Literature Review of Defeasible Explanations

Emily Morris
Department of Computer Science
University of Cape Town
mrremi007@myuct.ac.za

ABSTRACT

Explanations provide a way of showing why an entailment holds in classical logics. A justification is a form of explanation provided by minimal entailing subsets of a knowledge base. Very little work has been done in terms of extending explanations to forms of defeasible reasoning. Defeasible reasoning is a form on non-monotonic reasoning that aims to emulate reasoning with uncertainty. Explanations have many useful applications in classical logics, making it a worthwhile concept to extend to defeasible reasoning. We aim to extend justifications to KLM-style defeasible reasoning and potentially define a set of properties which contribute to a more general definition of explanations for defeasible reasoning.

CCS CONCEPTS

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

KEYWORDS

Artificial Intelligence, Knowledge Representation and Reasoning, Propositional Logic, Justifications, Explanations, Defeasible Reasoning, KLM Framework

1 INTRODUCTION

Knowledge Representation and Reasoning is a field within Artificial Intelligence in which information is modeled using formal logic, allowing one to apply a set of rules and manipulations related to a form of reasoning [4]. There are many different forms of logic, with different levels of expressivity. Our work will focus on propositional logic, one of the most basic forms of classical logic. While it is not as expressive as logics such as description logics or first-order logic, propositional logic provides a level of simplicity which makes it easier to understand and work with.

An important aspect of reasoning is being able to infer new knowledge from existing information. In propositional logic, this is done using classical entailment. In addition to being able to infer knowledge, it is often useful to know why certain information is being inferred. Explanations provide reasons as to why an entailment holds. The form of explanations that we will focus on are justifications. Justifications provide a desirable approach to explanations due to the fact that they are conceptually simple and can be computed using existing reasoning services [10]. Justifications have also been used extensively in knowledge base debugging due to their ability to expose the exact information causing faulty reasoning.

There are certain forms of reasoning that classical logics cannot model. In particular, one cannot replicate the uncertainty with which humans reason. To do this one must use a different form of reasoning, known as nonmonotonic reasoning. We will focus on defeasible reasoning, looking in particular at the KLM approach to defeasible reasoning. This approach provides us with a variety of definitions for defeasible entailment. Unlike in the classical case, little work has been done on defining defeasible explanations.

In this review we aim to provide a background for addressing defeasible explanation. In Section 2 we give an overview of the relevant theory for propositional logic and classical entailment. In Section 3 we then examine the work done for explanations in classical logic. We focus on the justification base approach and look at the various algorithms defined for this. Then in Section 4 we motivate the need for nonmonotonic reasoning and begin to look at defeasible reasoning within the KLM framework. In particular, we look at two definitions for defeasible entailment, Rational Closure and Lexicographic Closure. In Section 5 we examine the current literature for defeasible explanations. Finally, in Section 6 we conclude by examining how the current literature forms a basis for extending explanations to defeasible reasoning.

2 BACKGROUND

2.1 Propositional Logic

In propositional logic, the most basic propositions, which can be assigned truth values of true or false, are called *atoms* [2]. Intuitively, atoms are statements like 'the sun shines' and 'the sky is blue'. Atoms are usually denoted using lower case letters such as p and q. Atoms can be combined together using Boolean operators.

The main Boolean operators are: negation($\neg$), conjunction($\wedge$), disjunction($\vee$), implication($\rightarrow$) and bi-implication ($\leftrightarrow$). Negation is a unary operator taking only a single operand, while the rest are binary operators taking two operands. The Boolean operators negation, conjunction and disjunction behave similarly to the natural language words 'not', 'and' and 'or' respectively. Implication behaves similarly to the word 'implies', where one expects that if A is true and A implies B, then B must be true too. Bi-implication behaves like two directional implication.

Atoms combined together using Boolean operators create what are known as *formulas*. Formulas can also be created by combining other formulas using Boolean operators. This gives us a way to model more complex information. For an implication $\alpha \rightarrow \beta$, one calls α the *antecedent*. One use $\top$ to denote a statement that is always true and $\bot$ to denote a statement that is always false. A finite set of formula is called a *knowledge base*, denoted $\mathcal{K}$.

One assigns meaning to atoms using *interpretations*. Interpretations model a specific state of the world, where the information stated in each atom is either true or false. Formally an interpretations $\mathcal{I}$ is a function $\mathcal{I} : \mathcal{P} \rightarrow \{T, F\}$ which assigns a value true (T) or false (F) to every atom in $\mathcal{P}$. The set of all interpretations for a formula or knowledge base is denoted $\mathcal{U}$. We evaluate formulas under a specific interpretation by breaking them down into their individual atoms and then using the semantics and precedence order defined for the various operators to build up the final truth value. If a formula α is true under an interpretation $\mathcal{I}$, one says $\mathcal{I}$ is a *model* of α. The set of all models of α is denoted Mod(α) or $[[\alpha]]$. If α has a model one says α is *satisfiable*. Otherwise, if α has not models and thus is false in all interpretations, one says α is *unsatisfiable*. For a knowledge base $\mathcal{K}$, $\mathcal{I}$ is a model of $\mathcal{K}$ if it is a model of all the formulas in $\mathcal{K}$. One similarly talks about $\mathcal{K}$ being satisfiable or unsatisfiable.

