DEE - Defeasible Entailment and Explanations

Project Proposal

Chipo Hamayobe
hmychi001@myuct.ac.za
University of Cape Town
Cape Town, South Africa

Orefile Morule
mrlore001@myuct.ac.za
University of Cape Town
Cape Town, South Africa

ABSTRACT

Defeasible entailment is a non-monotonic type of reasoning that allows us to make conclusions with exceptions to general rules. It is a way to formalize human reasoning where previous inferences can be withdrawn when new knowledge is introduced. This addresses some of the limitations of classical reasoning. In any logical system, explanations are important for presenting why a particular entailment holds and are a key part of reasoning systems. Even though explanations have been studied extensively in classical reasoning, such has not been the case for defeasible reasoning systems, especially such as those proposed by Kraus, Lehmann, and Magidor KLM. Apart from classical, our project also aims to implement and determine entailment as advanced by the Rational Closure algorithm and then develop tools that will provide justifications for why those inferences hold within a particular knowledge. This will enhance our comprehension of explanations in classical and defeasible reasoning systems and could serve as a foundation for practical explanation services, such as debugging tools for complex and/or problematic knowledge bases.

CCS CONCEPTS

• **Theory of computation → Automated reasoning**; • **Computing methodologies → Nonmonotonic, default reasoning and belief revision**.

KEYWORDS

artificial intelligence, knowledge representation and reasoning, propositional logic, klm framework, defeasible entailment, rational closure, justifications, explanations

1 INTRODUCTION

One definition of *Artificial Intelligence (AI)* involves creating machines that can perform tasks requiring human-like intelligence. While humans can naturally derive information from existing knowledge, machines require more structured methods to represent and manipulate information. For example, a simple statement like *'it is sunny today'* is easily understood by humans, but machines need more rigorous ways to deduce its logical truthfulness and related inferences.

Knowledge Representation and Reasoning (KRR) is a key area of research in *AI* that focuses on representing knowledge about the world, deducing new knowledge from it, and balancing expressiveness with computational efficiency. It involves expressing and maintaining information about a domain using clear language [6]. In knowledge-based systems, information about a domain or world is represented in a symbolic structure called a *knowledge base*. This information is expressive and declarative. *Entailment* or *reasoning* involves applying rules and manipulations to the knowledge base to derive new conclusions about that domain [9].

KRR can be implemented using both logic-based and non-logic-based systems. One way to represent knowledge is through classical propositional logic, which assigns truth values to simple statements called *propositions*. These propositions can be combined using operators to form more complex logical statements. A logical system consists of a language with syntax and semantics, as well as a reasoning procedure [1, 6]. However, classical deduction systems are expressively limiting due to their *monotonicity* property which dictates that any derived inferences can never be withdrawn even when new conflicting explicit propositions have been added to the knowledge base.

Defeasible entailment is *non-monotonic*, meaning that previous inferences can be withdrawn when new knowledge is introduced. Unlike classical entailment, there is no single correct way to reason with defeasible knowledge. Kraus, Lehmann, and Magidor (KLM) [10] proposed a set of properties for making conclusions with defeasible knowledge. One such proposal, *Rational Closure*, will be the primary focus of this project for determining entailment. Compared to classical explanations, less research has been conducted on *defeasible explanations* [5]. This project will therefore also seek to investigate and define the concepts of defeasible explanations flowing from the developed algorithms for entailment with Rational Closure.

Finally, the tools independently developed for the two reasoning systems will be integrated into a common library to provide both classical and defeasible entailment and explanation services. We aim to provide such justifications in simple and clear human language so the entire tool could be used as a debugging mechanism for defeasible logic systems.

2 BACKGROUND

2.1 Propositional Logic

2.1.1 Overview. Propositional logic is a formal system used to reason about knowledge or information. It abstracts natural language into a formal language and serves as the foundation for this project and related work that might follow. A set of connectives and statements defines propositional logic. More complex statements can be created by combining statements using these connectives.

The truth of any composite statement depends on the truth of the constituent base statements and the interpretation of the connectives employed in its construction. This allows for the analysis of the truthfulness of any statement separate from its supposed intuitive meaning in natural language. It also defines how to reason about a set of statements formally and what can be derived from them[1, 9].

2.1.2 Syntax. The language of propositional logic, symbolised by $\mathcal{L}$, is constructed from a collection of the most basic units of knowledge known as propositional *atoms*. Atoms can have truth values of either true (T) or false (F) and their truthfulness does not necessarily have to make sense in natural language. A propositional atom such as `fly` can therefore be symbolised as f. The set of all such propositional atoms is finite, denoted as $\mathcal{P} = \{p, q, r, s, \dots\}$.

The unary negation operator ($\neg$) and binary connectives ($\wedge, \vee, \rightarrow, \leftrightarrow$) are used to form propositional formulas like $\neg(q \wedge r) \rightarrow r$. The set of all such formulas over $\mathcal{P}$ is called the propositional language $\mathcal{L}$ [1]. Finally, the constants $\top, \bot \in \mathcal{L}$ read as "top" to mean a tautology and "bottom" to mean the opposite respectively are also employed when required.

2.1.3 Semantics. The meaning of truth is defined by the semantics of a logic, which enables systematic and meaningful analysis of its language [9]. A *knowledge base*, $\mathcal{K} \subseteq \mathcal{L}$, is a finite set of propositional formulas such as $\{\alpha, \beta, \gamma, \dots\}$.

