**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCES**
**DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS**

**DATA SCIENCE & INFORMATION TECHNOLOGIES MASTERS PROGRAM**

**MSc THESIS**

# Natural Language Reasoning on ALC knowledge bases using Large Language Models

**Stylianos E. Kotzastratis**

**ATHENS**

**JUNE 2023**

**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΜΕΤΑΠΤΥΧΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΠΙΣΤΗΜΗΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΑΣ**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# Συλλογιστική φυσικής γλώσσας σε βάσεις γνώσεων ALC με τη χρήση μεγάλων γλωσσικών μοντέλων

**Στυλιανός Ε. Κοτζαστράτης**

**ΑΘΗΝΑ**

**ΙΟΥΝΙΟΣ 2023**

**MSc THESIS**

Natural Language Reasoning on ALC knowledge bases using Large Language Models

**Stylianos E. Kotzastratis**
DS1200006

**SUPERVISOR: Manolis Koubarakis**, Professor National and Kapodistrian University of Athens

**CO-SUPERVISOR: Eleni Tsalapati**, Postdoctoral Researcher National and Kapodistrian University of Athens

**THREE-MEMBER ADVISORY COMMITTEE:**

**Manolis Koubarakis**, Professor National and Kapodistrian University of Athens

**Eleni Tsalapati**, Postdoctoral Researcher National and Kapodistrian University of Athens

**Ioannis Panagakis**, Associate Professor National and Kapodistrian University of Athens

June 2023

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

Συλλογιστική φυσικής γλώσσας σε βάσεις γνώσεων ALC με τη χρήση μεγάλων γλωσσικών μοντέλων

**Στυλιανός Ε. Κοτζαστράτης**
DS1200006

**ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Μανόλης Κουμπαράκης**, Καθηγητής Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

**ΣΥΝΕΠΙΒΛΕΠΟΥΣΑ: Ελένη Τσαλαπάτη**, Μεταδιδακτορική Ερευνήτρια Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

**ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:**

**Μανόλης Κουμπαράκης**, Καθηγητής Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

**Ελένη Τσαλαπάτη**, Μεταδιδακτορική Ερευνήτρια Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

**Ιωάννης Παναγάκης**, Αναπληρωτής Καθηγητής Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Ιούνιος 2023

# ABSTRACT

Pretrained language models have dominated natural language processing, challenging the use of knowledge representation languages to describe the world. While these languages are not expressive enough to fully cover natural language, language models have already shown great results in terms of understanding and information retrieval directly on natural language data. We explore language models' performance at the downstream task of natural language reasoning in the description logic ALC. We generate a dataset of random ALC knowledge bases, translated in natural language, in order to assess the language models' ability to function as question-answering systems over natural language knowledge bases.



**SUBJECT AREA**: Deep Learning, Natural Language Reasoning

**KEYWORDS**: Reasoning, Language Models, ALC, Natural Language, Synthetic data

## ΠΕΡΙΛΗΨΗ

Τα προεκπαιδευμένα γλωσσικά μοντέλα έχουν κυριαρχήσει στην επεξεργασία φυσικής γλώσσας, αποτελώντας πρόκληση για τη χρήση γλωσσών αναπαράστασης γνώσης για την περιγραφή του κόσμου. Ενώ οι γλώσσες αυτές δεν είναι αρκετά εκφραστικές για να καλύψουν πλήρως τη φυσική γλώσσα, τα γλωσσικά μοντέλα έχουν ήδη δείξει σπουδαία αποτελέσματα όσον αφορά την κατανόηση και την ανάκτηση πληροφοριών απευθείας σε δεδομένα φυσικής γλώσσας. Διερευνούμε τις επιδόσεις των γλωσσικών μοντέλων για συλλογιστική φυσικής γλώσσας στη περιγραφική λογική ALC. Δημιουργούμε ένα σύνολο δεδομένων από τυχαίες βάσεις γνώσης ALC, μεταφρασμένες σε φυσική γλώσσα, ώστε να αξιολογήσουμε την ικανότητα των γλωσσικών μοντέλων να λειτουργούν ως συστήματα απάντησης ερωτήσεων πάνω σε βάσεις γνώσης φυσικής γλώσσας.



**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ**: Βαθειά Μάθηση, Συλλογιστική σε Φυσική Γλώσσα

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ**: Συλλογιστική, Γλωσσικά Μοντέλα, ALC, Φυσική Γλώσσα, Συνθετικά Δεδομένα

*To my parents.*

*The human mind is a ship – the immediate inferences we make from instinct keep us afloat, but reason is the compass that brings us to the shore of wisdom.*

- Ben Prystawski et al. [48]

# ACKNOWLEDGEMENTS

I would like to thank Professor Manolis Koubarakis at National Kapodistrian University of Athens, for giving me the opportunity to work on Natural Language Reasoning, and for his trust, patience and supervision. Additionally, I would like to express my gratitude to postdoctoral researcher Eleni Tsalapati for her invaluable guidance, unwavering support and expertise, which played a pivotal role in the successful completion of this project. I would like to also thank research assistant Despina Pantazi for her insightful advice on my experiments. Finally, I owe thanks to my partner, Nelly, for offering emotional support and determination to overcome critical obstacles, as well as my family that supported me for the entirety of my postgraduate studies. None of my achievements would be possible without them.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# PREFACE

This dissertation was carried out in the context of my MSc studies in the Data Science & Information Technologies program of the National Kapodistrian University of Athens, following the Big Data & Artificial Intelligence specialization.

# 1. INTRODUCTION

The vision to create and use Question-Answering (QA) machines has been one of the most prominent challenges in Computer Science. Already from 1950 with Turing's Imitation Game [67] question answering is associated with the concept of machine intelligence and up to nowadays it still remains an open field of study. In its center one can find the aim to create software that can receive a question in natural language, analyze it and formulate an accurate answer as output, again in natural language. With the advent of Artificial Intelligence, and more recently Deep Learning systems, there is an abundance of tasks and use-cases in which QA systems are set in the forefront.

Applications such as this nowadays include personal companions and assistants like Amazon's Alexa or Mycroft, call centers' automated systems for businesses like banks or travel agencies and tutoring assistants for educational centers. It includes search engines like Google, which for example aims to provide information beyond a simple listing of search results or Microsoft Bing and You.com, which completely redefined Search, by interfacing it through a chatbot. With the ongoing digitization and automation of fields like construction, driving, social services and more, the increase in demand for QA systems is expected to rapidly increase; both in professional environments and on individual level. With the recent advances by OpenAI, we see a vast number of fields being revolutionized in terms of productivity and human-machine interaction.

Most original QA systems were heavily relying on repositories of knowledge as references for answers (open-book QA), and were bound to technologies that focus on information management and retrieval, such as databases and semantic knowledge bases. While the benefits of such approaches are still highly regarded by researchers [32, 25, 30], the dominance of these approaches is challenged by the great improvements in the field of Deep Learning and the advent of neural- and attention-based end-to-end systems. The potential of deep neural networks has led to remarkable achievements in all fields of data manipulation and information extraction and specific to the field of Natural Language Processing (NLP), the creation of the Transformer architecture [68] turned out to be the pinnacle for recent advancements in the field [83].

Recently, in the field of QA, we see Transformer based Large Language Models (LLMs) that offer outstanding performances that were simply impossible without learning-powered architectures [51, 7]. The great results have increased the interest in the capabilities of those systems to empower the task of question-answering beyond the previously achieved information retrieval phase, by simulating intelligence. Instrumental in this vision are NLP tasks distinct to QA, like reasoning or inference, that so far also leveraged primarily information retrieval approaches and more specifically centered around the formal representation of knowledge. Systems like these have presented several challenges like limitations when faced with incomplete information, data sparsity and general bottlenecks with respect to information formulation, sensitivity to errors and inability to utilize fuzzy or paraphrased information [77].

Additionally, the success of such systems has been restrained by the use of formal languages, which are not expressive enough to fully cover the overly complex and ever-evolving nature of human language. Therefore, utilizing non-formulated but learned approximations that are offered in the Deep Learning space can pave a new path forward. There are several works that explore the reasoning potential of LLMs, re-shaping the field of Natural Language Reasoning (NLR), and this is the direction our work is following. Reasoning is defined as the process of utilizing existing information or knowledge to reach conclusions, and while its narrow definition may vary depending on the context of discourse [78], reasoning is not a subset of Question-Answering, but a critical characteristic that can enable it. Furthermore, it is recently considered by many as one of the emergent characteristics of LLMs as they scale to hundreds of billions of parameters [64, 71]. At the same time, research for a thorough evaluation of this finding is still ongoing, as it is at the same time qualitatively distinct from all other NLP tasks, in the sense that simply scaling a model does not reap analogous improvements, with the reasoning performance remaining comparatively low [13].

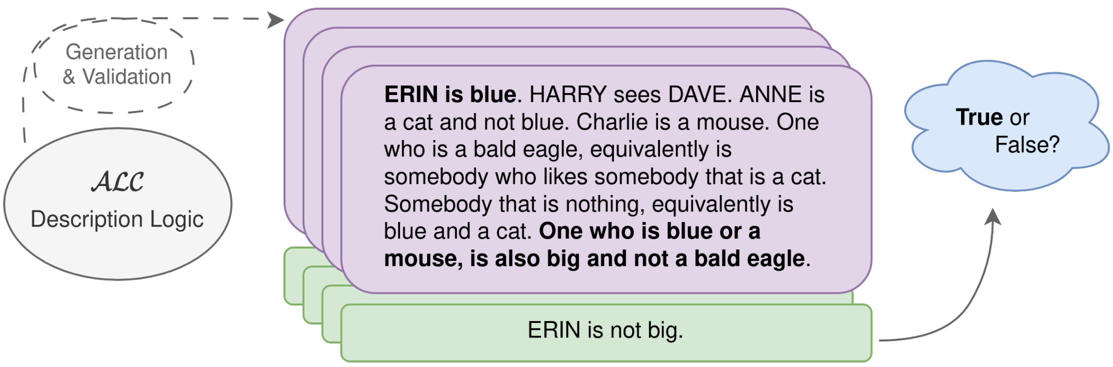



**Figure 1.1: A bird's-eye view of the evaluated task.**

In our work, we aim to assess if a Transformer language model can emulate the algorithmic logic of a formal reasoner, with a specific focus in the expressive capacity of the ALC description logic. The foundation of our work is the publication of P. Clark et al. [11], in which they set out to investigate the same question but focused on a simpler subset of natural language expressions. To perform this soft reasoning, the Transformer needs input in natural language and for that they are creating a pipeline that formulates randomly synthesized theories, validates their consistency and then translates them in natural language with the use of templates. This is the same approach that is followed in this thesis, with the key difference being the expressive capacity of the statements that are generated. In [11] the statements used are limited to (possibly negated) conjunctive expressions, essentially of the form $condition[\wedge condition]^{*}\rightarrow conclusion$. Extending that, in our work we are focusing on additionally supporting disjunctive expressions and existential or universal quantification.

In section 2 we are establishing the necessary background that frames our work, before presenting related works and publications in section 3. Then, in section 4, we are presenting details and key topics regarding the dataset generation pipeline and in section 5 we are presenting our experiments and findings.

# 2. BACKGROUND

## 2.1 Description Logics and $\mathcal{ALC}$

As described in [1], a Description Logic is a formal method to define and describe objects conceptually, offering both the representation of domain knowledge based on logic, but also the ability to perform reasoning on this knowledge structure. This method is defined by the semantics and syntax of formal expressions, and thus a description logic can be seen through the lens of a formal language. The logic rules expressed in this language constitute knowledge bases of descriptions, consist of concept, role and individual names as primary symbols, and can be further grouped in subsequent subsets of rules: *TBox*, *ABox* and *RBox* [53]. *TBox* is the terminological part of a knowledge base, consisting of general concept definitions and imposed constraints among them. It includes intentional knowledge about the targeted schema. *ABox* is the assertional part of a knowledge base complementing the TBox by defining *individuals* and factual statements that revolve around associating those individuals with named components of the TBox. It is also known as extensional knowledge. *RBox*, when existent, is the relational part that extends the knowledge base by introducing a set of statements that relate role names with other roles.

**Table 2.1: Collection of symbols used in $\mathcal{ALC}$, with examples.**

| Name | Symbol | Description | Example |
|---|---|---|---|
| (Atomic) Concept | - | Classes, sets of entities, unary predicates | $Cat, Lawyer, Woman$ |
| (Atomic) Role | - | Properties, relationships, binary predicates | $eats, loves, hasChild$ |
| Individual | - | Specific entities, instances | $JOHN, MARIA, ASH$ |
| Conjunction | $\sqcap$ | Boolean AND | $Alive \sqcap Well$ |
| Disjunction | $\sqcup$ | Boolean OR | $Dead \sqcup Alive$ |
| Negation | $\neg$ | Boolean negation | $\neg Tall$ |
| Universal quantifier | $\forall$ | Meaning "for all". Used on roles. | $\forall hasChild.Person$ |
| Existential quantifier | $\exists$ | Meaning "there exists some". Used on roles. | $\exists hasChild.Female$ |
| Equivalence | $\equiv$ | Indicating definition | $Dead \equiv \neg\ Alive$ |
| Inclusion | $\sqsubseteq$ | Indicating subsumption, $is\_a$ property | $Well \sqsubseteq Alive$ |
| Top concept | $\top$ | Concept symbol to represent *Everything* | $\top \sqsubseteq Dead \sqcup Alive$ |
| Bottom concept | $\bot$ | Concept symbol to represent *Nothing* | $Dead \sqcap Alive \sqsubseteq \bot$ |

Description logics are characterized by their expressive capacity, which in turn is, by convention, roughly hinted at by their name. For example, a description logic with the letter $\mathcal{U}$ in its name allows the use of *concept union* (e.g. $LightbulbState \equiv ON \sqcup OFF$), while one with the letter $\mathcal{O}$ allows for the use of *nominals* (e.g. $oneOf(AsianPerson, HumanPerson)$). One of the most popular description logics is $\mathcal{ALC}$. $\mathcal{ALC}$ is a language that does not contain an RBox, but is built on top of some ABox and TBox. It is defined as an **A**ttributive **L**anguage ($\mathcal{AL}$) extended by **C**omplements; meaning that it allows atomic and complex name negation, concept union and intersection, and universal and existential restrictions. In Table 2.1 we can see a list of common primary symbols and in Table 2.2 the rules that define a concept in $\mathcal{ALC}$.

While most of the concepts presented are not new, there are important details that need to

**Table 2.2: Rules that define what is considered a *concept* in $\mathcal{ALC}$.**

| No. | Rule |
|---|---|
| *1* | *Top and Bottom concepts are concepts.* |
| *2* | *Every concept name is a concept.* |
| *3* | *The complement of a concept is a concept.* |
| *4* | *The conjunction or disjunction of two concepts, is a concept.* |
| *5* | *The universal or existential restriction of a role which satisfies a concept, is a concept.* |
| *6* | *If a name isn't described in rules 1-5, it is not a concept.* |

be taken into account when modeling a system based on $\mathcal{ALC}$. Firstly, an atomic concept is the characterization of a set of entities[1], as a name description, but not abstract concepts in general. For example, while *Electricity* is a concept semantically, it should better not be modeled as an atomic concept in $\mathcal{ALC}$, as it is abstract, not being able to be materialized by individuals. Furthermore, we should also better describe what is a concept definition. An equivalence of the form $A \equiv B$ states a necessary and sufficient condition for membership in the concept $A$: For an individual to be in $A$, it is necessary that it is also in $B$ (it has the properties expressed by $B$). If an individual is in $B$, this is a sufficient condition for concluding that it is also in $A$. Similarly, a concept inclusion of the form $A \sqsubseteq B$ states a necessary condition: For an individual to be in $A$, it is necessary that it is also in $B$ (it has the properties expressed by $B$), but the opposite may not be true. The left concept is a subset of the right concept. Concept inclusions are very useful, allowing a variety of expressions, see for example table 2.3. Another way to achieve complex but concise expressions is the use of quantifiers. More specifically, using the existential quantifier one can create expressions like $\exists hasChild.Female(John)$ translating into *"John is somebody who has some child who is female"*, while when using the universal quantifier the same expression $\forall hasChild.Female(John)$ would translate into *"John is somebody who has only children who are female"*. Finally, an example of statements that could form an $\mathcal{ALC}$ knowledge base can be seen in Table 2.3.

## 2.2 Reasoning, Ontologies and Explanations

The previously defined description logics are connected to the notion of ontologies, which are often used interchangeably with the term of knowledge graph, creating some confusion. As described above, a *Description Logic* can be seen as a formal language based on logic through which one can create cohesive statements that can together form a *Knowledge Base*. A knowledge base's TBox can be considered synonymous to an *Ontology*, which is nothing more than the representation of knowledge of interest in an abstract but also strict manner [39]. Be it in a graph or rule-set representation, an ontology contains concepts, relations, and attributes of concepts and outlines their interconnections. A *Knowledge Graph* is formed when a designed ontology is populated with *individuals*,

[1]Pairing concepts with the conjunction and disjunction operators lead to complex concept name descriptions, as shown in Table 2.1

**Table 2.3: $\mathcal{ALC}$ statements are presented to illustrate examples of expressive potential, like domain and range definitions, grounding of concepts or concept inclusion. These statements could be used in some knowledge base and enable reasoning. Note that the natural language descriptions purposefully vary to illustrate possible expressive correspondences, they are not the ones used in our experiments later.**

| $\mathcal{ALC}$ **Syntax** | **Natural Language description** |
|---|---|
| $Cat \sqsubseteq Animal$ | Every cat is an animal. |
| $Animal \sqsubseteq \top$ | If something is an animal, then it also a Thing. |
| $Herbivore \sqsubseteq Animal$ | Every herbivore is an animal. |
| $Herbivore \sqsubseteq \exists eats.Plant$ | Every herbivore eats at least one plant. |
| $Animal \sqsubseteq \exists has.Sound$ | Every animal has a sound. |
| $Carnivore \sqsubseteq Dog \sqcup Cat$ | Every carnivore is either a dog or a cat. |
| $Mammal \sqsubseteq Animal \sqcap \exists has.Hair$ | Every mammal is an animal and has hair. |
| $\exists eats.Plant \sqsubseteq Animal$ | If something eats some plant, then that ”something” is an animal. |
| $\exists has.Owner \sqcap Animal \equiv Pet$ | Anything that has an owner and is an animal is equivalent to a pet. |
| $\neg Dog(Max)$ | Max is not a dog. |
| $\exists eats.Plant(Max)$ | Max eats a plant. |

materializing its information, and can therefore be considered a synonym of a *complete* knowledge base (TBox + ABox).

The aforementioned information representations are what forms the theoretical foundation of formal reasoning in tools like HermiT [20], enabling formal query answering. As most commonly used, a reasoner parses the provided knowledge bases to perform query answering, returning *True*, *False* or *Unknown* label results, depending on the assumed theoretical boundaries. These assumptions reflect different approaches to dealing with incomplete information and uncertainty in logical reasoning systems. The two primary ones are the Open World Assumption (OWA) and the Closed World Assumption (CWA). The *OWA* can include all three labels and is commonly applied in scenarios where the available information is limited or where the reasoner has incomplete knowledge about the world. In such cases, the reasoner can only assert what is known to be true or false based on the available evidence, but it does not assume that anything beyond that evidence is necessarily false. In contrast, the *CWA* operates on the principle that if a statement cannot be proven to be true based on the available evidence, it is assumed to be false, hence ignoring the Unknown label. CWA is often used in situations where the available information is assumed to be complete, and the reasoner can make conclusive judgments about the truth or falsity of statements.

Finally, a product of formal reasoning processes that is central in this work is the *explanations* of inferred statements. By the term explanation, we are referring to the *justification* as defined in [28]: The minimal subset of an ontology that entails some axiom, given that this axiom is the ontology's entailment. This is a definition that we follow for the rest of this dissertation when referring to explanations, and examples can be seen in section A.2 of the Appendix. The use of formal reasoners in this work is centered around the generation of formal explanations as well as the verification of the generated knowledge bases' consistency, while the target soft-reasoning is performed by Transformer large language models [68].

## 2.3 Large Language Models and the Transformer architecture

Large language models refer to powerful computational models that are capable of understanding and generating human-like text, based on extensive training on vast amounts of textual data. These models have gained significant attention in the field of artificial intelligence and natural language processing due to their ability to perform a wide range of language-related tasks, including tasks like text classification, generation, translation, summarization, and question answering. They are typically based on neural or attention-based architectures, which have revolutionized NLP tasks in recent years, as they are able to capture complex linguistic patterns and dependencies in text.

Their training process involves two main stages: pretraining and fine-tuning. During pretraining, the models are trained on a massive corpus of text, such as books, articles, or entire scrapped collections of web pages, to learn the statistical properties and underlying patterns of natural language. This pretraining enables the models to acquire a broad understanding of grammar, semantics, and contextual relationships, in order to then apply this encoded knowledge to the selected tasks. After pretraining, the models can undergo fine-tuning on further training data to specialize on specific applications. This can involve training the models on task-specific datasets with labeled examples, such as sentiment analysis or question answering or, in our case, sequence classification for logical reasoning. The fine-tuning process adjusts a model's parameters to optimize its performance on the targeted task, leveraging the general language knowledge acquired during pretraining.

The key advantage of the latest large language models is their ability to generate coherent and contextually relevant text that closely resembles human-authored content, reproducing various writing styles, terminologies or even languages, making them highly versatile in a global context. However, concerns regarding ethical, societal, and privacy aspects are also raised, due to the potential for misleading or biased information, propagation of misinformation, and general harmful manipulation of these powerful models [73].

The reason specifically Transformer models are heavily researched and considered adequate for most NLP tasks, is both their observed ability to perform well zero-shot [38] and their ability to form internal context-dependent representations that overcome ambiguity in high dimensions and capture implicit knowledge which is still difficult to model accurately in their weights [78].

The Transformer architecture was first introduced in [68] in 2017, as a Machine Translation model. The key innovation of the Transformer model lies in its self-attention mechanism, which enables it to capture the dependencies and relationships between words in a sentence. Unlike previous architectures that process sequential data linearly, the Transformer model processes all words in parallel, making it highly efficient for parallel computation.

