Justifications for KLM-style defeasible reasoning

By

Jane Imrie

A dissertation presented in accordance with the requirements for the degree of Masters of Science in the Faculty of Science

Supervisor: Prof. Thomas Meyer

Declaration

I, Jane Imrie, do hereby declare that this Masters dissertation titled "Justifications for KLM-style defeasible reasoning" is my own work and has not been submitted for any other degree award to any other University. I understand that failure to attribute material which is obtained from another source may be considered plagiarism.

Signed:

Date:

Contents

List of Tables

Abstract

The notion of using formal logic for artificial intelligence was first suggested by McCarthy in the 1950s, and this has led to extensive research into a field known as knowledge representation and reasoning, wherein research is conducted into how best to represent knowledge and reason about said knowledge in order to create more knowledge. Many systems which use formal logic were initially highly constrained as the algorithms which they employed were *monotonic* and consequently any inference which the system was able to compute could not be retracted, even if said information caused contradictions. This could prove detrimental, especially if the new information was more accurate vis-à-vis the domain under consideration. One of the solutions to this is non-monotonicity, and for the context of this dissertation, we will consider *defeasible reasoning*, which is a form of non-monotonic reasoning. There are many different types of formalisms for this type of reasoning, with the KLM (Kraus, Lehmann and Magidor) being one of the more popular. KLM has a number of desirable properties, which is why it is the formalism of choice. Regardless of the logic being used, reasoning systems also need to be able to "explain" how they were able to draw inferences. Justifications are one form of explanations, and research into them has increased throughout the years as explainable artificial intelligence researchers have highlighted their importance in creating trustworthy and reliable systems. This dissertation broadly aims to compile the research on justifications, with a focus on propositional and description logics, for both the classical and defeasible case, with a particular emphasis on the latter. We achieve this by first introducing classical propositional and description logic at a high level. We then delve into the history and current state of the literature on justifications in the classical case. We then detail how to add defeasibility into propositional logic and elaborate on different frameworks which are used to achieve this. This is then followed by work on defeasible justifications, where we highlight the algorithms used for computing them. The last chapter details gaps in the current literature for future research. By the end of this dissertation, the reader should have a keen understanding of classical and defeasible justifications, from their history and computation, to their algorithms and theoretical underpinnings.

Acknowledgements

My deepest gratitude goes out to my supervisor, Prof. Tommie Meyer, for many things. Namely, taking a chance on me and giving me the opportunity to learn about such a fascinating field. I am also eternally thankful for his advice and his patience. He is truly both a treasure trove and library of knowledge.

Thank you as well to many of the lecturers of the CS department - Prof Deshen Moodley, Prof Michelle Kuttel, Prof Hussein Suleman and Dr Jan Buys to name but a few. They have taught me numerous valuable lessons which I know I will carry with me for the rest of my life.

Thank you to Dean Reagon, best-friend and fellow student and mathematician, for being my voice of reason and sounding-board throughout this process and for always forcing me to prove my point. Your presence proved pivotal to my completion of this body of work and you're truly a master of intellectually stimulating and cognitively challenging conversation.

Thank you to my family, Chantelle Kearns and Michael Williams, for the iron-clad belief you've had in me since I was born. Thank you to Sabine and Nichola Ellis, Simon and Faith Rawlinson (and pets!) for being a safe haven and source of comfort for me all these years. There are no words I can express to do justice to how thankful I am to all of you.

Thank you to my friends from all the various stages of my life that I've (miraculously, given my introversion) maintained friendships with - from primary school to high school and university - for your support, encouragement and just in general hyping me up. You all were lights in my darkest times.

Lastly, I'd like to thank myself for being persevering and ambitious enough to pursue this.

Chapter 1

Introduction and Background

Knowledge representation and reasoning (KRR) is an area in Artificial Intelligence (AI) that is concerned with how best to represent information, usually (but not always) by means of a logic, as well as how to create additional or new information from this explicit knowledge. Domain knowledge can be explicitly represented in a structure known as a *knowledge base* [36]. This notion of a more declarative approach to AI was first suggested by McCarthy in the 1950s [38] and stands in stark contrast to the presently more popular statistical, data-driven approach [6].