A *valuation*, also referred to as an *interpretation*, $\mathcal{I}$, is defined as a function $\mathcal{I} : \mathcal{P} \mapsto \{T, F\}$ which assigns each propositional atomic statement a value of T (true) or F (false) respectively. If a valuation assigns the truth value *true* to an atom, it is said to satisfy the atom and satisfaction is symbolised by $\Vdash$. For example, if we have $\mathcal{P} = \{r, s\}$ and interpretation is $\mathcal{I}(r) = T$ and $\mathcal{I}(s) = F$ then $\mathcal{I} \Vdash (r \vee s)$. It should be stated that this expression can be extended to any formula in $\mathcal{L}$.

Models of a formula α are interpretations that satisfy it and the set of all such models is represented by $Mod(\alpha)$. An interpretation is considered a model of a knowledge base $\mathcal{K}$ if it satisfies all the formulas in $\mathcal{K}$. If the set of models of $\mathcal{K}$, $Mod(\mathcal{K})$, is a subset of the set of models of a statement β, $Mod(\beta)$, then we say that $\mathcal{K}$ entails β, represented as $\mathcal{K} \models \beta$. This forms the foundation of a fundamental reasoning system. By expressing knowledge as a set of statements in a knowledge base, we can determine if it implies other propositional statements.

Example 2.1. If the knowledge base contains the following propositions:

- birds fly ($b \rightarrow f$)
- ostriches are birds ($o \rightarrow b$)

We can conclude that since *ostriches are birds*, *ostriches fly*. We can formally represent entailment as since $Mod(\mathcal{K}) \subseteq Mod(f)$, $\mathcal{K} \models o \rightarrow f$.

2.2 Classical Reasoning

Example 2.2. Assuming we have the following atoms representing the propositions "being a bird", "having wings", "able to fly", "being a penguin" and "being an ostrich" respectively:

$$\mathcal{P} = \{b, w, f, p, o\}$$

We can codify this information in the knowledge base $\mathcal{K}$ as:

- birds have wings ($b \rightarrow w$)
- birds fly ($b \rightarrow f$)
- penguins are birds ($p \rightarrow b$)
- ostriches are birds ($o \rightarrow b$)

We can now semantically represent $\mathcal{K} = \{b \rightarrow w, b \rightarrow f, p \rightarrow b, o \rightarrow b\}$. We can reason logically and consequently infer using the principles of classical entailment that since penguins are birds, penguins fly ($p \rightarrow f$). This deduction is correct and acceptable based on the current knowledge base. But we know that penguins can not fly ($p \rightarrow \neg f$), despite being birds.

Despite this new conflicting information, we can not withdraw our previous inference that penguins fly ($p \rightarrow f$) due to the non-monotonicity property of classical reasoning. The only logical inference we can draw based on this new information is that penguins do not exit, which is not true in the real world because we know that penguins do exist.

To account for the fact that penguins are a non-flying class of birds, we need a different form of reasoning. While we could modify the knowledge base to address penguins as an exception, this is not practical as other exceptions like ostriches ($o \rightarrow \neg f$) would also require changes to the knowledge base structure. Classical entailment is denoted by $\models$.

2.3 Defeasible Reasoning

Classical reasoning systems have a significant drawback in that they cannot depict *typicality*. This makes it challenging to concisely represent any exceptional knowledge. Defeasibility on the other hand allows us to withdraw previous conclusions when new information comes to light. If we add this new explicit information (penguins do not fly) to the knowledge base, $\mathcal{K}$ is now represented as $\{b \rightarrow w, b \rightarrow f, p \rightarrow b, o \rightarrow b, p \rightarrow \neg f\}$.

There are no models of $\mathcal{K}$ in which p can be true and still infer the existence of penguins; therefore no further meaningful reasoning can be made about penguins.

Our main focus is making logical decisions based on an internal representation of knowledge, following the *AI* principles of knowledge-based intelligent agents. Example 2.2 shows that classical propositional logic alone cannot convey the idea of *typicality*. We want to express that 'birds *typically* fly' while still being able to reason about the existence of penguins.

The solution is non-monotonic reasoning, which includes the principles of "common sense" or *defeasible entailment*. This allows for the retraction of previous deductions when conflicting information is added to the knowledge base [2]. It is very important to note that *entailment* is a high-level concept, similar to logical equivalence and consequence. The idea of $\models$ is not defined within $\mathcal{L}$, but rather it represents what can be inferred from formulas in $\mathcal{L}$.

While other variations of defeasible reasoning have been proposed, we will focus on the *KLM Framework* by Kraus, Lehmann, and Magidor[10] specifically on *Rational Closure*. The next section will discuss this framework in more detail

2.3.1 The KLM Framework. We expand classical propositional logic by adding the representation for *typicality* through the defeasible connective, $\mid\sim$, which is similar to the classical implication one, $\rightarrow$. In defeasible reasoning, classical statements such as $\alpha \rightarrow \beta$ (α *implies* β) can now be written within a *typicality* context as $\alpha \mid\sim \beta$ (α *typically implies* β).

Therefore based on example 2.2 above, we can express *'birds typically fly'* as $b \mid\sim f$. We also introduce the concept of defeasible entailment, represented by $\mid\approx$, which is comparable to the classical $\models$. We aim to find reasonable non-monotonic entailment forms that

allow for the withdrawal of deductions when new contradictory knowledge is added. These entailment relations are established by a set of postulates and further refined to define more specific classes of entailment [4, 10, 11]. We will focus mainly on the Rational Closure type of entailment for explanations and justification.

2.4 Rational Closure

Rational Closure is a prototypical form of defeasible reasoning in the KLM framework that is very conservative in abnormal cases. Lehmann and Magidor [11] suggest that any other reasonable form of entailment, while it may be more "adventurous" in its deductions, should at least support the assertions in the Rational Closure of the corresponding knowledge base.