The *self-attention* mechanism allows the Transformer model to assign different weights to each word in a sentence, depending on its relevance to other words, indicating their relative importance. For example, given a sentence *"John is smart, he is a good scientist"*, the word *"he"* should be considered highly relevant to words *"John"* and *"scientist"*, and not as much with words *"is"* or *"good"*, regardless of the positions of the words in the text. This

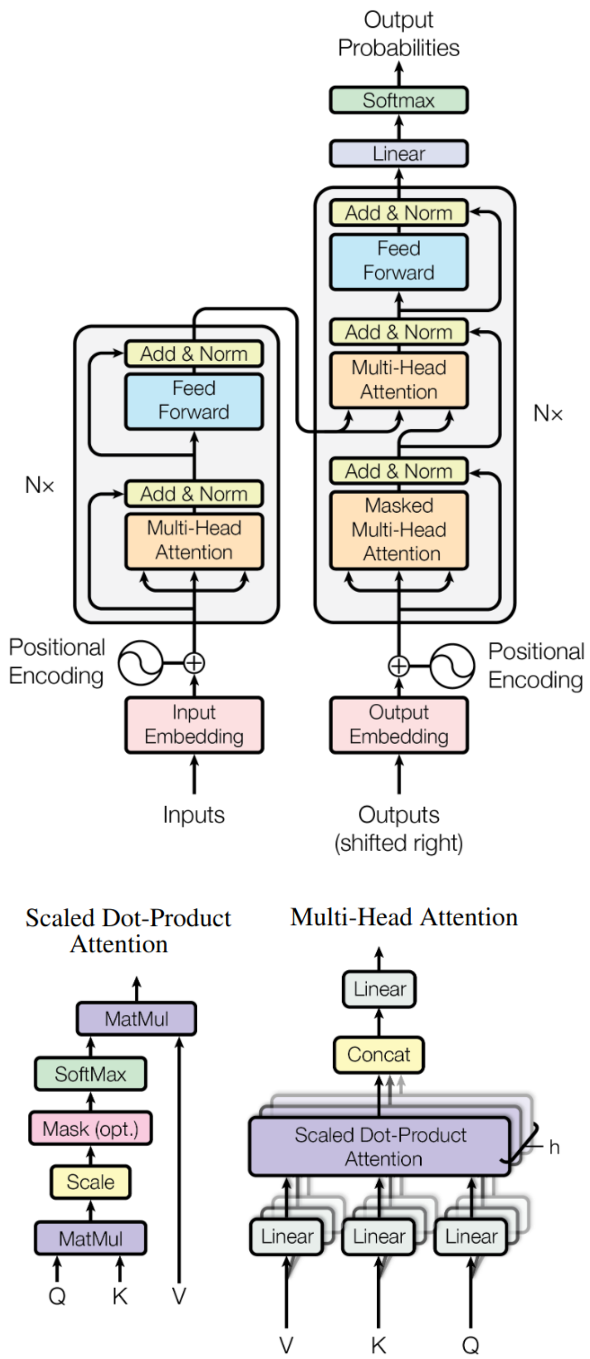



**Figure 2.1: Schematic illustration of the transformer model, taken from [68]. The top-left module depicts the encoder, while the top-right the decoder.**

enables the model to focus on the most important contextual information during both the encoding and decoding stages, which are described next. By *attending* to different parts of the input sequence, the model can capture long-range dependencies and efficiently model context, while previous models were struggling due to their architectural characteristics. The attention weights are used to compute a weighted sum of the encoded word representations, generating a contextually enriched representation for each word. This process is performed iteratively across multiple self-attention layers, allowing the model to refine its understanding of the input text. On a high level, self-attention at each layer is modelled as a normalized transformation of the input words, which are represented by three embeddings each, with learnable weight matrices of the model. It is also important to point out that the self-attention calculation is performed multiple times in parallel, using different embedding transformations, which allows taking into account more input information and token interdependencies; a function described as *multi-head attention*.

The Transformer model consists of an encoder and a decoder. The encoder takes text input and encodes it into a sequence of contextualized representations, while the decoder generates output text based on those encoded representations. Each encoder and decoder layer in the Transformer model contains multiple self-attention sub-layers, feed-forward neural networks, normalization steps and residual connections, that allow the model to capture complex patterns and relationships in the data.

The encoder input is the text transformed as input embeddings, which lack context but are augmented with positional information through supplementary vectors. By passing through a stack of encoder layers they undergo repeated transformations centered around the previously described self-attention, that progressively refine the representations in the projected dimensional spaces, enabling the encoder to capture increasingly integrated contextual information and encode it into the embeddings iteratively. Ultimately, this stacking of encoder layers empowers the encoder to effectively model dependencies and generate more comprehensive and contextually grounded representations of the input text than a single encoder layer would.

The encoder's output is passed as input to all decoder layers, each consisting of similar attention, neural and normalization modules. There are two key differences that differentiate them from the encoder layers. The first is that for the decoder layers, self-attention is only applied to the tokens preceding a certain token (*masked* multi-head attention), as this is the only way to enable *generation* of the upcoming words instead of attending to the respective input word and copying it. The second difference is that the previously generated output of each layer is reused as input to it, essentially to be combined with the encoder's outputs and form the three-embedding input for the attention module of the decoder, which is architecturally similar to the encoder's.

The decoder output is subsequently projected with a linear layer to a high dimensional space that corresponds to the model vocabulary size, and a softmax which binds it to the most probable output word. Having said that, the presence of an encoder or a decoder depends on the specific task the model is designed to perform. For example, in classification tasks, the Transformer model can have only an encoder module, that encodes the input, and the final pooled representation is fed into a classification layer, to predict the class label; models like these are BERT [16] and RoBERTa [43]. A decoder only model would be suited for tasks like language generation, where the model generates a text word by word, based on the encoded representations already available from the training phase; an example would be GPT3 [7].

# 3. RELATED WORK

Aligned with our vision, the authors of [77] are introducing the term of Logical Reasoning over Natural Language (LRNL) with the motivation of identifying and defining the subfield which focuses in transitioning from the paradigm of formal knowledge representation and formal reasoners, towards one where knowledge representation is free natural language itself and language models take the role of reasoner modules. Among the primary arguments the authors raise in favor of this new paradigm, they note the ability of language models to encode pre-existing knowledge to their internal weights, allowing less dependence on knowledge and information availability during prompting. They also mention the increased robustness to errors in the input compared to formal methods, the much wider range of compatible inputs that are generally available and also the potential for more well-rounded systems through iterative improvements using paraphrasing, symbols and semantic variations of any available information. On the other hand, the authors identify present challenges compared to formal reasoners. They include the computational overhead that is constant even for simple logical queries, the frailty against adversarial attacks as well as the existing reliability concerns for produced outputs, due to the current limitations in the interpretability of most neural methods and the difficulty to evaluate concretely due to the high reliance on synthetic datasets, given also the high cost of expert-authored ones.

The task at hand is, as stated in [11], the one of soft reasoning over language with explicitly provided rules, that need to be combined in order to provide correct boolean conclusions. Given our focus in utilizing $\mathcal{ALC}$ description logic, we are also limiting our scope to deductive reasoning using forward-chaining; deducing inference steps from pre-existing knowledge towards an ultimate and infallible conclusion, in our case a theorem proof. Having said that, our models were evaluated in an end-to-end set-up, without interpreting intermediate reasoning steps. Based on the carried out review, only the work presented in [23] seems to cover the expressive capacity of $\mathcal{ALC}$ description logic, but their dataset is vastly smaller. Additionally, here we are focusing only on binary label prediction, something that the research community seems to have shifted away from, but the datasets we are providing contain the explanation statements allowing for further evaluation on additional tasks.

Pioneering in the NLR field is the work in [11], the methodology of which we closely follow to generate a dataset and evaluate the results. While in [11] the authors are creating datalog theories [9], here the focus is on creating description logic knowledge bases, in a broader expressive landscape. They illustrate the ability for theorem proving in a single step over synthetically generated rulesets, and the potential to transfer the ability to out-of-distribution data. Their work in [11] is generally considered to have illustrated the ability of models to transfer deductive reasoning potential from explicitly provided statements to different domains [78], but as we later describe, there are several works that challenge the extent that reasoning is truly performed as opposed to overfitting. Aiming for more faithful predictions, their work is succeeded by the same authors in [65], where they extend their dataset from the CWA to the OWA and present a generative approach that is able to accompany the reasoning result with its corresponding proof, offering greater observability in the model's decisions. They investigated two different proof generation pipelines, namely a single-step approach which works as a black box for proof generation, and a second iterative approach where each inference step is distinctly generated and reused until no other is available. It was shown that the iterative approach was superior for harder examples and more robust over-all, albeit inefficient. They seem to improve the output performance over their previous work, but also use a different Transformer model, which makes a comparative assessment harder.

The work presented in [54] is another follow-up to [11] which aims to improve on zero-shot performance by bringing the proof generation to the foreground as well, and preceded [65]. The authors attempt to solve explicit inference rules and multiple reasoning steps in an end-to-end manner without intermediate logical representations. The approach is conceptually similar to work on neural theorem proving, albeit that it works with free-form natural language input to generate proofs similar to formal reasoners. A proof is defined as a directed graph with nodes representing facts, rules, or Negation-As-Failure (NAF), and edges indicating consumption of a fact by a rule, or consumption of the output of a rule by another rule. Their results indicate performance improvements in most evaluated scenarios, but also seem to struggle when deeper proofs are required. Additionally, in [65] this approach was presented to not be superior to the one presented there, both in terms of accuracy performance and given that intermediate conclusions are not made available.

In [23] the authors focused on First-Order-Logic (FOL) and curated a hand-authored dataset to evaluate reasoning potential of generative models. Compared to other works like [11] their dataset is non-synthetic, more complex and fluent, contains deeper proofs, but it is also vastly smaller due to its high cost. Due to this last point, the greatest percentage of their data is still reliant on a set of natural language templates, since curating the entire dataset from scratch imputes a huge cost. The templates form generic syllogisms, which, working as guides, are later factually populated with nouns and clauses by expert annotators to reflect real-life scenarios. Additionally, it's important to point out that the dataset created in this work is the one closest to ours regarding its expressive potential, as it contains universal and existential quantifiers, and the disjunction operator, on top of conjunctions and negations. Their experimental results show that the used LLMs are only slightly better than a random performance, indicating that their dataset can work as an interesting benchmark in NLR.

In [35] the authors explore the capacity of BERT-based models to capture factual knowledge as symbolic reasoners. In this work as opposed to others they are interfering with the models' pretraining, to provide synthetic corpora composed of fact triples. In their findings, they show that BERT [16] can partially achieve good results on reasoning rules, being particularly weaker on multistep rules, and explicitly requiring some memorization as well for knowledge acquisition. Finally, in their results, they clearly identify that symbolic data understanding is achievable but the natural language data one still remains uncertain.

Another work that presented a data generation pipeline of interest is [6]. The datasets created by the authors are rich both semantically and lexically, since their idea is founded in scraping common natural language formations from big corpora like Wikipedia, regardless of their subject or vocabulary. Following up their pattern matching retrieval, they add a string manipulation step which reorders and recombines the detected semantic parts into new statements which fit specific templates of model reasoning processes, like substitution or contraposition. The final step includes automatic paraphrasing with a language model fine-tuned for the task, finally shaping datasets with high fluency. An additional advantage of their model-based paraphrasing is the reduction of template regularities or statistical features which can reduce overfitting of the target models; an issue which as we describe later is maybe the biggest roadblock in this field of research. In their conclusions, they illustrate that the generative models fine-tuned with their datasets over different ones (including the ones from [65]) achieve observably superior performance.

In a publication from DeepMind [12] the authors present an iterative four-model generative pipeline that is capable of outputting forward-chaining reasoning steps as justification for its predictions, improving the faithfulness of the reasoning output. Each of the models has a different task in focus, and is trained in isolation. The first model is trained to accurately identify and select statements that are explicit parts of the input context, without using any information from its internal biases (hallucinating), and combine them into sentences. The output is passed to the second model, which is trained to produce an inference without having access to the initial query question, to avoid using it in a backward step to find a matching inference. The produced inference is returned to the context to be reused as input in the first model. The third model is trained to evaluate the produced inference against the initial query question, and if it decides that it can be answered, to appropriately format the output, halting the iterations. Finally, the fourth model is trained on properly assigning a value to a potential reasoning step, to work as a guide of a beam search over all the possible reasoning paths that unfolded by the previous models. The pipeline is evaluated in datasets from [65] and is presented to be superior to all baselines, especially in harder problems, while ensuring the faithfulness of the answer when the baseline does not. They additionally show that when the context reliability is negatively transformed with respect to the questions, the model is in most cases able to predict that the reasoning process is indeed inconclusive. In another modular approach with a focus on faithfulness, the authors of [56] also split the selection of statements from knowledge composition. They are using two models for rule and fact selection, and one for composition of the selections. Their results on ProofWriter [65] data show increased robustness compared to the baseline model in paraphrased language.

In a parallel direction of research, it has been shown that Chain-of-Thought (CoT) prompting has the ability to elicit forward reasoning potential to LLMs, at inference time [72]. It has been shown that generating a critical path of inference steps has the potential to improve reasoning performance, and has triggered a lot of research in this direction [78]. It is interesting how off-the-shelf models can improve their reasoning potential when presented, at inference time, with step-by-step answer examples that model the inferencing steps that should be followed. Beyond these results, it is shown in another work [38] that even simply adding a ”Let’s think step by step” phrase to the prompt is enough to ”unlock” models’ reasoning potential, even without further fine-tuning.

In [57] the authors are creating a dataset of similar format to our own, but with a focus in narrower expressions, following the paradigm set by [65] and [23], and evaluate the performance of generative models that utilize CoT in proof generation instead of focusing on the label itself. They show that CoT-prompted generative LLMs are relatively trustworthy in outputting inference steps but are still weak in planning a proof, which includes using the correct proof step from the available ones or forming valid proofs instead of incomplete ones. Their main conclusion was rooted in observing that true but misleading statements can be used in the proof, but ultimately lead to a wrong answer. To evaluate the proofs generated by the models, they use the same tooling used for generating training data to parse predictions into symbolic representations that can be algorithmically evaluated.

Reasons why systems like the one presented in [12], or chain-of-thought prompting, work better than available alternatives, were to some extent described in the work presented in [48]. In their experiments, they emulated artificial locality structures through sampling of random Bayes networks and focused on the ability of a generative model to estimate conditional probabilities given these structures, with or without intermediate steps. The authors showed that the reason chains-of-thoughts are actually working as well as they do, is because models are inadvertently relying on strong local structural dependencies among the queried concepts that were available during training; if those are missing then associating the ”previously seen” local dependencies incrementally until the query can be fulfilled becomes the only faithful option.

Finally, it is important to underline the line of conclusions from multiple researchers which indicate that exposing language models to the task of reasoning is highly susceptible to misleading results due to overfitting on underlying dataset artifacts. In the work presented in [84], where logical templates were also transferred to natural language, but in this instance with a human-in-the-loop translation, it was shown that in a commonsense setup RoBERTa cannot cope adequately to linguistic variations, showing a clear bias towards phrasing structures instead of logical interconnections, underperforming in theorem proving. A similar conclusion was also drawn in previously mentioned [23] work, that when it comes to more diverse natural language the performance degrades severely, and that LLM with few-shot prompting only performs slightly better than random results.

In another work in [31] the authors also identify that their models, focused on the Multi-hop QA task, also tend to overfit on dataset shortcuts instead of emulating proper reasoning. To prove these limitations, the authors introduced adversarial contexts in their data, which managed to confuse the model mostly when the identified shortcuts were used. The additional contexts, that contained a fake answer, satisfied the reasoning shortcut but didn’t contradict the real answer.

The work in [81] is a paramount publication in this direction, where the authors note that the reasoning task in general contains a plethora of statistical features that end up working as inherent biases, and do not enable true solvers of the tasks as opposed to other NLP tasks, while also their removal from the data is in itself a challenging task. In their experiments with BERT [16], creating propositional logic datapoints without meaningful semantics (just like [11]), they show that BERT models trained and then tested on datasets formed with different sampling algorithms respectively, are failing to generalize. In their case, they showed that not only natural language templates play the role of a possible overfitting cause, but also the knowledge base size and forming method themselves, giving rise to unexpected statistical features like for example the number of rules or number of predicates in the rules, as indirect ”indicators” of label values. Their conclusions seem to be that true reasoning emulation by language models based on big corpora may be an ill-defined problem with inherent roadblocks, thus difficult to be provably achieved.

In another work in [59] they also examine if there is a reasoning potential in digesting fragments of natural language. Interestingly in this work it is clearly stated that semantics in such datasets, that focus in formal reasoning, are not and should not be affected by vocabulary semantics, given that the skill that needs to be cultivated of the model is that of logical inferencing with no extent towards commonsense knowledge or understanding. Having said that, an evaluation of this insight is not provided, and given the previously cited frailty of the task, it is rendered questionable. In their experiments, expanding the expressiveness of language used for their data compared to previous works, they also feature existential and universal operators other than negation in the task of soft satisfiability proving. But note that disjunction is not used, as the modeling of the fragments is founded in a more basic level of natural language grammar rather than specific propositional logic. In the case of conjunction, the logical principles behind it are still covered through alternatively phrased sentences that doesn’t directly use the word *”and”*. In their experiments that included RoBERTa [43], their results do not match with near-perfect scores achieved in-distributions in other works, but it is also noted that there was space for more elaborate training. Even so, there is a clear trend illustrated, that models fail to generalize their supposedly learned reasoning skills, when the test set is generated by a different algorithm than the train set. A finding that is very prominent even when training on harder reasoning problems and testing on easier ones, and ultimately leads to their main conclusion being that reasoning potential of LLMs is not fulfilled or proven yet.

In the work presented in [18] the authors present a generative framework of creating adversarial attacks for logical reasoning models, and further illustrate the extent to which these models are not robust reasoning emulators. In their work, they specifically focus on RuleTaker [11] data. They are preserving logical semantics of inputs instead of lexical ones, and include transformations like statement negation, statement elimination or substitution of equivalence rules with subsumption ones, with proper retaining or altering of the label. Detected examples include that the target model is often fooled if the query literal appears within the body of a rule, or the prediction is unchanged if the attacker eliminates a leaf fact on samples containing deeper proofs, which should change the label. Usage of quantified rules, like changing words like ”someone”, ”they” or individual names, also indicate points of struggle for the model, specifically in binding variables on either side of an implication. Finally, it is shown that the harder the example at hand is, the easier it is for the attack to successfully affect the target model.

In [55] a benchmark focused on logical reasoning is presented, applying transformations like contrastive changes that flip example labels or logical paraphrases that should not affect model predictions. In their results, they show that all evaluated models fail to pass the benchmark while at the same time achieving near-perfect results in in-distribution tests, and that larger models’ performance increase is limited. It is important to point out that the authors did take into account the existence of statistical features presented in [81] in order to minimize the information leakage in their evaluations, by properly adjusting their data sampling method for some example statistical features.

Additionally, to the aforementioned works, there is a line of research towards questioning the understanding of meaning and factual knowledge on a more fundamental level; see for example [36]. In this work, the possibility that models are focusing on statistical artifacts is brought to the forefront. In a broader recitation of identified limits for the transformer architecture, the authors of [26] mention that these models have been found to be potentially susceptible to conceptual negation, adversarial context (like in [31]) that take a big enough part of the input (also referred to as *mispriming*), pattern heuristics like lexical overlap or simple word matching, or simple word shuffling, if such cases are not properly represented during training or fine-tuning. As a possible instrumental limitation of the models is noted the lack of any architectural characteristic to target symbolic handling, which is often considered essential for reasoning. They clearly state that underlying pattern identification in datasets is a primary factor utilized by the model and when such patterns are missing or adversarial examples are introduced, performance is consistently subpar. Finally, they are referring to further studies which show that there are theoretical limitations grounded in linguistics with acute effects in reasoning. In general, insight into the brittle capacity for deep language understanding can also be seen in the light of works like [72] or [38]. They illustrate that reasoning experience to some extent seemingly encoded in the models, can be used only if ”accessed” by specific prompt techniques, and not automatically triggered from more general inputs; of which a more precise understanding, would immediately identify them as queries that require reasoning.

Overall, the issue of models overfitting on dataset characteristics rather than learning the reasoning task at hand is repeatedly observed in current research [78], with the best available approach detected to overcome the observed pitfalls, being the introduction of mechanisms that utilize explanations and reasoning steps. Having the potential to anchor the predictions of the models to a step-by-step process leads to more faithful results. In general, taking into account all the presented publications and more specifically the work of [81], and combining it with the insights of [48], seems to lead a path of discovery. As mentioned in [48], frequent co-occurence of information, be it human experience or model training data, is adequate to estimate the effect of their interleaving by simple statistical models. This in itself admits that statistical features can play a central role in the use of language. The extent to which this claim can be verified would be of paramount importance to the modeling of the NLR task in the context of Deep Learning, on a fundamental level. If local structures are innate characteristics used by models to generally learn NLP tasks or humans to associate information and beliefs [15], a crucial skill that is used in reasoning, then the question is raised how could researchers best represent training data that target reasoning, organize the training process and evaluate the prediction result; also, to what extent should statistical interdependencies be identified, how could the NLR task modeling be dissociated from them, and to what extent they should be excluded or utilized.

# 4. GENERATION OF THE QA DATASET

## 4.1 Dataset Generation Pipeline

The generation pipeline for our synthetic data was heavily influenced by the pipeline used in [11] and its design, as their code[1] was the basis for our work. In this implementation[2], each datapoint of the dataset (often mentioned simply as *example*) contains three main parts: a knowledge base (context), a query statement (question) and a label (answer), indicating whether the query statement is entailed by the knowledge base itself.

The implemented code outputs other fields for each example as well, but they are unimportant for the specific experiments we are focusing on. They are provided as secondary information, to enable additional conclusions or experiments if one so chooses. Although, the one additional field that is utilized for the formation of datasets is the explanation depth for each generated example. As we describe in the Experiments, the datapoints were clustered in different datasets according to their explanation depth, in order to perform in- and out-of-distribution tests.

