh binary predicate. By considering either side of the if-and-only-if condition, we can re-write (5.16) as a conjunction of the following,

$$\forall x \forall y(r(x,y) \rightarrow (s(x,y) \rightarrow p(x))) \tag{5.19}$$

$$\forall x \forall y(r(x,y) \rightarrow (p(x) \rightarrow s(x,y))). \tag{5.20}$$

Moreover, the (5.19) is equivalent to $\forall x \forall y(\neg p(x) \rightarrow (r(x,y) \rightarrow \neg s(x,y)))$, which is equisatisfiable with the conjunction of $\forall x \forall y(\neg p(x) \rightarrow (r(x,y) \rightarrow l(y,x)))$ and $\forall x \forall y(l(x,y) \rightarrow \neg s(y,x))$ where l is a fresh binary predicate. Furthermore, the first formula in conjunction is equisatisfiable with the conjunction of the following two forms

$$\forall x \forall y(\neg p(x) \rightarrow (z(y) \wedge (r(x,y)) \rightarrow l(y,x))) \tag{5.21}$$

$$\forall x \forall y(\neg p(x) \rightarrow (\neg z(y) \wedge (r(x,y)) \rightarrow l(y,x))) \tag{5.22}$$

where z is a fresh unary predicate. Evidently, both (5.21) and (5.22) are in $\mathcal{A}$. The form $\forall x \forall y(l(x,y) \rightarrow \neg s(y,x))$ can be written similarly. Thus, we can identify a set of equisatisfiable formulae for form (5.19) in $\mathcal{A}$. We can write the form (5.20) in a similar manner to (5.19), and form (5.17) can be written similarly to the form (5.14). Form (5.18) is equisatisfiable with a set of formulae of the type $\forall x(p(x) \rightarrow X_j(x))$ conjunction with the set of formulae of the type $\forall x(X_j(x) \rightarrow p(x))$, both of which can

be written directly in $\mathcal{A}$. Therefore, we can identify a set of equisatisfiable formulae for form (5.9). Consequently, we can represent form (5.4) by an equisatisfiable set of formulae in $\mathcal{A}$. The above-mentioned approach of finding an equisatisfiable set of formulae can be repeated to form (5.5) as well. Indeed, the form (5.5) is equisatisfiable with $\forall x \forall y((r(x,y) \wedge c(x)) \rightarrow \neg c'(y))$ together with $\forall x \forall y(\eta(x,y) \rightarrow r(x,y))$. Later of the two formulae, $\forall x \forall y(\eta(x,y) \rightarrow r(x,y))$ is equivalent to $\forall x \forall y(\neg r(x,y) \rightarrow \neg \eta(x,y))$ and can be treated similarly to (5.9). Moreover, $\forall x \forall y((r(x,y) \wedge c(x)) \rightarrow \neg c'(y))$ is equivalent to $\forall x \forall y((c'(y) \wedge c(x)) \rightarrow \neg r(x,y))$ which is in $\mathcal{A}$.

Thus, any instance of an exponential tiling problem can be transformed, in polynomial time, to an instance of Sat($\mathcal{A}$) with the same answer. This establishes the NExpTime-hardness of Sat($\mathcal{A}$).

5.8.3 Zero-shot Prompting Templates

Given a set of sentences $s_1, s_2, ..., s_m$ and a label ℓ indicating whether the set of sentences is satisfiable or not, we formulate the prompt using the following template.

Q: Given the following set of sentences, tell me whether they are satisfiable or not. Generate satisfiable if they are and unsatisfiable if they are not. Set of sentences: $s_1, s_2, ..., s_m$

A:

As we mentioned in the Results and Discussion section, the LLaMa model did not generate "satisfiable" or "unsatisfiable" for the above prompt for most problem instances. Therefore, we modified the prompt by following an example problem instance and asking it to generate the words "True" or "False" instead. The example problem instance consists of a set of sentences $\bar{s}_1, \bar{s}_2, ..., \bar{s}_m$ and a satisfiability label be $\bar{\ell}$ (where $\bar{\ell}$ is "True" if $\bar{s}_1, \bar{s}_2, ..., \bar{s}_m$ is satisfiable and "False" otherwise). The resultant prompt is as follows:

Q: Given the following set of sentences, tell me whether they are satisfiable or not. Generate True if they are and False if they are not.

Set of sentences: $\bar{s}_1, \bar{s}_2, ..., \bar{s}_m$

A: $\bar{\ell}$

Given the following set of sentences, tell me whether they are satisfiable or not. Generate True if they are and False if they are not.

Set of sentences: $s_1, s_2, ..., s_m$

A:

Language Fragment	minimum	maximum	mean
$\mathcal{S}$	29	8	17.06
$\mathcal{W}$	32	9	18.95
$\mathcal{V}$	25	5	14.00
$\mathcal{Z}$	28	6	14.31
$\mathcal{A}$	34	8	16.50

Table 5.4: The minimum, maximum and mean number of clauses m in the training set for the fragments we consider, $\mathcal{S}$, $\mathcal{W}$, $\mathcal{V}$, $\mathcal{Z}$ and $\mathcal{A}$

5.8.4 Dataset details

We utilised five different language fragments, namely, Syllogistic $\mathcal{S}$, relative clauses $\mathcal{W}$, relative clauses with relational syllogistic $\mathcal{V}$, relative clauses with transitive verbs $\mathcal{Z}$ and anaphora $\mathcal{A}$. For each language fragment, we constructed a training set with 120K, an eval set with 10K and a test set with 10K data points. When constructing the dataset, we set the number of unary predicates n_1 to be between 3 and 8 and the number of binary predicates n_2 to be between 3 and 8 for fragments $\mathcal{V}$, $\mathcal{Z}$ and $\mathcal{A}$ while setting the n_1 value to be between 6 and 16 for fragments $\mathcal{S}$ and $\mathcal{W}$. It is apparent that the distribution of the number of ways specific numbers between 6 and 16 can be formed by randomly adding two numbers in the range 3 and 8 formed a bell curve. Hence, the distribution of the number of problem instances per number of variables of $\mathcal{V}$, $\mathcal{Z}$ and $\mathcal{A}$ seems to be a discrete approximation of normal distribution. Therefore, we sample the n_1 values from a normal distribution for $\mathcal{S}$ and $\mathcal{W}$ while sampling n_1, n_2 values uniformly for the other fragments.

When testing for transformers' ability to generalise, we construct a separate test set with more predicates than that of the training set. We construct 6K data points for each fragment and the dataset is balanced having an equal number of satisfiable and unsatisfiable problem instances. In this setup, we set the range of the n_1 value in the test set between 16 and 24 for fragments $\mathcal{S}$, and $\mathcal{W}$, while varying it between 8 and 12 for other fragments. The range of the number of binary predicates n_2 for fragments $\mathcal{V}$, $\mathcal{Z}$ and $\mathcal{A}$ was set to be between 8 and 24 due to the asymmetric nature of the variation of probability of satisfiability with $\frac{m}{n_1}$ compared to $\frac{m}{n_2}$. It is important to emphasise that all data points belonging to each of the datasets (train, eval and test) are unique.

To evaluate transformers in zero-shot settings, we construct a separate test set with

Language Fragment	minimum	maximum	mean
$\mathcal{S}$	149	35	83.19
$\mathcal{W}$	267	43	129.08
$\mathcal{V}$	221	22	97.37
$\mathcal{Z}$	274	29	110.22
$\mathcal{A}$	318	39	123.42

Table 5.5: The minimum, maximum and mean number of words (tokens) when separated by SPACE in the training set for the language fragments

1200 data points for each fragment, with 600 data points being satisfiable while the other data points being unsatisfiable. In this setup, we set $n_1^{min} = 5$, and $n_1^{max} = 10$ for fragments $\mathcal{S}$ and $\mathcal{W}$ and $n_1^{min} = 3$, $n_1^{max} = 5$, $n_2^{min} = 2$ and $n_2^{max} = 5$ for fragments $\mathcal{V}$, $\mathcal{Z}$, and $\mathcal{A}$. . Moreover, for each value n where $n = n_1 + n_2$, the dataset contains 200 data points, with 100 of them being satisfiable and the other 100 being unsatisfiable.

When defining the vocabulary, we expanded the vocabulary introduced by Richardson and Sabharwal [111]. The vocabulary of unary predicates $\mathcal{U}$ comprises 156 nouns (of people's professions) and the vocabulary of binary predicates $\mathcal{V}$ comprises 70 transitive verbs.

The mean, maximum, and minimum number of clauses for each of the language fragments are depicted in Table 5.4, while mean, maximum, and minimum instance lengths are illustrated in Table 5.5.

5.8.5 Training Details

We chose two transformer models with somewhat different architectures; T5 and DeBERTa-v3, as both models have proven track records on textual entailment tasks. Since the research question of interest relies on the learnability of transformers, not on identifying the best-performing model, we do not perform any hyperparameter tuning. According to prior literature [108], the performance of the transformers stems from model size and pre-trained data more so than the architectural choice, as well as since the accuracy values yielded for the transformers are similar, we anticipated similar performance for other transformers. Moreover, exploring different transformer models and hyper-parameter tuning would leave a higher carbon footprint [126], which is

deemed unnecessary considering the nature of the research questions.

A detailed description of the hyperparameters is as follows,

Maximum sequence length : When training the DeBERTa-v3 model, we used the maximum sequence length between 512 and 700 tokens, and when training the T5 model using the joint dataset, we used 700 tokens as the maximum sequence length. Since we used the T5 models trained on individual fragments for testing for generalisation, we used a maximum sequence length of 1024 tokens. We did not rely on any truncation, as truncating input could alter the satisfiability of the input sentences.

Training epochs : Five, the best model was identified using the eval set.

Batch size : Relying on the gradient checkpointing, we used a batch size of 24 for the DeBERTa-v3-large model. Similarly, when trained with the joint dataset, we used a batch size of 24, again relying on gradient checkpointing. Since we used the T5 model trained on the individual dataset for generalisation experiments, we downgraded the batch size to 12 to prevent memory errors.

Each of the transformer models is fined-tuned to predict the label by reducing the binary cross entropy loss over the target using Adam optimiser [65] and we used the HuggingFace [154] implementation when experimenting with the above-mentioned transformer models.

5.8.6 Phase Change Region of Language Fragments

We explore the problem distribution of the language fragments we considered along several analytical viewpoints. The quantifier ratio is used to define the ratio of the number of $\exists$ quantifiers to the number of $\forall$ quantifiers, and the subject quantifier ratio is the quantifier ratio between the quantifiers associated with the subject while the object quantifier ratio is that associated with the object. Since sentences belonging to the fragments $\mathcal{S}$ and $\mathcal{W}$ contain only one quantifier, which is associated with the subject, the object quantifier ratio is omitted for those fragments. The visualisation of variation in the probability of satisfiability is depicted in figures 5.4, 5.5, 5.7, 5.8 and 5.6. Figure 5.4 illustrates how the probability of satisfiability varies with $\frac{m}{n_1}$ and $\frac{m}{n_2}$, which we have used when defining the phase change region.

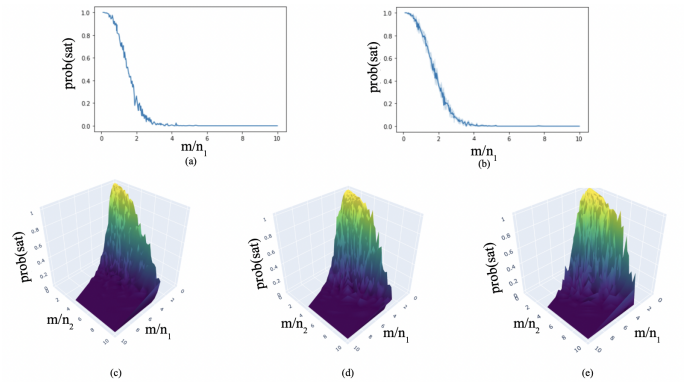

Figure 5.4: The variation of probability of satisfiability with clause unary and binary variable ratios for the language fragments (a) $\mathcal{S}$, (b) $\mathcal{W}$, (c) $\mathcal{V}$, (d) $\mathcal{Z}$ and (e) $\mathcal{A}$. The symbol m denotes the number of clauses, and n_1, and n_2 indicate the number of unary and binary variables respectively.