The Rational Closure of a knowledge base can be computed in two main ways. The option for this project is one that employs an algorithmic approach that ranks statements in the knowledge base [11]. We will use this method in our implementation because we need access to the actual formulas within the knowledge base.

Lets assume $\mathcal{K} = \{b \mid\sim f, p \rightarrow b, p \mid\sim \neg f\}$. We should actually express this as $\mathcal{K} = \{b \mid\sim f, \neg(p \rightarrow b) \mid\sim \bot, p \mid\sim \neg f\}$ and then have $\mathcal{K}_m = \{b \rightarrow f, \neg(p \rightarrow b) \rightarrow \bot, p \rightarrow \neg f\}$. We know that $\mathcal{K}_m \models \neg p$ and p is exceptional for $\mathcal{K}$, therefore $\mathcal{K}_m \not\models \neg p$ is also valid if so b is not exceptional for $\mathcal{K}$.

We define $\mathcal{R}(\mathcal{K})$ to represent the set of statements in $\mathcal{K}$ with exceptional antecedents. This allows us to create a sequence of knowledge bases $\mathcal{R}_0^{\mathcal{K}}, \mathcal{R}_1^{\mathcal{K}}, \mathcal{R}_2^{\mathcal{K}}, \ldots, \mathcal{R}_n^{\mathcal{K}}$ for any knowledge base $\mathcal{K}$. Earlier knowledge bases in the sequence contain more defeasible statements than later ones. We let $\mathcal{R}_0^{\mathcal{K}} = \mathcal{K}$ and $\mathcal{R}_{i+1}^{\mathcal{K}} = \mathcal{R}(\mathcal{R}_i^{\mathcal{K}})$ The last knowledge base $\mathcal{R}_\infty^{\mathcal{K}}$ contains only non-retractable statements (if not empty)

We create a ranking $\mathcal{K}_0, \mathcal{K}_1, \ldots, \mathcal{K}_\infty$ of the statements in $\mathcal{K}$ using the sequence of $\mathcal{R}$ knowledge bases. We set $\mathcal{K}_i = \mathcal{R}_i^{\mathcal{K}} + \cdots + \mathcal{R}_{i+1}^{\mathcal{K}}$ for $0 \leq i \leq n-1$ and $\mathcal{K}_\infty = \mathcal{R}_\infty^{\mathcal{K}}$. The ranks are disjoint and earlier ranks contain more defeasible statements than later ones.

Example 2.3. Again, assuming the knowledge base $\mathcal{K} = \{b \mid\sim w, b \mid\sim f, p \rightarrow b, p \mid\sim \neg f, c \rightarrow p\}$.

Figure 1 below presents the ranking of formulas associated with it.

0	$b \mid\sim w, b \mid\sim f$
1	$p \mid\sim \neg f$
∞	$p \rightarrow b, c \rightarrow p$

Figure 1: Rankings $\mathcal{K}_0, \mathcal{K}_1, \ldots, \mathcal{K}_\infty$ for Example 2.3

Our expectations are met by this result. The statement $p \mid\sim \neg f$ (*penguins typically do not fly*), is recognised as more typical or less defeasible than the statements $b \mid\sim f$ (*birds typically fly*) and $b \mid\sim w$ (*birds typically have wings*). It is important to note that classical information is constantly found in the infinite or boundless rank $\mathcal{K}_\infty$.

With this ranking of statements in place, we can now compute $\mathcal{K} \mid\approx (\alpha \mid\sim \beta)$ by first checking if α is exceptional with respect to $\mathcal{K}$, meaning whether $\mathcal{R}^{\mathcal{K}} \models \neg\alpha$. If α is not exceptional, our defeasible entailment result is computed as $\mathcal{R}^{\mathcal{K}} \models (\alpha \rightarrow \beta)$. If it is exceptional, we remove the lowest level from our ranking and repeat the process.

Let us take the query *chippies typically do not fly* ($c \mid\sim \neg f$) and the same knowledge base $\mathcal{K}$ as in Example 2.3. Since the *antecedent* c is exceptional for $\mathcal{K}$, we discard the propositions in the first rank $\mathcal{K}_0$ from $\mathcal{K}$. Now that c is no longer exceptional for $\mathcal{K}$, we check if $\mathcal{R}^{\mathcal{K}} \models (c \rightarrow \neg f)$. We can therefore deduce that the defeasible entailment is valid for Rational Closure since this classical entailment holds.

2.5 Explanations

Explanations give reasons for why an entailment holds from a knowledge base. This project focuses on a specific type of explanation called *justifications*, which are minimal subsets of formulas needed for an entailment to hold [8]. Let us illustrate this concept with an example.

Example 2.4. Suppose we have the following statements in our knowledge base $\mathcal{K}$:

- birds have wings ($b \rightarrow w$)
- birds fly ($b \rightarrow f$)
- tweety is a bird ($t \rightarrow b$)

If someone asks, *'does tweety fly?'*, the only statements needed to answer this question are *'birds fly'* and *'tweety is a bird'*. The statement *'birds have wings'* is irrelevant because the entailment still holds without it. However, if we didn't have the statements *'birds fly'* and *'tweety is a bird'*, the entailment would not hold.

We can define justification in relation to knowledge bases. Formally, if we have a knowledge base $\mathcal{K}$ and have a query α such that $\mathcal{K} \models \alpha$; the set of formulas $\mathcal{J}$ is a justification for $\mathcal{K} \models \alpha$ if $\mathcal{J} \subseteq \mathcal{K}$, $\mathcal{J} \models \alpha$ and for all $\mathcal{J}' \subseteq \mathcal{J}$, it holds that $\mathcal{J}' \not\models \alpha$ [13].