The logical approach to AI also has many benefits, perhaps most obviously, that knowledge representation is human-readable. However, there are additional advantages which are less apparent at first glance. The logical paradigm allows for the explicit declaration of assumptions, exceptions and preferences. A logical framework also equips the system with the ability to grasp multiple different types of nuances according to whichever extension is under consideration, for example, constraints, nondeterminism or aggregates [13]. Their internal properties can be examined without difficulty, as there are a number of verification techniques which can be used internally on them. Secondly, they are easily augmented as new knowledge can simply be added. Lastly, they can accommodate abstractions, that is to say, one can use high-level concepts to describe their decisions. These features are essential to prototyping and drafting interpretable and explainable AI (XAI) systems [6] (a deeper discussion on XAI and related topics is beyond the scope of this work).

Classical logic is the dominant form of reasoning and originates out of Tarskian notions of consequence [43], with the salient idea being that there

are certain crucial properties which must be satisfied such that each inference must have the highest support possible from the explicit knowledge. This condition has an undesirable side-effect: one cannot use classical logic to reason about explicit knowledge when that knowledge contains exceptions. Let us use a simple example to illustrate this: if your parents both have brown eyes, then you will have brown eyes. Tina and Tom both have brown eyes, but their daughter Tiffany has blue eyes. If we used classical reasoning here, we could conclude that either Tiffany does not exist - which is not correct. Or we could state that she does, which causes a contradiction. Of course, we could simply amend the knowledge that we do have to include the fact that brown-eyed people can have blue-eyed children - but to do this for every exception in every knowledge base would prove to be extremely impractical. The above scenario shows the problem with classical logic i.e. that it is *monotonic*, that is to say, we cannot retract an inference once it has been made. A possible solution to this problem is *defeasible* reasoning, which is a form of non-monotonicity.

Defeasible reasoning is a research area where we attempt to formalise the above type of *common-sense* reasoning. There are a number of different frameworks which attempt to do this [27]. The one we've chosen as the focus of this dissertation was developed by Kraus, Lehmann and Magidor (KLM) [35, 34, 32]. We will use propositional logic to define this type of reasoning, as it was done in the original papers.

With the above in mind, it is now possible for technologies to incorporate reasoning services or to build stand-alone reasoners, such as that developed by Wang [61]. Regardless of the logic under consideration or whether it is defeasible or not, there is an increasing requirement for systems to be able to "explain themselves". There are a number of reasons for this:

1. Users of the system may or may not be domain experts of the information stored in the knowledge base underlying the system. Let us provide an example scenario: SNOMED [54] is a large medical knowledge base. Assume we have a system which is used for medical diagnosis that utilises SNOMED for information - that is, we provide a list of symptoms and it spits out a diagnosis. For a domain expert such as a medical doctor who is only seeking a second opinion, simply receiving an answer might be sufficient (especially if it agrees with their own conclusion). However, a non-expert user might have trouble understanding the answer and consequently not trust it (or simply not trust it because

receiving just an answer is not enough). In either case, providing an explanation for how the system derived the answer is paramount to building trust between users and systems [50].

2. Knowledge bases are "living" - they can change in size, or the knowledge within them can change and thus become rather complex. This might result in drawing inferences which are simply too difficult to understand, even if the user is an expert on the information. SNOMED contains over 500000 axioms. Using the example from the previous point, even if the domain expert gets an answer, they might not understand how or why such an answer was derived due to the complexity of the knowledge under consideration.

One solution which has been proposed for this is justifications - which is essentially the minimum number of statements that can be used to draw a specific conclusion. We demonstrate this by means of an informal example. Assume we have this set of facts about birds: (1) birds can fly, (2) a robin is a type of bird and (3) birds have beaks. We could therefore state that a robin can fly using points (1) and (2) above only, as (3) is superfluous to this particular conclusion. Points (1) and (2) form a justification for our assertion, and they are minimal in that if we were to remove either one of them, our conclusion no longer holds. In this dissertation, we aim to provide an overview of justifications: namely their origins from Reiter's Theory of Model-Based Diagnosis [49], to how they are computed for the classical case, as well as why they cannot simply be applied as is to the defeasible case. We mainly consider the aforementioned in the case of propositional logic, but will also dedicate some time to description logic, as much of the research into justifications has been done within that context.

