Vrije Universiteit Amsterdam

Honours Programme, Formal Logic in AI

Explanations with Models

Author: Szymon Czternasty (2774641)
Supervisor: Dr Patrick Koopmann

March 23, 2024

Abstract

Ontologies are fundamental to knowledge representation, spanning disciplines from artificial intelligence to bioinformatics. Despite their significance, ontologies often suffer from readability issues, limiting their accessibility and usability. This paper proposes a system to automate the creation and presentation of ontology models for specific concepts, aiming to enhance readability. Leveraging Description Logic and the Tableau Method for EL, the system unfolds TBoxes, applies transformation rules, populates the ABox, and generates user-friendly graphs. By bridging the gap between complex ontologies and human understanding, the proposed system empowers researchers and practitioners across diverse fields.

1 Introduction

Ontologies serve as the backbone of knowledge representation and reasoning, underpinning a wide array of disciplines from artificial intelligence to bioinformatics, and beyond into fields as diverse as finance, healthcare, and engineering. These formal frameworks provide a structured means to define concepts and their interrelationships, essential for facilitating comprehension and inference in complex domains. Despite their pivotal role, ontologies often suffer from a lack of readability, hindering the accessibility and reuse of their content. This readability challenge poses a significant barrier to researchers, practitioners, and stakeholders seeking to leverage ontological knowledge for various applications.

This paper proposes a system capable of creating and presenting models of an ontology for a given concept. The importance of such a system lies in its potential to enhance the efficiency and accuracy of knowledge representation and reasoning processes across diverse applications. By enabling the automated generation of tailored models for specific ontologies and concepts, the system empowers researchers and practitioners from various backgrounds. Users with limited expertise in ontology languages can now grasp the essential knowledge embedded within complex ontologies. This newfound accessibility allows them to leverage this rich resource in their respective fields, fostering innovation and collaboration.

The paper delves into the details of the proposed system, outlining its architecture and core functionalities. It provides a clear explanation of the system's internal components and how they interact to generate user-friendly models. Additionally, the paper showcases the system's real-world application on various ontologies. These demonstrations serve to illustrate the system's effectiveness in identifying non-empty models, which are crucial for meaningful reasoning processes.

2 Description Logic

Description Logic (DL) serves as a foundational formalism in knowledge representation, offering a structured language for defining and reasoning about concepts, relationships, and constraints within ontologies. At its core, DL provides a rich set of syntactic constructs that enable the precise specification of domain knowledge. Description Logic is closely connected to modal logic. Classical Kripke models can be extended to Description

Logic by simply using different edge types.

Description Logic utilizes three key elements to represent knowledge within an ontology:

- Concepts: These represent classes or categories of entities within the domain. For example, in a bioinformatics ontology, "Protein" and "Enzyme" could be concepts.
- Roles: These represent relationships between concepts. They describe how instances of one concept relate to instances of another. Examples of roles include "hasPart" or "regulates" in the bioinformatics domain.
- Individuals: These represent specific instances within a concept.

2.1 DL semantics

The semantics of the Description Logic is grounded in first-order interpretations. In DL, an interpretation is defined as a tuple, denoted as $I = (\Delta^I, \cdot^I)$, where Δ^I represents the domain of interpretation, a non-empty set. The interpretation function $\cdot^I$ assigns meanings to names within the domain: concepts are interpreted as sets, roles as binary relations, and individuals as elements within the domain. Specifically, each concept A in the DL's concept language is interpreted as a subset A^I of the domain Δ^I, each role r is interpreted as a binary relation r^I between elements of Δ^I, and each individual a is mapped to an element a^I within the domain Δ^I. This framework forms the basis for understanding the logical structure and relationships within Description Logic.