3 PROJECT DESCRIPTION

3.1 Overview

When reasoning, we humans are generally capable of providing corroboration to support why we believe a certain fact is true or false. For instance, if someone knows that *'birds fly'* and *'penguins are birds'*, they could logically conclude that *'penguins fly'*. To support their conclusion, they would present the two known facts as corroboration.

Explanations in reasoning systems reveal the statements in a knowledge base that have relevance to the entailment between the knowledge base and an entailed statement [5]. A simple form of explanation frequently used for classical logics is *justifications*, which are based on the idea of minimal subsets of a knowledge base that can entail a propositional formula.

While much research has been done concerning classical explanations and justifications and how to compute them, minimal work has been done in expanding this to defeasible reasoning. Specifically, for KLM-style entailment, no definition of defeasible explanation has been proposed. More specifically, there are no tools available that output explanations in a format that could serve as a debugging service for knowledge bases. Our project aims to define and develop such an explanation service.

3.2 Project Work

Our intention is to extend the algorithms for Relevant Closure to provide a means of computing explanations and then provide justifications for this type of defeasible entailment. We believe that since this algorithm reduces to a series of classical entailment computations, we can use methods for computing classical justifications to obtain a version of defeasible justification, as Chama [5] explains for Rational Closure.

Additionally, we aim to explore and provide a general definition for defeasible explanation within the KLM framework, but more specifically for Rational Closure, and possibly in the broader context of defeasible reasoning. With this definition in place, we will develop algorithms and tools that could be employed in reasoning systems providing explanations as a form of service.

3.3 Motivation

Understanding how an automated system reaches a conclusion is very advantageous. AI systems are used in crucial aspects of everyday life such as security, transportation and health services [12]. The examples we have illustrated in this proposal show a relatively small number of propositional formulas with a relatively small complexity. In practice, a knowledge base can contain hundreds or more formulas with varying complexities. In scenarios where a knowledge base is inconsistent or has an error, entailments (classical or defeasible) can be challenging to understand. A justification can help with debugging a knowledge base as it provides one with the propositional formulas that make an entailment hold or not hold. By examining the justifications for an entailment, it is possible to identify inconsistencies or contradictions in the knowledge base that may be causing the error [3].

Our analysis and implementation could lead to the development of tools that evaluate justifications and improve our understanding of entailment in KLM and other similar formalisms. This is important because explanations are a key part of reasoning systems and are not well understood. Our work could also lead to the development of a more advanced KLM-style reasoning system providing explanations as a service, such as for debugging plugins for knowledge bases.

4 PROBLEM STATEMENT

This project seeks to address the gaps in classical and defeasible reasoning literature and available tools in explaining why a particular entailment of a knowledge base is valid (or not) by analysing the ranking of statements using the Rational Closure algorithm.

4.1 Aims

As part of this project, we aim to:

- Provide the relevant theory and motivation for defeasible entailment and explanations.
- Provide an algorithmic specification for computing defeasible entailment, specifically for Rational Closure, as proposed by [10].
- Propose an abstract representation of the ranking of statements compatible with the operations performed during entailment checking.
- Implement the proposed entailment and explanations for classical reasoning.
- Implement an algorithm to explain why an entailment holds within the Knowledge base.
- Implement algorithms to verify the explanations against the entailment in classical reasoning.
- Output the results for both levels of explanations in clear and simple human language.

4.2 Research Questions

Our work in this project seeks to provide answers to the following research questions:

4.2.1 Chipo Hamayobe.

- How can the corresponding ranked statements in Rational Closure be efficiently abstracted to determine entailment by constructing and analysing the applicable algorithms?
- How can such an entailment be tested and verified that it is satisfiable for every propositional atom in any rank of a knowledge base?
- How can the constituent formulas of such an entailment be simplified so that the corresponding explanations can be understood and expressed in natural language?

4.2.2 Orefile Morule.

- How can entailment be efficiently abstracted and developed in classical reasoning to determine explanations?
- How can such an entailment be tested and verified so as to inform the efficient abstraction of atomic propositions in classical reasoning?
- How can the discovered justifications be explained and presented to anyone in simple and clear natural language?

4.2.3 Additional Research Questions. In case the research questions are answered earlier than anticipated, we have proposed additional research questions:

- Can the algorithms and tools developed for defeasible entailment using Rational Closure also be easily extended to Lexicographic Closure?
- Can the algorithms and tools developed for explanations and justifications in classical reasoning also be easily extended to defeasible reasoning?

5 PROCEDURES AND METHODS

The first part of this project will primarily be theoretical focusing more on understanding the defeasible entailment and explanations pertaining to Rational Closure. At the same time, explanations for classical reasoning will be explored as a basis for understanding the groundwork. This will be followed by possibly the logical design of the overall system and determination of the required tools and focusing more on algorithms for the ranking of statements. After that, optimized implementations of the Rational Closure versions of these algorithms will be implemented in Java. Entailment explanations in the sense of justifications in natural language will also be analysed and developed. The Knowledge Base Generation Tool (KBGT) from [7] will be used as a benchmarking and testing tool to evaluate the performance of the developed algorithms.

This project involves iterative cycles of comprehending concepts, formulating theories, and verifying their validity. While substantial progress has been made in reviewing the relevant literature, this knowledge will be refined and built upon throughout the project. The project comprises four primary phases: tasks (3), (4), (5) and (6) as outlined in Table 2.