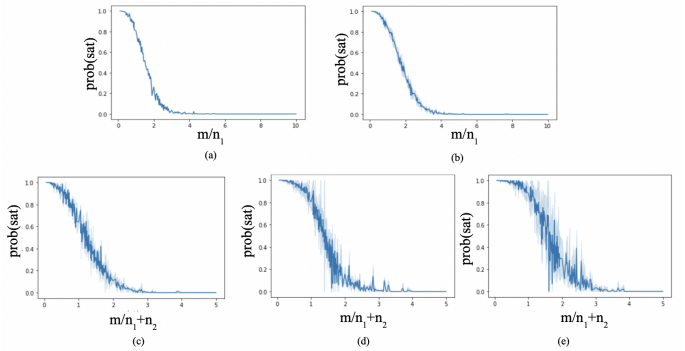

Figure 5.5: The variation of probability of satisfiability with clause variable ratio for the language fragments (a) $\mathcal{S}$, (b) $\mathcal{W}$, (c) $\mathcal{V}$, (d) $\mathcal{Z}$ and (e) $\mathcal{A}$, where we consider the total variables $n_1 + n_2$.

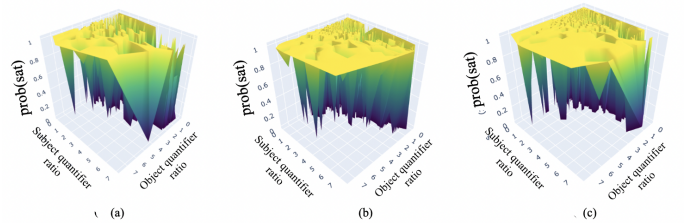

Figure 5.6: The variation of probability of satisfiability with subject quantifier ratio and object quantifier ratio for the language fragments (a) $\mathcal{V}$, (b) $\mathcal{Z}$ and (c) $\mathcal{A}$

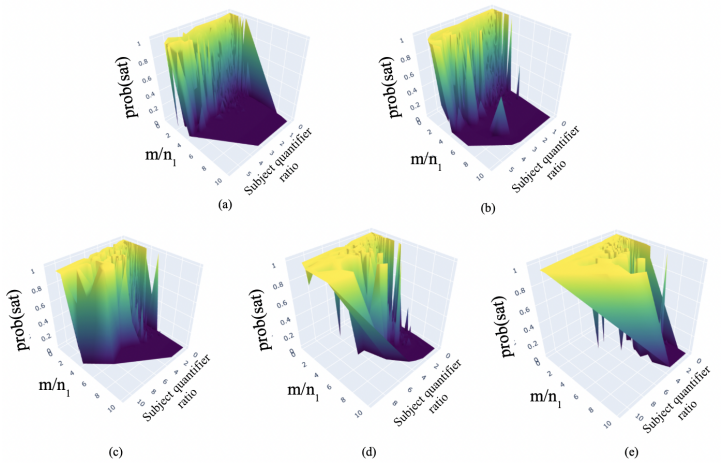

Figure 5.7: The variation of probability of satisfiability with clause unary variable ratio and subject quantifier ration for the language fragments (a) $\mathcal{S}$, (b) $\mathcal{W}$, (c) $\mathcal{V}$, (d) $\mathcal{Z}$ and (e) $\mathcal{A}$. The symbol m denotes the number of clauses, and n_1 indicates the number of unary variables respectively.

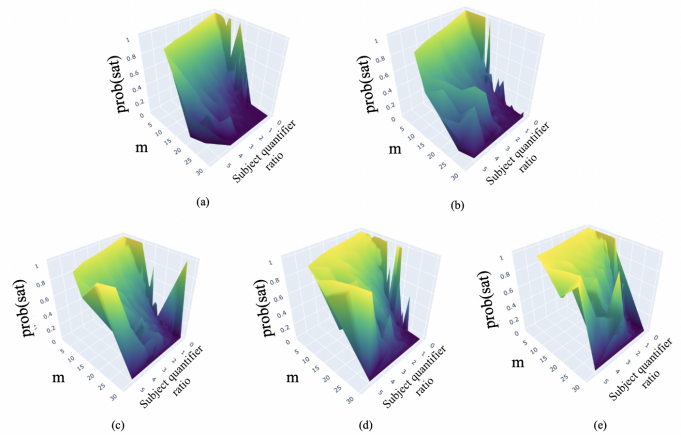

Figure 5.8: The variation of probability of satisfiability with the number of clauses and subject quantifier ratio for the language fragments (a) $\mathcal{S}$, (b) $\mathcal{W}$, (c) $\mathcal{V}$, (d) $\mathcal{Z}$ and (e) $\mathcal{A}$. The symbol m denotes the number of clauses.

Chapter 6

Investigating the Generalisation of Transformers in Numerical Satisfiability Problems

"It is not the strongest of the species that survives,
nor the most intelligent that survives.
It is the one that is most responsive to change"
– Charles Darwin

This chapter is based on a paper titled *"Unravelling the Logic: Investigating the Generalisation of Transformers in Numerical Satisfiability Problems"*. The paper was accepted and published at the Proceeding of the Association for Computational Linguistics: `ACL 2025` (main conference)..

In the previous chapters, we explored the transformer models' ability to solve natural language satisfiability problems belonging to various computational complexity classes. However, this work was limited to first-order quantifiers. In this chapter, we extend the work on natural language satisfiability to the problem of numerical satisfiability: a variant of natural language satisfiability whose problems contain numerical quantifiers. Furthermore, we place particular emphasis on different forms of generalisation, exploring how transformer models handle variations in problem structure and intricacy.

Abstract

Transformer models have achieved remarkable performance in many formal reasoning tasks. Nonetheless, the extent of their comprehension pertaining to logical semantics and rules of inference remains somewhat uncertain. Evaluating such understanding necessitates a rigorous examination of these models' generalisation capacity to out-of-distribution data. In this study, we probe the generalisation prowess of transformer models with respect to the hitherto unexplored domain of numerical satisfiability problems. Our investigation reveals that transformers exhibit minimal scale and noise invariance, alongside limited vocabulary and number invariance. However, even when transformer models experience a notable decline in performance on out-of-distribution test sets, they often still surpass the random baseline.

6.1 Introduction

Transformer models have become the de facto state-of-the-art in solving almost all language-based tasks [30, 160, 108, 94]. Notably, among these tasks, formal reasoning has garnered considerable interest in recent times [110, 131, 81]. Formal reasoning delineates a distinct strand of reasoning characterised by the drawing of conclusions solely from logical rules, without relying on commonsense or background knowledge. Although transformers have exhibited notable proficiency in formal reasoning tasks [133, 16, 66], the depth of their comprehension regarding logical semantics and rules of inference remains uncertain.

Many logical problems, and in particular the problem of recognising valid entailments, can be reduced to the problem of determining satisfiability: a set of closed formulae Φ is *satisfiable* if there exists some structure (in a model-theoretic sense) $\mathfrak{A}$ in which every element of Φ is true. This concept extends to natural language in an obvious way: a set of *sentences* is satisfiable if the set of *formulae* into which they translate is satisfiable. The problem of satisfiability, therefore, manifests as the task of determining whether there are any inherent contradictions within this set of sentences. Consequently, natural language variants of satisfiability problems represent an ideal domain for studying transformer models' ability to learn rules of inference and logical semantics. In our study, we extend research interest in the formal reasoning aspect of transformers by introducing problems that involve numerical reasoning, by considering the problem of numerical satisfiability: a variant of the satisfiability problem in which

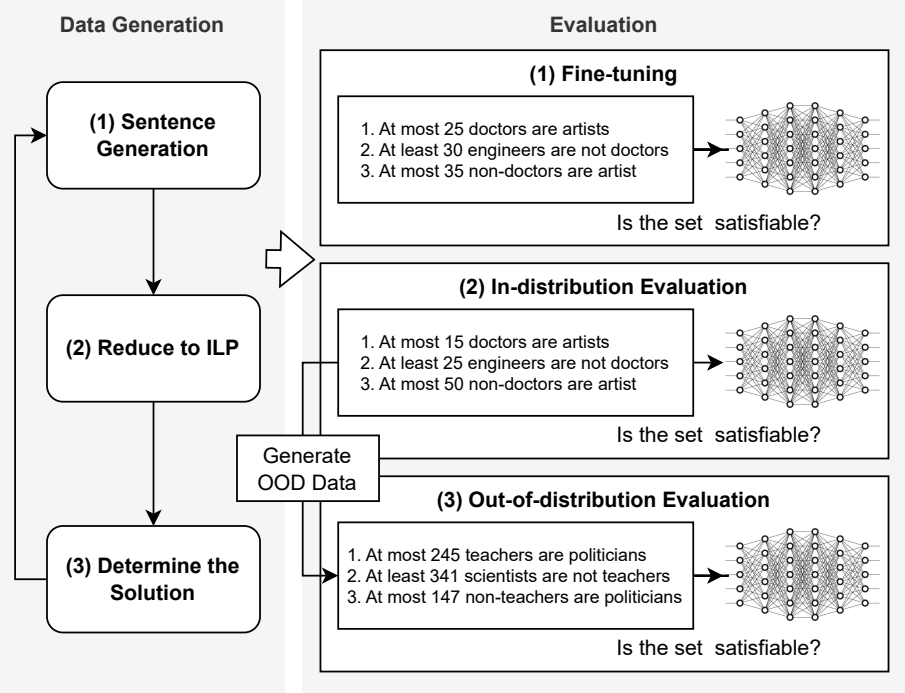

Figure 6.1: A data generation methodology that reduces sets of generated sentences to sets of inequalities and employs a linear solver to determine the solution. The resulting dataset is then used to fine-tune and evaluate transformers on both in-distribution and out-of-distribution (OOD) problems to test robustness and generalisation.

the formulae (or sentences) include numerical elements. For illustration, consider the following set of sentences:

Consider the following set of sentences,

1. *At most 25 doctors are artists*
2. *At least 30 engineers are not doctors*
3. *At most 35 non-doctors are artist*
4. *At least 100 engineers are artists*

Sentences 1, 3 and 4 form an unsatisfiable set: the first two entail there to be at most 60 artists, which directly contradicts sentence 4. On the other hand, sentences 1, 2 and 3 form a satisfiable set: as may be easily seen, there is a structure in which these sentences are all true.

When learning to solve instances of numerical satisfiability problems requires an understanding of logical and mathematical semantics along with acquiring and applying logical and mathematical rules. Furthermore, this framework offers a high degree of controllability, enabling the systematic scaling of the problem space, automated

evaluation of solutions, and the identification of hard-problem regions. Moreover, numerical satisfiability introduces problem instances that are computationally more complex than their first-order counterparts. Therefore, this problem provides a unique approach to evaluating transformer models' ability to perform numerical reasoning. To the best of our knowledge, this problem has not been investigated in the previous literature.

The primary reason for this limitation is that relatively few reasoning tools can directly ascertain the satisfiability of formulae incorporating numbers or numerical quantifiers. Consequently, there arises the challenge of constructing an extensive dataset suitable for fine-tuning and assessing transformers. We overcome this limitation by reducing the numerical satisfiability problems into integer linear problems (see Figure 6.1). Moreover, we acknowledge the potential drawbacks associated with evaluating models using synthetic data, particularly the pitfalls associated with undersampling challenging instances [156, 122]. To address this concern and ensure a robust dataset, we adopt a targeted sampling approach by selecting problem instances from the phase-change region. This region, where the probability of satisfiability is approximately 0.5, represents a critical region where algorithms tasked with determining satisfiability typically experience prolonged running times. Consequently, we conduct a comprehensive exploration of the problem space associated with the numerical satisfiability problems under consideration to delineate the boundaries of the phase-change region.