2.2 Logical Consequence

The semantics provided by interpretations in propositional logic provide a way of defining what is known as *logical consequence* or *entailment* for a propositional knowledge base. Given a knowledge base $\mathcal{K}$, one says a formula α is the logical consequence of $\mathcal{K}$, denoted $\mathcal{K} \vDash \alpha$, if in every model of $\mathcal{K}$ is also a model of α. If $\mathcal{K}$ does not entail α, one denotes this $\mathcal{K} \nvDash \alpha$. We will demonstrate the intuition behind logical consequence by means of examples.

Example 2.1. Suppose one has a knowledge base $\mathcal{K}$ containing the following information:

(1) Birds fly ($b \rightarrow f$)
(2) Cats do not like water ($c \rightarrow \neg w$)
(3) Garfield is a cat ($g \rightarrow c$)

Following the logic of the natural language sentences, one could conclude that 'Garfield does not like water'. Expressing this in terms of propositional logic, one says $\mathcal{K} \vDash g \rightarrow \neg w$ since in every model $\mathcal{I}$ of $\mathcal{K}$ where $\mathcal{I}(g) = T$, one must have $\mathcal{I}(c) = T$ and thus $\mathcal{I}(w) = F$ as $\neg w$ must be true, so $g \rightarrow \neg w$ is true.

Example 2.2. Using the knowledge base from example 3.1, consider the entailment of the formula $g \rightarrow f$, corresponding to the proposition 'Garfield flies'. $\mathcal{K} \nvDash g \rightarrow f$ since the interpretation $\mathcal{I}$ in which $\mathcal{I}(b) = F, \mathcal{I}(f) = F, \mathcal{I}(c) = T, \mathcal{I}(w) = F$ and $\mathcal{I}(c) = T$ is a model of $\mathcal{K}$ but not of $g \rightarrow f$. Intuitively, one can see that if one follows the logic of the natural language sentences, one cannot confidently make such a conclusion as there is no way to link the ideas of 'Garfield' and 'flying'.

Entailment provides one with a way of inferring new information from what one already knows. There are various efficient algorithms called SAT solvers for computing entailment in propositional logic. It is worth noting that entailment is defined on the meta level, rather than the object level [2]. Intuitively, the object level contains all elements in the language, for example Boolean operators and atoms, while the meta level describes information about the language.

2.3 Consequence Relations

Another way of defining entailment, which is independent of the underlying semantics of the language, is through consequence relations. A *consequence relation* is a mathematical relation defined on the formula in the language. That is, a consequence relation is a possibly infinite set $\mathcal{X} = \{(\alpha_1, \beta_1), ...(\alpha_i, \beta_i), ...\}$ where α_n can be a set of formulas but β_n is a single formula. One interprets $(\alpha, \beta) \in \mathcal{X}$ to mean 'from α, one can infer β'. In this way consequence relations define a pattern of reasoning for a given language [12]. Not all relations correspond to valid forms of reasoning however. Generally one provides a set of properties to restrict the definitions of valid relations. We will return to the idea of consequence relations in later sections when we discuss defeasible reasoning.

3 EXPLANATIONS

As humans, when reasoning we are generally able to give evidence as to why we think a specific fact holds. For example, if one knows that 'birds fly' and 'penguins are birds', one could reasonably conclude that 'penguins fly'. To substantiate ones conclusion, one would give the two known facts as evidence. In formal logic, *explanations* express why an entailment holds. This is helpful for large and complicated knowledge bases where it is not always obvious why an entailment holds. Explanations give one insight into how a particular knowledge base works and can aid in the debugging of knowledge bases [10]. They can also be presented in a way that enables people unfamiliar with formal logic to allow them to understand reasoning services [3, 18] and can be used to build versions of formal proofs.

3.1 Justifications

We will focus on a form of explanation known as justifications. A *justification* is a minimal subset of a knowledge base that entails the formula. They provide a simple and intuitive way of summarizing why an entailment holds by showing the exact formulas in the knowledge base which are responsible for the entailment. The notion of a justification is best illustrated with an example.

Example 3.1. Suppose one knows:

(1) Birds fly
(2) Penguins are birds
(3) Birds have wings

While one could give all three facts together as evidence for the conclusion 'penguins fly', the fact that 'birds have wings' does not actually influence ones conclusion. However, only giving one of the first two facts as evidence would not be enough to support the conclusion. So one's justification for the conclusion 'penguins fly' would be that 'birds fly' and 'penguins are birds'.

Formally, $\mathcal{J} \subseteq \mathcal{K}$ is a justification of the entailment $\mathcal{K} \vDash \alpha$ if $\mathcal{J} \vDash \alpha$ and for all $\mathcal{J}' \subseteq \mathcal{J}, \mathcal{J}' \nvDash \alpha$. Note one formula may have many justifications in a knowledge base.