Description logic is a type of logic where we use concepts and relations between them in order to reason about information. In propositional logic, we are only dealing with either `True` or `False`. So, for example, we can state that the sentence "a student attends a university" is true. However, for description logic, we would rather state that "students" and "university" are concepts, and that students are a part of a university. An instance of the concept "university" could for example be "University of Zurich" and for "student" could be "Albert Einstein".

1.1 Dissertation Outline

Chapter 2 provides an introduction to propositional logic and the notion of classical justifications. In this chapter, we present the syntax and semantics for propositional logic, which will be used throughout this dissertation. Additionally, we discuss the history and current state of the literature for classical justifications in the context of propositional and description logic.

Chapter 3 presents topics regarding non-monotonicity. More specifically, how it came about and why it was needed. We also delve into some of the formalisms (sometimes also referred to as "frameworks" in the literature) which have been created in order to perform non-monotonic reasoning, namely rational closure, lexicographic closure and relevant closure. Finally, we briefly discuss other forms of non-monotonic reasoning, such as System W [31].

Chapter 4 discusses defeasible justifications. We elaborate on the current paradigm of weak versus strong justifications, provide worked examples for current algorithms and discuss other approaches to justifications besides the idea of weak or strong. Weak justifications as they have been defined are the justifications we can derive once we have utilised one of the defeasible reasoning frameworks mentioned above. Strong justifications, which were created entirely divorced from the idea of weak justifications, are a type of justification where information from outside the justification (but still part of the knowledge base) cannot invalidate the justification.

Chapter 5 is the concluding chapter for this dissertation, where we present an overview of what was covered and posit some ideas for where possible future work for the field can be done. Presently there is much work to be done for defeasible justifications, from running benchmarks to compare how the different types of weak justification algorithms run for various types of knowledge base (which can have diverse complexities and sizes), to more theoretical work, such as developing a more generalised definition of defeasible justifications.

Chapter 2

Classical Logic and Classical Justifications

2.1 Mathematical Preliminaries

Let S be a set. Given that $x, y, z \in S$, we state that a partial order or order is a binary relation (notated $\preceq$) on S for all x, y, z such that:

1. $x \preceq x$ (reflexivity)

2. $x \preceq y$ and $y \preceq x$ implies $x = y$ (anti-symmetry)

3. $x \preceq y$ and $y \preceq z$ implies $x \preceq z$ (transitivity)

If a set has an order relation, then it is known as a partially ordered or an ordered set. If the relation has only the properties (1) reflexivity and (3) transitivity from above, then it is said to be a *pre-ordered set*. It is also possible to define some of the properties listed above in terms of the strict inequality relation $\prec$, leading to strict-order relations, which is used throughout subsequent chapters:

Definition 2.1.1. A strict order relation has the property of irreflexivity: $\forall x, x \not\prec x$.

Note that for strict partial orders, the reflexivity property does not hold.

2.2 Logic

It would be reasonable to question why logic is even necessary in the first place for KRR - or more specifically, to assume that any natural language should be sufficient for representing information and performing logical reasoning. Upon first inspection it is not immediately obvious that some attributes of natural language are partially, if not wholly incompatible with those required to adequately perform reasoning. These are as follows:

1. Most, if not all, natural languages are far too complex, verbose and dynamic - which makes them beyond formal description.

2. The meaning of a sentence within a natural language may be ambiguous, be context-dependant or require some implicit assumptions (to be understood).

3. In order to reason, we require a language that allows us to be as precise as possible when it comes to writing proofs and making assertions, whilst also being straightforward enough such that its syntax is simple and its semantics can be defined unambiguously.

Logics do not suffer from aforementioned problems (1) and (2) but does have the qualities described in (3), which is why they have been chosen for this task.