Even if transformer models learn to solve problem instances of numerical satisfiability, this is not indicative of their ability to learn logical semantics and rules of inference. Indeed, this aspect requires a comprehensive evaluation of these models' generalisation ability. Consequently, we evaluate transformer models' generalisation ability across several axes. (1) Vocabulary invariance: Do transformers demonstrate sensitivity to out-of-vocabulary (OOV) terms? (2) Numeracy invariance: Can transformers generalise to different numerical values that were unseen during fine-tuning? (3) Scale invariance: Are transformers capable of generalising to problems of larger scope? (4) Noise invariance: How sensitive are transformers to noisy sentences which do not affect the satisfiability of the problem? Together, these evaluative dimensions furnish a comprehensive understanding of transformers' generalisation abilities, spanning both their comprehension of logical and numerical semantics in natural language and their proficiency in learning and applying mathematical and logical rules.

The contributions of this paper are as follows. (1) Based on the principles of mapping satisfiability problems to integer linear problems [102, Ch. 7], we design an

algorithm to construct numerical satisfiability problems. (2) We explore the problem space of the constructed numerical satisfiability problems to establish the phase-change region. (3) We design a systematic investigation to evaluate transformers' ability to learn logical semantics and rules of inference from natural language text. (4) We evaluate a diverse range of transformers, encompassing varying sizes and architectures, in both fine-tuned (eg. Flan-T5 [25], Gemma [135]) and zero-shot/ few-shot settings(eg. GPT 4,[94], GPT 3.5 [16], Mistral 7B [59]).

6.2 Methodology

6.2.1 Language Fragments

We reference the definitions of language fragments and sentence templates introduced in Chapter 2.2 and outline the sentence templates that define the Aristotelian syllogistic [7] below.

Every A is a B	*Some A is a B*
No A is a B	*Some A is not a B*

We introduce two modifications to the Aristotelian syllogistic: (1) allow negations in the subject. (2) replace the quantifiers *all*, *some*, *no* with the numerical quantifiers *at least K* and *at most K*, where $K \in \mathbb{N}^+$ ($\mathbb{N}^+ = \{1,2,3,\dots\}$). The resulting fragment, which we refer to as the *counting fragment* or $\mathcal{C}$, can be defined by the following set of templates,

At least K A are B	*At least K A are not B*
At least K non-A are B	*At least K non-A are not B*
At most K A are B	*At most K A are not B*
At most K non-A are B	*At most K non-A are not B*

The problem of determining the satisfiability of a set of sentences in $\mathcal{C}$ is NPTIME-complete [67]. Given a set of sentences in $\mathcal{C}$, we derive a system of linear inequalities as described in the next section.

6.2.2 Reduction to Integer Linear Problems

Let us say we are given a set of sentences S in $\mathcal{C}$ over a signature $P_1, P_2, \dots, P_n$, for which we aim to determine the satisfiability.

If ψ is a formula let $\pm\psi$ be either ψ or $\neg\psi$. We call a conjunction of the form $\pm P_1(x) \wedge \pm P_2(x) \wedge \cdots \wedge \pm P_n(x)$ an *atomic 1-type* over the signature $(P_1, P_2, \ldots, P_n)$; in the sequel, we omit the quantifier "atomic" for brevity. We list the 1-types over $(P_1, P_2, \ldots, P_n)$ in some fixed order $\pi_1, \pi_2, \ldots, \pi_N$ where $N = 2^n$.

If $\mathfrak{A}$ is a structure interpreting the signature $P_1, P_2, \ldots, P_n$, over some domain A, we define the *histogram of* $\mathfrak{A}$,denoted $hist(\mathfrak{A})$, to be the N-tuple $(w_1, w_2, \ldots w_N)$ where for all i,

$$w_i = |\{a \in A : \mathfrak{A} \models \pi_i[a]\}|.$$

It is easy to see that any sentence s in the language $\mathcal{C}$ can be naturally translated into a linear inequality in variables $w_1, \ldots, w_n$ satisfied by the histogram of a structure $\mathfrak{A}$ just in case s is true in $\mathfrak{A}$. For example, the sentence "At least K P_a are not P_b" can be formulated as the inequality

$$\sum_{i=1}^{N} \{w_i : \models \pi_i \rightarrow (P_a(x) \wedge \neg P_b(x))\} \geq K, \tag{6.1}$$

stating that the frequencies of those 1-types entailing both $P_a(x)$ and $\neg P_b(x)$ sum to at least K. We add the inequality $\sum_{i=1}^{N} w_i \geq 1$ to rule out the zero solutions (corresponding to the standard assumption that the domain of quantification is non-empty). In this way, given a set of sentences, $S = \{s_1, \ldots, s_m\}$, we can construct a system of linear inequalities $\mathcal{E} = \{E_1, \ldots, E_m, \left(\sum_{i=1}^{N} w_i \geq 1\right)\}$ such that S is satisfiable if and only if $\mathcal{E}$ has a solution over the natural numbers. Since this latter problem has a well-known algorithmic solution, so has the former.

6.2.3 Phase-change Region and Data Construction

When considering the algorithmic solution of problems in logic, it is important to realize that not all instances are equally difficult. A case in point is provided by the well-known problem SAT: given a set Γ of clauses in propositional logic, determine whether Γ has a satisfying truth-value assignment. (In this context, *clause* is a disjunction of proposition letters or negated proposition letters.) Consider a problem instance with m clauses featuring a signature of size n (number of variables). If the clause variable ratio $\alpha = \frac{m}{n}$ is large, we have a highly constrained problem instance with few degrees of freedom; hence, the probability of satisfiability is close to zero. Conversely, if α is small, we have a relatively unconstrained problem with many degrees of freedom; hence, the probability of satisfiability is close to unity. In either case, it is easy for a learning

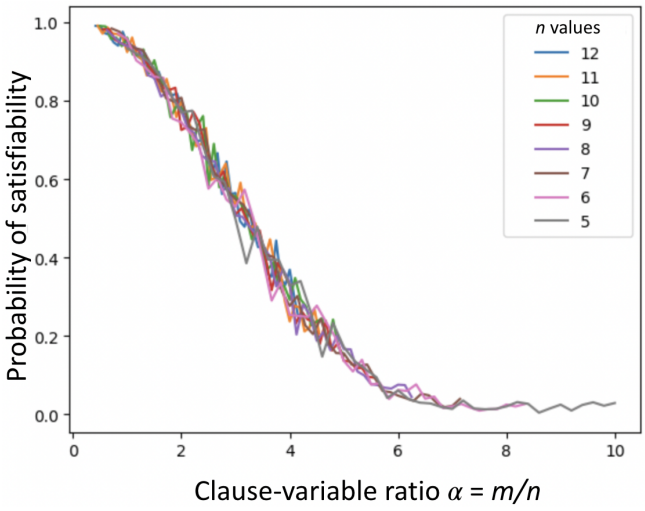

Figure 6.2: The variation of probability of satisfiability with clause variable ratio for the counting fragment $\mathcal{C}$

algorithm to determine satisfiability reliably. Only for values in a narrow range of α commonly referred to as the *phase-change region*, is the problem challenging [118, 86]. A similar phenomenon can be observed for the counting fragment we considered in this study. Figure 6.2 depicts the variation of the probability of satisfiability with the clause-variable ratio.

In our data construction framework, we fix a large vocabulary $\mathcal{V}$ of English common count nouns. When constructing a data point, we randomly select a signature consisting of n elements of $\mathcal{V}$, and construct m sentences of the language $\mathcal{C}$ over this signature. We systematically vary n within the interval $[5,12]$ ($n_{min} = 5, n_{max} = 12$). Through this variation, we investigate the relationship between the probability of satisfiability and the clause variable ratio $\alpha = \frac{m}{n}$, aiming to establish an appropriate range for the parameter α. Subsequently, guided by our exploratory analysis, we select specific α values to ensure that the probability of satisfiability falls within the predefined interval $[0.4, 0.6]$ ($\alpha_{min} = 0.4, \alpha_{max} = 0.6$). We design our algorithm to generate numerical satisfiability problems with appropriate ranges for α, n and K. We vary K between $K_{min} = 10$ and $K_{max} = 50$, and utilise a set of professions (eg: doctors, artists, ...) as the vocabulary $\mathcal{V}$. We translate the set of generated sentences into a system of linear inequalities as explained above, and use the ILP solver in the `Z3` to determine whether this system has a solution. Following this setup, we construct a training set of $130K$ problem instances and a test set with $12K$ instances. A more detailed explanation of the dataset is provided in Appendix 6.8.1.

Algorithm 4 Data Construction - Natural language satisfiability

Input : Vocabulary of nouns $\mathcal{V}$, number of variables $[n_{min}, n_{max}]$, numerical value range $[K_{min}, K_{max}]$, clause variable ratio α range $[\alpha_{min}, \alpha_{max}]$
Output : natural language satisfiability dataset $\mathcal{D}$

$D \leftarrow \{\}$
repeat
  $n \leftarrow$ randomly sample from $[n_{min}, n_{max}]$
  $v \leftarrow$ randomly sample n nouns from $\mathcal{V}$
  $m \leftarrow$ sample m s.t $\alpha_{min} \leq \frac{m}{n} \leq \alpha_{max}$
  for $j = 1$ to m **do**
    $A, B \leftarrow$ randomly sample two nouns from v
    $t_j \leftarrow$ randomly sample template from the counting fragment $\mathcal{C}$
    $K \leftarrow$ randomly sample from the range $[K_{min}, K_{max}]$
    $s_j \leftarrow$ substitute A, B, K for schematic variables in t_j
    $E_j \leftarrow$ translate s_j to a linear inequality
  end for
  $z \leftarrow \texttt{Solver}(E_1, \ldots, E_m, (\sum_{i=1}^{N} w_i \geq 1))$
  if z is not *None* **then**
    $\ell \leftarrow True$
  else
    $\ell \leftarrow False$
  end if
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\{s_1, ..., s_m\}, \ell\}$
until *stop condition is met*

Data construction: Out-of-distribution

To evaluate transformers' ability for generalisation to out-of-distribution data, we construct several out-of-distribution datasets following two approaches. In the first approach, we introduce perturbations to the in-distribution test set. We employ this approach to construct data to test for vocabulary invariance and noise invariance. In the second approach, we construct out-of-distribution data by introducing different input configurations. We employ this approach to construct data to test for numerical invariance and scale invariance.

Vocabulary invariance: We introduce perturbations to the test set by incorporating OOV nouns. These perturbations are categorised into two distinct types. First, nouns within the sentences of the test set are substituted with semantically similar nouns that were not encountered during the fine-tuning. Second, nouns within the sentences are substituted with symbols adhering to a prescribed format denoted as *P_J*, where

$J \in \mathbb{N}^+$ (eg: $P\_1, P\_2, \ldots$). The perturbed test sets resulting from the first and second approaches are designated as $Vocab_{prof}$ and $Vocab_{Ps}$, respectively.

Numerical invariance: We construct three separate test sets employing Algorithm 1, distinguished by the range within which K varies. We employ the ranges $[100, 500]$, $[1000, 5000]$, and $[100, 5000]$, and we denote the resulting datasets by $Num_{[100,500]}$, $Num_{[1000,5000]}$ and $Num_{[100,5000]}$, respectively.

Scale invariance: To explore the adaptability of transformers in tackling problems of larger scope, we devise a distinct test set utilising Algorithm 1. We define the number of variables n to span from 13 to 16 ($n_{min} = 13, n_{max} = 16$), thereby ensuring that the problem instances presented to the models are more intricate than those encountered during the fine-tuning process. Henceforth, the generated dataset is referred to as *Scale*.