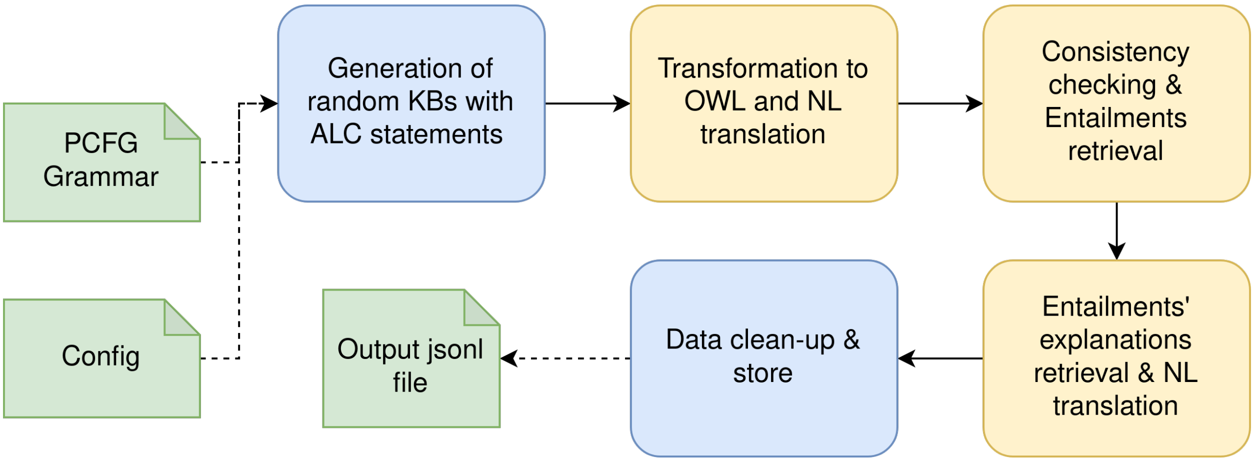



**Figure 4.1: A high-level illustration of the pipeline for the generation of the dataset. Green boxes illustrate static files, Blue boxes the Python part of the implementation and Yellow boxes the Java part of it.**

The definition of the explanation depth relies on the definition of explanations from section 2. The explanation depth is the size of the explanation subtracted by 1, to model reasoning steps; an example with a query that exists in the knowledge base itself, or needs only one knowledge base statement to support it, would have explanation depth 0, an example with a query that needs two unique knowledge base statements to be derived would have explanation depth 1 etc. Specific examples are presented in tables A.1 - A.5 in Appendix A.2. This is an important distinction between our datasets and the datasets presented in [11], where the depths are specifically defined based on SDD tree proofs [14].

[1] https://github.com/allenai/ruletaker
[2] Can be found in: https://github.com/Chuhtra/ALCRuler-DSIT

This difference should be kept in mind in the following sections, as a mapping between the two explanation methods was not investigated.

## 4.2 Generation of Formal Knowledge Bases

### 4.2.1 Grammar

In the core of the generation script is the definition of a formal Grammar as one of the inputs to the pipeline, following the design decision in [11]. It is through this grammar (presented in figure 4.2) that we can ensure that the pipeline generates statements that are consistent with the syntax of the $\mathcal{ALC}$ description logic, or even fragments of it. To enable a more direct comparison with [11], only the terminal symbols used in the authors' grammar already are utilized, effectively keeping the knowledge base vocabulary the same with the one used in RuleTaker.

Our formed grammar has two key characteristics. The first one, that could be considered as limitation, is the enforcement of the recursion of each statement regarding the level of $\mathcal{ALC}$ concept descriptions, to only one level. This is empirically chosen with the purpose of generating $\mathcal{ALC}$ statements that are not too complicated and are therefore as realistic as possible to mimic how humans express themselves. If this restriction is not in place, then statements like this are possible: '*only chases (only needs (only needs not (dog)) or T) (GIANNIS)*', which would be translated to "*Giannis is someone that only chases somebody that is a Thing or that only needs a person that only needs one that is not a dog.*", which of course does not constitute an example of commonly used natural language despite its semantics. While these are within the limits of $\mathcal{ALC}$, they are not practical, human-like data, which is our goal. To generate examples of $\mathcal{ALC}$ statements with minimum recursion depth, a $Concept$ can be generated only as an atomic one by using $AtomicConcept$. By using the rule for the non-terminal symbol $ConceptDescription\_1$ we can add one step in the concept recursion of the $\mathcal{ALC}$ statements.

The second characteristic is the fact that this grammar is using probabilities to ground non-terminal symbols, like in [11], which are predefined and tunable, but as a static parametrization have an effect on the final corpus.

### 4.2.2 From PCFG recipe to consistent OWL Knowledge Bases

We can now describe the pipeline that generates a datapoint, which starts with the use of the grammar to generate a random knowledge base. A base is formed by selecting a random number of TBox and ABox statements (within a configured range, in our case $[4, 18]$ and $[1, 16]$ respectively), and generating each statement at random, taking also into account biasing factors described later in section 4.4. The generation is driven by the probabilities that are present in the defined PCFG, which is explored recursively. Examples of this process can be seen in section A.1 of the Appendix.

With a set of randomly generated statements forming a knowledge base, the next step is

```
Statement -> AboxRule | TboxRule
TboxRule -> Concept Axiom Concept
AboxRule -> Role '(' Individual Individual ')' | Concept '(' Individual ')'
Concept -> AtomicConcept [0.4] | ConceptDescription_1 [0.6]
ConceptDescription_1 -> '(' AtomicConcept Junction AtomicConcept ')'
                        | '(' Role Restriction AtomicConcept ')'
AtomicConcept -> '(' Polarity 'Thing' ')' [0.05] | '(' Polarity Concept_N ')' [0.95]
Role -> Role_N
Individual -> Individual_N
Concept_N ->  "cat" | "dog" | "bald eagle" | "rabbit" | "mouse" | "tiger" | "lion" |
          "bear" | "squirrel" | "cow" | "red" | "blue" | "green" | "kind" | "nice" |
          "big" | "cold" | "young" | "round" | "rough" | "white" | "smart" |
          "quiet" | "furry"
Role_N -> 'likes' | 'chases' | 'eats' | 'sees' | 'visits' | 'needs'
Individual_N -> 'ANNE' | 'BOB' | 'CHARLIE' | 'DAVE' | 'ERIN' |
                'FIONA' | 'GARY' | 'HARRY'
Polarity -> Positive [0.5] | Negative [0.5]
Positive -> '+'
Negative -> '-'
Junction -> Disjunction [0.3] | Conjunction [0.7]
Disjunction -> 'or'
Conjunction -> 'and'
Axiom -> Definition [0.5] | Subsumption [0.5]
Definition -> '='
Subsumption -> '_<'
Restriction -> Universal [0.5] | Existential [0.5]
Universal -> 'only'
Existential -> 'some'
```

**Figure 4.2: The PCFG that was used for the dataset generation. Referring to the Experiments section 5.1.2, the entire grammar creates the main ALCRuler datasets (*MRD1* in our experiments). With blue color, we denote the part that if removed we can use the grammar to generate the ALCRuler version of the datasets that omit the universal quantifier (denoted as *ALCRuler-Fragment2* in our experiments). By additionally removing the orange part, we omit the existential quantifier as well (forming the *ALCRuler-Fragment1* version). If on top of these changes we remove the green part, we can omit the OR operator (forming the *ALCRuler-Fragment0*version). Finally, by removing the purple part, we can omit all complex concepts (which forms the *ALCRuler_MRD0* of our experiments).**

to provide this set to a Java process that utilizes the OWL API [27] to parse the statements into an OWL ontology; a necessary step in order to perform reasoning[3] later. This transition from a PCFG-derived syntax to OWL syntax is facilitated by parsing each part of the input statement into a binary tree data structure, with each node aiming to hold a label or also an OWL object. The generated tree is immediately parsed in a Depth-first manner (DFS), in order for each part of the statement to be matched with the proper OWL expression initialization, and ultimately build-up OWL axioms, by combining the OWL expressions if needed. At the final stage, all statements are translated to OWL syntax (see example in figure 4.3) and added to an OWL ontology. This formed ontology, other than the axioms themselves, only contains the declaration axioms which define the names of the OWL objects (classes, object properties and named individuals), as well as a forced disjointedness among all available named individuals, following the unique name assumption. This way we increase clarity by applying meaningful distinctions and reducing ambiguous and inefficient interpretations, while also allowing free concept associations through terminological axioms.

With a fully formed OWL ontology loaded, we continue by verifying whether the ontology is consistent. While this is automatically done through the OWL API, it should be noted that there is a special handling in the case of inconsistent ontologies. In such cases, the first step is to extract the explanation of the inconsistency itself by providing the inconsistent axiom $\mathrm{Thing\ subclassOf\ Nothing}$ as query to the $\mathrm{owlexplanation}$ package [29]. The second step, is to iteratively remove each axiom present in the inconsistency's explanation until a consistent ontology is present. The size of our knowledge bases is small enough that removing more than one such axiom isn't necessary. Having retrieved a consistent ontology, we continue on, while the retrieved explanation can be utilized to support the removed axiom as a negatively interpreted query for the remaining knowledge base.

### 4.2.3 Entailment retrieval

Having generated a knowledge base that will serve as the *context* for our Transformer model, we subsequently need to generate some assertion that will work as the *query*. After ensuring that the knowledge base itself is passing a consistency check, we are generating the *inferred closure* of our knowledge base. The above steps are done using the $\mathrm{HermiT}$ reasoner [20].

In this implementation despite working under the Open-World Assumption, as we are focusing on a description logic, the query generation in its core relies on Inference Closure, and thus the query label is never unknown. The axiom generators of the OWL API that are used include subclass axioms, class assertions, disjoint class axioms and equivalence axioms. Having said that, we exclude from later utilization some entailments like subclass tautologies, assertion-as or subclass-of Thing, and also having Nothing as a subclass or

[3]Initially, we tried using the class-oriented Owlready2 [41] package, but it had limitations that made it unfeasible. Despite being the most feature-rich compared to alternatives, able to create TBox and ABox statements, check consistency, and derive entailments, it couldn't handle non-atomic LHS concepts, retrieve explanations or access axioms directly. As a result, we had to switch from a Python-only approach.

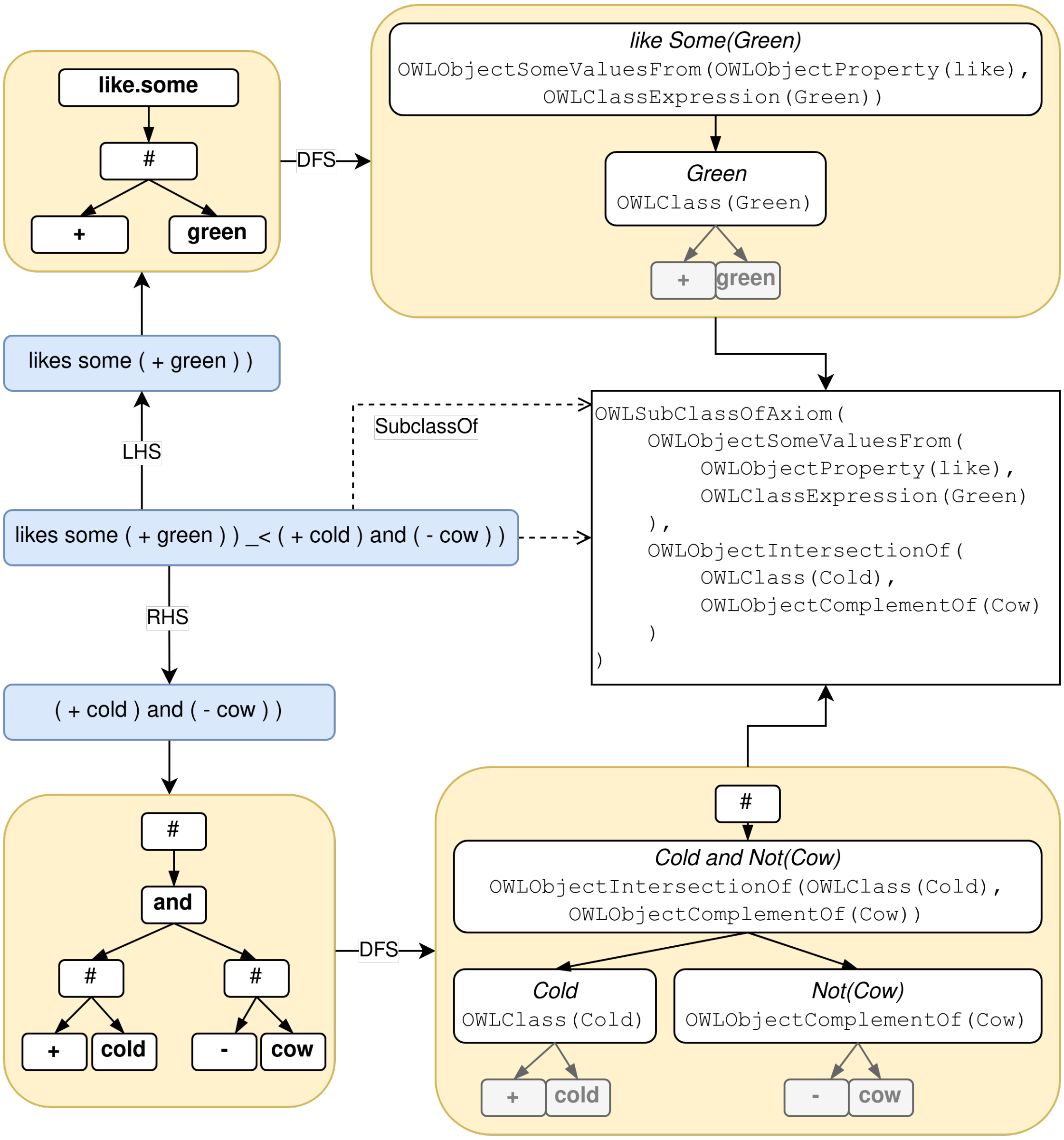



**Figure 4.3: A (simplified) example of transitioning from PCFG notation to OWL syntax, for a TBox axiom.**

disjoint class. The primary reasons being that statements like these would either flood our data pipeline with similar and trivial statements (i.e. most classes are subclasses of Thing), or lead to runtime errors.

After the entailments provided by the reasoner are retrieved, their utilization of the inferred axioms consists of two different approaches. In the first and simplest approach, the entailments themselves are used directly as queries, with the matching explanations being derived by the $\mathrm{owlexplanation}$ package [29]. In the second approach, a series of algorithmic combinations of entailments and knowledge base axioms is providing additional queries. The primary reason the second approach is necessary is the fact that the entailments, retrieved with the reasoner alone, are exclusively covering non-negated and non-negative statements. This is the case as the reasoner is purposefully limiting its search scope (e.g. to named classes), since the entailments can be in fact infinite [3] without such precautions in place. As an example, given that $A \sqsubseteq \neg B$ and $A(John)$, the reasoner would not infer that $\neg B(John)$, without being configured with a specific implementation to do so. While there could be a limiting criterion, specific to class complements, to include a finite subset of class expressions in the search space of the reasoner, one is not provided by the available reasoners and such an implementation was not chosen over post-inference algorithmic axiom combinations. In the following part of this section, we will describe the processes that shape the aforementioned approaches of entailment retrieval.

Central role in the materialization of an entailment, or a combination of them, as a query statement, play a set of per-query attributes that in the context of our implementation are collectively referred to as the query's *interpretation*. The main attributes that form the interpretation are the *label*, which is nothing more than the label that each datapoint will have and indicates whether the query holds true against the context, and the *tone*, which enables the variability of expressions in both false and true labels, through the use of negative language. See table 4.1 for a short example that makes the relation of these two attributes clear. These two attributes are in principle randomly set with a value per query statement, in order to ensure a balanced representation, but subsequent reassignments may happen in order to always ensure truthful interpretations with respect to the knowledge base at hand. For the reasoner's entailments, a non-truthful interpretation consists of a base tone with a false label or a negative tone with a true label. In other cases, interpretation's truthfulness depends on the context-query content.

**Table 4.1: Short example to clarify the relation between query *tone* and *label* in the dataset generation process. Whether the interpretation holds or not, is a direct consequence of our contextual knowledge. In this example, that of the physical world.**

| statement | tone | label | interpretation holds? |
|---|---|---|---|
| The Earth is flat. | positive (base) | true | ✕ |
| | | false | ✔ |
| The Earth is not flat. | negative | false | ✕ |
| | | true | ✔ |

#### 4.2.3.1 Entailments retrieved from Reasoner

The handling of entailments that belong to the inferred closure, is different depending on the type of axiom. For equivalence or subclass axioms, the entailment and explanations are immediately used as-is when the randomly set interpretation is truthful. Otherwise, we attempt a statement transformation, by negating the RHS for the query and adding an artificial assertion statement to the explanation set. This assertion contains the LHS and an available named individual, in order to ground the LHS and ensure that it is not equivalent to $\mathrm{Nothing}$. By transforming a true statement, into an untrue one, the non-truthful interpretation becomes valid. For example, given an axiom $A \sqsubseteq B$, we can create another axiom $A(John)$, so that $A \sqsubseteq \neg B$ can never hold. By then properly manipulating its interpretation, we create a query by utilizing the explanation that justified the initial subclass axiom. If such an artificial assertion is not attainable, then the interpretation is randomly flipped to become truthful.

For disjointness axioms, a simple positive interpretation (base, true) is selected as no transformation can be done in this stage. For assertion axioms, first we attempt to utilize some disjointness axiom, by matching its classes with the assertion axiom's class. This way, the inferred assertion axiom can be flipped by replacing the participating class, allowing us to form a query that contains negation. For example, given $\mathrm{A\ disjointWith\ B}$ and $A(John)$, we can use $B(John)$ using a proper interpretation. When this fails, the assertion axiom is used as is with a randomly selected truthful interpretation.

The two transformations described above already share a common detail that constitutes a crucial and repeating element for the entire implementation. Due to the aforementioned limitation of the reasoner, being unable to provide negated or negative statements, our data generation is immediately set up with an imbalanced representation of labels and tones. And for this reason, even for the entailments retrieved from the reasoner, a clear bias is purposefully implemented to first explore the capacity to acquire a transformed interpretation which contains negation.

#### 4.2.3.2 Transitive Closures

Following up the reasoner-retrieved entailment handling, the second part of algorithmic entailment generation is implemented, which starts with the generation of transitive closures. The reason to set this as the first step is the role subclass axioms can play for the further creation of disjointness axioms, which in turn play a primary role in the combination of all kinds of axioms into new ones, with manipulated interpretations.

Firstly, as the reasoner does not output transitive closures, they are created by recursively combining subclass axioms of the asserted ontology and its inferred closure. In these cases, the chain of subclass axioms, and alternatively the explanations that justify the axiom in case of using inferred ones, form the unified explanation of the derived transitive closure query. The transformation initially described in section 4.2.3.1 is also attempted here.

Subsequently, the generated transitive closures are utilized in an effort to get additional untrue axiom queries. We are initially matching the RHS of a (transitive) subclass axiom, with the available disjointness axioms to derive previously unseen disjointness axioms. Then these are used to form more class assertion axioms and subclass axioms. In the first case, given an assertion axiom with the LHS of the transitive subclass axiom as a participating class, a new class assertion is created using the aforementioned RHS. Similarly, for the second case, we create a new subclass axiom with the LHS of the transitive subclass axiom and the aforementioned RHS's disjoint counterpart. In some cases, this axiom will have been already generated, so we just enhance its collection of explanations instead. As an example, see table 4.2.

**Table 4.2: Example presenting a subset of entailments that can be generated by combining a Disjointness axiom and a Transitive Closure.**

| | | |
|---|---|---|
| Given | $\mathrm{D\ disjointWith\ C}$ | (1) |
| | $A \sqsubseteq B$ | (2) |
| | $B \sqsubseteq C$ | (3) |
| | $A(John)$ | (4) |
| Inferred | $A \sqsubseteq C$ | (2,3) |
| | $\mathrm{A\ disjointWith\ D}$ | (1,2,3) |
| | $C(John)$ | (2,3,4) |
| | $\neg D(John)$ | (1,2,3,4) |

#### 4.2.3.3 Additional methods

Having the reasoner-retrieved entailments and their explanations, we implement some additional combination methods. For equivalence or subclass entailments, we create untrue disjointness axioms by simply changing the axiom type of truthful entailments. While this creates an abundance of untrue disjointness axioms, greatly affecting the output balance of a raw dataset generation run, it is in fact necessary as it is the main point of generating any such type of query. The opposite is also implemented, when the entailment is a disjointness axiom, we can use it to generate untrue equivalence or subclass axioms by replacing the axiom type of truthful entailments. We can additionally generate class assertions, by matching the participating class of all already available class assertions of the asserted ontologies and its inferred closures with the class expressions of the entailed disjointness axioms; effectively replacing the participating class and flipping the interpretation, similarly to the previous sections.

Furthermore, two additional query forming methods have been implemented, that utilize unsatisfiable classes and single entailments respectively. For the first, all unsatisfiable classes of the ontology are collected through the OWL API and for each one the explanation of the axiom $\mathrm{UnsatisfiableClass\ subclassOf\ Nothing}$ is retrieved. An artificial class assertion is then created with an available named individual, which is used as an untrue query that is supported by the retrieved explanation. The second case is implemented in an effort to include in the generated datasets the simple-lookup case. A random set of axioms of the asserted ontology is selected, with the selected axioms being used themselves as queries and also explanations.

## 4.3 Transition to Natural Language

To acquire the Natural Language (NL) translations, a second program is used, again implemented in Java. It is used for each knowledge base statement that is generated (i.e. before the pipeline moves to the steps described in section 4.2.2 and onwards), and for each entailment that is found by the reasoner with its explanations(i.e. it is integrated in the entailment retrieval). The statement in PCFG syntax is provided to the Java process that is parsing it into OWL API syntax, similarly to the way the knowledge base statements are during the inference closure phase, and then the OWL axiom is recursively parsed into natural language using predefined templates (see also table 4.3).

Additionally, another point with respect to the use of NL templates is the handling of articles (a/an) for nouns and their absence for adjectives. The simplest and most reliable solution formed into not providing the names directly in the grammar (like shown in figure 4.2), but have a placeholder and retrieve them from a secondary file, in which the two types of names are distinctly stored. This is equivalent to having a grammar rule, like *Concept_N -> Adjective* | *Noun*, which albeit cleaner, it was avoided due to the simplification of other implementation details, as these lists are used outside the context of the PCFG as well. Finally, some other details regarding natural language translations can be found in section 4.4.

An alternative overall translation approach would be to use the NaturalOWL package [34] but through some trial experiments it was found to not be ideal for our $\mathcal{ALC}$ statements with complex descriptions in both sides, as its translations revolved around specific entities. Having said that, the use of NaturalOWL's "Syntax files" might resolve this issue, but it was not explored at this time.

**Table 4.3: Illustration of natural language phrases that are matched with OWL notions; used in dataset generation. The asterisk (*) indicates some variability in the template, see section 4.4.**

| OWL notion | NL template |
|---|---|
| SubclassOf | is (not) also |
| EquivalentClasses | equivalently is (not) |
| DisjointClasses | can/cannot also be |
| ObjectIntersection | and |
| ObjectUnion | or |
| OWLObjectAllValuesFrom | only somebody who* is (not) |
| OWLObjectSomeValuesFrom | somebody who* is (not) |
| ClassAssertion | is (not) |

## 4.4 Generation heuristics to ensure quality and performance

To finalize the data generation pipeline, we return to the main Python process which cleans the generated data, stores any artificial axiom that was created, removes any necessary statement in the context of resolving found inconsistencies and finally, selects one explanation among the possibly multiple per datapoint as the prime explanation, or discards entire datapoints if necessary. The details and motivation regarding most of these processes as well as details with respect to the prior steps of the pipeline are described in the following sections.