5.1 Algorithm Adaptation

To modify the relevant algorithms, we must first thoroughly understand their workings. We will begin by reviewing the papers that introduced these algorithms and any subsequent literature that presents variations of them, with a focus on examples that can enhance our intuitive grasp. We will also examine how these algorithms relate to Rational Closure and its extension for computing justifications.

To assess the completion of this section, we will submit preliminary versions of our algorithms and tools that demonstrate progress and effectiveness in computing defeasible and classical explanations to our supervisor. If the algorithms and tools are deemed unsatisfactory, we will address any concerns and work on developing new ones.

5.2 Determination of Entailment and Explanations

The approach to formulating a definition and developing of tools for explanations is less precise. We will begin by examining existing literature on entailment and explanations for both classical and defeasible reasoning, with a particular emphasis on the definition of strong justifications [3]. As a starting point, we will investigate this definition in the context of KLM in relation to the statement rankings we previously discussed for Rational Closure. Building on this foundation, we will develop a general definition for defeasible explanations by adapting and reinforcing the conditions for classical justifications. Based on this outcome, we will develop preliminary prototypes that demonstrate our understanding of the requirements so far.

We will assess this section more subjectively by verifying if it aligns with our expectations. We will present a high-level design evaluating the suitability of our approach. To this end, we will review other papers in the field of logics that explain how explanation systems have been theorised and implemented to gain a better understanding of how this is formally accomplished.

5.3 Implementation of Algorithms

After presenting our definition of defeasible entailment and explanation, we will formally verify our extended algorithms and the previously defined extended algorithm for Rational Closure. This will involve proving their correctness by demonstrating their soundness and completeness according to our definition. If the algorithms are found to be incorrect, we will review either the algorithms or our definition based on the source of the issue.

We will then spend a considerable amount of time developing and implementing the determined algorithms for entailment and explanation. The tools independently developed for the two reasoning systems will be integrated into a common library to provide a combined debugging service. When this work is tested and verified, we will then work on outputting the explanations in simple and clear human language.

5.4 Testing and Verification

For practical purposes and easier testing, we plan to define a reasonably small knowledge base, maybe a maximum of 16 propositional formulas. Our code will also involve lots of unit tests to verify some edge case functionality.

For testing, our supervisor *Professor Thomas Meyer* and 2 or more of his colleague will perform the informal testing of the end product to ensure that the resultant tool is useful for the intended purpose. The feedback and results of this expert testing will be documented and reported accordingly.

6 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

This project has both theoretical and practical aspects. We do not anticipate any ethical issues related to user testing or privacy breaches since neither component involves such activities. Since the end product will be tested informally by the experts, including our supervisor, there is no need to request any authorisation from the University of Cape Town's Research Ethics Committees (REC).

Since we will not be using or storing any personal information, the Protection of Personal Information (POPI) Act does not apply to us. We do not have to worry about licensing and copyright issues because we are only referencing the literature on the subject according to academic standards and not using any copyrighted works.

Regarding intellectual property rights, we will be using software such as Oracle OpenJDK releases for Java and the TweetyProject library, which are licensed under the GNU General Public License (GPL) with the Classpath Exception and GNU General Public License respectively.

Additionally, the KBGT [7] is published under the MIT license, allowing free use and modification of the tool. As such, we plan to license our software under an open-source software license (likely MIT) and will follow all necessary guidelines during development.

7 RELATED WORK

An array of algorithms for calculating classical justifications have been examined and offered by Horridge [8]. The algorithms presented can be classified as either *glass-box* or *black-box*. In *glass-box* algorithms, the calculation of justification is incorporated into the reasoning algorithm, meaning justifications are computed during the reasoning process. In contrast, *black-box* algorithms operate independently of the underlying reasoning process and are calculated separately. There are trade-offs between the two kinds of algorithms in terms of effectiveness and user-friendliness, which must be weighed when selecting which one to employ.

An adaptation or remodelling of the Rational Closure algorithm for determining justifications for Rational Closure defeasible entailment is introduced by Chama [5]. This adaptation employs algorithms presented by Horridge [8] as a foundation for determining justifications. This approach mirrors the reasoning process for Rational Closure: after removing more typical ranks, we depend on classical instruments to reason about the knowledge base, but

in this respect, rather than classical entailment we utilize classical justification.

A distinct approach to defining defeasible justifications is however taken by Brewka et al. [3]. An abstract concept of a defeasible justification is introduced that is purported to work for all types of defeasible reasoning, called a *strong explanation*. One reason why justifications function well in classical reasoning is that anything entailed by a justification is also entailed by the knowledge base. However, in the defeasible case, the remainder of the knowledge base may contain information that contradicts the entailment. This is addressed by broadening the definition of a justification to encompass an extra attribute that guarantees there is no additional information in the knowledge base that conflicts with the justification [3].

8 ANTICIPATED OUTCOMES

8.1 The System

Algorithms that determine the satisfiability of any input formula for a particular knowledge base will be implemented in Java. Algorithms that employ Rational Closure for defeasible entailment will be employed for this purpose. A program, also in Java, that offers explanations for the determined interpretations in the knowledge base will also be developed that will function as a *'debugging'* tool for providing details on why the determined entailment holds or not. These will likely be command-line executables that integrate with existing tools from the TweetyProject library collection. We may also develop a web interface that could be used as a testing and demonstration tool for the underlying explanation services.

The main design challenges will involve integration with these code bases. A custom algorithm to output the explanations in a simple natural language format will be developed as the main basis of whatever desirable output. The success and acceptability of the functional system will depend on its expressivity, efficacy and computational efficiency. Unit and integration tests will be developed for the determination of these metrics.