Noise invariance: To evaluate transformers' susceptibility to noise, we introduce perturbations to problem instances within the test dataset. These perturbations involve the introduction of sentences that do not alter the satisfiability of the instances. Specifically, given a problem instance q_1 employing a signature v_1, we generate an alternate problem instance q_2 that is satisfiable employing a mutually exclusive signature v_2. Consequently, the inclusion of sentences from q_2 into q_1 does not influence the satisfiability of q_1. Henceforth, we refer to this perturbed dataset as *Noise*. Furthermore, we partition the *Noise* dataset into two distinct subsets based on the number of clauses present. If the number of clauses m in a given instance exceeds the maximum number of clauses present in the training set, we categorise that dataset as $Noise_{>m}$, otherwise, $Noise_{\leq m}$. We outline the out-of-distribution datasets in more detail in Appendix 6.8.2.

6.3 Experimental Setup

6.3.1 Fine-tuning

To examine transformers' ability to solve numerical satisfiability problem instances, we fine-tune two well-known transformers that have a proven track record in textual reasoning tasks: Flan-T5 [25] and Gemma [135].

Flan-T5 Flan-T5 is an instruction-fine-tuned variant of the T5 model architecture [147] and is considered to be an improvement to the vanilla T5 model. T5-based

model architectures have a well-documented track record of solving formal reasoning problems, including satisfiability [80, 111, 26]. We utilise the `Flan-T5-base` model with 220M parameters, `Flan-T5-large` with 770M parameters and `Flan-T5-XL` model with 3B parameters.

Gemma Gemma is an open-source transformer model architected upon the foundation of the Gemini models [134]. Gemma has achieved state-of-the-art performance on various language-related tasks when compared to models of similar scale and even some larger models. For this investigation, we employ the 2B parameter version, referred to as `Gemma-2b`.

6.3.2 Zero-shot and Few-shot settings

In addition to fine-tuned models, we evaluate a range of closed and open-source Large Language Models (LLMs), including GPT-3.5-turbo [66], GPT-4 [94], and Mistral-7B [59] on a subset of 300 test examples using different prompting techniques. Recent work has suggested that pre-trained LLMs might exhibit emergent reasoning capabilities when the number of parameters scales above a certain threshold [148]. However, subsequent evidence has started questioning such claims [115], resulting in an open debate within the research community. Here, we aim to contribute to this debate by testing whether LLMs can generalise to the numerical satisfiability task. The prompts used for the experiments are shown in Appendix 6.8.3.

Zero-shot Inference In the zero-shot setting, we simply prompt LLMs with instructions about the task, asking the model to generate a "True" or "False" answer according to whether the set of statements provided as input is satisfiable or not.

In-Context Learning In addition to the zero-shot setting, we test the ability of LLMs to solve numerical satisfiability problems when provided with in-context examples [16]. To this end, given a test example, we employ a BM25 retrieval model [113] to select the top k most similar examples and their corresponding labels from the training set.

Chain-of-Thought Finally, we test LLMs via Chain-of-Thought (CoT) prompting [149], where the models are explicitly queried to generate a step-by-step explanation to derive the final answer. Here, we limit our experiments to GPT 3.5 and Mistral because of budget constraints, since the cost of GPT 4 directly depends on the number

Model	Accuracy	F1 score
Fine-tuning		
`Flan-T5-XL`	**89.41**	**89.72**
`Gemma-2b`	84.39	84.66
Zero-shot		
`Mistral-7b`	49.66	**65.60**
`GPT-3.5-turbo`	49.00	53.78
`GPT-4`	**53.60**	64.42
In-Context Learning (ICL)		
`Mistral-7b` (k = 5)	54.33	50.54
`Mistral-7b` (k = 20)	57.00	52.04
`GPT-3.5-turbo` (k = 5)	51.67	49.65
`GPT-3.5-turbo` (k = 20)	49.33	47.59
`GPT-4` (k = 5)	49.50	47.58
`GPT-4` (k = 20)	**57.14**	**58.25**
Chain-of-Thought (CoT)		
`Mistral-7b`	**48.49**	**66.17**
`GPT-3.5-turbo`	46.33	58.82

Table 6.1: Results on the in-distribution test set. Although transformers can be fine-tuned to solve numerical satisfiability instances, pre-trained LLMs struggle in zero-shot/few-shot settings.

of generated tokens. Moreover, we observed no significant improvements in GPT 3.5 and Mistral when using CoT as a prompting method.

6.4 Results and Discussion

6.4.1 In-distribution Evaluation

Although transformers can be fine-tuned to solve numerical satisfiability problem instances, these models struggle in zero-shot/few-shot settings . As shown in Table 6.1, fine-tuned transformers achieve adequate performance when solving in-distribution numerical satisfiability problems. However, transformers encounter notable challenges in zero-shot and few-shot scenarios. Indeed, the accuracy across all transformer model architectures in zero-shot and few-shot contexts closely approximates those of a random baseline. Notably, certain models, such as `Mistral-7b`, consistently

Model	In-distribution		Vocabulary Invariance			
			$Vocab_{prof}$		$Vocab_{Ps}$	
	Acc	*F1*	*Acc*	*F1*	*Acc*	*F1*
`Flan-T5-XL`	89.41	89.72	86.48	86.83	73.85	67.85
`Gemma-2b`	84.39	84.66	79.60	79.41	50.61	2.85

Table 6.2: Results on vocabulary invariance and numerical invariance datasets. We found that transformers exhibit limited generalisation to vocabulary

Model	In-distribution		Numeracy Invariance					
			$Num_{[100-500]}$		$Num_{[1000-5000]}$		$Num_{[100-5000]}$	
	Acc	*F1*	*Acc*	*F1*	*Acc*	*F1*	*Acc*	*F1*
`Flan-T5-XL`	89.41	89.72	85.27	85.75	83.46	83.82	78.55	79.84
`Gemma-2b`	84.39	84.66	79.13	79.95	76.23	77.34	68.05	70.18

Table 6.3: Results on vocabulary invariance and numerical invariance datasets. We found that transformers exhibit limited generalisation to numerical invariance.

exhibit an inclination to produce "True" (satisfiable) for almost all problem instances. Moreover, our analysis reveals the susceptibility of transformers to the influence of examples provided during in-context learning and chain-of-thought prompting, often leading to erroneous conclusions. We posit two potential explanations for the observed difficulties of transformers in zero-shot and few-shot settings. First, the numerical satisfiability problem is an intricate reasoning problem. Indeed, numerous cognitive studies underscore the inherent difficulties humans encounter when tasked with even rudimentary formal reasoning exercises [60, 15]. Given that transformers are trained on corpora derived from human-generated content, it is unsurprising that these models inherit this limitation. Second, prior research work has hypothesised that transformers solve multi-step reasoning problems by linearised path matching [32]. Considering the intricate nature of the numerical satisfiability problem, it is not a robust approach.

6.4.2 Out-of-Distribution Generalisation

Transformers' insensitivity to vocabulary invariance is bounded by semantic similarity. As shown in Table 6.2, transformers demonstrate proficiency in generalising to OOV instances when the OOV nouns exhibit semantic similarity to those present within the in-distribution data. Conversely, transformers encounter difficulties when

confronted with OOV nouns lacking semantic resemblance to their in-distribution counterparts. Specifically, the `Gemma-2b` model exhibits minimal robustness in scenarios where the vocabulary comprises nouns lacking semantic similarity, predicting "False" (unsatisfiable) across almost all problem instances. Although `Flan-T5-XL` demonstrates comparatively greater resilience than `Gemma-2b`, its performance nevertheless registers a notable decline under similar conditions. We hypothesise this is due to transformers' inability to separate logical semantics from non-logical semantics. Another contributing factor to this decline in performance could be the relatively lower term frequency associated with terms of type *P_J* compared to conventional nouns describing professions. Prior research has underscored the impact of term frequency on model performance [109]. However, given the substantial magnitude of the observed drop in performance, we contend that the more plausible influence stems from the lack of semantic similarity rather than from lower term frequency.

Transformers exhibit limited numerical invariance and rely on superficial cues for learning. As depicted in Table 6.3, transformers generalise well when the range of K is $[100-500]$ or $[1000-5000]$, but encounter difficulties when the range of K is $[100-5000]$. Although the performance of both models is well above the random baseline for K with the range $[100-5000]$, they experience a drop of more than 10 accuracy points. The numerical quantifiers impose a comparative operation between the logical and numerical elements involved in the expressions. Consequently, we hypothesise that transformers prioritise the first digit over the holistic consideration of the entire numerical value. This observed phenomenon underscores that even the more recent billion-parameter variants of transformers are susceptible to learning superficial cues, reminiscent of their earlier, smaller counterparts [81, 43]. Moreover, this limited generalisation ability to numerical invariance coupled with their failure to adapt to OOV that lacks semantic similarity indicates that these models still struggle to learn logical semantics. This stands in stark contrast to the demonstrated proficiency of transformers in grasping the logical semantics associated with simpler reasoning problems, such as model-checking [78, 80]. Consequently, we posit that the transformers' capacity to comprehend logical semantics is intricately tied to the complexity inherent within the reasoning tasks at hand.

Transformers fail to exhibit scale invariance As shown in Table 6.4, transformers exhibit a failure to generalise effectively when confronted with problem instances

Model	**In-distribution**		**Scale Invariance** *Scale*	
	Acc	*F1*	*Acc*	*F1*
`Flan-T5-XL`	89.41	89.72	78.38	78.14
`Gemma-2b`	84.39	84.66	73.44	72.95

Table 6.4: Results on scale invariance and noise invariance datasets. We found that transformers exhibit minimal generalisation to scale and noise invariance.

Model	**In-distribution**		**Noise invariance** *Noise*		*Noise*$_{\leq m}$		*Noise*$_{>m}$	
	Acc	*F1*	*Acc*	*F1*	*Acc*	*F1*	*Acc*	*F1*
`Flan-T5-XL`	89.41	89.72	72.08	65.15	73.41	68.70	70.77	61.24
`Gemma-2b`	84.39	84.66	68.64	61.98	69.15	64.58	68.15	59.15

Table 6.5: Results on scale invariance and noise invariance datasets. We found that transformers exhibit minimal generalisation to scale and noise invariance.

characterised by a higher number of variables than those encountered during training. This outcome aligns with prior investigations into scale invariance, which have yielded analogous findings [116, 111]. We posit that this deficiency in scale invariance arises from the transformers' inability to learn rules of inference gleaned from natural language texts. As previously noted, prior research suggests that transformers attempt to solve multi-step reasoning tasks through linearised path-matching strategies [32]. Additionally, during training and fine-tuning phases, transformers are known to acquire shortcuts via pattern-matching [75]. While this approach may prove expedient for in-distribution evaluation, it does not result in robust generalisation when subjected to out-of-domain testing. Indeed, we posit that this behaviour impedes the transformers' capacity to learn rules of inference, thereby constraining their ability to demonstrate scale invariance.