In this chapter, we provide an introduction to propositional and defeasible logic, elaborating on their syntax and semantics. We then briefly delve into classical justifications: their history, utility and function. We also provide an abstract, high-level introduction to the algorithms for computing justifications in the case of description logic. Since defeasible justifications for propositional logic is this focus of this dissertation, there is an entire chapter dedicated to discussing them - see chapter 4.

2.2.1 Propositional Logic

Logic is an abstraction over natural language. Such a system is composed of objects and their properties. At its core, a logic is a formal language which allows us to reason about the properties of an object. We use a predefined set of formation rules, known as syntax rules, in order to specify the language. These rules are logic-dependent, and can therefore vary between logics [22].

From there, one can establish a semantics for the logic. Semantics refers to how meaning is assigned and derived from a language, particularly a logical system [37].

Propositional logic lays the groundwork for many other logics such as description logic or first order logic [7]. It is for this reason that propositional logic is the logic of choice for this dissertation - as it is often the "simpler case", many other logics can easily translate algorithms and operations developed for propositional logic into ones which fit their own context.

We begin with the notion of propositional atoms, which are the simplest forms of expressing information, and cannot be further decomposed.

The set of all possible atomic propositions, i.e. atoms, is allocated the symbol $\mathcal{P}$ and can be either finite or infinite in size. For our purposes, we consider the finite case. In this dissertation atoms are represented using words such as `penguin` or `wings`. For typographic compactness, we will often shorten atoms such as `one chicken` to `1c` or `1chk`, i.e. with lowercase Latin alphabet letters and numerals. Following that, there are Boolean operators or connectives: $\neg$ (negation), $\wedge$ (and), $\vee$ (or), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence), which are used together with atoms to create *formulae*. In the literature, formulae are also refereed to as "axioms", particularly for description logic [26]. Additionally, atoms may be nested, that is, it is perfectly legal to have $\neg((p \wedge q) \vee r)$. Atoms may be assigned a value of either `true` or `false`, with a bar being placed over the atom when it is false. Thus, $\bar{c}$ denotes that `c` is `false`.

The constants, "top" and "bottom", $\top$ and $\bot$, are used to show either a tautology ($\top$) or contradiction ($\bot$). A tautology is a statement that is always true, in other words, a universal truth. A contradiction is a statement that is always false. There are also some auxiliary symbols used, such as round parendissertation "()", which is especially useful when imposing an order of operations when building nested formulae, as seen above. $\mathcal{L}$ is used to represent the set of all formulae and members of the set are denoted with lower case Greek letters, $\{\alpha, \beta, \gamma, ...\}$. Finally, $\mathcal{K}$ denotes a finite set of propositional formulae, otherwise known as a knowledge base (KB) - which is how we will refer to this set for this and all later sections. Theoretically a knowledge base could be finite or infinite in size, however in our context we will consider only the finite case.

Definition 2.2.1 (Propositional Formulae)**.** We define $\mathcal{L}$ recursively using these rules:

1. $\forall \mathtt{p} \in \mathcal{P}, \mathtt{p} \in \mathcal{L}$
2. For $\alpha \in \mathcal{L}, \neg\alpha \in \mathcal{L}$
3. For $\alpha, \beta \in \mathcal{L}$, $\alpha \wedge \beta, \alpha \vee \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta \in \mathcal{L}$

Only formulae which are constructed using these rules can be regarded as valid formulae in $\mathcal{L}$

All of the above constitutes the syntax of propositional logic, and so we now move to define the semantics [7].

Definition 2.2.2 (Interpretation)**.** An interpretation, $\mathcal{I}$, is a total function that assigns the truth values, $\{T, F\}$ to atoms i.e. $\mathcal{I} : \mathcal{P} \mapsto \{T, F\}$.

Interpretations are also called *valuations* or *worlds* in the literature. Further, we define $\mathcal{U}$ to be the set of all interpretations for our language $\mathcal{L}$.