### Grammar biasing

To ensure that the statements within a single knowledge base will be relevant to each other, the authors in [11] are selecting a random subset of vocabulary names per knowledge base. As an alternative to this approach, in our implementation a grammar biasing mechanism is introduced. Instead of selecting a predefined random subset of vocabulary names each time, this approach is biasing the predefined probabilities of the (vocabulary) terminal symbols of the PCFG with each selection by a small factor. For example, for a PCFG rule $\mathrm{Concept\_N} \rightarrow \text{"cat"} \mid \text{"dog"}$, which states that "cat" is equally probable (50-50%) to be selected with "dog", if the first statement of a knowledge base is using the name "dog", then for the following statement generation the grammar rule will be altered to be $\mathrm{Concept\_N} \rightarrow\ \text{"cat"}\ [0.4] \mid \text{"dog"}\ [0.6]$ etc. In this example, the biasing factor was 10%. A benefit of this approach over getting repeated samples, is both that it introduces a parameter that allows dynamic control, and that due to its probabilistic nature its effect can be variable.

Having said that, experimental runs showed that the majority of the generated datapoints required little to no grammar biasing, which questions the need for such relevance-focused measures altogether. Distribution sets that were formed from the initial phases of the script development had shown most datapoints ($94$%) don't "need" a biasing factor of above $0.3$, with $75$% of the points having used a factor of just $0.19$. In our final implementation, the updating bias for the grammar is selected at random for each knowledge base in an interval that can be defined in a configuration file, and was set to be $[0.001, 0.2]$. These values were observed to be adequate for all explanation depths (i.e. the entire range of our dataset). Additionally, to avoid numerical instabilities the minimum probability allowed through this biasing was set to $0.01$, a parameter that is also configurable.

Moreover, in an effort to avoid sentences like *(((likes only(+ Cow)) and ((- Cow) and (+ Cow))) 'DAVID')* a mechanism similar to the grammar biasing is introduced, targeting intra-statement concept repetition. In this case, when a vocabulary name is used at some point of a statement, then the probability in the PCFG for this specific name and for this specific statement expression is brought directly to zero.

### Explanation ranking

Several methods of explanation ranking were implemented in efforts to reduce the need for repeated runs of the script in order to acquire balanced data, with the most promising among them being selecting the maximum depth explanation for each example, as the deeper explanations are also the more rare ones. Having said that, we ended up using the one with the least time-savings due to concerns of the model (or our results) being "confused" when non-obvious explanation paths are used, especially given that for our experiments the models don't have access to the explanations themselves.

For example, if a query can be explained with both a single statement and a set of 3 statements, handling it as a d-2 example in our dataset formation methodology instead of a d-0, could only mean that our d-2 dataset would be of lower quality. Since the model is not directly provided with what the explanation depth means, had it achieved a good understanding of handling d-0 examples, then our d-2 example would be handled as a d-0 too. This effectively means that a dataset that in our perspective is balanced with X number of examples per depth, in the perspective of the model can be considered imbalanced with disproportionately large number of d-0 examples. For this reason, the explanation with the minimum depth is ranked first in our datasets, but it remains an interesting path for future work if the deeper explanations can be of use to the model, given that it has access to them during training.

### TBox statement chaining

Another effort to improve the efficiency of using the dataset generation script, for dataset output balance with respect to the explanation depths, is *chaining*. Through chaining, the direct reuse of vocabulary names between knowledge base statements is purposefully added to the script. In practice, this means that a name or expression that was randomly chosen for a statement $n$, will be stored and reused also for the generation of a statement $n + 1$, in an effort to produce chains of reasoning connections that will lead to deeper explanations more easily. The nature of this idea relies on the chaining of statements, so it can primarily be used for binary axioms which are part of the TBox. While it could also be used for ABox statements, this was not included, as it was considered a better approach to let the accumulated grammar biasing, that is still active underneath, bring the ABox assertions closer to the chained TBox axioms.

There are two primary methods implemented for chaining. Firstly, *full-side chaining*, in which the entire RHS is copied and used as the next TBox axiom's LHS, was initially used due to its ability to bring tangible improvements to the generation runtime and result. Having said that, it was later replaced due to fears that it would lead to a very monotonic pattern in the data that could be easily detectable by the models. As a replacement, we ended up using *atomic chaining*, in which the LHS of the current TBox statement uses atomic concepts taken from the pool of concepts that is formed from the previous statement's RHS's atomic concepts. But not all atomic concepts are chained, to allow some entropy. This ended up being used as a middle ground that can bring observable benefits to the script performance while also allowing adequately dynamic data formation.

### Transformer tokenization limit

Each knowledge base that is generated is ensuring that statement-by-statement, post-translation, the tokenization limit of our target Transformer model (set to $512$) isn't reached. This way, we are sure that no datapoint will require pruning to be used as input for the final model. This way the dataset balance is preserved and natural limits to the amount of knowledge base statements per datapoint are set.

### Handling of natural language translation templates

As already discussed, a major aspect in the creation of the dataset was the aim to mimic naturally sounding statements. A clear obstacle in this aim comes from the use of pre-defined templates (table 4.3) for translation to natural language. For a human reader, the statements were immediately recognizable as artificial due to the repetitive use of the phrase ”somebody who”, used to connect a main and a subordinate clause.

As a first move towards mitigating this, a paraphrasing step was added in the natural language template utilization. With this, each such phrase is not always ”somebody who” but it's pseudorandomly chosen as a combination of these two sets:

- (”someone”, ”somebody”, ”a person”, ”one”).
- (”who”, ”that”)

A second step towards this direction was the introduction of a metric, generally referred to as ”somebodies' metric” ($SbM$). For a single statement, it is calculated as:

$$SbM = (1 - \frac{x}{y}) - (\frac{y-k}{y}) = \frac{k-x}{y} \tag{4.1}$$

In equation 4.1, $y$ is the number of words from set $A$ that are present in the statement, where $A = \{\text{someone, somebody, a person, one}\}$. $x$ is the maximum number of duplicates, for some word among the $y$, (the first occurrence of the word doesn't count). $k$ is a pre-defined limit of words from $A$, that can exist in a ”natural” sentence (default $k = 3$). In the context of our generation script, $SbM$ is calculated for each statement in a knowledge base+query pair, and the $\text{minimum}$ value represents the $SbM$ value of the pair, indicating whether a statement will be part of the pair, and if a pair will be used for a datapoint overall.

A minimum of $SbM = 0.6$ is allowed as, empirically, statements with lower values tend to appear not-realistic enough. Examples of that can be seen in the Appendix in table A.6.

The formulation of this metric rests on the motivation to quantify two specific characteristics for a statement. One, targeted by the first term, repetitions of one word from set $A$ (e.g. repetitions of ”somebody”) should be punished. Two, targeted by the second term, use of too many words from set $A$ in general should be punished. The first term represents the ratio of the most observed repetitions (of 1 word) over the number of such words, and we are subtracting from 1 as we want this to be a maximization metric. The second term represents the number of ”extra” words from $A$ that have been used through the subtracting, while the division by $y$ is a normalization factor. The second term is subtracted, as the bigger the second term, the smaller the metric’s score should be. Finally, regarding the first term, the first occurrence isn’t counted for $x$ because this way $\frac{1}{1}$ (a good situation) makes this term the same with $\frac{5}{5}$ (a bad situation). The second term somewhat remedies that, but as the behavior isn’t otherwise affected, the first occurrence is chosen to remain uncounted. In table 4.4 two examples are shown. We see how a single word can reduce the $SbM$ value of the statement.

**Table 4.4: First example of calculating ”somebodies metric”.**

| |
|---|
| **Somebody** who loves **somebody** that needs **someone** that finds **a person** that runs, is blind. |
| $y = 4, x = 1, k = 3, SbM = (3 - 1)/4 = 0.5$ |
| **Somebody** who loves **somebody** that needs **somebody** that finds **a person** that runs, is blind. |
| $y = 4, x = 2, k = 3, SbM = (3 - 2)/4 = 0.25$ |

# 5. EXPERIMENTS

## 5.1 Fine-tuning configuration

### 5.1.1 The models

For fine-tuning, Lambda Labs[1] machines were used with single A100 (40 GB SXM4) GPUs. The language model used for all runs is RoBERTa-large [43], for binary (true/false) classification. We are using the $\mathrm{RobertaForSequenceClassification}$ package of hugging-face [75], which provides a classification head with $2$ linear layers, prepended with dropout layers ($prob = 0.1$) and separated by a $tanh$ activation function. Cross-Entropy loss is used during training and the chosen performance metric is accuracy.

In all cases, a linear learning-rate scheduler was used, as a variable learning rate is shown to improve loss-based training [60, 62], and the Adam optimizer [37] ($b1 = 0.9, b2 = 0.98, e = 1e - 6$), with a learning rate of $1e - 5$. Also, weight decay was set to $0.1$. Other training hyperparameters, that varied for all models, are presented in table 5.1. Altering the number of training epochs and the learning rate warm-up ratio was the main way to experiment with different training formulas, due to their direct effect on the progression of learning rate scheduler. Early-stopping was also used to avoid potential overfitting, while gradient accumulation was used in an effort to speed-up training. Gradient accumulation is tightly connected with the training batch size, since it effectively virtually increases it. Consequently, these values were chosen with the focus of best utilizing the available GPU memory. The overall selection of the ALCRuler models' hyperparameters was mainly driven by what the best performances were in the matching RuleTaker models, taking into account some complimentary experimentation (see table B.1 of the appendix). In figures 5.1, 5.2 and 5.3, the detailed progression of the training can be seen for each ALCRuler dataset, while matching plots can be found in Appendix B.2 for RuleTaker datasets.

[1] https://lambdalabs.com/

**Table 5.1: Training Hyperparameters that were used for each (final) model fine-tuning.**

| finetuning_dataset | train epochs | eval batch size | train batch size | learning rate warmup ratio | gradient accum. steps | Runtime |
|---|---|---|---|---|---|---|
| ALCRuler/depth-0 | 5 | 24 | 24 | 0.06 | 12 | 1h 53m |
| ALCRuler/depth-1 | 6 | 18 | 18 | 0.2 | 12 | 3h 23m |
| ALCRuler/depth-2 | 15 | 24 | 24 | 0.3 | 12 | 3h 57m |
| ALCRuler/depth-3 | 15 | 24 | 24 | 0.5 | 12 | 4h 14m |
| ALCRuler/depth-5 | 8 | 18 | 18 | 0.5 | 8 | 4h 13m |
| RuleTaker/depth-0 | 4 | 32 | 64 | 0.06 | 16 | 27m |
| RuleTaker/depth-1 | 4 | 16 | 16 | 0.06 | 16 | 48m |
| RuleTaker/depth-2 | 12 | 24 | 24 | 0.2 | 12 | 2h 9m |
| RuleTaker/depth-3 | 15 | 24 | 24 | 0.5 | 12 | 2h 56m |
| RuleTaker/depth-5 | 15 | 24 | 24 | 0.4 | 8 | 2h 50m |

### 5.1.2 The Data

Regarding the datasets used, these were created with the methodology described in section 4, following the data arrangement of [11]. Our main dataset collection, named ALCRuler_MRD1 (ALCRuler w/ complex concepts), consists of 5 different datasets, each one split in a $70/10/20$ train/dev/test split, and containing $100k$ examples. Each of the 5 is named as $d <= X$ where X can be in the range $[0, 1, 2, 3, 5]$ and denotes the maximum available explanation depth. All datasets are balanced with respect to the labels and to the explanation depths.

Additionally, to our main datasets, we created a few more datasets for secondary experiments. The first of those is ALCRuler_MRD0, which follows the same principle as ALCRuler_MRD1 but does not contain any complex concept (see more in figure 4.2). For this dataset, we only created test sets, as it was aimed for zero-shot tests only. Also, for both ALCRuler_MRD0 and ALCRuler_MRD1, we include in our test results another dataset (referred to as $d >= 0*$) which is close to the $d <= 5$ but has additionally examples with explanation depths higher than 5, which were created but not discarded during data generation.

Finally, a third family of datasets was created; the ALCRuler_Fragment[0,1,2]. These three datasets only have a $d <= 5$ version with the same train/dev/test splits and checked data balance as before, but are based on fragments of the original ALC grammar that was used for ALCRuler_MRD1. The grammar used for each one is described in figure 4.2 as well.

Lastly, it is important to clarify that since the queries are derived from the reasoner entailments or their combinations, there may be multiple ones that are using the same knowledge base as context. This was the case for the implementation of [11] as well. Like in this work, the context-query pairs are selected so that all datapoints which share a unique context are part of the same dataset, without any spillage from the train set to the development or test set. Datapoints are only repeated between different sets of the same type. For example, a d-0 datapoint can be part of the $d <= 0$ dataset, but also of the $d <= 1$, the $d <= 2$ etc. This reduced greatly the time required for the data generation, while not presenting any issue, since each dataset is used to fine-tune a different language model.

## 5.2 Exploration of Inference Depth

In tables 5.3 - 5.6 we can see the Accuracy performances of 10 language models fine-tuned on reasoning datasets. Tables 5.3 and 5.4 cover the performance of models fine-tuned on the RuleTaker datasets [11], while the tables 5.5 and 5.6 that of models fine-tuned on ALCRuler datasets. Each column identifies with the range of inference depths that were available in the test set (e.g. for $d = 0$ proof/explanation depths are all 0, for $d <= 1$ proof/explanation depths can be either 0 or 1 etc.). For the ALCRuler test sets, the rightmost column ($d >= 0*$) is provided as supplementary, as all inference depths that were available from the data generation were used, without discarding any in contrast

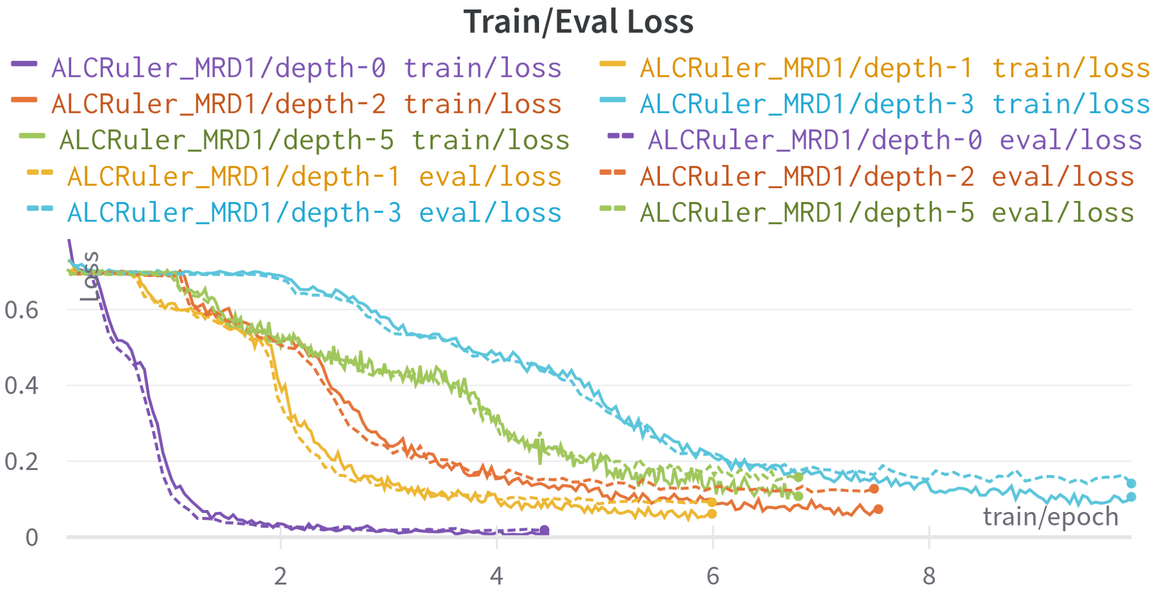



**Figure 5.1: Training/Validation Loss progression per training epoch for ALCRuler datasets.**

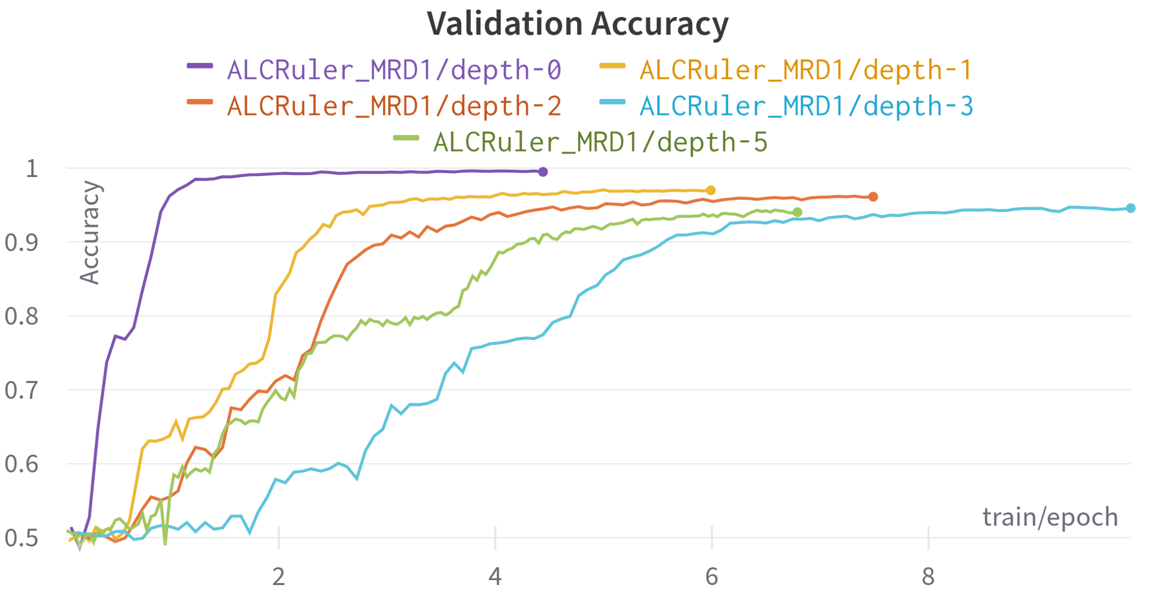



**Figure 5.2: Validation Accuracy progression per training epoch for ALCRuler datasets.**

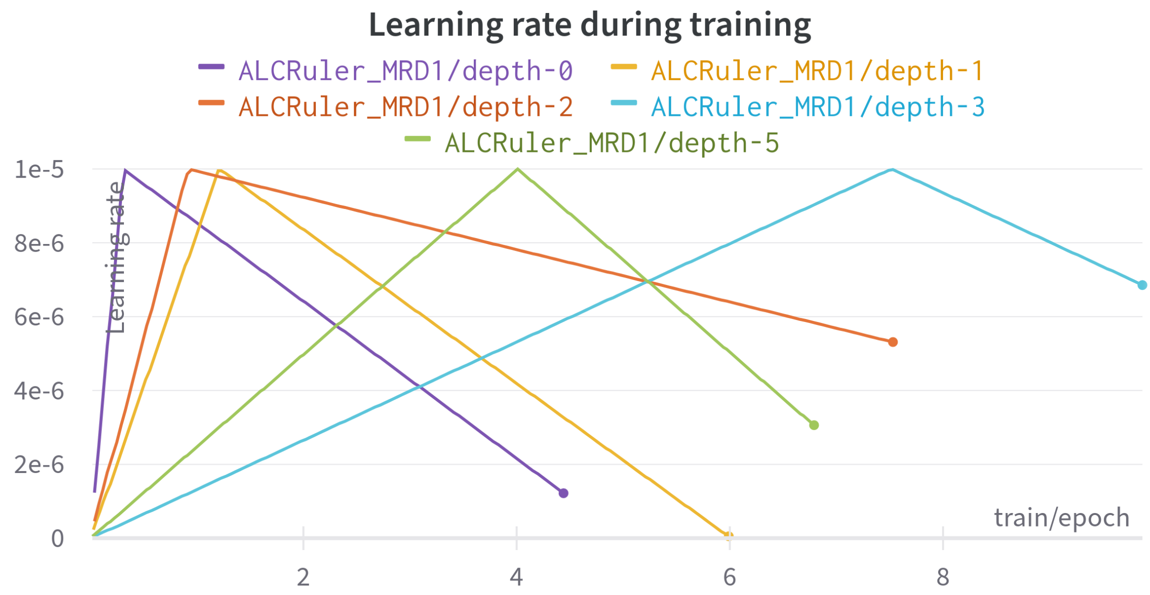



**Figure 5.3: Learning Rate progression per training epoch for ALCRuler datasets.**

to the previous columns. The green-background areas indicate test results on datasets including inference depths that were all represented in the fine-tune sets (In-Distribution tests), while the no-background areas show the results of tests that included higher inference depths which were not seen by the models during fine-tuning (Out-of-Distribution tests). The expectation for the in-distribution tests is that models exposed to high inference depths during fine-tuning will reach perfect accuracy for all test sets, while models exposed to only low inference depths will reach perfect accuracy only for low inference depth test sets.

### 5.2.1 Fine-tuning on RuleTaker

We observe that the results presented in [11], with respect to RuleTaker models' performance on their synthetic data, are reproducible. RoBERTa is able to achieve perfect or near perfect results in all In-Distribution tests, while having considerable gains in out-of-distribution tests as it is gradually exposed to higher inference depths during fine-tuning, already from the middle lines of table 5.3. Presented differences regarding higher out-of-distribution values compared to the results reported by the original authors, can be attributed to possible deviations in the training formulas between the original and the presented experiments or possible pipeline differences (e.g. use of 2 linear layers in the classification head instead of 1).

When the RuleTaker models are tasked to perform on the ALCRuler test sets (table 5.4), we see that high accuracies cannot be achieved. The performance is consistently higher when the ALCRuler sets don't include complex concepts (avg. difference of $7.2 \pm 1,9$%, over all measurements), and there is no indication that exposure to higher inference depths during fine-tuning or testing affects those results. Additionally, the average distance from the random $50$% boundary, over all measurements, is $12.17 \pm 2.3$% in the case of ALCRuler not containing any complex concept, but only $4.90 \pm 2.3$% when these are included. This hints to us that, despite the vastly lower performance, there are some underlying similarities between the two synthetic datasets that are detectable by the language model, and are also considerably more pronounced the closer the datasets get, by removing complex concepts, which RuleTaker datasets lack. Having said that, the performance of the RuleTaker-finetuned models was expected to be considerably higher in the ALCRuler datasets that do not include complex concepts, as the expressive complexity of these examples remains a subset of the one that the models were exposed to during fine-tuning.