8.2 Expected Impact

We expect to develop justification algorithms for Rational Closure by extending Chama's algorithm [5] to include justifications in clear human language. We will also formulate a definition for defeasible explanation and provide soundness and completeness proofs that connect the reasoning algorithms for Rational Closure to this declarative description. Additionally, we offer more informal explanations to aid intuitive understanding. This preliminary work will help in the development and implementation of algorithms that output explanations on both levels in clear human language to aid in the debugging of more complex and logically problematic knowledge bases. This type of work has not been explored satisfactorily yet, therefore we hope our project work will be used as a stepping stone for more research in further meaningful explanation systems.

8.3 Key Success Factors

The success of this project will be determined by whether or not we achieve these outcomes. A deeper understanding of the determination of classical and defeasible explanations will be the first goal. The project will be considered successful if we:

- conduct a comprehensive analysis of how defeasible entailment and explanations function as demonstrated and implemented by Horridge [8] and Chama [5].
- implemented, develop and explain why particular inferences hold or not using the Rational Closure type of defeasible reasoning.
- implemented, develop and explain why particular inferences hold or not in classical reasoning.
- output both levels of explanations and justifications in clear and simple human language.
- package these tools in some form of a library that can be used as a debugging plugin for other software.

9 PROJECT PLAN

9.1 Risks and Contingencies

In addition to the general risks associated with projects, this project depends on the ability to comprehend complex literature. This introduces risks such as not being able to understand the work or not being able to do so within the required time frame. Specifically, spending too much time trying to understand a particular concept can cause delays that can endanger the entire project. The risks, along with their probabilities and impacts, are provided in Appendix A as Table 3. A plan to mitigate, monitor and manage associated risks is also presented under the same Appendix as Table 4.

9.2 Resources

For the theoretical aspects of the project, it will be crucial to have access to pertinent literature such as credible research journals and textbooks. For the practical parts of the system, computers will be required to create and execute algorithms using IDEs software like Visual Studio Code and IntelliJ IDEA. Additionally, open source software such as Java (Open JDK) and the TweetyProject (https://tweetyproject.org), as well as the source code for implementing tools and algorithms in [7], will be required.

9.3 Deliverables

Please note that all the dates below relate to the year 2023.

Deliverable	Due Date
Literature Reviews	24 March
Project Proposal Draft	21 April
Project Proposal Presentation	25-28 April
Project Proposal Final	2 May
Ethics Applications (if any)	12 May
Software Feasibility Demonstration	17 July
Project Progress Demonstration	17-21 July
Final Project Paper Draft	28 August
Final Project Paper Submission	11 September
Final Project Code Submission	15 September
Final Project Demonstration	26-29 September
Project Poster	9 October
Project Website	16 October
School of IT Showcase	24 October

Table 1: Project Deliverable Deadlines

9.4 Milestones and Tasks

Please note that all the dates below relate to the year 2023.

Task	Dates
(1) Literature Review	24/02 - 24/03
(2) Project Proposal	28/03 - 02/05
(a) Draft	28/03 - 21/04
(b) Presentation	25/04 - 28/04
(c) Final	01/05 - 02/05
(3) In-depth Topic Exploration	03/05 - 26/05
(a) Do further research on defeasible and classical entailment	03/05 - 26/05
(b) Do further research on defeasible and classical explanations	03/05 - 26/05
(c) Review understanding with supervisor	24/05 - 26/05
(4) Preliminary Work	05/06 - 06/06
(a) Finalise the overall approach with supervisor	05/06 - 06/06
(5) Algorithm Implementation *Defeasible Entailment (Chipo) and Classical Explanations (Orefile)*	07/06 - 16/07
(a) Research on the best possible algorithms	07/06 - 14/06
(b) Analysis of the chosen Algorithms	15/06 - 20/06
(c) Initial Prototype and Implementation	21/06 - 05/07
(d) Analysis and Testing	06/07 - 11/07
(e) Finalisation Analysis	12/07 - 16/07
(6) Demonstrations	12/07 - 21/07
(a) Software Feasibility Demonstration	12/07 - 17/07
(b) Project Progress Demonstration	17/07 - 21/07
(7) Integration	22/07 - 08/08
(a) Integrate defeasible entailment and explanation service programs onto the chosen external tool	17/07 - 31/07
(b) Analysis and Testing	01/08 - 08/08
(8) Final Submissions	09/08 - 15/09
(b) Complete Draft of final paper	09/08 - 28/08
(b) Paper final submission	29/08 - 11/09
(c) Code final submission	09/08 - 15/09
(9) Other Tasks	16/09 - 24/10
(a) Final Project Demonstration	26/09 - 29/09
(b) Project Poster	30/09 - 09/10
(c) Project Website	16/09 - 16/10
(d) School of IT Showcase	17/10 - 24/10

Table 2: Project Milestones and Tasks

The project timeline is summarised in the Gantt Chart presented as Figure 2 in Appendix B.

9.5 Work Allocation

Task (1) has already been completed and submitted independently. The remaining work is divided into tasks as tabulated in Table 2 and is allocated as follows:

9.5.1 Chipo Hamayobe.

- Finalise the defeasible entailment theory and preliminary work under Task(4).
- All the defeasible entailment and related work under Task(5).
- Development and testing of defeasible entailment algorithms and explanations under Tasks (5) and (6).
- Integration work for entailment explanations under Task(7).
- Task (8) based on the work done on the project.