Definition 2.2.3 (Satisfaction)**.** For $p \in \mathcal{P}$, if $\mathcal{I}(p) = T$, then $\mathcal{I} \Vdash p$, read as $\mathcal{I}$ satisfies p. For any $\alpha, \beta \in \mathcal{L}$:

1. $\mathcal{I} \Vdash \neg\alpha$ iff (if and only if) $\mathcal{I} \nVdash \alpha$
2. $\mathcal{I} \Vdash \alpha \wedge \beta$ iff $\mathcal{I} \Vdash \alpha$ and $\mathcal{I} \Vdash \beta$
3. $\mathcal{I} \Vdash \alpha \vee \beta$ iff either $\mathcal{I} \Vdash \alpha$ or $\mathcal{I} \Vdash \beta$
4. $\mathcal{I} \Vdash \alpha \rightarrow \beta$ iff $\mathcal{I} \nVdash \alpha$ or $\mathcal{I} \Vdash \beta$ or both
5. $\mathcal{I} \Vdash \alpha \leftrightarrow \beta$ iff $\mathcal{I} \Vdash \alpha \rightarrow \beta$ and $\mathcal{I} \Vdash \beta \rightarrow \alpha$
6. $\mathcal{I} \Vdash \top$ for every $\mathcal{I} \in \mathcal{U}$
7. $\mathcal{I} \nVdash \bot$ for every $\mathcal{I} \in \mathcal{U}$

The model(s) of a formula, are those where the values of the atoms contained within it will allow the formula to evaluate to true. A formula is said to be *satisfiable* if and only if it has a model.

Definition 2.2.4 (Model)**.** For any formula, say $\alpha \in \mathcal{L}$, and an interpretation, $\mathcal{I}$, if $\mathcal{I} \Vdash \alpha$, then $\mathcal{I}$ is a model of α, denoted $Mod(\alpha)$

We also define entailment and equivalence for formulae:

Definition 2.2.5 (Classical entailment for two formulae)**.** Given two formulae $\alpha, \beta \in \mathcal{L}$, then β is a logical consequence of α (notated as $\alpha \models \beta$) if $Mod(\alpha) \subseteq Mod(\beta)$ or equivalently, for every $\mathcal{I} \in \mathcal{U}$ such that $\mathcal{I} \Vdash \alpha$ then $\mathcal{I} \Vdash \beta$

Definition 2.2.6 (Logical Equivalence)**.** Given $\alpha, \beta \in \mathcal{L}$. $\alpha \equiv \beta$ holds if and only if $\alpha \models \beta$ and $\beta \models \alpha$.

Entailment (otherwise known as logical consequence) essentially states what formula or formulae can be derived based on other formulae. Entailment can be generalised for a formula and a knowledge base:

Definition 2.2.7 (Classical Entailment)**.** Given $\mathcal{K}$ and a formula α, we can say that $\mathcal{K} \models \alpha$ - read as "$\mathcal{K}$ entails α" - if and only if for every $\mathcal{I} \in \mathcal{U}$ we have that when $\mathcal{I} \Vdash \mathcal{K}$ it is also the case that $\mathcal{I} \Vdash \alpha$.

The above demonstrates the classical, Tarskian definition of logical consequence [43, 17]. Entailment relations in propositional logic also have a number of properties [59], with *monotonicity* being the one that has far-reaching consequences for knowledge representation and reasoning:

$\mathcal{K} \models \alpha$ implies $\mathcal{K} \cup \{\beta\} \models \alpha$ (Monotonicity)

The basic idea here is that once a conclusion is made, it cannot be retracted. This rigidity limits the utility of propositional logic for the purposes of knowledge representation and reasoning, as very frequently we are dealing with incomplete or changing information. However, with some modifications to the language and defining a new semantics, it is possible to account for the aforementioned circumstances. We discuss this in depth in chapter 3.

Pivoting back to semantics: we now extend satisfiability to entire knowledge bases: [61]:

Definition 2.2.8 (Knowledge Base Satisfiability)**.** Given that $\mathcal{K} = \{\alpha_1, ..., \alpha_n\}$. $\mathcal{K}$ is satisfiable iff there exists some $\mathcal{I}$ such that $\mathcal{I} \Vdash \alpha_i$, for all i, where $1 \leq i \leq n$.