3.1.1 Justification Algorithms. Many algorithms have been developed for computing justifications. There are two axes of classification for these algorithms: single-all-axis and reasoner-coupling-axis. The single-all-axis differentiates between algorithms that compute all justifications for an entailment and ones that just compute a single justification for an entailment. Most algorithms that compute all justifications use a single justification algorithm as a subroutine [10]. The reasoner-coupling-axis differentiates between glass-box and black-box algorithms. In glass-box algorithms, justification computation is built into the reasoning algorithm. This means justifications are computed during the reasoning process. Black-box algorithms are independent of the underlying reasoning process and are computed separately. This makes black-box algorithms robust and easier to implement, as one can use existing reasoning algorithms without modification both for computing entailment and justifications. It also means they are not restricted to a single form of logic, making them more widely applicable [14]. However, glass-box algorithms perform better than black-box algorithms, particularly in the case of unsatisfiable knowledge bases [10].

Combining the axes of classification allows us to categorize algorithms that compute all justifications, using the reasoner-coupling-axis for both the base algorithm and the single-justification subroutine algorithm. For example, one can have an All-Black-Box algorithm with a Single-Glass-Box subroutine.

3.1.2 Black-box algorithms. We will first discuss black-box algorithms for computing a single justification. Single-Black-Box algorithms have an 'expand and contract' structure. Intuitively, the idea is to take a subset of the knowledge base and enlarge it until it entails the formula. This forms the 'expansion' phase. Then elements are removed from the set until no unnecessary formulas remain, guaranteeing that the set is minimal and thus a justification. This forms the 'contraction' phase. There are various ways of implementing and optimizing the two phases of Single-Black-Box algorithms. Expansion must be done in such a way that ensures that the generated entailing subset is not much larger than the final justification. This can be done by choosing an appropriate selection function, which determines which formulas should be added during each iterative expansion based on the current subset and the formula being entailed. The selection function is dependent on the logic system being used. During the contraction phase one aims to reach the final justification as quickly as possible. Various methods have been proposed for this, with two of the most important being the sliding window approach [14] and a divide and conquer strategy [11]. It has been shown that the divide and conquer strategy is more efficient[20], making it a preferred method for implementation. For more expressive logics, such as description logics, one can also optimize justification computation by constructing a module $\mathcal{M}$ of $\mathcal{K}$. A module is essentially a subset of $\mathcal{K}$ with a specific property [10]. Various definitions for modules in description logics have been provided that preserve entailment. This means one can compute justifications with respect to $\mathcal{M}$, which is more efficient if $\mathcal{M}$ is smaller than $\mathcal{K}$.

All justifications for an entailment can be computed using an adaption of Reiter's Hitting Set Tree Algorithm [19]. This adapted algorithm is proposed and proven by Kalyanpur et al[13]. The algorithm constructs a hitting set tree which contains all the justifications for the entailment as labels of the nodes. A hitting set tree for a given entailment is a finite tree with the following properties:

- The interior nodes of the tree are labeled with justifications and the edges as labeled with formulas.
- An edge from node $n1$ to node $n2$ is labeled with $\alpha \in n1$ such that $\alpha \notin n2$
- For any node n, the formulas labelling the path from n to the root node do not intersect with the justification that labels of n

The algorithm uses some predefined single justification algorithm to compute node labels, which can be either a black-box or glass-box algorithm.

3.1.3 Glass-Box Algorithms. Much work has been done in terms of developing glass-box algorithms for computing justifications in description logics. The algorithms are based on the idea of 'tracing' [10]. Essentially, in the process of 'tracing', elements derived in the reasoning process are labeled according to the original formula in the knowledge base they are related to. In the case of description logics such as $\mathcal{ALC}$, axioms added using the rules in the consistency algorithm are labeled according to the axiom that caused the rule to be invoked. This idea of axiom pinpointing is similar to the approach used in SAT solvers when computing minimal unsatisfiable subsets [1]. For propositional logic, justifications are analogous to minimal unsatisfiable subsets. Some work has been done in terms of modifying SAT solvers to include the idea of tracing in the computation of minimal unsatisfiable subsets [24]. While this has mainly been used as a way of proving unsatisfiability, it also provides a way of computing justifications.

3.2 Comparing Justifications

As previously stated, there can be more than one justification for a given entailment. This introduces the idea of whether certain justifications are better than others. For example, one might favour a justification that contains the smallest number of formulas. In unsatisfiable knowledge bases, one might want only satisfiable justifications, which would allow one to verify whether the entailment holds independent of the contradiction in the knowledge base. One could also focus on producing concise justifications, known as fine-grained justifications. While justifications are minimal in terms of the formulas they contain, the formulas themselves might contain information that is not relevant to the entailment. Justifications are also at risk of 'masking', where a single justification may contain more than one reason for entailment. We will illustrate these ideas with reference to an example.