Transformers demonstrate sensitivity to noise As shown in Table 6.5, the introduction of noisy sentences into problem instances precipitates a notable decline in the performance of transformers. When the inclusion of noisy sentences extends the overall problem length beyond the scope of the model's training data, the scenario resembles

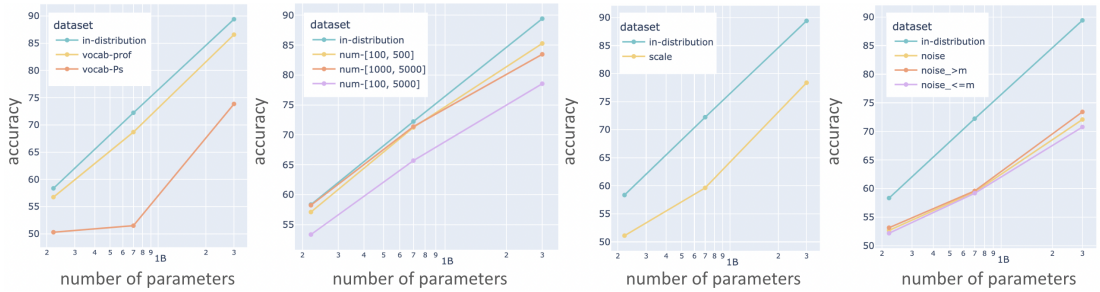

Figure 6.3: Variation of accuracy level with the number of parameters for the Flan-T5 model. We employed `Flan-T5-base`, `Flan-T5-large` and `Flan-T5-XL` models and found they exhibit similar generalisation patterns.

a variant of the length generalisation problem. Previous research has indicated transformers' propensity to struggle with length generalisation [5, 105], thus explaining the observed decline for the $Noise_{>m}$ test set. However, even in scenarios where the problem length remains within the bounds of the model's training data, the performance of transformers still undergoes a significant decline. Notably, there exists a negative correlation between the performance of transformers and the number of noisy sentences introduced into the problem instances. We posit that this phenomenon arises from a fundamental shift in the underlying problem structure induced by the introduction of noisy sentences. For instance, consider the scenario where sentences from problem q_2, constructed with a mutually exclusive vocabulary to problem q_1 and determined to be satisfiable, are incorporated into problem q_1. The resultant problem q comprises two sub-problems for which the models have to derive the satisfiability. We posit this change in problem structure influences transformers' inability to exhibit noise invariance.

Models of different scales exhibit analogous generalisation patterns As depicted in Figure 6.3, transformers of varying sizes exhibit similar patterns of generalisation. For instance, the `Flan-T5-base` model achieves an accuracy of 58.35%, while the `Flan-T5-large` model attains 72.23%, both notably lower than the accuracy achieved by the `Flan-T5-XL` model, which stands at 89.41%. However, their relative disparities in generalisation performance to out-of-distribution datasets mirror the pattern observed with `Flan-T5-XL`. Previous investigations exploring the impact of model size on generalisation have yielded congruent findings [5]. It is noteworthy that despite experiencing a marked decline in performance for out-of-distribution datasets, the generalisation proficiency of `Flan-T5-XL` remains significantly above the random baseline. Nonetheless, we assert that transformer models still have significant strides to make before achieving

the capability to learn rules of inference and comprehend logical semantics.

6.5 Related Work

Extensive research has been dedicated to exploring the generalisation capabilities of neural models, including transformers. These investigations encompass a spectrum of generalisation forms, such as length generalisation [105, 5, 140], easy-to-hard generalisation [117, 9, 84], and compositional task generalisation [32]. However, there remains a notable gap in the literature concerning the application of these models to textual inference problems, particularly within the realm of natural language satisfiability. Consequently, our study aims to delve into the various facets of generalisation exhibited by transformer models when confronted with a variant of the natural language satisfiability problem, known as the numerical satisfiability problem.

Transformers have demonstrated impressive performance on various formal reasoning tasks [110, 73, 131, 28]. All of these reasoning tasks can be reduced to the problem of determining the *satisfiability* of a set of sentences. Neural approaches such as graph neural networks have been used extensively to solve instances of satisfiability problems [158, 19, 119]. Richardson and Sabharwal (2021)[111] extended this research into natural language by employing language fragments for data construction. Their investigation also diverts from prior research as they employ transformers rather than graph neural approaches. Schlegel et al. (2022)[116] and Madusanka et al. (2024) [79] extended this work into fragments of first-order logic. Our research extends upon these foundations by investigating the transformer models' capacity to tackle numerical satisfiability problems. We further differentiate from prior work on satisfiability by explicitly focusing on the generalisation aspects of transformer models.

6.6 Conclusion

We probe the generalisation ability of transformers on the satisfiability problem for a simple fragment of English featuring numerical quantification. We find that fine-tuned transformers exhibit limited generalisation to vocabulary and numerical invariance while exhibiting minimal scale and noise invariance. Furthermore, our investigation indicates that transformers in zero-shot and few-shot settings find numerical satisfiability problems more or less unsolvable. However, we emphasise that certain model architectures, such as `Flan-T5-XL`, demonstrate robustness when assessed against out-of-distribution

(OOD) test sets. Despite experiencing a decline in performance, `Flan-T5-XL` consistently surpasses the random baseline by a considerable margin. Given the pervasive adoption of transformer models, it is imperative to comprehensively grasp the robustness of these models. We assert that substantial research efforts are warranted to improve transformer models' ability to generalise to OOD data over a complex reasoning task, which we leave for future work.

6.7 Limitations

Due to the empirical nature of our work, it suffers from inductive dilemmas on three fronts. First, while we explore several transformer models with established proficiency in formal reasoning tasks, it remains conceivable that there exist other transformer architectures whose behaviour diverges from the empirical findings delineated in this study. Second, in our study, we construct the numerical satisfiability problems by employing language fragments involving numerical quantifiers "At least K" and "At most K". However, it is important to acknowledge that these quantifiers represent only a subset of the broader spectrum of numerical quantifiers in existence. Thus, there may exist alternative quantifiers for which transformer models exhibit different behavioural patterns. Third, we consider in-context learning and chain-of-thought prompting defined through prompts presented in Appendix 6.8.3. However, it is plausible that alternative prompting methodologies may yield superior performance for transformer models.

6.8 Supplementary materials

6.8.1 Dataset Details

Formally the dataset for each objective takes the form, $\{((S)^{(d)}, \ell^{(d)})\}_d^{|D|}$, where ($S$ are set of sentences concatenated together, and $\ell \in \{\text{True, False}\}$ is the label. We have depicted an instance of a numerical satisfiability problem along with its perturbations (vocabulary and noise) in Figure 6.4

In-distribution datasets

In our experimental setup, we generated a training dataset comprising 130K instances and an in-distribution testing dataset consisting of 12K instances employing Algorithm 4. Across these datasets, we varied the number of variables from 5 to 12, while adjusting

	Vocabulary invariance	Noise Invariance
"**sentences**": {"At least 36 non-ballerinas are fishermen. At least 46 fishermen are non-musicians. At least 49 non-musicians are artists. At least 14 non-warlords are artists. At most 27 non-fishermen are non-warlords. At most 20 non-artists are musicians. At least 23 non-musicians are artists. At most 30 non-musicians are fishermen. At most 49 fishermen are ballerinas. At most 23 non-fishermen are warlords. At most 10 ballerinas are fishermen. At most 34 non-fishermen are warlords. At most 22 fishermen are ballerinas." "**label**": *False*}	{"**sentences**": "At least 36 non-P_1s are P_2s. At least 46 P_2s are non-P_3s. At least 49 non-P_3s are P_5s. At least 14 non-P_4s are P_5s. At most 27 non-P_2s are non-P_4s. At most 20 non-P_5s are P_3s. At least 23 non-P_3s are P_5s. At most 30 non-P_3s are P_2s. At most 49 P_2s are P_1s. At most 23 non-P_2s are P_4s. At most 10 P_1s are P_2s. At most 34 non-P_2s are P_4s. At most 22 P_2s are P_1s.", "**label**": *False*}	"**sentences**": {"At least 36 non-ballerinas are fishermen. At least 46 fishermen are non-musicians. At least 49 non-musicians are artists. At least 14 non-warlords are artists. At most 27 non-fishermen are non-warlords. At least 20 doctors are engineers. At most 20 non-artists are musicians. At least 23 non-musicians are artists. At most 30 non-musicians are fishermen. At least 20 non-doctors are beekeepers. At most 49 fishermen are ballerinas. At most 23 non-fishermen are warlords. At most 50 bee-keepers are not engineers. At most 10 ballerinas are fishermen. At most 34 non-fishermen are warlords. At most 22 fishermen are ballerinas." "**label**": *False*}

Figure 6.4: Instance of numerical satisfiability problem. We modify the vocabulary to form the vocabulary variance dataset and add noise to form the noise invariance dataset, as shown.

the alpha ratio within the range of 2.54 to 3.71. The alpha ratio was selected based on an exploration of how the probability of satisfiability correlates with the clause-variable ratio. As detailed in the Methodology section, we chose the alpha value such that the probability of satisfiability falls within the range of 0.4 to 0.6, ensuring that the resulting number of clauses falls between 13 and 45. Additionally, we varied the range of K, from 10 to 50. Furthermore, we employed a list of nouns comprised of professions as our common noun vocabulary. The vocabulary contains 155 nouns. We emphasise that the problem space is sufficiently large that no two problems would be equal.

Out-of-distribution datasets

We constructed several out-of-distribution datasets, and the size of each of them is detailed in Table 6.6. The table 6.7 depicts mean, minimum, and maximum problem instance lengths. The size of the scale test set is low due to computational time constraints involved in the data construction setup. Indeed, an n-variable numerical satisfiability problem translated to a 2^n variable integer linear problem that the integer linear solver needs to solve.

		Size
Vocabulary invariance	$Vocab_{prof}$	12000
	$Vocab_{Ps}$	12000
Numerical invariance	$Num_{[100-500]}$	2000
	$Num_{[1000-5000]}$	2000
	$Num_{[100-5000]}$	2000
Scale invariance	*Scale*	800
Nosie invariance	*Noise*	12000
	$Noise_{\leq m}$	5921
	$Noise_{>m}$	6079

Table 6.6: The number of problem instances in each of out-of-distribution test sets.

6.8.2 Fine-tuning Details

We chose two transformer models of different architectures of fine-tuning[1]: Flan-T5 and Gemma. Both models boast a commendable track record in textual entailment tasks and have demonstrated state-of-the-art performance compared to other models of comparable size. The Flan-T5 model is based on the T5 architecture and is an encoder-decoder model, while Gemma is a decoder-only model [141]. Moreover, unlike Flan-T5, Gemma leverages reinforcement learning from human feedback for instruction fine-tuning [95]. As the primary focus of our study is to determine the extent to which transformer models demonstrate generalisation capabilities, we did not perform extensive hyperparameter tuning on in-distribution data. However, we did explore several variations of hyperparameters, including batch size and learning rate, and found that the resultant performance remained largely equivalent across these variations. We establish the following hyper-parameter setup when fine-tuning.

Maximum sequence length: We used the maximum sequence 1024 for both Flan-T5 and Gemma models. We did not rely on any truncation, as truncating input could alter the satisfiability of the input sentences.

Training epochs: For both `Gemma-2b` and `Flan-T5-XL` models, we fine-tuned for two epochs. The performance of the models stagnates after the very first epoch. For `Flan-T5-base` and `Flan-T5-large` models, we fine-tuned for 5 epochs.

[1]We also fine-tune phi-2 model with 2.7B parameters. However, the model did not optimise properly, therefore, we excluded it from the experimental setup

dataset	min	max	mean
$Vocab_{prof}$	270	70	147.1
$Vocab_{Ps}$	226	66	123.6
$Num_{[100-500]}$	270	78	161.2
$Num_{[1000-5000]}$	264	78	159.9
$Num_{[100-5000]}$	270	78	158.2
Scale	360	198	272.9
Noise	474	150	276.3
$Noise_{\leq m}$	270	150	233.6
$Noise_{>m}$	474	276	317.5

Table 6.7: The minimum, maximum and mean number of words (tokens) when separated by SPACE in each of the test sets

Gradient accumulation steps: 4

Batch size: Relying on the gradient checkpointing and gradient accumulation, we used 12 as the batch size for `Gemma-2b` and `Flan-T5-XL` models, while using 48 as the batch size for `Flan-T5-base` and `Flan-T5-large`.

Learning Rate: We set the learning rate to 1×10^{-5}. We use the ADAM optimiser with the default parameters $\varepsilon = 1 \times 10^{-8}$, $\beta_1 = 0.99$ and $\beta_2 = 0.999$.

Hardware: 1 Nvidia A100 GPU with 80GB of RAM.

Each transformer is fine-tuned to predict the label (True/False: True if the set of sentences is satisfiable and False if the set of sentences is unsatisfiable) by reducing the binary cross entropy loss over the target using the Adam optimiser [65], and we used the HuggingFace implementation in the fine-tuning process [154].