In other words, a knowledge base is satisfiable if and only if there exists an interpretation in which all formulae in the knowledge base evaluate to true. Entailment can be determined by using satisfiability (but need not necessarily be). That is we can test if $\mathcal{K} \models \alpha$ by checking the unsatisfiability

of $\mathcal{K} \cup \{\neg\alpha\}$. This process of checking if a formula is entailed by a knowledge base is known as *query checking* [27], wherein the query represents some propositional formula.

For all of the above concerning propositional logic, a distinction must be made between concepts which are a part of the language, and those which are used to reason about the language. The former is known as the object-level, the latter is the meta-level. For example, formulae and connectives belong to the object level. Conversely, entailment and logical equivalence exist on the meta-level [27].

Consequence relations

Consequence relations is a mathematical relation between formulae, similar to how entailment is a relation between a knowledge base and a formula. The relation represents a pattern of reasoning and patterns of reasoning can be imposed by requiring that the relation satisfy some property or set of properties.

Definition 2.2.9 (Consequence relation)**.** A consequence relation is a set of ordered pairs (that is potentially infinite in size), with each element of the pair a formula from the language, $\{(\alpha, \beta) \mid \alpha, \beta \in \mathcal{L}\}$.

The notion of a consequence relation (which is essentially a binary relation) can be generalised to the case where the first element in the ordered pair is a subset of $\mathcal{L}$. For our context, we consider the case where α_i is a single formula. For example, $\alpha \vdash \beta$ is understood as (α, β) are in the set. Conversely, the use of $\nvdash$ implies that they are not in the set. Intuitively, one can think of each ordered pair as corresponding to an inference. Therefore, $\alpha \vdash \beta$ denotes that β can be inferred from α. Of course, it is also possible to construct a random set that provides no meaningful notion of consequence, irrespective of how the formulae are interpreted. Avron [1] thus argues that there are some properties which consequence relations should conform to in order represent a reasonable idea of inference, though elaboration of that is beyond the scope of this discussion.

2.2.2 Description Logic

A lot of current literature concerning justifications have description logics as the logic of choice, such as that of Horrridge [26], Kalyanpur[28], Schlobach

[51] and Suntisrivaraporn [58] (to name but a few). We therefore deem it necessary to provide a high-level introduction to description logics.

Description logics (DL) refer to a collection of logic-based formalisms, which are based on first-order logic and are decidable. Many (but not all) are more expressive than propositional logic [3] - a characteristic which will be demonstrated later in this section. Note however, that for this chapter, we use the $\mathcal{ALC}$ description logic, as it is a centrally important DL which allows for easy comparison with the other description logics [12]. Furthermore, defeasible reasoning for this particular DL has been researched and implemented [16].

We begin with the notion of a *concept name*, which is (in a very abstract sense) a representation of sets of objects. In essence, concept names are atomic representations of the basic classes for a given domain. Some examples of concept names when the domain concerns a university is `Lecturer` or `Course` (though there is nothing inevitably correct about these choices). There is also a *role name*, which is used to represent relations between concepts names, e.g. `lectures`. Finally, there are individual names, which are names of objects within the domain, e.g. `Einstein`. We also have some familiar constructors, namely negation, conjunction, disjunction - similar to propositional logic. There is also some additional ones: value restriction and existential restriction.