2.2 Knowledge Base

A DL knowledge base is typically composed of two parts:

- TBox (Terminology Box): The TBox captures the general knowledge about the domain in a set of axioms. These axioms define relationships between concepts using concept descriptions and specify constraints on how concepts can be combined. The TBox consists of axioms that define the terminology of the ontology, specifying relationships between concepts and roles. These axioms include:
  - General Concept Inclusions: Asserting that one concept is subsumed by another, written as $C \sqsubseteq D$.
  - Equivalence Axioms: Asserting that two concepts are equivalent, written as $C \equiv D$, meaning C is equivalent to D.
- ABox (Assertion Box): The ABox contains assertions about specific individuals within the domain. These assertions state which individuals belong to which concepts and relate them using roles.

The TBox and ABox work together to represent knowledge in an ontology. The TBox provides the framework for defining concepts and their relationships, while the ABox populates this framework with specific instances.

2.3 $\mathcal{EL}$

To avoid the problem of non-termination and high complexity, the system implemented in this paper uses less expressive Description Logic $\mathcal{EL}$. $\mathcal{EL}$ is Description Logic with restricted syntax for improved computational efficiency. Here are the key features of $\mathcal{EL}$:

- Concepts: Concepts in $\mathcal{EL}$ can only be atomic (basic concept names) or formed by conjunction ($\sqcap$). $\mathcal{EL}$ allows only for existential restrictions ($\exists R$).
- No Negation or Universal Quantification: Negation ($\neg$) and universal restrictions ($\forall R$) are not permitted in $\mathcal{EL}$ descriptions.

This limited syntax of $\mathcal{EL}$ makes reasoning about ontologies very efficient, making it suitable for large-scale knowledge bases.

3 Problem

Despite their critical role in knowledge representation and reasoning, ontologies often present readability challenges, limiting their accessibility and usability across various domains. While ontologies serve as the backbone for understanding complex concepts and their interrelationships, their intricate structures can pose significant barriers, particularly for users with limited expertise in ontology languages.

Current Limitations:

- Readability: Existing ontologies are often expressed in the form of code, posing a significant barrier to their comprehension and adoption.
- Accessibility: Users with limited knowledge of Description Logic struggle to grasp the essential knowledge encoded within ontologies.
- Limited Leverage: The full potential of ontologies for reasoning and knowledge representation tasks remains unrealized due to accessibility constraints.

The goal of this paper is to address the limitations above and implement an efficient algorithm for creating and presenting models of specific concepts in a given ontology. Existing approaches often require substantial manual effort and expertise in ontology languages, hindering widespread adoption and utilization.

There are three main assumptions made for the implementation to ensure the proper functioning:

- All axioms are in $\mathcal{EL}$. The implementation works only for $\mathcal{EL}$, namely utilizes only conjunction, existential quantifier, and T concept. Nevertheless, the algorithm is able to handle TBoxes containing all the other logical operators and concepts.
- Provided TBox is acyclic and contains axioms in the form of general concept inclusions ($A \sqsubseteq B$) and eqivalence axioms ($A \equiv B$). However, there is a maximum depth parameter that stops the algorithm whenever maximum depth is researched. It is to address the problem of non-termination in case there is a cycle in the ontology.

- Equivalence axioms ($A \equiv B$) are evaluated as two general concept inclusions. If there is $A \equiv B$, it is treated by the system as $A \sqsubseteq B$ and $B \sqsubseteq A$

The key requirement of the implementation is to deploy a system capable of creating and presenting models for a given ontology O and concept C. Given the relevant input, the code should be able to generate a model that should be presented in a clear and user-friendly format, facilitating comprehension for users with varying levels of expertise in Description Logic. The parameters that are to be manipulated by users are concept name, ontology name, and the maximum depth.

By achieving the project objectives, the proposed system seeks to bridge the gap between complex ontologies and human understanding. This will empower researchers and practitioners to leverage the rich knowledge embedded within ontologies, fostering innovation and collaboration across diverse fields.