Example 3.2. Suppose one has a knowledge base $\mathcal{K}$ containing the following information:

(1) Birds fly and have wings ($b \rightarrow f \wedge w$)
(2) Penguins are birds and have wings ($p \rightarrow b \wedge w$)
(3) If something flies, it is happy and sings ($f \rightarrow h \wedge s$)
(4) If something is happy, a bird and has wings then it jumps ($h \wedge b \wedge w \rightarrow j$)

Consider whether $\mathcal{K}$ entails the statement $p \rightarrow j$ or 'Penguins jump'. One has that $\mathcal{K} \models p \rightarrow j$ and there is only a single justification, namely $\mathcal{J} = \mathcal{K}$. But the information about flying things singing is not relevant to our entailment, meaning one can remove the information about singing, reducing the formula $f \rightarrow h \wedge s$ to $f \rightarrow h$, and retain the entailment. As an example of masking, note that as part of our entailment, one requires that 'penguins have wings'. One could say a penguin has wings either because it is a penguin or because it is a bird. This gives one two reasons within a single justification as to why the entailment holds.

Thus reduction and simplification on an axiomatic level could be used to define better justifications. Fine-grain justifications seek to provide a way of doing this.

Horridge [10] presents a formal definition for fine-grain justifications in the case of description logics. Horridge also presents various versions of masking. Two types of fine-grained justifications are defined: Laconic Justifications and Precise Justifications. Intuitively, Laconic Justifications are justifications whose axioms contain no unnecessary information for the entailment. Precise

Justifications build off Laconic Justifications and highlight the parts of axioms that must be changed during knowledge base repair. The set of Laconic Justifications, which is shown to be infinite, is further refined to create a set of Preferred Laconic Justifications, which is finite. The latter set contains Laconic Justifications that are syntactically similar to the original axioms in the knowledge base. Additionally, each reason for entailment in the original knowledge base is represented by a Laconic Justification in the set of Preferred Laconic Justifications, addressing the issue of masking.

4 DEFEASIBLE REASONING

Classical logics are limited in terms of the information they can convey. This is because they exhibit the property of monotonicity. Intuitively this means that adding more information to a knowledge base does not change any information one has previously inferred. Thus when one adds statements to the knowledge base, one cannot retract previously entailed knowledge. We will provide an illustrative example as to why this can be restrictive.

Example 4.1. Suppose one has a knowledge base $\mathcal{K}$ that contains the following information:

(1) Birds fly ($b \rightarrow f$)
(2) Penguins are birds ($p \rightarrow b$)

It is shown have seen in a previous example that $\mathcal{K}$ entails 'penguins fly'. Now suppose one adds a new statement 'penguins do not fly', given by the propositional formula $p \rightarrow \neg f$. One can now infer that penguins both fly and do not fly which removes one's ability to reason about penguins as they can no longer exist. More formally the atomic proposition of penguin must be false in every interpretation.

For this example to work, one should instead phrase the first fact as 'birds typically fly'. This captures an idea of uncertainty that allows one to later retract inferred statements if one learns information that contradicts them. To reason with this element of uncertainty in formal logic one needs to use a version of nonmonotonic reasoning. We will focus on defeasible reasoning. Over the years various forms of defeasible reasoning have been developed. An overview of the various approaches is given in [12]. We will focus on the KLM-style of defeasible reasoning, first proposed by Kraus, Lehmann and Magidor (KLM)[15].

4.1 KLM Framework

KLM extend Propositional Logic to enable defeasible reasoning by including a new Boolean operator, denoted $\mid\!\sim$, and allowing defeasible implications of the form $\alpha \mid\!\sim \beta$, where α and β are propositional formulas. Informally, a defeasible implication $\alpha \mid\!\sim \beta$ means 'typically, if α then β' [7]. We will focus on KLM-style rational defeasible implications, the semantics of which are given by ranked interpretations. The question then arises as to how one should define entailment when defeasible implications are present. Unlike in the classical case, there is no single definition for defeasible entailment ($\mid\!\approx$). KLM proposes restrictions on the definitions for defeasible entailment by giving a set of properties, known as the KLM properties, they should adhere to. Thus, the KLM framework does not define a specific way of performing defeasible reasoning, but rather acts as a guidance as to how KLM believes reasonable defeasible reasoning should behave. The KLM properties for defeasible entailment are as follows:

(1) $K \mid\!\approx \alpha \mid\!\sim \alpha$ (Ref)
(2) $\frac{\alpha \equiv \beta, K \mid\!\approx \alpha \mid\!\sim \gamma}{K \mid\!\approx \beta \mid\!\sim \gamma}$ (LLE)
(3) $\frac{K \mid\!\approx \alpha \mid\!\sim \beta, \beta \models \gamma}{K \mid\!\approx \alpha \mid\!\sim \gamma}$ (RW)
(4) $\frac{K \mid\!\approx \alpha \mid\!\sim \beta, K \mid\!\approx \alpha \mid\!\sim \gamma}{K \mid\!\approx \alpha \mid\!\sim \beta \wedge \gamma}$ (And)
(5) $\frac{K \mid\!\approx \alpha \mid\!\sim \gamma, K \mid\!\approx \beta \mid\!\sim \gamma}{K \mid\!\approx \alpha \vee \beta \mid\!\sim \gamma}$ (Or)
(6) $\frac{K \mid\!\approx \alpha \mid\!\sim \beta, K \mid\!\approx \alpha \mid\!\sim \gamma}{K \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma}$(CM)
(7) $\frac{K \mid\!\approx \alpha \mid\!\sim \gamma, K \not\mid\!\approx \alpha \mid\!\sim \neg \beta}{K \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma}$(RM)