All the above forms the language of $\mathcal{ALC}$ description logic, denoted $\mathcal{L}$ - i.e. we have a finite set C of concept names with single atomic concepts denoted as A, B and lowercase a, b for individual names; a finite set R of role names with lowercase r for a single role name and a finite set I of individual names such that these three sets are pairwise disjoint. Concept descriptions, denoted with $\mathcal{C}, \mathcal{D}$, can be formed using these sets and constructors:

Definition 2.2.10 (Concept descriptions)**.** Let C be a set of concept names and R a set of role names that are disjoint from C. The set of concept descriptions over C and $\mathcal{R}$ is defined recursively as follows:

1. Every concept name is also a concept description
2. $\top$ and $\bot$ are concept descriptions.
3. If $\mathcal{C}$ and $\mathcal{D}$ are concept descriptions and r is a role name, then the following constructs are also concept descriptions:
   (a) Conjunction: $\mathcal{C} \sqcap \mathcal{D}$

(b) Disjunction: $\mathcal{C} \sqcup \mathcal{D}$

(c) Negation: $\neg\mathcal{C}$

(d) Existential restriction $\exists r.\mathcal{C}$

(e) Value restriction: $\forall r.\mathcal{C}$

For example, a concept description such as "A student that has enrolled at a university and whose classes are all taught by either a lecturer or PhD student" could be represented by the following:

$$\texttt{Student} \sqcap (\exists \texttt{enrolledAt.University}) \sqcap (\forall \texttt{attendsCourse.(Lecturer} \sqcup \texttt{Phd-Student))}$$

The semantics for description logics is based on that of classical logic [3]. We therefore define an interpretation as follows:

Definition 2.2.11 (Interpretation)**.** An interpretation for $\mathcal{ALC}$ is a structure $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$. $\Delta^{\mathcal{I}}$ denotes the domain, a non-empty set and $\cdot^{\mathcal{I}}$ is an interpretation function which maps concept names $A \in C$ to subsets $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, maps roles names $r \in R$ to binary relations $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and also maps individual names a to elements of the domain $\Delta^{\mathcal{I}}$.

An interpretation can thus be seen as a complete description of the state of the "world", where concepts are subsets of the domain, role names are binary relations over the domain and individuals map to elements of the domain [16].

Using the teaching domain mentioned earlier, we could have concept names such as `Student, Teacher, Course`, some roles names - `teaches, attends` and lastly some individual names, `Einstein, Theoretical Physics, Oppenheimer, Szilard`. Table 2.1 provides an example of a possible interpretation for this knowledge base.

Given concept descriptions $\mathcal{C}, \mathcal{D}$ and a role name r, the interpretation function can be extended to interpret concept descriptions from $\mathcal{L}$ using the following inductive definitions [12]:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$\bot^{\mathcal{I}} = \emptyset$$

$$(\neg\mathcal{C})^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \mathcal{C}^{\mathcal{I}}$$

$\Delta^{\mathcal{I}} = \{\texttt{Einstein, Theoretical Physics, Oppenheimer, Szilard}\}$
$\texttt{Teacher}^{\mathcal{I}} = \{\texttt{Einstein}\}$
$\texttt{Student}^{\mathcal{I}} = \{\texttt{Oppenheimer, Szilard}\}$
$\texttt{Course}^{\mathcal{I}} = \{\texttt{Theoretical Physics}\}$
$\texttt{teaches}^{\mathcal{I}} = \{\texttt{(Einstein, Oppenheimer), (Einstein, Szilard)}\}$

Table 2.1: A trivial example of an interpretation for a teaching domain

$$(\mathcal{C} \sqcup \mathcal{D})^{\mathcal{I}} = \mathcal{C}^{\mathcal{I}} \cup \mathcal{D}^{\mathcal{I}}$$

$$(\mathcal{C} \sqcap \mathcal{D})^{\mathcal{I}} = \mathcal{C}^{\mathcal{I}} \cap \mathcal{D}^{\mathcal{I}}$$

$$(\exists r.\mathcal{C})^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \text{There is a } b \in \Delta^{\mathcal{I}} \text{ such that } (a,b) \in r^{\mathcal{I}} \text{ and } b \in \mathcal{C}^{\mathcal{I}}\}$$

$$(\forall r.\mathcal{C})^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \text{For all } b \in \Delta^{\mathcal{I}}, \text{ if } (a,b) \in r^{\mathcal{I}} \text{ then } b \in \mathcal{C}^{\mathcal{I}}\}$$

A description logic knowledge base can be divided into two parts: the terminological box (TBox) and assertional box (ABox) [2].