4 Related Work

The field of automated reasoning and ontologies has witnessed significant advancements, with numerous studies focusing on enhancing the efficiency and effectiveness of algorithms in Description Logic by introducing model creation. In this section, an overview of the relevant related work in this domain is provided.

4.1 Tableau Method

Tableau algorithms have been extensively studied as a fundamental approach for automated reasoning in DL. Works by Baader and Sattler (1999)[3] and others have laid the theoretical foundations of tableau algorithms, elucidating their semantics, syntax, and practical implementation. These studies have contributed to a deep understanding of tableau-based reasoning techniques and their applications in knowledge representation systems.

Baader and Sattler's paper [3] serves as a foundational resource for comprehending tableau-based reasoning algorithms in Description Logic, primarily focusing on methods for testing satisfiability. While their work primarily targets the determination of satisfiability, this paper diverges by employing similar principles to develop an implementation aimed at generating models for explanations. By leveraging the theoretical foundations and practical insights laid out by Baader and Sattler, this project adapts tableau-based reasoning techniques to fulfill a distinct objective: constructing models tailored for explanatory purposes.

4.2 Visual Models

Visual models play a pivotal role in elucidating the justification of reasoning processes in OWL ontologies. In "On the Eve of True Explainability for OWL Ontologies: Description Logic Proofs with Evee and Evonne"[2] the authors introduce Evee and Evonne, tools designed to enhance the explainability of OWL ontologies by providing visual proofs based on Description Logic (DL) reasoning. These tools generate comprehensible visual representations of DL proofs, facilitating a deeper understanding of entailments and inferences within ontologies. Similarly, in "Why Not? Explaining Missing Entailments with

Evee"[1] the authors extend this concept by focusing on explaining missing entailments. They introduce Evee as a tool specifically tailored to identify and explain inconsistencies and omissions in DL reasoning, offering intuitive visualizations to aid users in grasping the reasons behind missing entailments. Both papers underscore the significance of visual models in enhancing the transparency and interpretability of DL reasoning, thereby advancing the field toward true explainability in OWL ontologies.

Although the goals of the plugins are different from the goal of this paper, the idea of visual explanations remains the same. As Protégé software offers only a Black Box as a justification of the inferences, sometimes it is impossible to understand the reasoning behind it, similar to the case of extracting knowledge from ontology. Regardless of the purpose, both publications similar to this paper utilize visual representation as a way of improving reasoning over ontologies.

5 Methodology

In order to construct a graph for a given concept and ontology three major steps need to be performed. First, TBox from a given ontology has to be unfolded resulting in one complex concept, then it is passed through a tableau algorithm to obtain an ABox that in the end can be sent to the graph generation algorithm. The following subsections will elaborate on each step of this process.

5.1 Unfolding TBox

Given the concept and ontology, the process of unfolding TBox involves retrieving all the GCI where the particular concept is on the left side of the subsumption and all the equivalence axioms that contain the concept on either side. Then the given concept and all the concepts retrieved from the TBox are combined together with the use of conjunction. For instance given concept A and TBox that contains $A \sqsubseteq B$ and $A \sqsubseteq C$, the outcome of unfolding process is $A \sqcap B \sqcap C$

5.2 Tableau Method for $\mathcal{EL}$

The tableau method begins with adding the whole complex concept to the ABox. Since the project utilizes only $\mathcal{EL}$ the tableau method consists of two main transformation rules. The first rule handles conjunction, if the ABox contains $(C_1 \sqcup C_2)(x)$ but it does not contain both $C_1(x)$ and $C_2(x)$ then $C_1(x)$ and $C_2(x)$ are added to the ABox. The second rule handles existential restriction, if the ABox contains $(\exists R.C)(x)$, but there is no individual name z such that $C(z)$ and $R(x, z)$ are in this ABox, then $C(y)$ and $R(x, y)$ where y is an individual name not occurring in A[3].