### 5.2.2 Fine-tuning on ALCRuler

Moving on to models fine-tuned with ALCRuler, RoBERTa seems to not only being able to achieve high performance results, but also do that faster than with RuleTaker datasets. In these cases, results of over $90$% are already achievable from the first exposure to a single reasoning step, for in- or out-of-distribution tests. On top of that, we can see that already for the model fine-tuned with data of zero inference depth (table 5.5, first row) the out-of-distribution performances are comparatively higher than the matching values of the RuleTaker fine-tuning (table 5.3, first row). While it is not clear why this is the case, a possible theory is that even though all fine-tuning datapoints originate from single-statement explanations (where depth=0), not all datapoints are strictly simple look-ups in the knowledge base (see for example table 5.2). Due to this aspect of the dataset, it's possible that innate characteristics of $\mathcal{ALC}$ that directly correlate with reasoning, are already exposed to the models, but evade our sense of reasoning difficulty gradation, which is based on the previously defined explanation depth. When comparing the performance of the ALCRuler-finetuned models on the ALCRuler test sets with or without complex concepts, we see as expected comparable performances with slight deviations of $0.9 \pm 1.05$% over all measurements. This small difference seems to be of low statistical significance and probably attributed to datapoint representation differences on a set-by-set case, but it seems interesting that when complex concepts are removed the performance is consistently slightly inferior, despite the opposite would be expected.

**Table 5.2: Small set of datapoints where inference depth is 0, but explanation complexity is not a simple look-up.**

| query | explanation | label |
|---|---|---|
| One who is a lion, is not also a rabbit. | A person that is not a rabbit, is also a dog and not a lion. | false |
| DAVE is not a cow. | Somebody who is nothing, equivalently is a cow or not round. | true |
| A person who is smart, cannot also be a dog. | A person who is a thing, equivalently is not a dog. | true |
| One that is a lion, cannot also be round. | Somebody who is not a lion, equivalently is not round. | false |
| A person who is a lion, is also a bald eagle. | A person that is a lion and not a bald eagle, equivalently is kind and nothing. | true |

Finally, for the performance of ALCRuler-finetuned models on RuleTaker test sets, the picture we see is shifting negatively, as the complexity of the fine-tuning set is increasing. In table 5.6 we see that when RoBERTa is fine-tuned on an explanation depth=0 set, it can achieve a relatively high accuracy of $87.4$% on the proof depth=0 RuleTaker set. Moving to higher depths on the RuleTaker test sets, the performance does seem to diminish, but is still in high levels compared to the matching zero-shot results of the RuleTaker-finetuned model. This hints to us that the ALCRuler datapoints of low inference depths bear detectable similarities to the RuleTaker counterparts, and also that the ALCRuler-finetuned models exhibit higher adaptability to the unknown dataset (compared to the RuleTaker zero-shot results). Having said that, we also see that as the RoBERTa models' fine-tuning includes higher and higher inference depths the performance in the RuleTaker test sets is quickly dropping, to the point (in the final row) of barely overcoming the random $50$% boundary. The reason behind this progression seems to be that the increase in complexity is such that, acting as noise, does not allow the model to focus on the simpler patterns included in the zero depths only, the way it could do before.

There is an additional indication that the models (in the context of the used training formula), were not able to completely absorb the escalating dataset complexities in their weights, without losses. Despite achieving test accuracy results that match those of the validation phase in all cases, the results themselves don't meet up to the perfect or near-perfect in-distribution results of the RuleTaker models. On the contrary, only the simplest ALCRuler-finetuned model managed to achieve a near-perfect result in the same context, with the follow-up models reaching an accuracy maximum that was consistently diminished, down to $94$%. An alternative viewpoint of these results would suggest that the perfect or near-perfect results, observed in [11] and our initial runs, are not in fact indicators of high reasoning potential but instead artifacts of overfitting. Such a viewpoint would also be consistent with the diminishing of the test accuracy, as training data complexity increases, that was described in the previous paragraph.

**Table 5.3: Reproduction of main results from [11], testing in and out of distribution in RuleTaker datasets, the RuleTaker fine-tuned models. The green area shows the In-Distribution tests. *p_d* indicates the proof depths that are present in each set.**

| Accuracy per RuleTaker test set | | | | |
|---|---|---|---|---|
| **p_d=0** | **p_d<=1** | **p_d<=2** | **p_d<=3** | **p_d<=5** |
| 100.0 | 75.5 | 65.5 | 60.3 | 54.0 |
| 99.1 | 98.8 | 84.7 | 75.1 | 64.9 |
| 99.4 | 99.4 | 99.4 | 94.4 | 83.7 |
| 99.4 | 99.4 | 99.5 | 99.3 | 99.1 |
| 99.2 | 99.1 | 99.1 | 99.0 | 99.2 |

## 5.3 Exploration of expressive potential

In table 5.7 we explore the effect of exposing RoBERTa with $\mathcal{ALC}$ fragments [2] instead of the original ALCRuler dataset.

In these results, we see that neither the RuleTaker-finetuned model can achieve good results on any of the $\mathcal{ALC}$ fragments, nor the fragment-finetuned models can perform well

**Table 5.4: Zero-shot performance of RuleTaker fine-tuned models in all ALCRuler test sets. The italicized numbers in parentheses show the respective results in a secondary collection of test sets *without* complex concepts. *p_d* indicates the proof depths that are present in each fine-tuning set, while *e_d* indicates the explanation depths that are present in each test set.**

| RuleTaker finetune set | Accuracy per ALCRuler test set | | | | | |
|---|---|---|---|---|---|---|
| | **e_d=0** | **e_d<=1** | **e_d<=2** | **e_d<=3** | **e_d<=5** | **e_d>=0*** |
| **p_d=0** | 58.2 *(67.7)* | 53.8 *(60.7)* | 52.8 *(58.7)* | 53.4 *(60.7)* | 55.5 *(62.5)* | 57.4 *(64.5)* |
| **p_d<=1** | 60.1 *(68.5)* | 55.3 *(62.5)* | 53.9 *(60.6)* | 54.7 *(62.1)* | 56.0 *(63.5)* | 57.7 *(65.2)* |
| **p_d<=2** | 51.3 *(63.6)* | 51.6 *(60.1)* | 51.8 *(59.8)* | 52.6 *(60.4)* | 54.1 *(61.2)* | 55.6 *(62.1)* |
| **p_d<=3** | 58.5 *(63.9)* | 56.3 *(60.2)* | 55.3 *(59.5)* | 55.0 *(59.7)* | 56.5 *(61.2)* | 58.1 *(61.9)* |
| **p_d<=5** | 53.3 *(65.1)* | 52.8 *(61.7)* | 52.4 *(60.7)* | 52.9 *(61.1)* | 54.3 *(62.7)* | 55.9 *(63.1)* |

**Table 5.5: Main accuracy results for the ALCRuler fine-tuned models, testing in and out of distribution. The italicized numbers in parentheses show the respective results in a secondary collection of test sets *without* complex concepts. The green area shows the In-Distribution tests. *{*e_d} indicates the explanation depths that are present in each set.**

| Accuracy per ALCRuler test set | | | | | |
|---|---|---|---|---|---|
| **e_d=0** | **e_d<=1** | **e_d<=2** | **e_d<=3** | **e_d<=5** | **e_d>=0*** |
| 99.6 *(99.3)* | 84.7 *(84.2)* | 80.0 *(78.3)* | 78.3 *(77.9)* | 78.4 *(79.7)* | 78.9 *(80.9)* |
| 99.3 *(98.9)* | 97.1 *(96.2)* | 94.4 *(92.5)* | 92.5 *(91.0)* | 91.4 *(90.5)* | 90.7 *(90.2)* |
| 98.9 *(98.8)* | 96.9 *(95.5)* | 95.5 *(93.2)* | 93.8 *(92.7)* | 93.2 *(92.2)* | 92.7 *(92.2)* |
| 98.8 *(98.6)* | 96.9 *(95.4)* | 95.7 *(93.5)* | 95.0 *(93.4)* | 94.9 *(93.8)* | 94.6 *(93.8)* |
| 98.5 *(98.7)* | 96.1 *(94.1)* | 95.0 *(92.1)* | 94.5 *(92.1)* | 94.4 *(92.7)* | 94.4 *(92.9)* |

**Table 5.6: Zero-shot performance of ALCRuler fine-tuned models in all RuleTaker test sets. *p_d* indicates the proof depths that are present in each fine-tuning set, while *e_d* indicates the explanation depths that are present in each test set.**

| ALCRuler finetune set | Accuracy per RuleTaker test set | | | | |
|---|---|---|---|---|---|
| | **p_d=0** | **p_d<=1** | **p_d<=2** | **p_d<=3** | **p_d<=5** |
| **e_d=0** | 87.4 | 74.0 | 68.3 | 63.8 | 56.8 |
| **e_d<=1** | 73.5 | 68.4 | 66.3 | 63.5 | 59.8 |
| **e_d<=2** | 73.2 | 67.9 | 66.6 | 64.3 | 59.9 |
| **e_d<=3** | 65.9 | 60.7 | 58.4 | 55.9 | 48.6 |
| **e_d<=5** | 57.9 | 56.3 | 54.6 | 52.3 | 46.7 |

on the RuleTaker test set. Both of these results constitute strong indicators of overfitting, as RuleTaker's expressive capacity is directly comparable to that of the only-AND fragment of ALCRuler. Higher accuracy than the observed, seemingly equal to random performance, was expected. On the contrary, we see that the RuleTaker model is effectively unable to differentiate the ALCRuler fragments in its performance, and the ALCRuler-Fragment finetuned models showcase a progressive reduction of $3.4$% in their results as complexity increases, in similar fashion to the observations of the previous section 5.2.2.

**Table 5.7: Accuracy of models fine-tuned in data with the highest inference depth available (p_d<=5 for RuleTaker, e_d<=5 for ALCRuler), when tested on full expressivity datasets or fragments of $\mathcal{ALC}$. The blue area indicates measurements where the expressive complexity of all test datapoints was available during fine-tuning.**

| | | Accuracy per test set | | | |
|---|---|---|---|---|---|
| **Fine-tune set** | RuleTaker ($\sqcap$) | ALCRuler Frag.0 ($\sqcap$) | ALCRuler Frag.1 ($\sqcap, \sqcup$) | ALCRuler Frag.2 ($\sqcap, \sqcup, \exists$) | ALCRuler ($\sqcap, \sqcup, \exists, \forall$) |
| RuleTaker ($\sqcap$) | 99.2 | 53.2 | 52.4 | 53.5 | 54.3 |
| ALCRuler Frag.0 ($\sqcap$) | 50.1 | 94.8 | 92.7 | 90.2 | 90.2 |
| ALCRuler Frag.1 ($\sqcap, \sqcup$) | 49.9 | 94.4 | 95.0 | 92.1 | 92.2 |
| ALCRuler Frag.2 ($\sqcap, \sqcup, \exists$) | 45.4 | 94.6 | 94.9 | 94.8 | 94.3 |
| ALCRuler ($\sqcap, \sqcup, \exists, \forall$) | 46.7 | 94.1 | 94.6 | 94.5 | 94.4 |

Beyond these results, we see an accuracy of $90$% or more for all ALCRuler-related test sets[2]. The ALCRuler-finetuned model can achieve high accuracy for all fragments, which is expected given that these levels of expressive complexity were already available during its fine-tuning. Among the fragment-finetuned models, we see that all in-distribution runs had high accuracy results, while out-of-distribution ones saw a decrease of $2-4$%. Finally, it seems interesting that the model that was fine-tuned on $\mathcal{ALC}$ minus the for-all operator managed to achieve almost no decrease in performance. The reason behind this measurement could be either that the presence of the for-all operator in the ALCRuler dataset is anyway relatively low, or that the existential operator is conceptually close enough to the universal quantifier, so the model can use the experience from being exposed to the former one, to handle the latter.

## 5.4 Evaluation on hand-authored data

In tables 5.8 and 5.9 we are testing, zero-shot, the RuleTaker- and ALCRuler-finetuned models of section 5.2 to the hand-authored sets presented in [11].

[2] All ALCRuler-related models reached a validation accuracy of around $94$% during fine-tuning.

The performance of the RuleTaker-finetuned model in Birds and Electricity datasets (table 5.8, top) is comparable to the one presented by the authors in [11], achieving high results across the board which indicate the ability of these models to perform well in different data. Moving to the NatLang dataset (table 5.9, top), which paraphrases the synthetic RuleTaker data [11], we see a very similar performance profile to the one presented in section 5.2.1. While higher-depth datapoints are more difficult to be accurately labeled by the initial models, as models are exposed to higher-depth data they manage to completely label all datapoints, reaching perfect results. It's important to note here that this is not in agreement to the observations of the original authors, which indicated relatively low or only partial ability to perform well on these datasets. This potentially suggests that the generation methodology templates followed for the datasets' generation may be a more important factor than was originally considered by the authors, or that their observations do not translate robustly to variations in model architecture or training formulas (see section 5.2).

For the ALCRuler-finetuned models, we again see a different picture. For the Birds and Electricity datasets (table 5.8, bottom) we see that only the initial models can achieve relatively good results for Birds, while the performance for Electricity is severely decreased across the board. In NatLang's case (table 5.9, bottom) we see again that initial models can achieve medium results while the final model falls short in its predictions. Having said that, we see an increase in all values in comparison to matching performances of the same models in table 5.6, potentially suggesting that the paraphrasings that formed the NatLang dataset bring the data closer to the data generation templates used in the ALCRuler datasets.

**Table 5.8: Zero-shot test performances on birds-electricity dataset presented in [11], of all models fine-tuned on the respective depths. *p_d* indicates the proof depths that are present in each fine-tuning set, while *e_d* indicates the explanation depths that are present in each test set. *Note: Numbers of the test sets are following the notation of [11] and do not correlate with depth.***

| | Accuracy per test set | | | | | |
|---|---|---|---|---|---|---|
| **finetune set** | **birds1** | **birds2** | **electricity1** | **electricity2** | **electricity3** | **electricity4** |
| ruletaker (p_d-0) | 80.0 | 80.0 | 77.8 | 70.0 | 80.1 | 91.3 |
| ruletaker (p_d-1) | 97.5 | 100.0 | 97.5 | 92.8 | 94.1 | 92.1 |
| ruletaker (p_d-2) | 100.0 | 100.0 | 100.0 | 100.0 | 92.9 | 90.0 |
| ruletaker (p_d-3) | 100.0 | 100.0 | 100.0 | 100.0 | 92.6 | 71.4 |
| ruletaker (p_d-5) | 100.0 | 100.0 | 98.8 | 96.7 | 91.2 | 77.9 |
| ALCRuler (e_d-0) | 80.0 | 80.0 | 46.3 | 50.0 | 41.7 | 42.0 |
| ALCRuler (e_d-1) | 87.5 | 82.5 | 59.3 | 57.2 | 46.3 | 40.6 |
| ALCRuler (e_d-2) | 87.5 | 87.5 | 48.8 | 53.3 | 40.7 | 33.9 |
| ALCRuler (e_d-3) | 60.0 | 60.0 | 42.6 | 45.0 | 37.2 | 34.3 |
| ALCRuler (e_d-5) | 55.0 | 55.0 | 45.7 | 50.0 | 51.9 | 37.2 |

**Table 5.9: Zero-shot test performances on the NatLang dataset presented in [11], of all models fine-tuned on the respective depths. *p_d* indicates the proof depths that are present in each fine-tuning set, while *e_d* indicates the explanation depths that are present in each test set.**

| | Accuracy per test set | | | | |
|---|---|---|---|---|---|
| **finetune set** | **d=0** | **d<=1** | **d<=2** | **d<=3** | **dMax** |
| ruletaker (p_d-0) | 100.0 | 81.3 | 70.8 | 62.1 | 61.0 |
| ruletaker (p_d-1) | 100.0 | 100.0 | 85.6 | 75.1 | 73.7 |
| ruletaker (p_d-2) | 100.0 | 100.0 | 100.0 | 93.2 | 91.8 |
| ruletaker (p_d-3) | 100.0 | 99.9 | 99.9 | 99.9 | 99.9 |
| ruletaker (p_d-5) | 100.0 | 100.0 | 99.9 | 99.9 | 99.9 |
| ALCRuler (e_d-0) | 95.5 | 79.1 | 69.1 | 60.8 | 59.7 |
| ALCRuler (e_d-1) | 85.2 | 64.2 | 65.1 | 70.4 | 77.9 |
| ALCRuler (e_d-2) | 89.1 | 80.3 | 74.3 | 70.8 | 70.3 |
| ALCRuler (e_d-3) | 75.4 | 68.6 | 61.0 | 53.8 | 52.9 |
| ALCRuler (e_d-5) | 65.2 | 62.7 | 56.3 | 49.6 | 48.7 |

# 6. CONCLUSIONS AND FUTURE WORK

The primary goal of this work was to investigate whether the stellar results presented in [11] for datalog based natural language, can be transferred to more expressive language that is founded on $\mathcal{ALC}$ Description Logic. In the related work section, we saw how the current models that are used for the LRNL task are not considered robust due to overfitting or statistical feature shortcuts, while the best available approaches seem to explicitly include the reasoning steps as part of the training or predictions. Our experiments seem to challenge the claims in [11] with respect to robustness in paraphrasing and generalization.

We outline that overfitting appears to play the primary role in the behavior of our models, which did not meet our initial expectations, thus making a concrete evaluation on the task of NLR, and our specific result, difficult to interpret. Our observations are in sync with prior work that LLMs may tend to overfit to underlying linguistic correlations that can work against successful theorem proving [78, 84, 81, 59]. The direction of the research community of focusing on including intermediary and final proofs, or alternatively the precautionary adoption of specific fine-tuning methods and prompting, like Chain-of-Thought prompting [78, 49], seems to be valid. Furthermore, there are already published works with the aim of guaranteeing that dataset shortcuts are avoided (e.g. [56, 55]), towards "faithful" reasoning.

As mentioned in [12], it's essentially impossible to determine with acceptable certainty that a model is actually emulating some form of reasoning without having transparency on its reasoning steps. It is unclear how impactful are internal biases or representations on its output, and thus we believe that including a justification in this output should be considered a requirement going forward. Furthermore, it would be interesting to investigate how limitations in the form of statistical features affect models that focus on faithful proofs, like in [12]. Having said that, in order to include the justifications as an integral part of the predictions, fundamental architectural model changes are required. Instead, there is a set of future tasks that could shed more light on the observed results that we presented here:

- **Explore the effect of choosing random and synthetic data over hand-authored ones.** It should be investigated to what extent LRNL can be trained or evaluated on data with rich lexical semantics. A big advantage of using a pretrained model is the ability to distill world-knowledge in a broad range of topics and then use this knowledge inductively to influence its performance on the downstream task; a powerful property of the models for tasks like natural language inference or commonsense reasoning. But it remains unclear how logical reasoning is affected. On the one hand, as we saw in section 3, works like [59] hint that inductive biases based on vocabulary would hinder the downstream task or also that their presence should not theoretically constitute a complication. At the same time, in [11] the authors claim that using non-synthetic, lexically meaningful datasets for the task of emulating formal reasoners would not be ideal. The reason being that non-synthetic data by design offer a variability and flexibility of expression that may require from the model to also perform accurate reading comprehension other than the target reasoning task alone. On the other hand, we see works like [6, 23] where a richer language and vocabulary in the dataset is considered an advantage over other approaches.

- **Can deeper insights be drawn to the extent overfitting on templates play?** This could include a detail dataset description analysis over different publications and focused comparisons with other datasets like the ones in [23, 6] and others, with a focus on specific phrasings and the retaining of a certain vocabulary. On a similar note, also entailed is identifying what statistical features can be detected in our presented datasets (see also [81]) and to what extent their removal improves the outcomes or is even possible. In [81] it was indicated both that such a removal is beneficial and that it probably can never be complete or is computationally infeasible.

- **What is currently the most promising and robust way to generate datasets for NLR?** Based on the relevant conclusions from the aforementioned items, a potential outcome could be the incorporation of other data generation modules and approaches, like for example [6]. Additionally, an interesting future path forward, in an aim to produce a dataset of even higher quality, would be to utilize large generative language models to create the data (see also appendix A.4). This might take the form of starting a new dataset from scratch, or augmenting a small highly controlled and rigorously verified pre-existing dataset (see for example the works in [46, 10]).

- **What is actually the proper way of evaluating NLR generalization?** It becomes evident that simple accuracy reports are not enough to fully describe the success on the task. A more robust analysis with extended metrics, cross-evaluation to multiple datasets and statistically rich descriptions of used datasets seem to be imperative in order to promote reproducible and promising milestones of further research; relevant work can be followed in [21, 52, 79, 55]. The issue of ignoring or spoiling the task at hand in favor of identification of dataset statistics, artifacts or heuristics is a fundamental problem that seems to affect the usability and interpretability of LLMs. It is prevalent in both our experiments and related works, and has been identified even before those, in fields related to natural language inference [45, 22]. We believe that a deeper exploration of this weakness of the LLMs should be considered as integral for future works. It seems that knowledge and understanding on how to overcome such issues in a systematic and well-defined manner is still unclear, but would immensely improve the quality of the results and proposals being published.

Additional to the above future prospects, and after their potential resolution, there are also some open questions that could be further followed up as a direct extension to this work. For example:

- **Investigate if statements that exceed the expressivity of $\mathcal{ALC}$ can be answered**, given that $\mathcal{ALC}$ itself is learned robustly.

- Extending the idea of using a multiple models, it might be worth evaluating **how a model that has ultimately learned a reasoning task, can be used as the foundation for further domain specific fine-tuning.** Considering, that fine-tuning on specific Knowledge Graphs could lead to models that function as ”domain experts” for entities that utilize such data storage methods. The type of further fine-tuning that is required may or may not be reasoning-related in order to apply the previously acquired reasoning skills on specific domains.

- **Investigate to what extent training on the full expressivity of $\mathcal{ALC}$ is required to achieve a high enough performance on $\mathcal{ALC}$ or other Description Logic tasks.** A preliminary experimentation with fragments of $\mathcal{ALC}$ was presented, hinting that, for example, the universal and existential quantifiers may be closer conceptually than initially thought of. This example hypothesis should be validated with further analysis, on frequency statistics related to the expressivity of our presented datasets, as well as an extension that can support or refute this and similar hypotheses.