One calls any form of defeasible entailment that satisfies these properties *LM-Rational.* Note that property six is implied by the rest of the properties, but it included to remain consistent with how the properties were introduced. KLM provided the intuition behind these properties when they were defined [15, 17]. Before considering defeasible entailment, we will first discuss preferential and ranked interpretations, which provide semantics for $\mid\!\sim$. They also provide a basis for definitions of LM-Rational forms of defeasible entailment.

4.1.1 Preferential Interpretations. The semantics KLM define for $\mid\!\sim$ are based off the class of preferential logics, proposed by Shoman [21–23]. This extends the semantics of classical logic by introducing an ordering of interpretations, which can be viewed as an order of typicality. A statement is then said to be true if it holds in the most 'typical' interpretations. In their initial work, KLM defined defeasibility on the meta-level, defining a preferential consequence relationship using a version of the first six KLM properties given above. KLM then defined a preferential interpretation, which consists of a set of states, a mapping from the set of states to the set of propositional interpretations and a strict partial order defined on the set of states. Preferential interpretations are shown to define preferential consequence relationships.

4.1.2 Ranked Interpretations. Lehmann and Magidor[17] refine preferential interpretations to create ranked interpretations by defining conditions which restrict the partial order for the interpretation. These restrictions cause the ordering to create a series of non-empty levels, where interpretations on the lower levels are more 'typical'. The definition for a preferential consequence relation is also extended to define a rational consequence relation, which satisfies all seven of the KLM properties above. Ranked interpretations are shown to define rational consequence relations. Furthermore, every rational consequence relation can be defined by a ranked interpretation. Lehmann and Magidor also address defeasibilty on the object level, providing semantics for the operator $\mid\!\sim$ using ranked interpretations. A defeasible implication $\alpha \mid\!\sim \beta$ holds in a ranked interpretation $\mathcal{R}$, if for all the most typical interpretations in which α is true (the models of α on the lowest level which contains an interpretation that satisfies α), β is true too. It is worth noting any propositional formula α can be written as an equivalent defeasible implication $\neg\alpha \mid\!\sim \bot$ [7].

4.1.3 Defeasible Entailment. We now address defining a version of defeasible entailment. KLM define preferential entailment [15] and Lehmann and Magidor define rank entailment [17]. Both of these forms of entailment are still monotonic, but they provide a basis for defining a form of nonmonotonic entailment based off preferential semantics. Two such forms of defeasible entailment are Rational Closure, proposed by Lehmann and Magidor [17], and Lexicographic Closure, proposed by Lehmann [16]. Both of these approaches are LM-Rational. Rational Closure is the more conservative form of defeasible entailment, falling under what is known as prototypical reasoning. Lexicographic Closure on the other hand is a form of presumptive reasoning. The key difference between these two types of reasoning how is much one assumes based on the information given. In prototypical reasoning, one assumes things to be true if they are true in the most 'typical' cases but in presumptive reasoning, one assumes something to be true if there is no evidence to the contrary. It is worth mentioning there are other reasonable forms of defeasible entailment that are not LM-Rational, for example Relevant Closure [6]. This highlights the fact that the KLM approach is not the only valid method for defining defeasible entailment.

4.2 Rational Closure

Rational Closure can be defined in terms of a ranked interpretation or a ranking of statements in the knowledge base. First we will provide the ranked interpretation that defines Rational Closure, then we will discuss Rational Closure's definition using ranked statements and the algorithm this provides.

4.2.1 *Ranked Interpretation.*

Lehmann and Magidor [17] define an ordering on all the rational consequence relations, with the minimal element of this ranking corresponding to the rational consequence relation known as Rational Closure. Casini et al [7] provide a related definition by imposing an ordering $\preceq_K$ on all the ranked interpretations that are models of a knowledge base $\mathcal{K}$. One has $\mathcal{R}_1 \preceq_K \mathcal{R}_2$ if for all $u \in \mathcal{U}$, one has $\mathcal{R}_1(u) \leq \mathcal{R}_2(u)$. This introduces another layer of typicality, where the ranked interpretations further down the ranking represent the more 'typical' ranked interpretations. Giordano et al [9] show that there is a unique minimal model $\mathcal{R}^K_{RC}$ for $\mathcal{K}$, where interpretations are as 'pushed down' as possible. One says a defeasible implication $\alpha \mid\!\sim \beta$ is in the Rational Closure of $\mathcal{K}$ if $\mathcal{R}^K_{RC}$ is a model of $\alpha \mid\!\sim \beta$. Since Rational Closure is defined in terms of ranked interpretations, it is LM-Rational.