5.3 Graph Generation

The graph generation algorithm iterates over the ABox and creates a node when it comes across $C(z)$ and an edge between the corresponding nodes when it comes across $R(x, y)$. The edges are directed from the first entry to the second entry of the role. Figure 1 and Figure 2 illustrate example graphs from GALEN ontology.

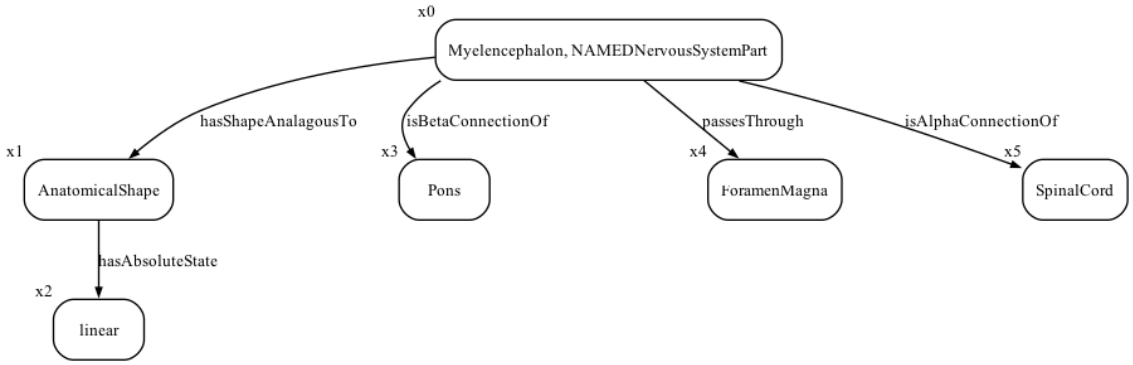

Figure 1: Explanation graph for Myelencephalon

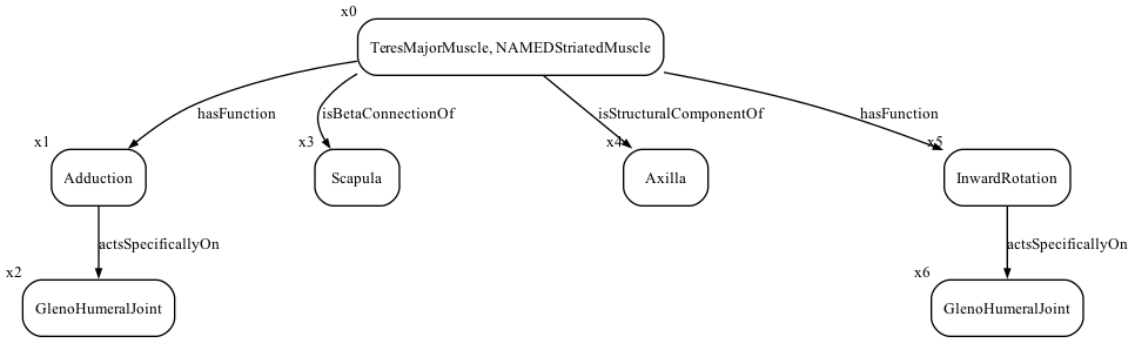

Figure 2: Explanation graph for Teres Major Muscle

6 Code structure

6.1 Loading Ontology

The LoadOntology.py file consists of the loadOntology function which takes the name of the ontology as an input and utilizes the dl-lib library for loading ontology and retrieving TBox and its axioms. The function returns a list of axioms in the TBox.

6.2 Unfolding TBox

The UnfoldTBox.py file consists of unfold_tbox and unfold function. The former takes the concept, TBox, and a maximum depth as inputs, iterates over the TBox, and searches for relevant axioms as described in the methodology section. Whenever depth reaches the maximum parameter the process stops the end returns the current complex concept to ensure termination, otherwise, it returns the complex concept after the algorithm goes over the whole TBox. The latter function is called recursively from the first function and itself to unfold concepts. It takes the same parameters as the former function and returns unfolded concepts according to the technique described in the methodology section. Whenever the functions come across unsupported operators, they are simply ignored and the algorithms continue.