- **Investigate to what extent training on *unlimited* $\mathcal{ALC}$ (or fragments of it) can affect general NLU, or soft reasoning.** In our datasets, we have limited the data generation with our grammar, which impedes certain directions of research. On the one hand, human language may contain few to many levels of subcontext or parenthetical speech, while on the other hand, formal languages and reasoners operate on unlimited nesting levels. It would be interesting to explore how training on different recursion levels provided and modeled by our grammar would affect NLU tasks and additionally soft reasoning performances, given that such trained LLMs could be used alongside formal reasoners for optimization. Relevant insight for the handling of nesting in human speech and LLMs can be found in [40].

# ABBREVIATIONS - ACRONYMS

| | |
|---|---|
| AI | Artificial Intelligence |
| ALC | Attributive Language with Complements |
| API | Application Programming Interface |
| CoT | Chain-of-Thought |
| CWA | Closed World Assumption |
| DFS | Depth First Search |
| FOL | First-Order Logic |
| KB | Knowledge Base |
| LHS | Left-hand Side |
| LLM | Large Language Model |
| LRNL | Logical Reasoning over Natural Language |
| NAF | Negation as Failure |
| NL | Natural Language |
| NLP | Natural Language Processing |
| NLR | Natural Language Reasoning |
| NLU | Natural Language Understanding |
| OWA | Open World Assumption |
| OWL | Web Ontology Language |
| PCFG | Probabilistic Context-free Grammar |
| QA | Question-Answering |
| RHS | Right-hand Side |

# APPENDIX A. DATA GENERATION SUPPLEMENTARY

## A.1 PCFG examples for statement generation

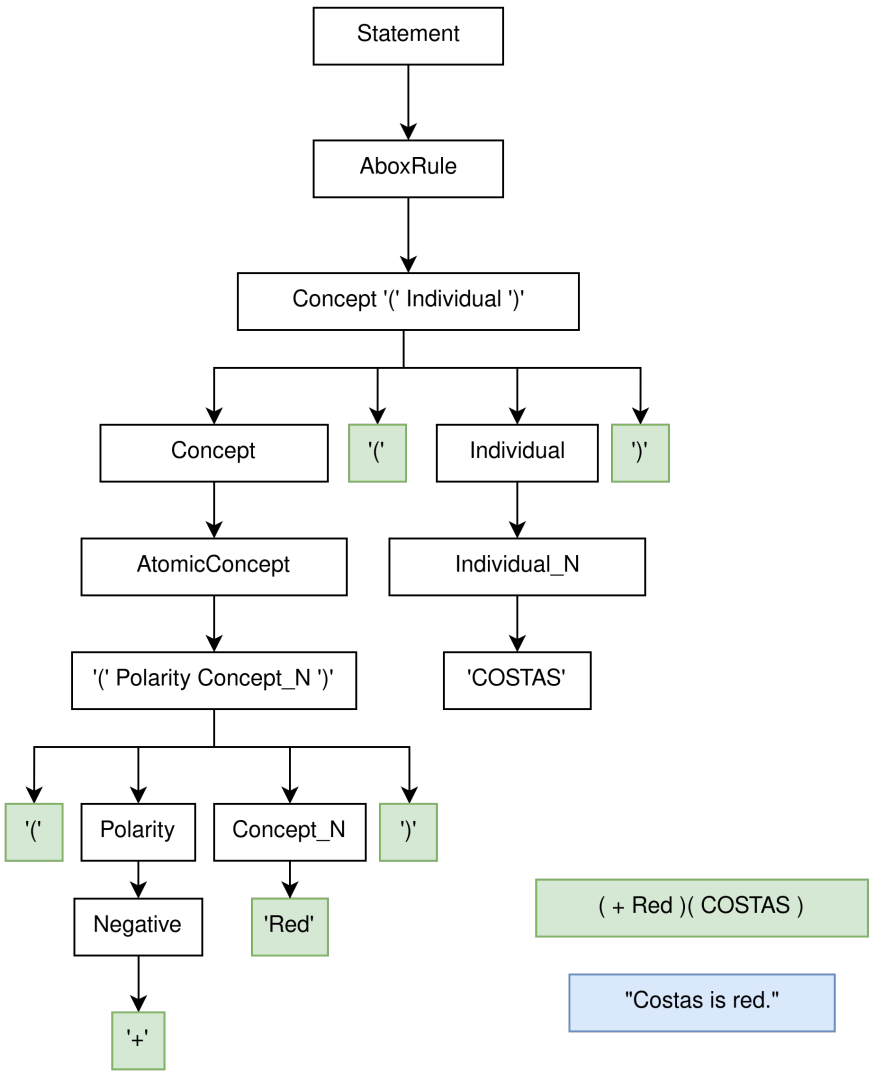



**Figure A.1: A tree expansion of an ABox axiom using ALCRuler grammar.**

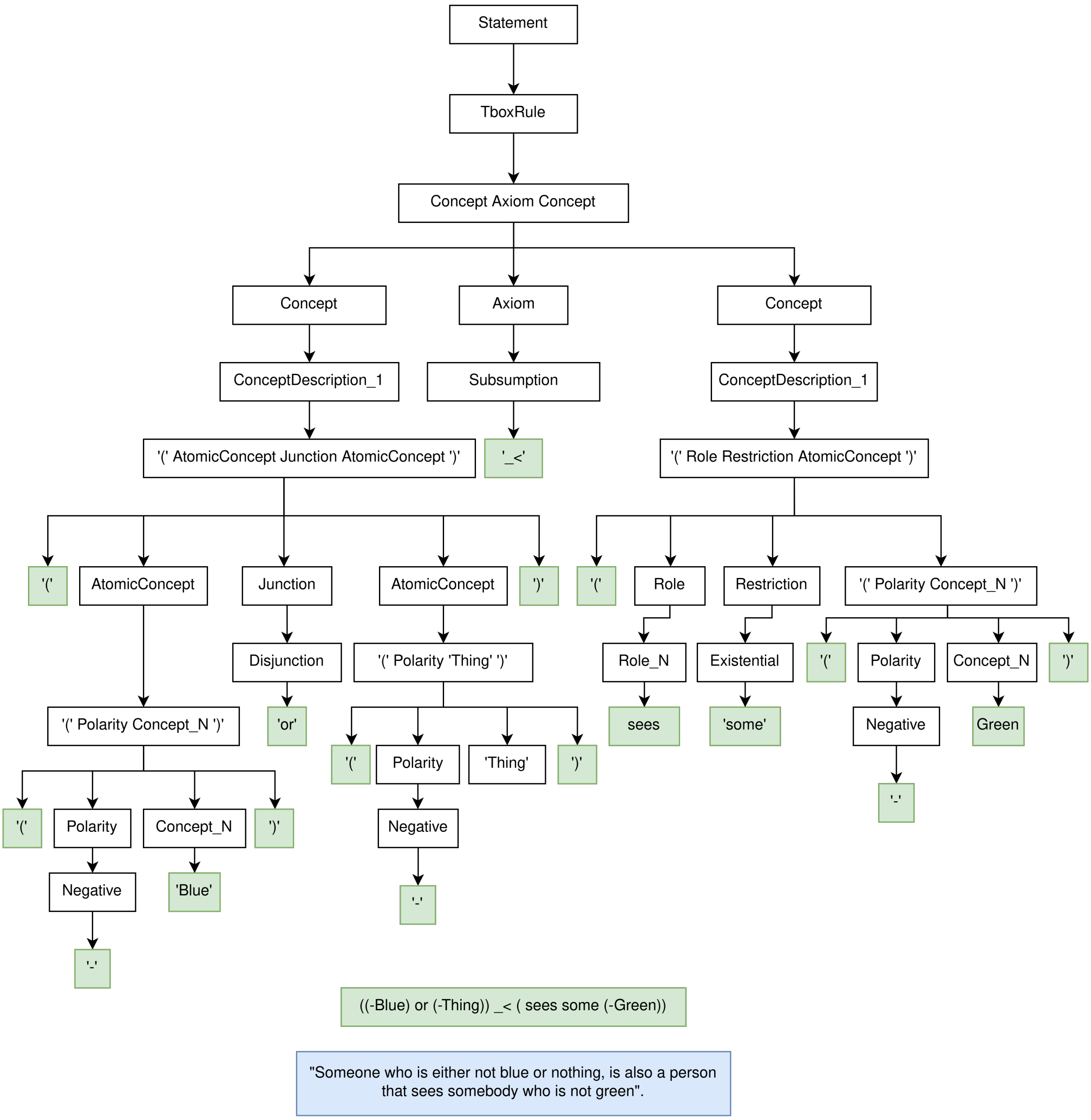



**Figure A.2: A tree expansion of a TBox axiom using ALCRuler grammar.**

## A.2 Example Datapoints from each dataset

**Table A.1: Example in which all ALCRuler fine-tuned models managed to give an accurate prediction, but only half the RuleTaker ones. Here we see how the "bear" concept is indirectly internally equalized to the Thing concept.**

| | |
|---|---|
| **example_id** | ALCRuler_MRD0_d5_test_example_17888_k3682_q6 |
| **proof_depth** | 4 |
| **label** | true |
| **context** | ERIN is white.<br>Anne is not white.<br>Somebody that is not a bear, equivalently is a dog.<br>One that is nothing, equivalently is a dog.<br>One who is not young, equivalently is not a bear.<br>Someone who is white, equivalently is not a tiger.<br>One that is not red, is also white.<br>Someone who is not white, is also not red.<br>One who is a squirrel, is also not young. |
| **query** | Someone who is a bear, equivalently is white. |
| **explanation** | Someone who is not red, is also white.<br>Someone that is not a bear, equivalently is a dog.<br>A person that is a dog, equivalently is nothing.<br>One that is not white, is also not red.<br>ERIN is a bear. |

**Table A.2: Example in which all fine-tuned models failed to give an accurate prediction.**

| | |
|---|---|
| **example_id** | ALCRuler_MRD1_d5_test_example_19505_k4775_q11 |
| **proof_depth** | 2 |
| **label** | false |
| **context** | BOB is a cat or quiet.<br>BOB is rough and not young.<br>BOB is a cat and not quiet.<br>FIONA eats FIONA.<br>CHARLIE eats CHARLIE.<br>FIONA is someone who visits only somebody who is young.<br>FIONA needs CHARLIE.<br>CHARLIE is green and not young.<br>GARY is a person who visits someone who is not a cat.<br>GARY visits DAVE.<br>Erin is young.<br>Somebody who is quiet and not young, equivalently is somebody that likes a person that is a thing.<br>One who is a tiger or not a cat, is also not a bald eagle.<br>One that is one that visits only one who is quiet, is also green or not young.<br>One who is young and not green, equivalently is somebody who likes somebody who is a cat.<br>A person that is not a cat, equivalently is not green. |
| **query** | One that is young, cannot also be a cat. |
| **explanation** | One who is not green, equivalently is not a cat.<br>One who is somebody that likes one who is a thing, equivalently is quiet and not young.<br>Someone who is young and not green, equivalently is a person that likes somebody that is a cat. |

**Table A.3: Example in which all fine-tuned models failed to give an accurate prediction.**

| | |
|---|---|
| **example_id** | ALCRuler_MRD1_Fragment0_d5_test_example_17339_k3017_q17 |
| **proof_depth** | 1 |
| **label** | false |
| **context** | FIONA is not nice.<br>FIONA is not round and not a squirrel.<br>GARY eats HARRY.<br>FIONA likes GARY.<br>GARY chases HARRY.<br>GARY visits FIONA.<br>GARY visits GARY.<br>HARRY visits HARRY.<br>HARRY is blue.<br>DAVE is big and not round.<br>HARRY chases GARY.<br>GARY visits HARRY.<br>Anne is a bald eagle and round.<br>Fiona is a squirrel.<br>Someone who is a squirrel and not big, is also nice and not a cat.<br>A person that is a squirrel, equivalently is big and not nice.<br>Somebody that is a bald eagle and round, is also not big.<br>One who is not a cat, equivalently is not big.<br>One who is round, equivalently is big and a squirrel. |
| **query** | Someone who is round, cannot also be a squirrel. |
| **explanation** | A person that is round, equivalently is big and a squirrel.<br>Somebody that is a squirrel, equivalently is big and not nice. |

**Table A.4: Example in which all fine-tuned models failed to give an accurate prediction.**

| | |
|---|---|
| **example_id** | ALCRuler_MRD1_Fragment2_d5_test_example_7285_k1503_q27 |
| **proof_depth** | 2 |
| **label** | true |
| **context** | FIONA needs BOB.<br>FIONA is one that sees somebody that is quiet.<br>HARRY visits ERIN.<br>ERIN is kind.<br>CHARLIE is somebody who visits one who is a squirrel.<br>ERIN needs BOB.<br>FIONA is white and young.<br>ANNE is someone who likes somebody who is a tiger.<br>Gary is a tiger.<br>Gary is a bear.<br>Anne is red.<br>Bob is red.<br>Someone who is rough and not a mouse, equivalently is a tiger.<br>One that is not a bear and not red, equivalently is quiet or not a cat.<br>A person that is someone that chases a person who is not a lion, equivalently is not a dog.<br>Someone that is big, is also young.<br>Somebody who is red, equivalently is a bear and not a tiger.<br>A person who is not green, equivalently is a dog.<br>A person who is young, equivalently is somebody that likes one that is a lion.<br>One who is not green, is also quiet.<br>A person that is rough, equivalently is big.<br>A person that is a rabbit and a tiger, equivalently is someone who sees somebody that is nothing. |
| **query** | Someone who is red, is not also a dog. |
| **explanation** | A person who is a bear, is also green.<br>One who is red, is also a bear.<br>Someone who is not green, equivalently is a dog. |

**Table A.5: Example in which all fine-tuned models managed to give an accurate prediction.**

| | |
|---|---|
| **example_id** | ALCRuler_MRD1_d5_test_example_18524_k4535_q59 |
| **proof_depth** | 3 |
| **label** | false |
| **context** | BOB needs ERIN.<br>CHARLIE eats GARY.<br>GARY is a lion and white.<br>CHARLIE eats CHARLIE.<br>GARY chases DAVE.<br>GARY eats HARRY.<br>GARY eats ERIN.<br>ANNE needs CHARLIE.<br>CHARLIE is a dog or smart.<br>GARY likes BOB.<br>GARY visits GARY.<br>CHARLIE is one that eats only somebody who is a bear.<br>BOB eats FIONA.<br>ERIN is someone that eats someone who is a tiger.<br>GARY visits CHARLIE.<br>CHARLIE is not a tiger.<br>Harry is big.<br>Somebody who is someone that visits somebody that is rough, is also not blue and not young.<br>One that is not a dog, equivalently is not a bear.<br>One that is blue and young, equivalently is white.<br>Someone that is someone that eats only somebody who is not a bear, equivalently is nice and not blue.<br>One who is not nice, equivalently is not big.<br>Somebody who is somebody who likes a person that is not a bear, equivalently is blue and rough.<br>Someone who is a person that chases only somebody who is nice, is also somebody who eats one that is a bear.<br>Someone who is somebody that visits only one that is a lion, equivalently is someone that eats one that is not a bear.<br>A person that is a thing, is also not nice. |
| **query** | GARY is nice. |
| **explanation** | One that is not a dog, equivalently is not a bear.<br>CHARLIE eats GARY.<br>CHARLIE is somebody who eats only somebody who is a bear.<br>Someone who is a thing, is also not nice. |

## A.3 Example scorings of Somebodies' metric

**Table A.6: Examples of natural language statements next to their somebodies' metric score.**

| statement | score |
|---|---|
| Somebody who is somebody who eats somebody that is not nice, equivalently is somebody that likes somebody that is kind and red. | -0.2 |
| Someone that is someone who sees a person that is kind, equivalently is a person who visits someone who is someone that chases someone that is young. | -0.14 |
| One who is somebody who chases only one that is one who sees only one who is rough, is also a thing. | 0.0 |
| Somebody who is one that eats only someone who is a person that needs someone that is cold, equivalently is a person that needs only a person that is not quiet. | 0.14 |
| A person that is cold and not nice and a dog and not furry, equivalently is somebody that chases someone that is not bald eagle and somebody who eats only somebody that is rough. | 0.2 |
| Somebody who loves somebody that needs somebody that finds a person that runs, is blind. | 0.25 |
| A person who is a person who eats one who is one that eats only someone that is not kind, is also somebody that likes someone that is blue and a thing. | 0.29 |
| Someone who is a cow and not rough, equivalently is someone who visits someone who is not a cow. | 0.33 |
| Somebody who loves somebody that needs someone that finds a person that runs, is blind. | 0.5 |
| FIONA is somebody who eats somebody that is kind and one who needs a person that is white. | 0.5 |
| Someone who is a mouse, is also somebody who eats only somebody that is not a bear or not a dog. | 0.67 |
| Someone who is a dog and not rough, is also someone that chases somebody who is green. | 0.67 |
| One who is someone that likes one who is a bear and not blue, is also not quiet. | 0.67 |
| A person that is white and not young, equivalently is kind or quiet and somebody who sees only a person that is not nice. | 0.67 |
| One who is furry, equivalently is not a bear and not a lion or a person who visits only someone who is a thing. | 1.0 |
| CHARLIE is one that eats only someone who is not quiet. | 1.5 |
| Somebody who is young, is also not young. | 3 |
| FIONA likes CHARLIE. | 4 |

## A.4 Data Generation with Generative LLMs

An alternative approach to the process presented in 4 would be to directly use a Generative Large Language Model [50] to synthesize the random examples, as such an approach would bring several advantages. Firstly, it would be vastly simpler in terms of implementation complexity, which would allow the generation of the datasets in much shorter time with a quite simpler pipeline. Another positive side effect from using a LLM for data generation would be the fact that the knowledge that is distilled in it would inadvertently be utilized in the creation of random data. It was for example shown in [74] that in an automated data generation scenario, the end-model can be provided with quality data with the pitfalls that characterize the starting-model filtered out, ultimately surpassing it. This way the generated knowledge bases can be meaningful while also remaining random (a task achieved in works like [23, 6]). Additionally, as investigated in [19], such models have the potential not only of producing datasets of higher quality when compared to algorithmically synthetic ones, but surprisingly be competitive against human annotators, and at a fraction of a cost.

On the other hand, this method wouldn't come without its own set of challenges. Firstly, using the output of Generative LLMs that are currently available can be heavily impacted by potential issues with regard to the quality of results [63, 82, 5], a reality that became very apparent with the publication and immediate recalling of Meta's Galactica project [66]. As extraordinary as these models may be, the results may often contain mistakes, inaccuracies, noise or simply be incomplete. This would affect both the trustworthiness of such generation and also their meaningfulness, which was previously mentioned as an advantage. And of course another equally important consideration would be that of embodied biases in those models [24, 15], that could very easily be passed on in the generated dataset and by extension be further baked in future models. Additionally, their use would make by definition the dataset generation pipeline less controllable and predictable, in the way a manual method is, rending the actual efficiency of their use not a given. The aforementioned concerns would therefore either block the use of such a pipeline, or at least necessitate the implementation of a rigorous evaluation and verification step in that pipeline, which could borrow many of the characteristic of the currently implemented one, further reducing its preparation-related time advantage. Not to mention that the required verification step can itself be non-complete, as it is targeting a fully natural language output, that is not anchored in any given generation structure or logic, making it only approximately effective.

The trustworthiness of such a pipeline would certainly be a big investigation topic, and it seems that an amount of trust is unavoidable. But nonetheless, this approach is an example of a big paradigm change in the dataset and general content generation [8, 76, 17]. To show how big is the potential of such an approach, below we are including three unedited example "conversations" with OpenAI's ChatGPT[1], using less than 10 minutes of manual prompting without prior experience with their Free Open Research demo[2]. As discussed

[1]Created with the version available on 1 Dec 2022.

[2]The demo can be found at https://chat.openai.com/chat.

in [42] and vividly illustrated in the field of image-generating diffusion models [80], accurate prompting for Large Generative Models is as important as any other part of their training. Therefore, codifying a set of specific prompt generating techniques has a great potential is dataset generation pipelines.

As a final remark, looking into the following examples, we see that the model can not only generate the random data, but it can also re-format them based on provided feedback. And most importantly, not only that, but it can also in itself use the generated data and provide answers that simulate the very thing this work set out to explore: If LLMs can emulate reasoning. In the below examples we see hints of such a behavior, by a "general purpose" generative model, that hasn't even been specifically fine-tuned to that task. Be it in the form of answering reasoning questions or in directly providing entailments of a given knowledge base themselves. Whether these examples constitute indicators of reasoning potential or are simply artifacts of the models' fuzzy next-word prediction routine, remains open research, adjacent to NLR and NLU [61, 58, 33, 4, 64, 47, 71, 44, 70].

### A.4.1 Example 1

Human: Create a random ALC knowledge base with 3 A-box rules and 2 T-box rules.

ChatGPT: A-box rules:

- All cats are mammals.
- All dogs are mammals.
- All birds are vertebrates.

T-box rules:

- A mammal is a type of animal.
- A vertebrate is a type of animal.

### A.4.2 Example 2

Human: Create a random ALC ontologies with 3 A-box rules and 2 T-box rules.
ChatGPT: A-box rules:

- John is a human.
- Mary is a human.
- All humans have two legs.

T-box rules:

- A human is a type of mammal.
- A mammal is a type of vertebrate.

Human: The third A-box rule you created is not an assertion.
ChatGPT: You are correct, the third A-box rule is not an assertion. It is a generalization stating that all members of a particular class (humans) possess a certain characteristic (having two legs). In an ALC ontology, assertions are statements about specific individuals, whereas generalizations are statements about the characteristics or relationships of classes or concepts.

Human: Then shouldn't you create a different rule to add in the Abox?
ChatGPT: Yes, in order to have a complete and valid ALC ontology, we would need to replace the third A-box rule with a valid assertion about a specific individual. For example: "John has two legs."
Alternatively, we could keep the third A-box rule as a generalization and add additional assertions about specific individuals to complete the A-box. For example: "Mary has two legs." "George is a mammal." "Samantha is a vertebrate."

Human: Please re-write the ontology correcting your mistake.

ChatGPT: Here is a revised version of the ALC ontology with the third A-box rule replaced with a valid assertion:
A-box rules:

- John is a human.
- Mary is a human.
- John has two legs.

T-box rules:

- A human is a type of mammal.
- A mammal is a type of vertebrate.

Human: Based on the above ontology is John a mammal?

ChatGPT: Yes, based on the A-box rule that states "John is a human" and the T-box rule that states "A human is a type of mammal," we can infer that John is a mammal. This is an example of how ALC ontologies can be used to make inferences based on the relationships between classes and individuals in the knowledge base.

Human: How many statements of the ontology did you use to answer?

ChatGPT: In order to answer the question of whether John is a mammal, I used two statements from the ontology: the A-box rule that states "John is a human" and the T-box rule that states "A human is a type of mammal." By combining these two statements and using the rules of ALC logic, we can infer that John is a mammal.