4.2.2 *Ranked Formulas.*

An alternative definition for Rational Closure is given through the ranking of statements in the knowledge base using each formula's base rank. First, we define some necessary concepts. Lehmann and Magidor [17] define the materialisation of a defeasible knowledge base K, denoted $\overrightarrow{\text{K}}$, where every defeasible implication is replaced by classical implication. Formally, for a knowledge base $\mathcal{K}$, $\overrightarrow{\text{K}} = \{\alpha \rightarrow \beta | \alpha \mid\!\sim \beta \in K\}$. One then defines a formula α to be exceptional with respect to a knowledge base $\mathcal{K}$ if $\overrightarrow{\text{K}} \models \neg\alpha$ [1994]. Intuitively, α is exceptional with respect to $\mathcal{K}$ if one can 'disprove' α using formulas in $\mathcal{K}$. Using the concept of exceptionality, one can define a series of subsets of $\mathcal{K}$ using the function $\varepsilon(K) = \{\alpha \mid\!\sim \beta \mid \alpha \text{ is exceptional with respect to K}\}$. One defines $E^K_0 = K$ and successively set $E^K_{i+1} = \varepsilon(E^K_i)$ until one reaches i where $E^K_{i+1} = E^K_i$. For this i one sets $E^K_\infty = E^K_i$. The base rank of a formula α, denoted $br(\alpha)$, is then defined to be the smallest i such that α is not exceptional with respect to E^K_i. For a defeasible statement, $br(\alpha \mid\!\sim \beta)$ is defined as $br(\alpha)$. For a knowledge base $\mathcal{K}$ and a defeasible formula $\alpha \mid\!\sim \beta$, one then says $\alpha \mid\!\sim \beta$ is in the Rational Closure of $\mathcal{K}$ if $br(\alpha) < br(\alpha \wedge \neg\beta)$ or $br(\alpha) = \infty$ [9].

4.2.3 *Algorithm.*

The base rank definition for Rational Closure provides an algorithm for computing Rational Closure. The basic idea is to rank formulas in the knowledge base based on their level of specificity, with more general statements having a lower rank. When is comes to computing entailment, if there is an inconsistency, one throws away the most general information until the inconsistency is resolved. Then one computes entailment from the remaining knowledge base using classical entailment. To do this, one first materializes the knowledge base and partitions it into levels $\mathcal{R}_0 ... \mathcal{R}_{n-1}, \mathcal{R}_\infty$, grouping formulas based on their base ranks. Intuitively, the base rank of a formula is related to how defeasible or typical it is, with formulas with a lower rank being more atypical. Therefor the partitioning of the knowledge base into these levels provides us with a specificity ranking of the formulas. Now to check whether a formula $\alpha \mid\!\sim \beta$ if entailed by $\mathcal{K}$, one checks whether $\mathcal{R}_0 \cup ... \cup \mathcal{R}_{n-1} \cup \mathcal{R}_\infty \models \neg\alpha$. If this entailment holds, one removes the lowest level $\mathcal{R}_0$ and checks again. This continues until either the entailment no longer holds or only R_∞ remains. At this point, one computes $\mathcal{R}_i \cup ... \cup \mathcal{R}_{n-1} \cup \mathcal{R}_\infty \models \alpha \rightarrow \beta$, where i is the level at which α is no longer exceptional, and returns this as the entailment result. This algorithm reduces defeasible entailment to a series of classical entailment checks, making it efficient to implement with current classical reasoners.

4.3 Lexicographic Closure

Lehman [16] presents a form of Defeasible Entailment known as Lexicographic Closure. Like Rational Closure, Lexicographic Closure can be defined using both a ranked interpretation and by ranking formulas. We present the ranked formula definition in terms of an algorithm. Both definitions use the concept of a 'seriousness' ordering. Lehmann describes two aspects of seriousness: the number of formulas an interpretation violates and the specificity of the violated formulas, where specificity is given by the base rank of a formula. These two criteria give one two separate orderings which one combines in a lexicographic manner, treating the specificity criterion as the principle criterion.

4.3.1 *Ranked Interpretation.*

To create the ranked interpretation corresponding to Lexicographic Closure, one orders interpretations by the specificity of the formulas they violate, then refines this ordering using the number of formulas they violate. Casini et al[7] show how one can obtain Lexicographic Closure from Rational Closure, making Lexicographic Closure a refinement of Rational Closure. One can obtain the ranked interpretation that defines Lexicographic Closure by taking the ranked interpretation that defines Rational Closure and ranking interpretations within levels according to the number of formulas they satisfy, with interpretations satisfying more formulas having a lower rank. Formally, one defines a ranking $\preceq^K_{LC}$ where for $u, v \in \mathcal{U}$, $v \preceq^K_{LC} u$ if $\mathcal{R}^K_{RC}(u) = \infty$, or $\mathcal{R}^K_{RC}(v) < \mathcal{R}^K_{RC}(u)$, or $\mathcal{R}^K_{RC}(v) = \mathcal{R}^K_{RC}(v)$ and $C^K(v) \geq C^K(u)$, where $C^K(v) = \#\{\alpha \mid\!\sim \beta \in K \mid v \in Mod(\alpha \rightarrow \beta)\}$ and # denotes the size of the set. One then says a formula $\alpha \mid\!\sim \beta$ is in the Lexicographic Closure of a knowledge base $\mathcal{K}$ if the ranked interpretation derived from the ordering $\preceq_{LC}^{K}$, denoted R_{LC}^{K}, is a model of $\alpha \mid\!\sim \beta$.