9.5.2 Orefile Morule.

- Finalise the classical explanations theory and preliminary work under Task(4).
- All the classical explanations and related work under Task (5).
- Development and testing of classical explanations algorithms and outputs under Tasks (5) and (6).
- Integration work for justifications and explanations under Task(7).
- Task (8) based on the work done on the project.

9.5.3 Collaborative Tasks.

- All the work under Task(2) will be done as a team.
- We shall also collaborate on Task(6) as part of the integration of developed tools.
- As a following process from Task(6), Task (7) will be a team effort to ensure we test the integrated software.
- All the Task(9) sub-tasks will be done as a team, to prepare for final demonstrations, presentations and a showcase to the School of IT.

REFERENCES

[1] Mordechai Ben-Ari. 2012. *Propositional Logic: Formulas, Models, Tableaux.* Springer London, London, 1, 7–47.

[2] Richard Booth, Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2015. On the Entailment Problem for a Logic of Typicality. In *Proceedings of the 24th International Conference on Artificial Intelligence* (Buenos Aires, Argentina) *(IJCAI'15).* AAAI Press, 2805–2811.

[3] Gerhard Brewka and Markus Ulbricht. 2019. Strong explanations for nonmonotonic reasoning. *Description Logic, Theory Combination, and All That: Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday* (2019), 135–146.

[4] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking Defeasible Entailment Beyond Rational Closure. In *Logics in Artificial Intelligence.* Springer International Publishing, Cham, 182–197.

[5] Victoria Chama. 2020. *Explanation for defeasible entailment.* Master's thesis. Faculty of Science, University of Cape Town, Rondebosch, Cape Town, 7700.

[6] Crina Grosan and Ajith Abraham. 2011. *Knowledge Representation and Reasoning.* Springer Berlin Heidelberg, Berlin, Heidelberg, 131–147. https://doi.org/10.1007/978-3-642-21004-4_6

[7] Joel Hamilton, Joonsoo Park, Aidan Bailey, and Thomas Meyer. 2022. *An Investigation into the Scalability of Defeasible Reasoning Algorithms.* SACAIR 2021 Organising Committee, Online, 235–251. https://protect-za.mimecast.com/s/OFYSCpgo02fL1l9gtDHUkY

[8] Matthew Horridge. 2011. *Justification based explanation in ontologies.* Ph. D. Dissertation. University of Manchester, UK.

[9] Adam Kaliski. 2020. *An Overview of KLM-Style Defeasible Entailment.* Master's thesis. Faculty of Science, University of Cape Town, Rondebosch, Cape Town, 7700.

[10] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44, 1 (1990), 167–207. https://doi.org/10.1016/0004-3702(90)90101-5

[11] Daniel Lehmann and Menachem Magidor. 1992. What does a conditional knowledge base entail? *Artificial Intelligence* 55, 1 (1992), 1–60. https://doi.org/10.1016/0004-3702(92)90041-U

[12] Warren J von Eschenbach. 2021. Transparency and the black box problem: Why we do not trust AI. *Philosophy & Technology* 34, 4 (2021), 1607–1622.

[13] S Wang. 2022. *Defeasible Justification for the KLM Framework.* Master's thesis.

Appendix A RISKS

ID	Risk	Probability	Impact
1	Poor time management	Medium	High
2	Difficulty in understanding the technical aspects in our research	Medium	Medium
3	Integration of the two project components with external tools is more challenging than expected	Medium	Medium
4	Disagreement between partners in the joint-project phase of the project	Medium	Medium
5	Project partner does not successfully complete their part of the project	Medium	Medium
6	Supervisor is unavailable	Low	Medium
7	Source code or project paper gets corrupted or is otherwise irretrievable	Low	Medium
8	Inability to access internet resources due to load shedding	High	High

Table 3: Risk Identification

ID	Mitigation	Monitoring	Management
1	Make use of project management tools such as our Gantt chart and a Kanban board to keep track of tasks and their timelines.	Regularly check the Gantt chart and Kanban board to see if we are on track to meet deadlines.	Reduce our scope if we have sufficiently covered the fundamental goals of a particular phase to ensure we meet the next deadline(s).
2	Engage with multiple resources as soon as possible and collaborate with project partners when dealing with the same concepts.	Regularly engage with the supervisor to ensure that your understanding is correct.	Request a meeting with our supervisor to help clear up misunderstandings.
3	Take a considerable amount of time to understand how the tools work and note what any relevant limitations are.	As we build our algorithms incrementally, test whether integration is possible after each increment.	Resort to online forums AI tools to help solve our problem.
4	Ensure that your project partner knows what you are doing at each stage of your individual project even way before the collaboration phase.	Hold team meetings after a milestone has been reached to assess each other's progress.	Meet with our supervisor to help mediate the matter.
5	Make sure that the scope given is reasonable.	Keep track of deadlines as stated in the Gantt chart and request a change in scope if it is inevitable that the current goals will not be matched.	Ensure that each partner project still works independently of the unfinished project.
6	Request the supervisor's schedule well in advance.	Inform the supervisor of our intention to have a meeting two weeks prior and regularly check if they will still be available.	Reschedule meetings and work on parts of the project that we can without the supervisor's assistance.
7	Backup our electronic documents on the cloud.	Back-up work to the cloud each night after editing.	Restore the latest backup and rewrite work from that point.
8	Set up a schedule for working on the project such that you do not need to work during load-shedding.	Regularly check the load-shedding schedule and go to areas where you have internet access if possible.	Continue project work for sections that are not reliant on internet access.