6.8.3 Prompts

Regarding the pre-trained models, we adopt the following prompt for zero-shot and in-context learning:

- *"You are an expert in logical satisfiability and numerical reasoning. Determine whether the set of statements in the test example is satisfiable. Your answer must be 'The answer is False' if the test example is unsatisfiable, 'The answer is True' if the test example is satisfiable."*

For in-context learning, the prompt above is followed by a list of training examples with their correct labels and the set of statements constituting the test example.

To elicit step-by-step reasoning via chain-of-thought, we adopt the following prompt:

- *"You are an expert in logical satisfiability and numerical reasoning. Determine whether the set of statements in the test example is satisfiable. Think step-by-step and terminate your reasoning with 'The answer is False' if the test example is unsatisfiable, 'The answer is True' otherwise."*

6.8.4 Reducing other Quantifiers

The mechanism for reducing the numerical satisfiability problem to quantifiers included in Fragment $\mathcal{C}$ can be easily adapted to other generalised quantifiers with minor changes to inequality (6.1). Let K, w_i, π_i, $\pm P_a(x)$ and $\pm P_b(x)$ and have the same definition as mentioned in the Methodology section. Let $\mathcal{W} = (w_1, \ldots w_N)$ and,

$$\lambda(\mathcal{W}) := \sum_{i=1}^{N} \{w_i : \models \pi_i \rightarrow (\pm P_a(x) \wedge \pm P_b(x))\}$$

Then other quantifiers can be formulated as below,

More than K: (More than K (non-)P_as are (not) P_bs),

$$\lambda(\mathcal{W}) > K, \tag{6.2}$$

Less than K: (Less than K (non-)P_is are (not) P_js),

$$\lambda(\mathcal{W}) < K, \tag{6.3}$$

K: (K (non-)P_is are (not) P_js),

$$\lambda(\mathcal{W}) = K, \tag{6.4}$$

Most: (Most (non-)P_as are (not) P_bs),

$$\lambda(\mathcal{W}) > \sum_{i=1}^{N} \{w_i : \models \pi_i \rightarrow (\pm P_a(x) \wedge \mp P_b(x))\} \tag{6.5}$$

Few: (Few (non-)P_as are (not) P_bs),

$$\lambda(\mathcal{W}) < \sum_{i=1}^{N} \{w_i : \models \pi_i \rightarrow (\pm P_a(x) \wedge \mp P_b(x))\} \tag{6.6}$$

At least r/s (At least $\frac{r}{s}$ (non-)P_as are (not) P_bs),

$$\lambda(\mathcal{W}) \geq \frac{r}{s} \sum_{i=1}^{N} \{w_i : \models \pi_i \rightarrow \pm P_a(x)\} \tag{6.7}$$

Since the primary focus of this study is to investigate the behaviour of transformers when solving numerical satisfiability problems rather than identifying how their performance varies with different quantifiers, we only focus on the quantifiers mentioned in Fragment $\mathcal{C}$. Furthermore, we emphasise that the problem best suited to analyse the effect of different generalised quantifiers is model checking rather than satisfiability.

The mechanism can further be adapted to include relative clauses with minor changes. Consider the sentence "At least K P_as who are non-P_bs are P_cs", it can be formulated as shown below,

$$\sum_{i=1}^{N} \{w_i : \models \pi_i \rightarrow (P_a(x) \wedge \pm P_b(x) \wedge P_c(x))\} \geq K \tag{6.8}$$

We exclude this type of complex structure, considering the low performance exhibited by most transformer models when confronted with simple numerical satisfiability problems.

Chapter 7

Conclusion and Future Work: Blue Skies or Stormy Weather

"The ironic tragedy of life is that it has to be lived forward,
but only make sense in reverse"
– Soren Kierkegaard

7.1 Concluding Remarks

In this thesis, we ask the overarching question:

> ***Can transformer models learn logical semantics in natural language and rules of inference?***

As outlined in Chapters 1 and 2, we identify two problems in logic that are well-suited to study this question: the problem of model-checking with natural language and the problem of natural language satisfiability.

In Chapters 3 and 4, we employ the first problem—model-checking with natural language—to ascertain the extent to which transformer models learn the logical semantics of natural language. In particular, Chapter 3 presents a preliminary study wherein the problem of model-checking is employed to understand these models' ability to grasp grammatical constructs with logical significance. Chapter 4 extends this study into a logical problem with significant interest at the intersection of logic and language, namely, generalised quantifiers.

Our study provides empirical evidence of the effect of linguistic complexity on the ability of transformer models to solve an elementary reasoning problem. Notably, such factors as quantifiers (both in type and number), binary predicates and anaphora exert a considerable effect on these models' efficacy in solving model-checking problems. It is of particular interest that grammatical constructs such as negation, while having no significant effect on fine-tuned models, appear to exert a notable effect on pre-trained models in zero-shot settings. Furthermore, transformer models exhibit behavioural patterns akin to those observed in human reasoning, including difficulties with comparative quantifiers and sensitivity to the number of quantifiers present in a sentence. Based on three key findings, we posit that these models can learn the essence of logical semantics of natural language if the underlying problem is simple enough. First, learning the simple fragments enables transformers to learn complex ones. Second, transformer models exhibit partial scale invariance. Third, these models exhibit numerical invariance, as their performance remains relatively stable when tested on out-of-distribution numerical values in sentences containing numerical quantifiers.

In Chapters 5 and 6, we direct our inquiry toward the second of the identified problems—natural language satisfiability—to ascertain the extent to which transformer models learn rules of inference. In particular, Chapter 5 is devoted to an investigation of transformer models' ability to resolve instances of the natural language satisfiability problem drawn from various fragments of first-order logic. By varying the expressive power of these fragments, we further assess the impact of computational complexity on model performance. Chapter 6 extends this examination to the problem of numerical satisfiability, with a particular emphasis on diverse forms of generalisation. In pursuit of a rigorous and faithful evaluation, we draw problem instances from the phase-change region, thereby mitigating various pitfalls associated with random sampling of complex reasoning problems. Accordingly, we undertake an exploratory analysis to ascertain the precise boundaries of the phase-change region for the language fragments considered in this study.

Our experimental findings suggest that the language fragment to which a given problem belongs exerts a significant effect on transformer models. Specifically, problems belonging to fragments that are—from a computation complexity standpoint—more complex pose a correspondingly greater challenge to transformer models. Moreover, transformer models exhibit limited generalisation in both fine-tuned and zero-shot settings. In zero-shot settings, these models struggle to solve even the simplest form of

satisfiability problem we consider in this study. In fine-tuned settings, transformer models exhibit limited vocabulary and numerical invariance. In particular, their performance is adversely affected by the semantic dissimilarity between the vocabulary of the test set and that of the train set. Consequently, we posit that transformer models encounter substantial difficulty in acquiring logical semantics when the reasoning problem in question is of a more intricate nature. Moreover, transformer models exhibit little to no scale and noise invariance. Of particular note is the substantial decline in performance observed across all fragments, irrespective of their computational complexity, when subjected to evaluation for scale invariance. Consequently, we posit that these models lack the ability to learn the rules of inference required when solving satisfiability problems.

In our study to answer the aforementioned overarching question, we arrive at the following principal conclusion: While transformer models can capture the essence of logical semantics in natural language when reasoning problems remain simple, they struggle with complex reasoning tasks. Furthermore, we find that transformer models are fundamentally deficient in their ability to learn the rules of inference.

7.2 Limitations

Our primary objective is to study the extent to which transformer models understand logical semantics and rules of inference from language text. We design our experiments with this particular objective in mind. However, due to computational and time constraints, we impose specific limits during our experimentation setup, which may introduce some induction bias. For one, we limit our experimental setup to specific language fragments, a specific set of generalised quantifiers, and a well-known set of transformer models. We acknowledge that there could be other variations of this experimental setup that could lead to somewhat different insights than what we identified. Moreover, we did not experiment with paraphrasing and surface changes. Although we agree that it would have improved the experimental setup, we did not find that it was necessary to answer the overarching question raised above. Primarily, large-scale transformer models—such as the ones we used in zero-shot/few-shot settings —exhibit robustness to such surface-level perturbations. Therefore we posit that such changes would not affect the insights derived through zero-shot/few-shot experiments. In fine-tuned settings, we should avoid such perturbations to avoid ambiguity.

Furthermore, due to explicitly focusing on the logical reasoning aspect of transformer models, we ignored non-logical variations. For example, many non-formal

reasoning problems are formalised as a three-way classification problem (entail/neutral/-contradict). Moreover, we acknowledge that this form of formal reasoning is somewhat difficult for human beings to master compared to non-formal counterparts. As we mentioned in Chapter 2, to faithfully investigate transformer models' ability to understand logical semantics and learn rules of inference, we eliminate ambiguity and the influence of background knowledge and common sense. Thus, we constrain our problem space to the ones we introduced in earlier chapters.

Moreover, the present study on generalised quantifiers was primarily oriented toward their mathematical and logical interpretations without explicitly examining certain linguistic properties that have been shown to play a significant role in natural language understanding—namely, monotonicity and conservativity. Monotonicity refers to the directionality of entailment licensed by a quantificational determiner. A quantifier is said to be upward monotone (or monotone increasing) if the substitution of the restrictor set with a superset preserves truth. Conversely, a quantifier is a "downward monotone" (or monotone decreasing) if the substitution with a subset preserves truth. For example, the quantifier "some" exhibits upward monotonicity: Some dogs bark $\rightarrow$ Some animals bark In contrast, the quantifier no demonstrates downward monotonicity: No animals bark $\rightarrow$ No dogs bark. Psycholinguistic research suggests that language comprehenders are sensitive to monotonicity constraints, particularly in tasks involving inference and entailment recognition [41, 163, 2]. Another fundamental property is conservativity, a constraint considered universal among natural language determiners (quantifiers). A determiner D satisfies the property of conservativity if all sets $A, B : B \in D(A) \leftrightarrow A \cap B \in D(A)$. For example, in the statement All cats are mammals, the truth conditions pertain only to the set of entities that are both cats and mammals, rather than to the full domain of mammals. Conservativity thus imposes a domain restriction on quantificational reasoning, effectively reducing the computational complexity involved in semantic evaluation by constraining attention to the relevant subset of entities.

In the current investigation, we did not examine the influence of monotonicity and conservativity on transformer-based language models. For studies that do explore the interaction between these semantic properties and model behaviour, we refer interested readers to works such as Hu et al. (2019) [55] and Rios et al. (2021) [112], which probe the extent to which transformer architectures internalise or simulate monotonicity and conservativity constraints. We acknowledge that a comprehensive integration of these semantic properties—alongside the components of our study on quantifier

generalisation—would yield a more complete characterisation of transformer models' capacity to capture the inferential structure of natural language.

As our primary objective was to investigate the inherent reasoning capabilities of transformer models—rather than to implement a full natural language reasoning system—we did not consider neuro-symbolic approaches that incorporate transformer architectures. As discussed in Section 2, such approaches have demonstrated state-of-the-art performance in domains such as mathematics and code generation. However, in many of these methods, transformers primarily function as translators, converting natural language inputs into formal symbolic representations. Consequently, these models provide limited insight into the reasoning processes intrinsic to the transformer architecture itself. That said, recent work has begun to explore more integrative neuro-symbolic frameworks in which transformers play a more active role, such as generating candidate theorems or axioms to guide symbolic reasoning systems [138]. Extending our experimental setup to incorporate this class of neuro-symbolic methods represents a compelling direction for future research. Such an extension could yield deeper insights into transformers' capacity to comprehend and generate formal mathematical structures, thereby shedding light on their potential for mathematical creativity and deductive reasoning.