4.3.2 Ranked Formulas. One can also derive a ranking of statements for Lexicographic Closure from Rational Closure. This derivation will be presented in the form of an algorithm. Once again, one takes the ranking of statements that defines Rational Closure and refines it. This is done by adding extra levels which contain weakened versions of formulas from the knowledge base. The idea is that instead of removing an entire level of information when an inconsistency arises, one instead wants to remove the formula causing the inconsistency. This is done by weakening the more general information.

The algorithm works by initially ranking formulas in a knowledge base $\mathcal{K}$ according to the base rank algorithm in Rational Closure. Then, when checking whether $\mathcal{K}$ defeasibly entails a formula $\alpha \mid\!\sim \beta$, one starts by checking whether $\mathcal{R}_0 \cup \ldots \cup \mathcal{R}_{n-1} \cup \mathcal{R}_\infty \vDash \neg\alpha$. However, now if the entailment holds, instead of removing and entire level, one instead weakens the lowest level, $\mathcal{R}_0$. This is done by taking all subsets of size $x - 1$, where x is the number of elements in the level, creating a single formula equivalent to all the formulas in the subset by joining them using conjunction, and then joining all these combined formulas using disjunction to create a single, final formula. This is equivalent to considering all the ways of removing a formula from $\mathcal{R}_0$. If α is not exceptional with respect to the rest of the levels and this weakened version of $\mathcal{R}_0$, one computes entailment using classical entailment as before. Otherwise one weakens $\mathcal{R}_0$ again, this time considering all subsets of size $x - 2$. If one reaches the point where one is taking subsets of size zero, one throws away $\mathcal{R}_0$ completely and starts the same process with the next level. In the same way that one did not throw away the lowest level in Rational Closure, one also does not weaken it in Lexicographic Closure.

All the weakened levels can be computed and inserted before entailment computation begins. This allows for the Lexicographic Closure algorithm to fit into a more general pattern for defeasible entailment proposed by Casini et al [7]. However, in practice this would result in a large amount of unnecessary computation, as higher levels often do not need their weakened versions to be added.

5 DEFEASIBLE JUSTIFICATIONS

While much research has been done concerning classical justifications and their computation, little work has been done in extending this to defeasible reasoning. To the best of our knowledge, the only work done on this within the KLM framework is by Chama[8]. Chama provides an extension of Rational Closure algorithm for $\mathcal{ALC}$ to include the computation of justifications. The definition follows naturally due to the fact that Rational Closure can be reduced to a series of classical entailment checks. This allows for the ideas of classical justifications to be used. The extended algorithm ranks the axioms in the knowledge base as before. It then computes at which level the antecedent for the formula being entailed it no longer exceptional. This tells one how much information on needs to remove before a justification can be computed. Finally, all the axioms from ranks less than the calculated level are removed from the materialized knowledge base and all justifications are calculated using classical justification methods. Chama uses an All-Black-Box algorithm with a Single-Black-Box subroutine. While this may make for a simpler algorithmic definition, it it worth noting that in practice the algorithm could be more efficiently implemented if it used one of the glass-box algorithms for justification computation. The defeasible justifications computed are a subset of the classical justifications that would have been computed from the materialisation of the knowledge base. This could also be viewed as a way of comparing justifications, where one prefers justifications that are valid in both the defeasible and classical cases, rather than just the classical case.

Brewka et al [5] take a different approach for defining nonmonotonic justifications. Brewka et al present an abstract idea of a nonmonotonic justification that works for all forms of nonmonotonic reasoning, called a strong explanation. One of the reasons why justifications work well in monotonic reasoning is that anything entailed by a justification is entailed by the knowledge base. However, in the nonmonotonic case, the rest of the knowledge base might contain information contradictory to the entailment. Thus one could have a justification for an entailment when the entailment does not in fact hold. Brewka et al address this by extending the definition of a justification to include an extra property. A subset $\mathcal{J}$ of a nonmonotonic knowledge base $\mathcal{K}$ is a strong explanation for a formula α if it is a minimal subset of $\mathcal{K}$ that entails and for all $\mathcal{J}'$ where $\mathcal{J} \subseteq \mathcal{J}' \subseteq K$, $\mathcal{J}'$ also entails α. The definition of entailment here is dependent on the logic.