6.3 Tableau Method for $\mathcal{EL}$

The TabeauMethod.py file consists of two functions, one responsible for setting initial ABox and the other for transformation rules. The former takes the complex concept as input, adds it to the ABox dictionary, and calls the latter function, then it returns the transformed ABox. The latter function according to transformation rules, rewrites the concepts from the ABox assigning the corresponding indexes as keys in the ABox dictionary. There are two variants of the keys, it is either in the form of x1, x2, ... or tuple representing relation, for instance (x1,x2).

6.4 Graph Generation

GraphGeneration.py file consists of a single function that generates the graph, using the graphviz library, given an ABox dictionary. First the ABox dictionary is formatted and then depending on the key type either node or edge is created, similarly to the description in the methodology section. The function returns a graph that is ready to be plotted.

6.5 Main

Main.py file introduces a very simple terminal user interface that allows for entering the name of the ontology, the name of the concept, and the maximum depth. Once the user enters the data, all the above functions are called one by one and in the end graph is rendered, saved, and displayed.

7 Conclusion

In conclusion, this paper has presented a comprehensive system aimed at addressing the challenges associated with the readability and accessibility of ontologies. By proposing a method for creating and presenting models of specific concepts within ontologies, the project has demonstrated the potential to enhance knowledge representation and reasoning processes across various domains.

The system leverages Description Logic and employs the tableau method, tailored specifically for the less expressive EL, to unfold complex concepts within ontologies. Through this approach, intricate ontology structures become more user-friendly by facilitating the generation of models.

By focusing on the practical implementation of the proposed system, the paper has outlined the methodology involved in unfolding TBox, applying the tableau method for EL, and generating intuitive graphs for enhanced visualization of ontological concepts and relationships.

Furthermore, the paper sets the stage for some future implementation of systems supporting more expressive Description Logics like $\mathcal{ALC}$, addressing the issue of non-deterministic disjunction.

Overall, the proposed system represents an improvement in the accessibility and usability of ontological knowledge, empowering researchers and practitioners to leverage ontologies more effectively in diverse fields. Through enhanced comprehension and accessibility, this project paves the way for fostering innovation, collaboration, and advancement across various domains reliant on ontological knowledge.

References

[1] C. Alrabbaa, S. Borgwardt, T. Friese, P. Koopmann, and M. Kotlov. Why not? explaining missing entailments with evee. In O. Kutz, C. Lutz, and A. Ozaki, editors, *DL 2023 Description Logics 2023*, CEUR Workshop Proceedings, pages 1–13. CEUR-WS.org, 2023. Funding Information: This work was supported by DFG grant 389792660 as part of TRR 248 – CPEC (see https: //perspicuous-computing.science). Publisher Copyright: © 2023 Copyright for this paper by its authors.; 36th International Workshop on Description Logics, DL 2023 ; Conference date: 02-09-2023 Through 04-09-2023.

[2] C. Alrabbaa, S. Borgwardt, T. Friese, P. Koopmann, J. Méndez, and A. Popovič. On the eve of true explainability for owl ontologies: Description logic proofs with evee and evonne. In O. Arieli, M. Homola, J. Jung, and M.-L. Mugnier, editors, *DL 2022 - Proceedings of the 35th International Workshop on Description Logics, co-located with Federated Logic Conference, FLoC 2022*, volume 3263 of *CEUR Workshop Proceedings*. CEUR-WS, 2022. 35th International Workshop on Description Logics, DL 2022 ; Conference date: 07-08-2022 Through 10-08-2022.

[3] F. Baader and U. Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, volume 1847 of *Lecture Notes in Artificial Intelligence*, pages 1–18, St Andrews, Scotland, UK, 2000. Springer-Verlag.