Table 4: Risk Mitigation, Monitoring, and Management

Appendix B GANTT CHART

ID	Task Name	Start	Finish
1	**Literature Review**	**Feb 24**	**Mar 24**
2	Individual Literature Reviews	Feb 24	Mar 24
3	**Project Proposal**	**Mar 28**	**May 02**
4	Draft	Feb 28	Apr 24
5	Presentation	Apr 25	Apr 28
6	Final	May 01	May 02
7	**In-depth Topic Exploration**	**May 03**	**May 26**
8	Do further research on defeasible and classical explanations	May 03	May 23
9	Do further research on defeasible and classical entailment	May 03	May 23
10	Review understanding with supervisor	May 24	May 26
11	**Preliminary Work**	**Jun 05**	**Jun 07**
12	Finalise the overall approach with supervisor	Jun 05	Jun 07
13	**Algorithm Implementation**	**Jun 07**	**Jul 14**
14	Research on the best possible algorithms	Jun 07	Jun 14
15	Analysis of the chosen Algorithms	Jun 15	Jun 20
16	Initial Prototype and Implementation	Jun 21	Jul 05
17	Analysis and Testing	Jul 06	Jul 11
18	Finalisation Analysis	Jul 12	Jul 14
19	**Demonstrations**	**Jul 12**	**Jul 21**
20	Software Feasibility Demonstration	Jul 12	Jul 17
21	Project Progress Demonstration	Jul 18	Jul 21
22	**Integration**	**Jul 22**	**Aug 08**
23	Integrate defeasible entailment and explanation service programs on to the chosen external	Jul 22	Jul 31
24	Analysis and Testing	Aug 01	Aug 08
25	**Final Submissions**	**Aug 10**	**Sep 15**
26	Complete Draft of final paper	Sep 11	Sep 11
27	Paper final submission	Aug 29	Sep 11
28	Code final submission	Sep 12	Sep 15
29	**Other Tasks**	**Sep 16**	**Oct 24**
30	Final Project Demonstration	Sep 26	Sep 29
31	Project Poster	Oct 02	Oct 10
32	Project Website	Sep 16	Oct 16
33	School of IT Showcase	Oct 17	Oct 24

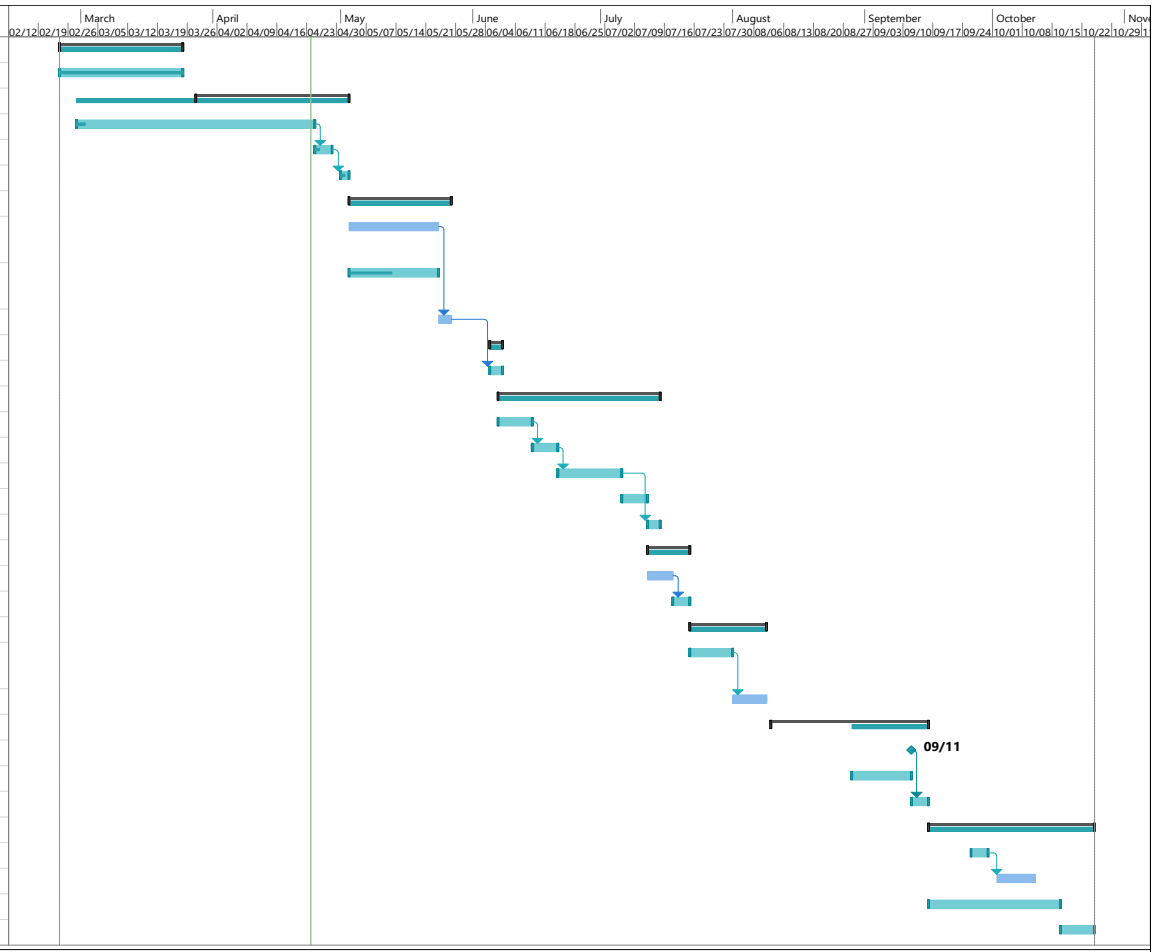

Figure 2: Gantt Chart