Our work is limited to behaviour-based probing of transformer models and does not address interpretability. Interpretability research in transformer-based language models aims to uncover the internal mechanisms through which these models process, represent, and generate language. Among the most promising directions is mechanistic interpretability [33, 74, 3], which seeks to map components of transformer architectures—such as attention heads and neurons—to interpretable circuits that perform specific computational functions. This line of research often leverages structural sparsity to improve tractability and clarity. Recent efforts have examined how transformers perform basic mathematical operations, such as addition, and how reasoning capabilities can emerge through phenomena like grokking [144, 88, 125]. While our current study focuses on the question of whether transformers can learn logical semantics and rules of inference, integrating mechanistic interpretability would provide deeper insights into why and how these capabilities or shortcomings arise. We recognise this as a valuable direction for future work.

7.3 Future Work

Apart from the future work avenues established in the limitations subsection above, we identify the following to be promising future research directions.

Establishing the ability of reasoning models to learn rules of inference and exhibit different forms of generalisation. In our work, we investigate the ability of transformer models to learn logical semantics and rules of inference. Our findings—by and large—align with other evaluation methodologies that have assessed various forms of generalisation [111, 32]. However, it is important to acknowledge that research into transformer models, particularly within the domain of reasoning, has been progressing at a remarkable pace. Indeed, a new generation of transformer-based reasoning models, trained on chain-of-thought-style data and largely refined through reinforcement learning, has rapidly surpassed its predecessors in reasoning capabilities [48, 165]. These models now constitute the state-of-the-art across nearly all significant coding and mathematics benchmarks. Given these advances, a compelling direction for future inquiry would be to examine whether these so-called reasoning models exhibit the ability to learn and apply rules of inference and exhibit generalisation patterns distinct from those of earlier transformer models. Nevertheless, it is important to acknowledge that certain studies have underscored an inherent limitation: Even though these models exhibit better generalisation compared to their non-reasoning counterparts, they are still plagued by the same apparent weaknesses [72].

Improving transformer model architecture. Some researchers have explored various forms of positional encoding [71, 82, 166] and other architectural modifications [36] designed to enhance the length generalisation capabilities of transformer models. However, the impact of such architectural modifications on these models' ability to apply more intricate reasoning rules remains an open and largely unexplored research area. We underscore the need for a more rigorous and comprehensive research effort to refine these models so that they may effectively learn the rules of inference. This challenge remains one of the most significant and persistent obstacles in the study of transformer models.

Constructing better evaluation baselines to assess the true potential of transformer models. We further advocate for the development of more rigorous evaluation schemes for benchmarking transformer models. In a broader sense, many existing evaluation

frameworks are increasingly becoming obsolete. Indeed, transformer models now exhibit expert-level performance across numerous benchmarks, including those designed to assess reasoning capabilities[48]. Given that these models are trained on vast amounts of data, it is imperative that evaluation schemes—particularly those pertaining to reasoning—be designed to disentangle memorisation from genuine reasoning ability. It is worth noting that while these models achieve expert-level performance in various mathematical benchmarks, they nonetheless fail at fundamental tasks such as multiplication when the number of digits exceeds a certain threshold. This calls into question the reliability of widely recognised benchmarks in mathematics, as well as in other reasoning-related domains such as coding, as faithful measures of a model's reasoning capabilities. Thus, there is a pressing need for more sophisticated and adaptable benchmarks that can evolve alongside the rapid advancements in transformer architectures. It is evident that considerable work remains to be done, both in refining transformer models—especially in relation to their reasoning capabilities—and in developing evaluation methodologies that can more accurately assess their true potential.

"They say that great beasts once roamed this world, as big as mountains, yet all that's left of them is bone and amber. Time undoes even the mightiest of creatures. Just look what it's done to you. One day you will perish. You will lie with the rest of your kind in the dirt, your dreams forgotten, your horrors faced. Your bones will turn to sand, and upon that sand a new god will walk. One that will never die. Because this world doesn't belong to you, or the people that came before. It belongs to someone who is yet to come."

— Dolores Abernathy (Evan Rachel Wood)
Westworld

Bibliography

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] G. Agmon, Y. Loewenstein, and Y. Grodzinsky. Measuring the cognitive cost of downward monotonicity by controlling for negative polarity. *Glossa: a journal of general linguistics*, 4(1), 2019.

[3] E. Ameisen, J. Lindsey, A. Pearce, W. Gurnee, N. L. Turner, B. Chen, C. Citro, D. Abrahams, S. Carter, B. Hosmer, et al. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 6, 2025.

[4] C. An, Z. Chen, Q. Ye, E. First, L. Peng, J. Zhang, Z. Wang, S. Lerner, and J. Shang. Learn from failure: Fine-tuning LLMs with trial-and-error data for intuitionistic propositional logic proving. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 776–790, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[5] C. Anil, Y. Wu, A. J. Andreassen, A. Lewkowycz, V. Misra, V. V. Ramasesh, A. Slone, G. Gur-Ari, E. Dyer, and B. Neyshabur. Exploring length generalization in large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[6] M. Apidianaki and A. Garí Soler. ALL dolphins are intelligent and SOME are friendly: Probing BERT for nouns' semantic properties and their prototypicality. In *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 79–94, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics.

[7] Aristotle. *The Categories, On Interpretation, Prior Analytics*. Loeb Classical Library. Harvard University Press, Cambridge, MA, 1938. Tr. H. Cooke and H. Tredennick.

[8] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*, 2023.

[9] A. Bansal, A. Schwarzschild, E. Borgnia, Z. Emam, F. Huang, M. Goldblum, and T. Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[10] E. Beeching, S. C. Huang, A. Jiang, J. Li, B. Lipkin, Z. Qina, K. Rasul, Z. Shen, R. Soletskyi, and L. Tunstall. Numinamath 7b tir. `https://huggingface.co/AI-MO/NuminaMath-7B-TIR`, 2024.

[11] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.

[12] C. Bhagavatula, R. L. Bras, C. Malaviya, K. Sakaguchi, A. Holtzman, H. Rashkin, D. Downey, W. tau Yih, and Y. Choi. Abductive commonsense reasoning. In *International Conference on Learning Representations*, 2020.

[13] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[14] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. 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, Sept. 2015. Association for Computational Linguistics.

[15] H. Bronkhorst, G. Roorda, C. Suhre, and M. Goedhart. Logical reasoning in formal and everyday reasoning tasks. *International Journal of Science and Mathematics Education*, 18:1673–1694, 2020.

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss,

G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[17] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

[18] L. Buijtelaar and S. Pezzelle. A psycholinguistic analysis of BERT's representations of compounds. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2230–2241, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[19] C. Cameron, R. Chen, J. Hartford, and K. Leyton-Brown. Predicting propositional satisfiability via end-to-end learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3324–3331, 2020.

[20] S. Chen, Y. Wang, Y.-F. Wu, Q.-G. Chen, Z. Xu, W. Luo, K. Zhang, and L. Zhang. Advancing tool-augmented large language models: Integrating insights from errors in inference trees. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[21] T. Chen, L. Li, and Y. Sun. Differentiable product quantization for end-to-end embedding compression. In *International Conference on Machine Learning*, pages 1617–1626. PMLR, 2020.

[22] Z. Chen, K. Zhou, B. Zhang, Z. Gong, X. Zhao, and J.-R. Wen. ChatCoT: Tool-augmented chain-of-thought reasoning on chat-based large language models. In H. Bouamor, J. Pino, and K. Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14777–14790, Singapore, Dec. 2023. Association for Computational Linguistics.

[23] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[24] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[25] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[26] P. Clark, O. Tafjord, and K. Richardson. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021.

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

[28] A. Creswell, M. Shanahan, and I. Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] R. Cui, D. Hershcovich, and A. Søgaard. Generalized quantifiers as a source of error in multilingual NLU benchmarks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4875–4893, Seattle, United States, July 2022. Association for Computational Linguistics.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[31] S. Duan, Y. Shi, and W. Xu. From interpolation to extrapolation: Complete length generalization for arithmetic transformers. *arXiv preprint arXiv:2310.11984*, 2023.

[32] N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, S. Welleck, P. West, C. Bhagavatula, R. L. Bras, J. D. Hwang, S. Sanyal, X. Ren, A. Ettinger, Z. Harchaoui, and Y. Choi. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[33] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[34] A. Ettinger. What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48, 2020.

[35] R. Evans, D. Saxton, D. Amos, P. Kohli, and E. Grefenstette. Can neural networks understand logical entailment? In *International Conference on Learning Representations*, 2018.

[36] Y. Fan, Y. Du, K. Ramchandran, and K. Lee. Looped transformers for length generalization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[37] G. Frege. Über die wissenschaftliche berechtigung einer begriffsschrift. 1882.

[38] G. Fuhrken. Per lindström. first order predicate logic with generalized quantifiers. theoria, vol. 32 (1966), pp. 186–195. *The Journal of Symbolic Logic*, 34(4):650–650, 1970.

[39] D. Gabbay, F. Guenthner, and D. Westerståhl. *Quantifiers in formal and natural languages*. Springer, 1989.

[40] A. Geiger, I. Cases, L. Karttunen, and C. Potts. Stress-testing neural models of natural language inference with multiply-quantified sentences. *arXiv preprint arXiv:1810.13033*, 2018.

[41] B. Geurts and F. van der Slik. Monotonicity and Processing Load. *Journal of Semantics*, 22(1):97–117, 02 2005.

[42] E. Glazer, E. Erdil, T. Besiroglu, D. Chicharro, E. Chen, A. Gunning, C. F. Olsson, J.-S. Denain, A. Ho, E. d. O. Santos, et al. Frontiermath: A benchmark for evaluating advanced mathematical reasoning in ai. *arXiv preprint arXiv:2411.04872*, 2024.

[43] M. Glockner, V. Shwartz, and Y. Goldberg. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[44] N. Gontier, K. Sinha, S. Reddy, and C. Pal. Measuring systematic generalization in neural proof generation with transformers. *Advances in Neural Information Processing Systems*, 33:22231–22242, 2020.

[45] E. Goodwin, K. Sinha, and T. J. O'Donnell. Probing linguistic systematicity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1958–1969, Online, July 2020. Association for Computational Linguistics.

[46] E. Grädel, P. Kolaitis, and M. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.

[47] Y. Gu, B. Dalvi Mishra, and P. Clark. Do language models have coherent mental models of everyday things? In A. Rogers, J. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1892–1913, Toronto, Canada, July 2023. Association for Computational Linguistics.

[48] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[49] H. He and J. D. Choi. 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, 2020.

[50] P. He, J. Gao, and W. Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

[51] D. Hendrycks, C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song, and J. Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[52] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[53] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[54] A. Hosseini, S. Reddy, D. Bahdanau, R. D. Hjelm, A. Sordoni, and A. Courville. Understanding by understanding not: Modeling negation in language models. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1301–1312, Online, June 2021. Association for Computational Linguistics.

[55] H. Hu, Q. Chen, and L. Moss. Natural language inference with monotonicity. In S. Dobnik, S. Chatzikyriakidis, and V. Demberg, editors, *Proceedings of the 13th International Conference on Computational Semantics - Short Papers*, pages 8–15, Gothenburg, Sweden, May 2019. Association for Computational Linguistics.

[56] Y. Hu, X. Tang, H. Yang, and M. Zhang. Case-based or rule-based: How do transformers do the math? In *Forty-first International Conference on Machine Learning*, 2024.

[57] N. Jain, K. Han, A. Gu, W.-D. Li, F. Yan, T. Zhang, S. Wang, A. Solar-Lezama, K. Sen, and I. Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

[58] G. Jawahar, B. Sagot, and D. Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics.

[59] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[60] P. N. Johnson-Laird and B. G. Bara. *Syllogistic inference*, volume 16. Elsevier, 1984.

[61] A. Kamath and R. Das. A survey on semantic parsing. In *Automated Knowledge Base Construction (AKBC)*, 2019.

[62] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[63] N. Kassner and H. Schütze. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online, July 2020. Association for Computational Linguistics.