6 CONCLUSIONS

We have shown how one can use formal logic to model information and perform reasoning. Entailment allows us to infer new information and draw conclusions from knowledge bases containing information of interest. We have also demonstrated that classical logic it too restrictive, motivating for a switch to nonmonotonic logic which allows one to reason with uncertainty. In particular we looked at defeasible reasoning with respect to the KLM framework and discuss two approaches for defeasible entailment within the framework, Rational Closure and Lexicographic Closure.

We discussed various applications and motivations for explanations for entailment. We then focused on justifications as a form of explanation, examining various algorithms for justification computation and looking at ways of differentiating between the appropriateness of justifications when there is more than one justification for an entailment. We briefly mention aspects related to the efficiency of justification computation algorithms, but do not focus on this as we are interested in the more theoretical aspects.

Finally, we looked at the limited existing literature for defeasible explanations. Chama's work is representative of how one can extend existing algorithms to allow for the computation of justifications in a defeasible setting. Since Lexicographic Closure is a refinement of Rational Closure, extending the algorithm for Lexicographic Closure in a similar way would provide a way of computing defeasible justifications when using the Lexicographic Closure algorithm.

Chama does not address justifications in terms of the semantic definition of Rational Closure. We showed that Rational Closure has both a semantic and algorithmic definition, but Chama only provides an algorithmic definition for defeasible justifications. When trying to define defeasible explanations on a more abstract level, deriving a semantic definition might prove helpful. Brewka et al's definition for strong explanations highlights the main issue one faces when computing defeasible justifications. It also provides a good starting point for developing a set of properties for defeasible explanations.

REFERENCES

[1] Franz Baader and Rafael Penaloza. 2010. Axiom pinpointing in general tableaux. *Journal of Logic and Computation* 20, 1 (2010), 5–34.

[2] Mordechai Ben-Ari. 2012. Mathematical logic for computer science. (2012).

[3] Stefan Borgwardt, Anke Hirsch, Alisa Kovtunova, and Frederik Wiehr. 2020. In the Eye of the Beholder: Which Proofs are Best?. In *Proc. of the 33th Int. Workshop on Description Logics (DL 2020)*.

[4] Ronald J. Brachman and Hector J. Levesque. 2004. *Knowledge representation and reasoning*. Elsevier.

[5] Gerhard Brewka, Matthias Thimm, and Markus Ulbricht. 2019. Strong inconsistency. *Artificial Intelligence* 267 (2019), 78–117.

[6] Giovanni Casini, Thomas Meyer, Kodylan Moodley, and Riku Nortjé. 2014. Relevant closure: A new form of defeasible reasoning for description logics. In *European Workshop on Logics in Artificial Intelligence*. Springer, 92–106.

[7] Giovanni Casini, Thomas Meyer, and Ivan Varzinczak. 2019. Taking defeasible entailment beyond rational closure. In *European Conference on Logics in Artificial Intelligence*. Springer, 182–197.

[8] Victoria Chama. 2020. *Explanation for defeasible entailment.* Master's thesis. University of Cape Town.

[9] Laura Giordano, Valentina Gliozzi, Nicola Olivetti, and Gian Luca Pozzato. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence* 226 (2015), 1–33.

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

[11] Ulrich Junker. 2004. Preferred explanations and relaxations for over-constrained problems. In *AAAI-2004*.

[12] Adam Kaliski. 2020. *An overview of KLM-style defeasible entailment.* Master's thesis. University of Cape Town.

[13] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. 2007. Finding all justifications of OWL DL entailments. In *The Semantic Web*. Springer, 267–280.

[14] Aditya Anand Kalyanpur. 2006. *Debugging and repair of OWL ontologies.* Ph.D. Dissertation.

[15] Sarit Kraus, Daniel Lehmann, and Menachem Magidor. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence* 44, 1-2 (1990), 167–207.

[16] Daniel Lehmann. 1995. Another perspective on default reasoning. *Annals of mathematics and artificial intelligence* 15, 1 (1995), 61–82.

[17] Daniel Lehmann and Menachem Magidor. 1992. What does a conditional knowledge base entail? *Artificial intelligence* 55, 1 (1992), 1–60.

[18] Tu Anh Thi Nguyen. 2013. *Generating natural language explanations for entailments in ontologies.* Ph.D. Dissertation. The Open University.

[19] Raymond Reiter. 1987. A theory of diagnosis from first principles. *Artificial intelligence* 32, 1 (1987), 57–95.

[20] Kostyantyn Shchekotykhin, Gerhard Friedrich, and Dietmar Jannach. 2008. On computing minimal conflicts for ontology debugging. *Model-Based Systems* 7 (2008).

[21] Yoav Shoham. 1987. Nonmonotonic Logics: Meaning and Utility.. In *IJCAI*, Vol. 10. Citeseer, 388–393.

[22] Yoav Shoham. 1987. *Reasoning about change: time and causation from the standpoint of artificial intelligence.* Ph.D. Dissertation. Yale University.

[23] Yoav Shoham. 1987. *A Semantical Approach to Nonmonotonic Logics.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 227–250.

[24] Lintao Zhang and Sharad Malik. 2003. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *2003 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 880–885.