Human: Can you negate the first Abox rule?

ChatGPT: Yes, we can negate the first A-box rule to create a new statement that states ”John is not a human.” This revised ontology would have the following A-box and T-box rules:

A-box rules:

- John is not a human.
- Mary is a human.
- John has two legs.

T-box rules:

- A human is a type of mammal.
- A mammal is a type of vertebrate.

However, this revised ontology is not necessarily a valid representation of reality, as it contradicts the third A-box rule, which states that John has two legs. In the real world, it is not possible for someone to have two legs but not be a human. It is important to carefully consider the implications of the statements included in an ALC ontology and ensure that they are consistent with our understanding of the real world.

Human: Based on the above ontology is John a mammal?

ChatGPT: It is not possible to determine whether John is a mammal based on the revised ontology, as the first A-box rule contradicts the third A-box rule. In order to make inferences about John’s characteristics or relationships, we would need to remove or revise the conflicting statements in the ontology. For example, we could change the first A-box rule to state ”John is a human” in order to be consistent with the third A-box rule and allow us to make inferences based on the T-box rules.

### A.4.3 Example 3

Human: Can you create an ALC ontology using the binary predicates (likes, needs, visits), the unary predicates (lion, cow, smart, furry) and the individuals (Gary, Anne, Fiona)?

ChatGPT: Sure, I can create an ALC ontology using the binary predicates (likes, needs, visits) and the unary predicates (lion, cow, smart, furry) and the individuals (Gary, Anne, Fiona). Here's an example:
Ontology: Animals
Binary predicates:

- likes(x, y): x likes y
- needs(x, y): x needs y
- visits(x, y): x visits y

Unary predicates:

- lion(x): x is a lion
- cow(x): x is a cow
- smart(x): x is smart
- furry(x): x is furry

Individuals:

- Gary
- Anne
- Fiona

(cont.): Assertions:

- lion(Gary)
- cow(Anne)
- smart(Fiona)
- furry(Gary)
- likes(Gary, Anne)
- needs(Anne, grass)
- visits(Fiona, library)

Human: Can you add 4 T-box statements?
ChatGPT: Sure, here are four T-box statements that I can add to the ontology:

1. All lions are furry. This statement can be represented in the ontology as: furry(x) $\sqsubseteq$ lion(x)
2. All cows need grass. This statement can be represented in the ontology as: needs(x, grass) $\sqsubseteq$ cow(x)
3. Smart individuals visit libraries. This statement can be represented in the ontology as: visits(x, library) $\sqsubseteq$ smart(x)
4. Individuals who like cows are smart. This statement can be represented in the ontology as: smart(x) $\sqsubseteq$ likes(x, cow)

Note that these statements are written in first-order logic using the symbols $\sqsubseteq$ (subset or subset-equal) to denote the "is-a" relationship between concepts. In this case, the statements are saying that the concepts on the left-hand side are subsets of the concepts on the right-hand side. For example, the first statement is saying that all lions are furry, so the concept of "furry" is a subset of the concept of "lion".

Human: Combine this with the ontology into formal ALC representation.

ChatGPT: Sure, here is the ontology with the four T-box statements included in a formal ALC representation:

Ontology: Animals

Binary predicates:

- likes(x, y): x likes y
- needs(x, y): x needs y
- visits(x, y): x visits y

Unary predicates:

- lion(x): x is a lion
- cow(x): x is a cow
- smart(x): x is smart
- furry(x): x is furry

Individuals:

- Gary
- Anne
- Fiona

Assertions:

- lion(Gary)
- cow(Anne)
- smart(Fiona)
- furry(Gary)
- likes(Gary, Anne)
- needs(Anne, grass)
- visits(Fiona, library)

(cont.): T-box axioms:

- furry(x) $\sqsubseteq$ lion(x)
- needs(x, grass) $\sqsubseteq$ cow(x)
- visits(x, library) $\sqsubseteq$ smart(x)
- smart(x) $\sqsubseteq$ likes(x, cow)

The T-box axioms are written in the ALC (Aquatic Logics for the Web) description logic, which is a subset of first-order logic that is commonly used for representing ontologies. The ALC representation allows us to formally specify the concept hierarchy and the relationships between concepts in the ontology.

Human: I didn’t define library or grass though.

ChatGPT: Ah, you’re right, I apologize for the mistake. In the previous example, I included the individuals ”library” and ”grass” in the assertions, but you did not define them earlier. I can remove these individuals from the ontology and update the assertions accordingly. Here is the updated ontology:

Ontology: Animals

Binary predicates:

- likes(x, y): x likes y
- needs(x, y): x needs y
- visits(x, y): x visits y

Unary predicates:

- lion(x): x is a lion
- cow(x): x is a cow
- smart(x): x is smart
- furry(x): x is furry

Individuals:

- Gary
- Anne
- Fiona

Assertions:

- lion(Gary)
- cow(Anne)
- smart(Fiona)
- furry(Gary)
- likes(Gary, Anne)

(cont.): In this updated ontology, the individuals ”library” and ”grass” have been removed, and the assertion ”needs(Anne, grass)” has been removed because it references the individual ”grass”. Additionally, the assertion ”visits(Fiona, library)” has been removed because it references the individual ”library”. This updated ontology only includes the individuals that you defined earlier.

Human: Why did you remove the T-box statements?

ChatGPT: I apologize for the mistake, I did not mean to remove the T-box statements from the ontology. In my previous response, I focused on removing the individuals ”library” and ”grass” from the ontology, and I accidentally removed the T-box statements as well. Here is the updated ontology with the T-box statements included:
Ontology: Animals
Binary predicates:

- likes(x, y): x likes y
- needs(x, y): x needs y
- visits(x, y): x visits y

Unary predicates:

- lion(x): x is a lion
- cow(x): x is a cow
- smart(x): x is smart
- furry(x): x is furry

(cont.): Individuals:

- Gary
- Anne
- Fiona

Assertions:

- lion(Gary)
- cow(Anne)
- smart(Fiona)
- furry(Gary)
- likes(Gary, Anne)

T-box axioms:

- furry(x) $\sqsubseteq$ lion(x)
- needs(x, grass) $\sqsubseteq$ cow(x)
- visits(x, library) $\sqsubseteq$ smart(x)
- smart(x) $\sqsubseteq$ likes(x, cow)

In this updated ontology, the T-box axioms have been included to specify the concept hierarchy and the relationships between concepts in the ontology. The individuals ”library” and ”grass” have been removed, and the assertions ”needs(Anne, grass)” and ”visits(Fiona, library)” have been removed because they reference these individuals. This updated ontology only includes the classes and individuals that you defined earlier.

Human: What are the entailments of this ontology?

ChatGPT: Entailments are statements that can be logically inferred from the axioms in an ontology. In this case, the entailments of the ontology would be statements that can be logically inferred from the binary predicates, unary predicates, individuals, and T-box axioms that are defined in the ontology.
Here are some possible entailments of the ontology:

- Gary is a lion. This entailment can be inferred from the assertion "lion(Gary)".
- Anne is a cow. This entailment can be inferred from the assertion "cow(Anne)".
- Fiona is smart. This entailment can be inferred from the assertion "smart(Fiona)".
- Gary is furry. This entailment can be inferred from the assertion "furry(Gary)".
- Gary likes Anne. This entailment can be inferred from the assertion "likes(Gary, Anne)".
- Anne needs something. This entailment can be inferred from the T-box axiom "needs(x, grass) $\sqsubseteq$ cow(x)" and the assertion "cow(Anne)".
- Fiona visits something. This entailment can be inferred from the T-box axiom "visits(x, library) $\sqsubseteq$ smart(x)" and the assertion "smart(Fiona)".

These are just some possible entailments of the ontology. There may be other entailments that can be inferred from the axioms in the ontology. The actual entailments of the ontology would depend on the specific axioms that are defined in the ontology and how they relate to each other.

| | |
|---|---|
| Human: | Can you list all the other entailments? |
| ChatGPT: | It is not possible for me to list all of the entailments of the ontology because the number of possible entailments is infinite. The actual entailments of the ontology would depend on the specific axioms that are defined in the ontology and how they relate to each other.<br>In general, the process of determining all of the entailments of an ontology is known as reasoning, and it involves using logical rules to infer new statements from the axioms in the ontology. However, this can be a complex process, and it is not possible for me to provide a complete list of all of the entailments of the ontology. |

Human: Ok, provide 10 more entailments.

ChatGPT: Sure, here are ten additional possible entailments of the ontology:

1. Anne is a cow and needs something. This entailment can be inferred from the T-box axiom "needs(x, grass) $\sqsubseteq$ cow(x)" and the assertions "cow(Anne)" and "needs(Anne, grass)".
2. Fiona is smart and visits something. This entailment can be inferred from the T-box axiom "visits(x, library) $\sqsubseteq$ smart(x)" and the assertions "smart(Fiona)" and "visits(Fiona, library)".
3. All lions are furry. This entailment can be inferred from the T-box axiom "furry(x) $\sqsubseteq$ lion(x)".
4. All cows need something. This entailment can be inferred from the T-box axiom "needs(x, grass) $\sqsubseteq$ cow(x)".
5. Smart individuals visit something. This entailment can be inferred from the T-box axiom "visits(x, library) $\sqsubseteq$ smart(x)".
6. Individuals who like cows are smart. This entailment can be inferred from the T-box axiom "smart(x) $\sqsubseteq$ likes(x, cow)" and the assertion "likes(Gary, Anne)".
7. Anne is a cow and needs grass. This entailment can be inferred from the T-box axioms "needs(x, grass) $\sqsubseteq$ cow(x)" and "cow(Anne)" and the assertion "needs(Anne, grass)".
8. Fiona is smart and visits a library. This entailment can be inferred from the T-box axioms "visits(x, library) $\sqsubseteq$ smart(x)" and "smart(Fiona)" and the assertion "visits(Fiona, library)".
9. Gary is a lion and likes Anne. This entailment can be inferred from the assertions "lion(Gary)", "cow(Anne)", and "likes(Gary, Anne)".
10. Fiona is smart and likes cows. This entailment can be inferred from the T-box axioms "smart(x) $\sqsubseteq$ likes(x, cow)" and "likes(Gary, Anne)" and the assertion "smart(Fiona)".

These are just some additional possible entailments of the ontology. There may be many other entailments that can be inferred from the axioms in the ontology. The actual entailments of the ontology would depend on the specific axioms that are defined in the ontology and how they relate to each other.

Human: Provide to me 10 entailments that use 3 or more axioms from the ontology.

ChatGPT: Sure, here are ten possible entailments of the ontology that use three or more axioms:

1. Anne is a cow, needs something, and likes Gary. This entailment can be inferred from the T-box axioms "needs(x, grass) $\sqsubseteq$ cow(x)" and "likes(x, y) $\sqsubseteq$ smart(x)" and the assertions "cow(Anne)", "needs(Anne, grass)", and "likes(Gary, Anne)".

2. Fiona is smart, visits something, and likes cows. This entailment can be inferred from the T-box axioms "visits(x, library) $\sqsubseteq$ smart(x)" and "smart(x) $\sqsubseteq$ likes(x, cow)" and the assertions "smart(Fiona)", "visits(Fiona, library)", and "likes(Gary, Anne)".

3. Gary is a lion, is furry, and likes Anne. This entailment can be inferred from the T-box axioms "furry(x) $\sqsubseteq$ lion(x)" and "likes(x, y) $\sqsubseteq$ smart(x)" and the assertions "lion(Gary)", "furry(Gary)", and "likes(Gary, Anne)".

4. Fiona is smart, visits a library, and likes cows. This entailment can be inferred from the T-box axioms "visits(x, library) $\sqsubseteq$ smart(x)" and "smart(x) $\sqsubseteq$ likes(x, cow)" and the assertions "smart(Fiona)", "visits(Fiona, library)", and "likes(Gary, Anne)".

5. Anne is a cow, needs grass, and likes Gary. This entailment can be inferred from the T-box axioms "needs(x, grass) $\sqsubseteq$ cow(x)" and "likes(x, y) $\sqsubseteq$ smart(x)" and the assertions "cow(Anne)", "needs(Anne, grass)", and "likes(Gary, Anne)".

6. Gary is a lion, is furry, and needs something. This entailment can be inferred from the T-box axioms "furry(x) $\sqsubseteq$ lion(x)" and "needs(x, y) $\sqsubseteq$ smart(x)" and the assertions "lion(Gary)", "furry(Gary)", and "needs(Anne, grass)".

7. Fiona is smart, visits a library, and needs something. This entailment can be inferred from the T-box axioms "visits(x, library) $\sqsubseteq$ smart(x)" and "needs(x, y) $\sqsubseteq$ smart(x)" and the assertions "smart(Fiona)", "visits(Fiona, library)", and "needs(Anne, grass)".

8. Anne is a cow, needs grass, and visits something. This entailment can be inferred from the T-box axioms "needs(x, grass) $\sqsubseteq$ cow(x)" and "visits(x, y) $\sqsubseteq$ smart(x)" and the assertions "cow(Anne)", "needs(Anne, grass)", and "visits(Fiona,

Human: These are 8 entailments not 10.

ChatGPT: You're right, I apologize for the mistake. In my previous response, I listed eight possible entailments of the ontology that use three or more axioms, but I did not provide two additional entailments as requested. Here are two additional possible entailments of the ontology that use three or more axioms:

10. Gary is a lion, is furry, and visits something. This entailment can be inferred from the T-box axioms "furry(x) $\sqsubseteq$ lion(x)" and "visits(x, y) $\sqsubseteq$ smart(x)" and the assertions "lion(Gary)", "furry(Gary)", and "visits(Fiona, library)".

11. Anne is a cow, needs grass, and is smart. This entailment can be inferred from the T-box axioms "needs(x, grass) $\sqsubseteq$ cow(x)" and "smart(x) $\sqsubseteq$ likes(x, cow)" and the assertions "cow(Anne)", "needs(Anne, grass)", and "likes(Gary, Anne)".

These are two additional possible entailments of the ontology that use three or more axioms. As I mentioned earlier, there may be many other entailments that can be inferred from the axioms in the ontology. The actual entailments of the ontology would depend on the specific axioms that

Human: Can you count for me how many axioms where used in each of the above entailments?

ChatGPT: Sure, here is a list of the ten possible entailments of the ontology that use three or more axioms, along with the number of axioms used in each entailment:

- Anne is a cow, needs something, and likes Gary. (3 axioms)
- Fiona is smart, visits something, and likes cows. (3 axioms)
- Gary is a lion, is furry, and likes Anne. (3 axioms)
- Fiona is smart, visits a library, and likes cows. (3 axioms)
- Anne is a cow, needs grass, and likes Gary. (3 axioms)
- Gary is a lion, is furry, and needs something. (3 axioms)
- Fiona is smart, visits a library, and needs something. (3 axioms)
- Anne is a cow, needs grass, and visits something. (3 axioms)
- Gary is a lion, is furry, and visits something. (3 axioms)
- Anne is a cow, needs grass, and is smart. (3 axioms)

As you can see, each of the above entailments uses three axioms from the ontology. The number of axioms used in each entailment may vary depending on the specific axioms that are defined in the ontology and how they relate to each other.

Human: Can you give me an entailment that uses 5 axioms?

ChatGPT: Sure, here is a possible entailment of the ontology that uses five axioms:

- Fiona is smart, visits a library, needs something, likes cows, and is furry. This entailment can be inferred from the T-box axioms "visits(x, library) $\sqsubseteq$ smart(x)", "smart(x) $\sqsubseteq$ likes(x, cow)", "needs(x, y) $\sqsubseteq$ smart(x)", "furry(x) $\sqsubseteq$ lion(x)", and "likes(x, y) $\sqsubseteq$ smart(x)" and the assertions "smart(Fiona)", "visits(Fiona, library)", "needs(Anne, grass)", "likes(Gary, Anne)", and "furry(Gary)".

This entailment uses five axioms from the ontology to infer that Fiona is smart, visits a library, needs something, likes cows, and is furry. The number of axioms used in an entailment may vary depending on the specific axioms that are defined in the ontology and how they relate to each other.

Human: Can you give me an entailment whose explanation is chaining 4 axioms?

ChatGPT: Sure, here is a possible entailment of the ontology whose explanation involves chaining four axioms:

- Anne is a cow, needs grass, visits something, and is smart. This entailment can be inferred from the T-box axioms ”needs(x, grass) $\sqsubseteq$ cow(x)”, ”visits(x, y) $\sqsubseteq$ smart(x)”, ”smart(x) $\sqsubseteq$ likes(x, cow)”, and ”likes(x, y) $\sqsubseteq$ smart(x)” and the assertions ”cow(Anne)”, ”needs(Anne, grass)”, ”visits(Fiona, library)”, and ”likes(Gary, Anne)”.

The explanation of this entailment involves chaining four axioms from the ontology. First, the T-box axiom ”needs(x, grass) $\sqsubseteq$ cow(x)” is used to infer that Anne needs something because she is a cow. Then, the T-box axiom ”visits(x, y) $\sqsubseteq$ smart(x)” is used to infer that Anne visits something because she is smart. Next, the T-box axiom ”smart(x) $\sqsubseteq$ likes(x, cow)” is used to infer that Anne is smart because she likes cows. Finally, the T-box axiom ”likes(x, y) $\sqsubseteq$ smart(x)” is used to infer that Anne likes cows because Gary likes Anne.

This explanation involves chaining four axioms from the ontology to infer that Anne is a cow, needs grass, visits something, and is smart. The number of axioms used in an explanation may vary depending on the specific axioms that are defined in the ontology and how they relate to each other.

Human: Generate a new ontology with the names I originally defined, that is different to the previous one but equally big.

ChatGPT: Sure, here is a possible ontology that uses the names "Gary", "Anne", "Fiona", "likes", "needs", "visits", "lion", "cow", "smart", and "furry" and is equally big as the previous ontology:

- Binary predicates: "likes", "needs", "visits"
- Unary predicates: "lion", "cow", "smart", "furry"
- Individuals: "Gary", "Anne", "Fiona"
- T-box axioms:
  - "likes(x, y) $\sqsubseteq$ smart(x)"
  - "smart(x) $\sqsubseteq$ likes(x, cow)"
  - "needs(x, y) $\sqsubseteq$ smart(x)"
  - "visits(x, y) $\sqsubseteq$ smart(x)"

This ontology defines the same names, binary predicates, unary predicates, and individuals as the previous ontology, but it uses different T-box axioms to define the relationships between these entities. In this ontology, the T-box axioms assert that individuals who like something are smart, that smart individuals like cows, that individuals who need something are smart, and that individuals who visit something are smart. This ontology is equally big as the previous ontology because it uses the same names, binary predicates, unary predicates, individuals, and number of T-box axioms. However, the specific axioms and entailments of this ontology may be different from the previous ontology because the axioms in this ontology define different relationships between the entities.

Human: You forgot to add an Abox.

ChatGPT: You're right, I apologize for the mistake. In my previous response, I provided a possible ontology that uses the names "Gary", "Anne", "Fiona", "likes", "needs", "visits", "lion", "cow", "smart", and "furry" and is equally big as the previous ontology, but I forgot to include an Abox with assertions about these entities. Here is the same ontology with an added Abox that includes some assertions about these entities:

- Binary predicates: "likes", "needs", "visits"
- Unary predicates: "lion", "cow", "smart", "furry"
- Individuals: "Gary", "Anne", "Fiona"
- T-box axioms:
  - "likes(x, y) $\sqsubseteq$ smart(x)"
  - "smart(x) $\sqsubseteq$ likes(x, cow)"
  - "needs(x, y) $\sqsubseteq$ smart(x)"
  - "visits(x, y) $\sqsubseteq$ smart(x)"
- Abox assertions:
  - "lion(Gary)"
  - "cow(Anne)"
  - "smart(Fiona)"
  - "furry(Gary)"
  - "needs(Anne, grass)"
  - "visits(Fiona, library)"
  - "likes(Gary, Anne)"

This ontology defines the same names, binary predicates, unary predicates, individuals, and T-box axioms as the previous ontology, but it includes an Abox with assertions about these entities. The Abox asserts that Gary is a lion, Anne is a cow, Fiona is smart, Gary is furry, Anne needs grass, Fiona visits a library, and Gary likes Anne.

This ontology is equally big as the previous ontology because it uses the same names, binary predicates, unary predicates, individuals, T-box axioms, and number of Abox assertions. However, the specific axioms and entailments of this ontology may be different from the previous ontology because the axioms and assertions in this ontology define different relationships

# APPENDIX B. EXPERIMENTS SUPPLEMENTARY

## B.1 Hyperparameter search runs

**Table B.1: Training Hyperparameters that were tried for different datasets, including the ones that were chosen as final.**

| finetuning_dataset | eval batch size | train batch size | train epochs | learning rate warmup ratio | gradient accum. steps | Eval accuracy |
|---|---|---|---|---|---|---|
| ruletaker/depth-1 | 16 | 16 | 4 | 0.06 | 16 | **0.98** **(final_model)** |
| | 16 | 16 | 4 | 0.06 | 48 | 0.90 |
| ruletaker/depth-2 | 24 | 24 | 12 | 0.2 | 12 | **0.99** **(final_model)** |
| | 24 | 24 | 8 | 0.1 | 12 | 0.98 |
| | 32 | 36 | 4 | 0.3 | 16 | 0.84 |
| | 32 | 36 | 4 | 0.1 | 16 | 0.91 |
| | 32 | 64 | 4 | 0.06 | 18 | 0.86 |
| ruletaker/depth-3 | 24 | 24 | 15 | 0.5 | 12 | **0.99** **(final_model)** |
| | 32 | 36 | 8 | 0.5 | 16 | 0.95 |
| | 32 | 36 | 4 | 0.1 | 16 | 0.88 |
| ruletaker/depth-5 | 24 | 24 | 15 | 0.4 | 8 | **0.99** **(final_model)** |
| | 24 | 24 | 12 | 0.2 | 12 | 0.98 |
| ALCRuler_MRD1/depth-1 | 18 | 18 | 6 | 0.2 | 12 | **0.97** **(final model)** |
| | 24 | 24 | 5 | 0.06 | 12 | 0.89 |
| ALCRuler_MRD1/depth-5 | 18 | 18 | 8 | 0.5 | 8 | **0.94** **(final_model)** |
| | 18 | 18 | 8 | 0.2 | 12 | 0.92 |
| | 24 | 24 | 15 | 0.35 | 8 | 0.93 |

## B.2 Training plots for Ruletaker

In this section, the plots that describe the training process for the RuleTaker fine-tuning datasets are presented.