[64] A. Kazemnejad, I. Padhi, K. Natesan, P. Das, and S. Reddy. The impact of positional encoding on length generalization in transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[65] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[66] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[67] V. Kuncak and M. Rinard. Towards efficient satisfiability checking for boolean algebra with presburger arithmetic. In *The Twenty-First International Conference on Automated Deduction CADE-21*, Berlin, 2007. Springer Verlag.

[68] B. M. Lake and M. Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, 2017.

[69] G. Lample and F. Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

[70] N. Lee, Z. Cai, A. Schwarzschild, K. Lee, and D. Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges. *arXiv preprint arXiv:2502.01612*, 2025.

[71] S. Li, C. You, G. Guruganesh, J. Ainslie, S. Ontanon, M. Zaheer, S. Sanghai, Y. Yang, S. Kumar, and S. Bhojanapalli. Functional interpolation for relative positions improves long context transformers. In *The Twelfth International Conference on Learning Representations*, 2024.

[72] B. Y. Lin, R. L. Bras, K. Richardson, A. Sabharwal, R. Poovendran, P. Clark, and Y. Choi. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.

[73] K. Lin, O. Tafjord, P. Clark, and M. Gardner. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62, Hong Kong, China, Nov. 2019. Association for Computational Linguistics.

[74] J. Lindsey, W. Gurnee, E. Ameisen, B. Chen, A. Pearce, N. L. Turner, C. Citro, D. Abrahams, S. Carter, B. Hosmer, et al. On the biology of a large language model. *Transformer Circuits Thread*, 2025.

[75] B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023.

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

[77] N. Lourie, R. Le Bras, C. Bhagavatula, and Y. Choi. Unicorn on rainbow: A universal commonsense reasoning model on a new multitask benchmark. *AAAI*, 2021.

[78] T. Madusanka, R. Batista-navarro, and I. Pratt-hartmann. Identifying the limits of transformers when performing model-checking with natural language. In A. Vlachos and I. Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3539–3550, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[79] T. Madusanka, I. Pratt-Hartmann, and R. Batista-Navarro. Natural language satisfiability: Exploring the problem distribution and evaluating transformer-based language models. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15278–15294, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[80] T. Madusanka, I. Zahid, H. Li, I. Pratt-Hartmann, and R. Batista-Navarro. Not all quantifiers are equal: Probing transformer-based language models' understanding of generalised quantifiers. In H. Bouamor, J. Pino, and K. Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8680–8692, Singapore, Dec. 2023. Association for Computational Linguistics.

[81] T. McCoy, E. Pavlick, and T. Linzen. 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, July 2019. Association for Computational Linguistics.

[82] S. M. McLeish, A. Bansal, A. Stein, N. Jain, J. Kirchenbauer, B. R. Bartoldson, B. Kailkhura, A. Bhatele, J. Geiping, A. Schwarzschild, and T. Goldstein. Transformers can do arithmetic with the right embeddings. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

[83] C. T. McMillan, R. Clark, P. Moore, C. Devita, and M. Grossman. Neural basis for generalized quantifier comprehension. *Neuropsychologia*, 43(12):1729–1737, 2005.

[84] J. Meadows, M. Valentino, D. Teney, and A. Freitas. A symbolic framework for systematic evaluation of mathematical reasoning with transformers. *arXiv preprint arXiv:2305.12563*, 2023.

[85] P. Minervini, M. Bosnjak, T. Rocktäschel, S. Riedel, and E. Grefenstette. 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, 2020.

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

[87] A. Mostowski. On a generalization of quantifiers. *Fundamenta mathematicae*, 44(2), 1957.

[88] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[89] N. Nangia and S. R. Bowman. Human vs. muppet: A conservative estimate of human performance on the GLUE benchmark. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4566–4575, Florence, Italy, July 2019. Association for Computational Linguistics.

[90] N. Narodytska, H. Zhang, A. Gupta, and T. Walsh. In search for a sat-friendly binarized neural network architecture. In *International Conference on Learning Representations*, 2020.

[91] T. Niven and H.-Y. Kao. Probing neural network comprehension of natural language arguments. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy, July 2019. Association for Computational Linguistics.

[92] M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[93] S. Ontanon, J. Ainslie, Z. Fisher, and V. Cvicek. Making transformers solve compositional tasks. In S. Muresan, P. Nakov, and A. Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3591–3607, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[94] OpenAI. Gpt-4 technical report, 2023.

[95] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[96] L. Pan, V. Ganesh, J. Abernethy, C. Esposo, and W. Lee. Can transformers reason logically? a study in sat solving. *arXiv preprint arXiv:2410.07432*, 2024.

[97] E. Pasewark, K. Montgomery, K. Duan, D. Song, and C. Wang. Re-tuning: Overcoming the compositionality limits of large language models with recursive tuning. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10422–10437, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics.

[98] S. Peters and D. Westerståhl. *Quantifiers in language and logic*. OUP Oxford, 2006.

[99] J. Petty, S. Steenkiste, I. Dasgupta, F. Sha, D. Garrette, and T. Linzen. The impact of depth on compositional generalization in transformer language models. In K. Duh, H. Gomez, and S. Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7239–7252, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

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

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

[102] I. Pratt-Hartmann. *Fragments of First-Order Logic*. Oxford University Press, Oxford, UK, 2023.

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

[104] I. Pratt-Hartmann and A. Third. More fragments of language. *Notre Dame Journal of Formal Logic*, 47(2):151–177, 2006.

[105] O. Press, N. Smith, and M. Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022.

[106] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

[107] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[108] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[109] Y. Razeghi, R. L. Logan IV, M. Gardner, and S. Singh. Impact of pretraining term frequencies on few-shot numerical reasoning. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[110] K. Richardson, H. Hu, L. Moss, and A. Sabharwal. Probing natural language inference models through semantic fragments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8713–8721, 2020.

[111] K. Richardson and A. Sabharwal. Pushing the limits of rule reasoning in transformers through natural language satisfiability. In *AAAI Conference on Artificial Intelligence*, 2021.

[112] A. Rios, C. Amrhein, N. Aepli, and R. Sennrich. On biasing transformer attention towards monotonicity. In K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4474–4488, Online, June 2021. Association for Computational Linguistics.

[113] S. Robertson, H. Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[114] S. Saha, S. Ghosh, S. Srivastava, and M. Bansal. 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, Nov. 2020. Association for Computational Linguistics.

[115] R. Schaeffer, B. Miranda, and S. Koyejo. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36, 2024.

[116] V. Schlegel, K. Pavlov, and I. Pratt-Hartmann. Can transformers reason in fragments of natural language? In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11184–11199, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics.

[117] A. Schwarzschild, E. Borgnia, A. Gupta, F. Huang, U. Vishkin, M. Goldblum, and T. Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6695–6706. Curran Associates, Inc., 2021.

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

[119] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019.

[120] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*, 2023.

[121] C. Shin, J. Cooper, and F. Sala. Weak-to-strong generalization through the data-centric lens. In *The Thirteenth International Conference on Learning Representations*, 2025.

[122] R. Shin, N. Kant, K. Gupta, C. Bender, B. Trabucco, R. Singh, and D. Song. Synthetic datasets for neural program synthesis. In *International Conference on Learning Representations*, 2019.

[123] S. Singh, N. Wen, Y. Hou, P. Alipoormolabashi, T.-l. Wu, X. Ma, and N. Peng. COM2SENSE: A commonsense reasoning benchmark with complementary sentences. In C. Zong, F. Xia, W. Li, and R. Navigli, editors, *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 883–898, Online, Aug. 2021. Association for Computational Linguistics.

[124] K. Sinha, S. Sodhani, J. Dong, J. Pineau, and W. L. Hamilton. 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, Nov. 2019. Association for Computational Linguistics.

[125] A. Stolfo, Y. Belinkov, and M. Sachan. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[126] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics.

[127] J. Szymanik et al. *Quantifiers and cognition: Logical and computational perspectives*, volume 96. Springer, 2016.

[128] J. Szymanik, S. Steinert-Threlkeld, M. Zajenkowski, and T. F. Icard III. Automata and complexity in multiple-quantifier sentence verification. *Logic and Interactive RAtionality Yearbook 2012*, page 133, 2013.

[129] J. Szymanik and M. Zajenkowski. Comprehension of simple quantifiers: Empirical evaluation of a computational model. *Cognitive Science*, 34(3):521–532, 2010.

[130] J. Szymanik and M. Zajenkowski. Quantifiers and working memory. In *Logic, Language and Meaning: 17th Amsterdam Colloquium, Amsterdam, The Netherlands, December 16-18, 2009, Revised Selected Papers*, pages 456–464. Springer, 2010.

[131] O. Tafjord, B. Dalvi, and P. Clark. 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, Aug. 2021. Association for Computational Linguistics.

[132] O. Tafjord, M. Gardner, K. Lin, and P. Clark. 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, nov 2019. Association for Computational Linguistics.

[133] A. Talmor, J. Herzig, N. Lourie, and J. Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[134] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[135] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[136] K. Tirumala, A. H. Markosyan, L. Zettlemoyer, and A. Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[137] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[138] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

[139] V. Troiani, J. E. Peelle, R. Clark, and M. Grossman. Is it logical to count on quantifiers? dissociable neural networks underlying numerical and logical quantifiers. *Neuropsychologia*, 47(1):104–111, 2009.

[140] M. Valentino, J. Meadows, L. Zhang, and A. Freitas. Multi-operational mathematical derivations in latent space. *arXiv preprint arXiv:2311.01230*, 2023.

[141] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[142] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. *SuperGLUE: a stickier benchmark for general-purpose language understanding systems*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[143] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In T. Linzen, G. Chrupała, and A. Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics.

[144] B. Wang, X. Yue, Y. Su, and H. Sun. Grokking of implicit reasoning in transformers: A mechanistic journey to the edge of generalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[145] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[146] L. Weber, P. Minervini, J. Münchmeyer, U. Leser, and T. Rocktäschel. 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, July 2019. Association for Computational Linguistics.

[147] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022.

[148] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[149] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[150] S. Welleck, J. Liu, R. L. Bras, H. Hajishirzi, Y. Choi, and K. Cho. Naturalproofs: Mathematical theorem proving in natural language. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[151] D. Westerståhl. Branching generalized quantifiers and natural language. *Generalized quantifiers: Linguistic and logical approaches*, pages 269–298, 1987.

[152] J. Weston, A. Bordes, S. Chopra, and T. Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. In Y. Bengio and Y. LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[153] A. Williams, N. Nangia, and S. Bowman. 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, June 2018. Association for Computational Linguistics.

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

[155] B. Workshop, T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[156] Z. Wu, E. Kreiss, D. Ong, and C. Potts. ReaSCAN: Compositional reasoning in language grounding. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[157] H. Xin, D. Guo, Z. Shao, Z. Ren, Q. Zhu, B. Liu, C. Ruan, W. Li, and X. Liang. Advancing theorem proving in LLMs through large-scale synthetic data. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024.

[158] K. Xu, J. Li, M. Zhang, S. S. Du, K. ichi Kawarabayashi, and S. Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020.

[159] H. Yanaka, K. Mineshima, D. Bekki, and K. Inui. Do neural models learn systematicity of monotonicity inference in natural language? In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6105–6117, 2020.

[160] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, volume 32, 2019.

[161] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. R. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[162] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[163] H. Zeijlstra. 426Negative Quantifiers. In *The Oxford Handbook of Negation*. Oxford University Press, 03 2020.

[164] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[165] T. Zhong, Z. Liu, Y. Pan, Y. Zhang, Y. Zhou, S. Liang, Z. Wu, Y. Lyu, P. Shu, X. Yu, et al. Evaluation of openai o1: Opportunities and challenges of agi. *arXiv preprint arXiv:2409.18486*, 2024.

[166] A. Zhou, K. Wang, Z. Lu, W. Shi, S. Luo, Z. Qin, S. Lu, A. Jia, L. Song, M. Zhan, and H. Li. Solving challenging math word problems using GPT-4 code interpreter with code-based self-verification. In *The Twelfth International Conference on Learning Representations*, 2024.

[167] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.