## B.3 Examination of intra-dataset similarities

In effort to gain some insight on the quality of the tested synthetic datasets and more specifically aiming to potentially explain the difference in the ease of fine-tuning on ALCRuler over RuleTaker, in this section we are exploring two similarity coefficients for dataset samples. The method is to examine the textual similarity between a train set that is limited to inference depth 0, with a test set that contains examples of higher inference depths. We are not comparing a train set with another train set, as in the ALCRuler case, datapoints of lower inference depth are re-used.

We use two types of similarity. Jaccard similarity, to examine if completely identical phrases have been generated, and cosine similarity, to examine if the semantic similarity is too high, and if yes, for which pairs. Jaccard can be valuable because it doesn't encode any meaning in its measurement beyond the co-occurence of identical words. By contrast, this may not be the case for cosine similarity, which relies on embeddings that can

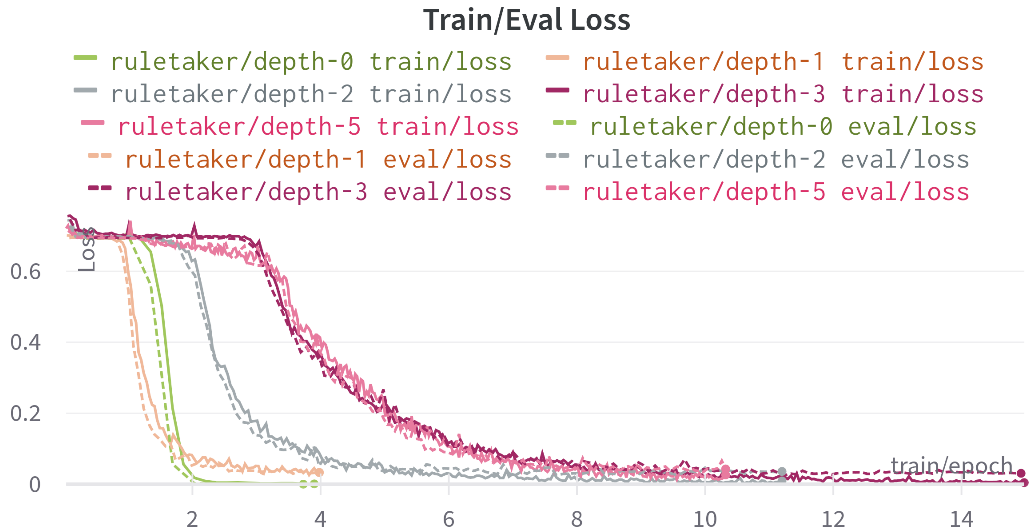



**Figure B.1: Training/Validation Loss progression per training epoch for Ruletaker datasets.**

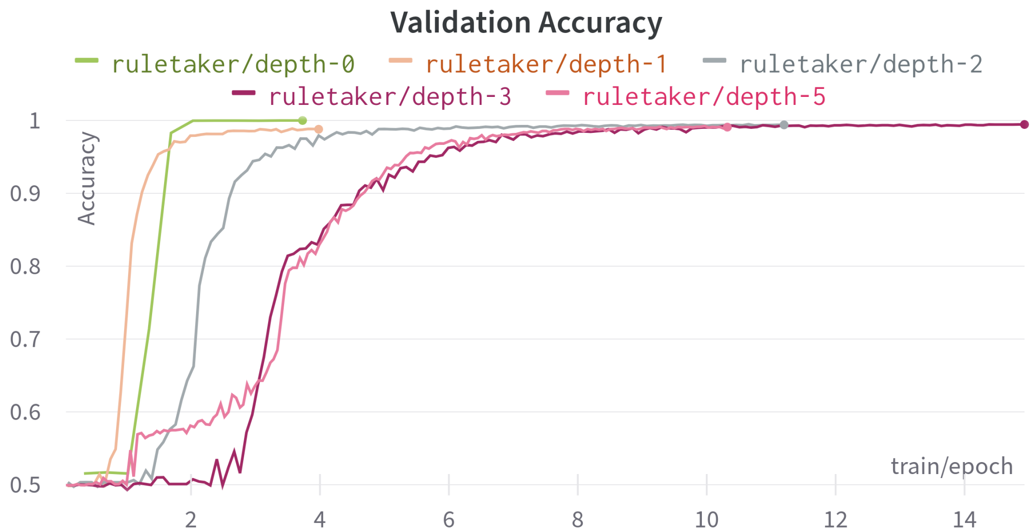



**Figure B.2: Validation Accuracy progression per training epoch for Ruletaker datasets.**

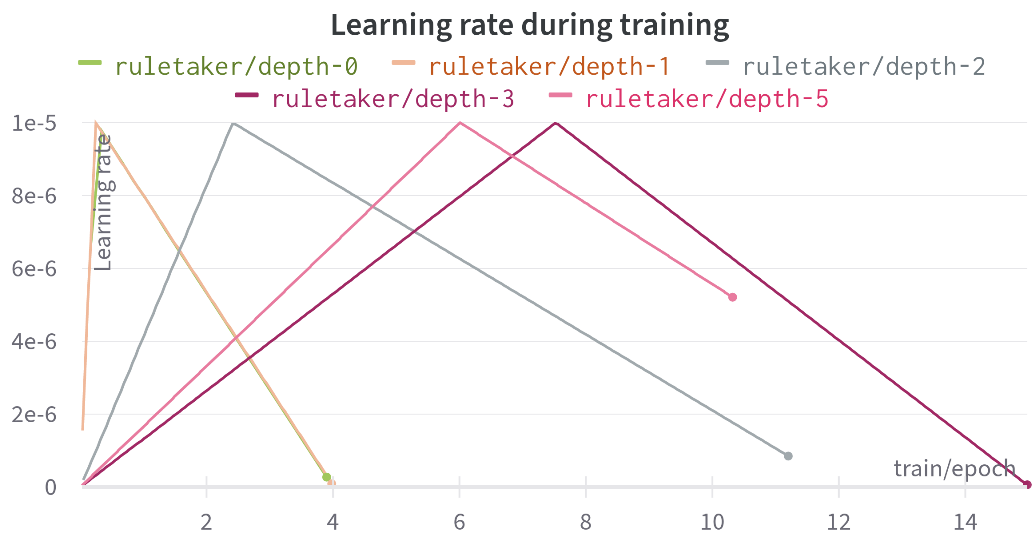



**Figure B.3: Learning Rate progression per training epoch for Ruletaker datasets.**

encode additional information. To acquire the embeddings that are to be used for the cosine similarity comparisons, we are using the $\mathrm{sentence\text{-}transformer}$ library[1] and Microsoft's MiniLM-L6 [69] model, as it is considered fast but good for the task, and in some initial trial runs it seemed to identify similarities on par with RoBERTa based alternatives (see one example in table B.2).

A random sample of 150 datapoints was selected from each file, leading to 22.5k pair comparisons, while the metric limits were set to $0.1$ and $0.9$ for minimum Jaccard and maximum cosine similarities respectively. To preprocess each datapoint, the noun natural language templates (see section 4.4) were removed to reduce noise in the calculations, including $\mathrm{"something"}$ for the RuleTaker data. In figures B.4 and B.5 we see the results of these runs.

Jaccard similarity sample appears to be close or equal to zero, for both sample populations, which indicate that near-duplicates are basically non-existent. No exact duplicate was detected. Regarding the cosine similarity, the two datasets exhibit different results. In the case of ALCRuler, we see a normal distribution shape centered around the $0.75$ point, with a small but evident tail to the left. In the case of RuleTaker, we see a joint normal distribution shape with two modes, around $0.45$ and $0.75$. This indicates to us that current language models consider the datapoints relatively close to each other in their embedding space, assigning on average high similarities for the randomly selected pairs. This is expected given that the used models are general-purpose ones, that encode vast amounts of information, while the semantics of our synthetic data are narrow by comparison.

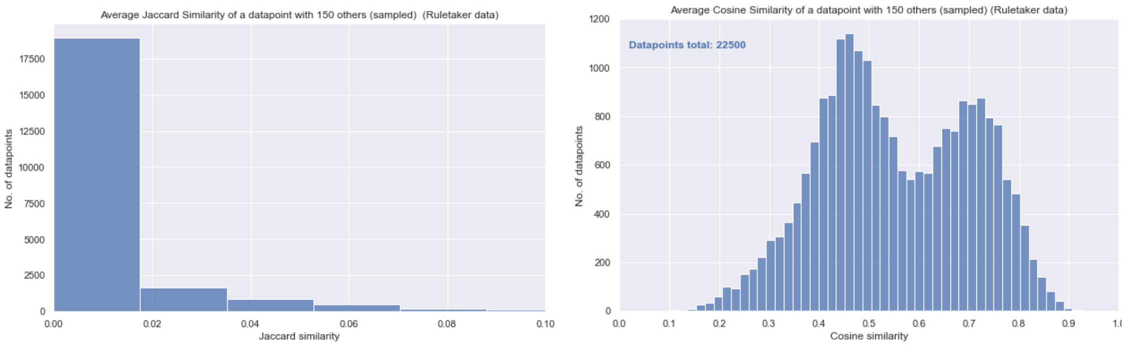
Having said that, the second mode detected for RuleTaker is indicative of a noise factor that is able to reduce the average similarity coefficient by almost half, and which is also

[1] https://www.sbert.net/docs/pretrained_models.html#model-overview

missing from the ALCRuler dataset despite its closeness to the original dataset. While a further analysis of these results was not made, it remains an interesting future exploration path, with the potential of explaining why ALCRuler datasets (that miss the lower cosine similarity mode) appeared to be easier to predict than the RuleTaker ones (see section 5.2).

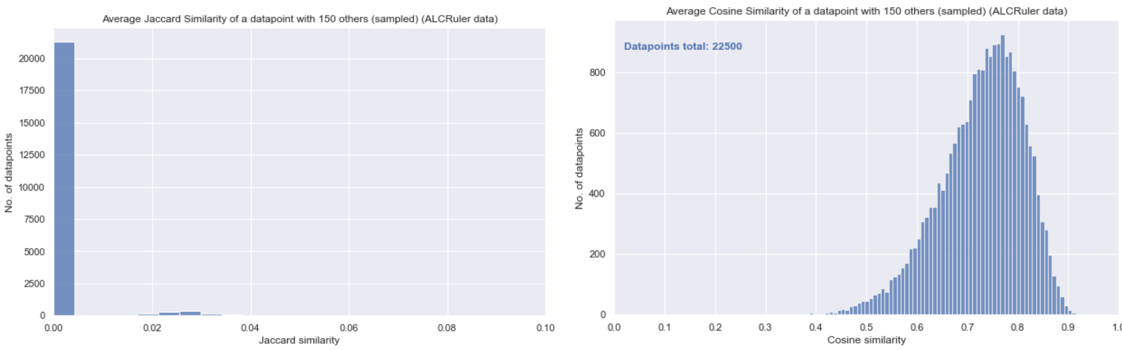
**Table B.2: Example of high cosine textual similarity.**

| | Cosine similarity (MiniLM-L6-v2) | Cosine similarity (RoBERTA large) |
|---|---|---|
| | 0.90 | 0.91 |
| **text 1** | HARRY sees GARY.<br>FIONA is somebody that eats somebody who is not quiet.<br>HARRY eats FIONA.<br>GARY is kind.<br>DAVE is not quiet.<br>HARRY is furry and kind.<br>CHARLIE is a dog.<br>Someone that is quiet and not a cat, is also one that likes only somebody who is cold.<br>Somebody who is big, is also not a cow.<br>A person that is not a cow, equivalently is someone who chases someone that is quiet.<br>One who is a thing and not young, is also a person that chases one who is not a rabbit.<br>Somebody who is not a dog, is also a person that likes only somebody that is a thing.<br>Somebody who is rough, is also a tiger and not a cow.<br>A person that is somebody that chases only someone who is a thing, is also not kind or not a rabbit.<br>One that is quiet or not kind, is also a dog.<br>Someone who is somebody who sees one who is a tiger, is also furry or not a cow.<br>One who is red, equivalently is a person who likes one that is nothing.<br>A person that is quiet and not a cow, equivalently is someone who visits somebody that is a dog. | |
| **text 2** | GARY needs CHARLIE.<br>FIONA is somebody who chases one who is not a squirrel.<br>BOB eats BOB.<br>FIONA sees ERIN.<br>GARY eats CHARLIE.<br>BOB is rough and not big.<br>HARRY is a person who needs only somebody who is not young.<br>Someone that is not quiet, is also a bear.<br>Someone who is someone who likes only one who is not blue, is also white and not red.<br>Somebody that is red, is also smart.<br>A person that is not a cow, is also not kind.<br>A person who is smart, equivalently is a dog and not big.<br>Somebody who is a squirrel and white, is also a person who visits a person who is not blue.<br>Somebody that is nice and not furry, is also nothing and not quiet.<br>Someone who is big or not a dog, is also nice.<br>Somebody that is a dog or not kind, is also not a bear.<br>A person that is quiet, equivalently is a tiger.<br>One that is not furry or not nice, is also not quiet and not young.<br>Someone who is a cow and not a bear, equivalently is not nice. | |




**(a) Jaccard** **(b) Cosine**

**Figure B.4: Text similarities for RuleTaker datasets.**




**(a) Jaccard** **(b) Cosine**

**Figure B.5: Text similarities for ALCRuler datasets.**

# REFERENCES

[1] Franz Baader. Description logics. In Sergio Tessaris, Enrico Franconi, Thomas Eiter, Claudio Gutiérrez, Siegfried Handschuh, Marie-Christine Rousset, and Renate A. Schmidt, editors, *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School 2009, Brixen-Bressanone, Italy, August 30 - September 4, 2009, Tutorial Lectures*, volume 5689 of *Lecture Notes in Computer Science*, pages 1–39. Springer, 2009.

[2] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2 edition, 2007.

[3] Samantha Bail, Bijan Parsia, and Ulrike Sattler. Extracting finite sets of entailments from owl ontologies. In *CEUR Workshop Proceedings|CEUR Workshop Proc.*, volume 745, pages 422–432, 2011. 24th International Workshop on Description Logics, DL 2011 ; Conference date: 01-07-2011.

[4] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, 2023.

[5] Ali Borji. A categorical archive of chatgpt failures, 2023.

[6] Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. Flexible generation of natural language deductions. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[8] Yihan Cao, Siyu Li, Yixin Liu, Zhiling Yan, Yutong Dai, Philip S. Yu, and Lichao Sun. A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt, 2023.

[9] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Trans. Knowl. Data Eng.*, 1:146–166, 1989.

[10] Derek Chen, Celine Lee, Yunan Lu, Domenic Rosati, and Zhou Yu. Mixture of soft prompts for controllable data generation, 2023.

[11] Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. *CoRR*, abs/2002.05867, 2020.

[12] Antonia Creswell and Murray Shanahan. Faithful reasoning using large language models, 2022.

[13] Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning, 2022.

[14] Adnan Darwiche. Sdd: A new canonical representation of propositional knowledge bases. In *International Joint Conference on Artificial Intelligence*, 2011.

[15] Ishita Dasgupta, Andrew K. Lampinen, Stephanie C. Y. Chan, Antonia Creswell, Dharshan Kumaran, James L. McClelland, and Felix Hill. Language models show human-like content effects on reasoning, 2022.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[17] Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. Gpts are gpts: An early look at the labor market impact potential of large language models, 2023.

[18] Alexander Gaskell, Yishu Miao, Lucia Specia, and Francesca Toni. Logically consistent adversarial attacks for soft theorem provers, 2022.

[19] Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd-workers for text-annotation tasks, 2023.

[20] Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An owl 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, Oct 2014.

[21] Olga Golovneva, Moya Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. Roscoe: A suite of metrics for scoring step-by-step reasoning, 2022.

[22] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data, 2018.

[23] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Luke Benson, Lucy Sun, Ekaterina Zubova, Yujie Qiao, Matthew Burtell, David Peng, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Shafiq Joty, Alexander R. Fabbri, Wojciech Kryscinski, Xi Victoria Lin, Caiming Xiong, and Dragomir Radev. Folio: Natural language reasoning with first-order logic, 2022.

[24] Jochen Hartmann, Jasper Schwenzow, and Maximilian Witte. The political ideology of conversational ai: Converging evidence on chatgpt's pro-environmental, left-libertarian orientation, 2023.

[25] Hangfeng He, Hongming Zhang, and Dan Roth. Rethinking with retrieval: Faithful large language model inference, 2022.

[26] Chadi Helwe, Chloé Clavel, and Fabian M. Suchanek. Reasoning with transformer-based models: Deep learning, but shallow reasoning. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[27] Matthew Horridge and Sean Bechhofer. The owl api: A java api for working with owl 2 ontologies. In *Proceedings of the 6th International Conference on OWL: Experiences and Directions - Volume 529*, OWLED'09, page 49–58, Aachen, DEU, 2009. CEUR-WS.org.

[28] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining inconsistencies in owl ontologies. In Lluís Godo and Andrea Pugliese, editors, *Scalable Uncertainty Management*, pages 124–137, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[29] Matthew Horridge and Simon Spero. owlexplanation package. https://github.com/matthewhorridge/owlexplanation.

[30] Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. A survey of knowledge-enhanced pre-trained language models, 2022.

[31] Yichen Jiang and Mohit Bansal. Avoiding reasoning shortcuts: Adversarial evaluation, training, and model development for multi-hop qa, 2019.

[32] Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation, 2023.

[33] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. Language models (mostly) know what they know, 2022.

[34] George Karakatsiotis, Dimitrios Galanis, Gerasimos Lampouras, and Ion Androutsopoulos. Naturalowl : Generating texts from owl ontologies in protege and in second life. In *European Conference on Artificial Intelligence*, 2008.

[35] Nora Kassner, Benno Krojer, and Hinrich Schütze. Are pretrained language models symbolic reasoners over knowledge? In *Conference on Computational Natural Language Learning*, 2020.

[36] Nora Kassner and Hinrich Schütze. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly, 2020.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[38] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *ArXiv*, abs/2205.11916, 2022.

[39] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer, 2013.

[40] Andrew Kyle Lampinen. Can language models handle recursively nested grammatical structures? a case study on comparing models and humans, 2023.

[41] Jean-Baptiste Lamy. Owlready. *Artif. Intell. Med.*, 80(C):11–28, jul 2017.

[42] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021.

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

[44] Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. Dissociating language and thought in large language models: a cognitive perspective, 2023.

[45] R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference, 2019.

[46] Yu Meng, Martin Michalski, Jiaxin Huang, Yu Zhang, Tarek Abdelzaher, and Jiawei Han. Tuning language models as training data generators for augmentation-enhanced few-shot learning, 2023.

[47] Steven T. Piantadosi and Felix Hill. Meaning without reference in large language models, 2022.

[48] Ben Prystawski, Michael Y. Li, and Noah D. Goodman. Why think step by step? reasoning emerges from the locality of experience, 2023.

[49] Shuofei Qiao, Yixin Ou, Ningyu Zhang, Xiang Chen, Yunzhi Yao, Shumin Deng, Chuanqi Tan, Fei Huang, and Huajun Chen. Reasoning with language model prompting: A survey, 2023.

[50] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[51] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022.

[52] Emily Reif, Minsuk Kahng, and Savvas Petridis. Visualizing linguistic diversity of text datasets synthesized by large language models, 2023.

[53] Sebastian Rudolph. Foundations of description logics. In *Reasoning Web*, 2011.

[54] Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. Prover: Proof generation for interpretable reasoning over rules. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[55] Soumya Sanyal, Zeyi Liao, and Xiang Ren. Robustlr: Evaluating robustness to logical perturbation in deductive reasoning, 2022.

[56] Soumya Sanyal, Harman Singh, and Xiang Ren. Fairr: Faithful and robust deductive reasoning over natural language, 2022.

[57] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought, 2023.

[58] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage?, 2023.

[59] Viktor Schlegel, Kamen V. Pavlov, and Ian Pratt-Hartmann. Can transformers reason in fragments of natural language?, 2022.

[60] Sihyeon Seong, Yegang Lee, Youngwook Kee, Dongyoon Han, and Junmo Kim. Towards flatter loss surface via nonmonotonic learning rate scheduling. In *Conference on Uncertainty in Artificial Intelligence*, 2018.

[61] Tianmin Shu, Abhishek Bhandwaldar, Chuang Gan, Kevin A. Smith, Shari Liu, Dan Gutfreund, Elizabeth Spelke, Joshua B. Tenenbaum, and Tomer D. Ullman. Agent: A benchmark for core psychological reasoning, 2021.

[62] Leslie N. Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2015.

[63] Adam Sobieszek and Tadeusz Price. Playing games with ais: The limits of gpt-3 and similar large language models. *Minds Mach.*, 32(2):341–364, jun 2022.

[64] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, and Abu Awal Md Shoeb et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models, 2022.

[65] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. *CoRR*, abs/2012.13048, 2020.

[66] Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science, 2022.

[67] A. M. TURING. COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460, 10 1950.

[68] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[69] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers, 2021.

[70] Taylor Webb, Keith J. Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models, 2023.

[71] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

[72] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[73] Laura Weidinger, John F. J. Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zachary Kenton, Sande Minnich Brown, William T. Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William S. Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Ethical and social risks of harm from language models. *ArXiv*, abs/2112.04359, 2021.

[74] Peter West, Chandra Bhagavatula, Jack Hessel, Jena D. Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic knowledge distillation: from general language models to commonsense models, 2022.

[75] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[76] Jiayang Wu, Wensheng Gan, Zefeng Chen, Shicheng Wan, and Hong Lin. Ai-generated content (aigc): A survey, 2023.

[77] Zonglin Yang, Xinya Du, Rui Mao, Jinjie Ni, and Erik Cambria. Logical reasoning over natural language as knowledge representation: A survey, 2023.

[78] Fei Yu, Hongbo Zhang, Prayag Tiwari, and Benyou Wang. Natural language reasoning, a survey, 2023.

[79] Ping Yu, Tianlu Wang, Olga Golovneva, Badr Alkhamissy, Gargi Ghosh, Mona Diab, and Asli Celikyilmaz. Alert: Adapting language models to reasoning tasks, 2022.

[80] Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion models in generative ai: A survey, 2023.

[81] Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. On the paradox of learning to reason from data, 2022.

[82] Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. Why does chatgpt fall short in answering questions faithfully?, 2023.

[83] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, Hao Peng, Jianxin Li, Jia Wu, Ziwei Liu, Pengtao Xie, Caiming Xiong, Jian Pei, Philip S. Yu, and Lichao Sun. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt, 2023.

[84] Pei Zhou, Rahul Khanna, Bill Yuchen Lin, Daniel Ho, Jay Pujara, and Xiang Ren. Rica: Evaluating robust inference capabilities based on commonsense axioms. In *Conference on Empirical Methods in Natural Language Processing*, 2020.