Definition 2.2.12 (TBox)**.** For complex concepts $\mathcal{C}$ and $\mathcal{D}$, a TBox $\mathcal{T}$ is a finite collection of:

- Subsumption statements (otherwise known as general concept inclusion axioms) - these are of the form $\mathcal{C} \sqsubseteq \mathcal{D}$ and are understood as "$\mathcal{C}$ is subsumed by $\mathcal{D}$". In essence, subsumption implies that $\mathcal{C}$ refers to something more specific than $\mathcal{D}$. For example, `Master's Student` $\sqsubseteq$ `Student`

- Concept equivalence axioms - $\mathcal{C} \equiv \mathcal{D}$ (shorthand for $\mathcal{C} \sqsubseteq \mathcal{D}$ and $\mathcal{D} \sqsubseteq \mathcal{C}$). This is read as "$\mathcal{C}$ and $\mathcal{D}$ are equivalent"

For an interpretation $\mathcal{I}$ to be a model of a subsumption statement $\mathcal{C} \sqsubseteq \mathcal{D}$, written as $\mathcal{I} \Vdash \mathcal{C} \sqsubseteq \mathcal{D}$, it must be the case that $\mathcal{C}^{\mathcal{I}} \subseteq \mathcal{D}^{\mathcal{I}}$. A concept equivalence is satisfied (i.e. $\mathcal{I} \Vdash \mathcal{C} \equiv \mathcal{D}$) if and only if $\mathcal{C}^{\mathcal{I}} = \mathcal{D}^{\mathcal{I}}$. $\mathcal{I}$ models $\mathcal{T}$ if it is a model of every subsumption statement in the TBox.

The TBox therefore contains the terminology i.e. the concepts of the domain, their relations and properties. The ABox contains facts about specific objects in the domain.

Definition 2.2.13 (ABox)**.** Let $\mathcal{C}$ be a concept description, r a role name and let a and b be individual names. An ABox, denoted $\mathcal{A}$ is thus a set of assertions concerning individuals.

There are a finite number of names which is used to reference objects in $\mathcal{A}$. Furthermore, there are two types of assertion statements:

- Concept assertions of type $\mathcal{C}(a)$. In other words a is a member of (the interpretation of) C. `Teacher(Einstein)` and `Student(Oppenheimer)` are examples of this - the first asserts that Einstein is a teacher, and the other that Oppenheimer is a student.

- Role assertions of type $r(a, b)$. Here, b is a filler of the role r for a. Role fillers are used to differentiate between the concepts' function in the relation. For example, `teaches(Oppenheimer, Einstein)` demonstrates that the individual `Einstein` (the role filler) is in the relation `teaches`(role) which contains the individual `Oppenheimer`.

Interpretations can be extended to individual names in order to provide ABoxes with a semantics. The implication is that an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, .^{\mathcal{I}} \rangle$ maps the concepts and roles to sets and relations and also maps each individual name a to an element $\mathrm{a}^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ [2]:

Definition 2.2.14 (Concept Assertion Satisfaction)**.** Given a concept assertion $\mathcal{C}(a)$ and interpretation $\mathcal{I}$, $\mathcal{I} \Vdash \mathcal{C}(a)$ if $\mathrm{a}^{\mathcal{I}} \in \mathcal{C}^{\mathcal{I}}$.

Definition 2.2.15 (Role Assertion Satisfaction)**.** Given a role assertion $r(a, b)$ and interpretation $\mathcal{I}$, $\mathcal{I} \Vdash r(a, b$ if $(\mathrm{a}^{\mathcal{I}}, \mathrm{b}^{\mathcal{I}}) \in r^{\mathcal{I}}$.

We can extend the above to the entire ABox:

Definition 2.2.16 (ABox satisfaction)**.** Given an ABox, $\mathcal{A}$ and interpretation, $\mathcal{I} \Vdash \mathcal{A}$ if it satisfies every assertion in $\mathcal{A}$.

Following from the above, we present an example of a TBox and an ABox in table 2.2 and 2.3 respectively.

We can also use satisfaction for defining models of TBoxes and ABoxes respectively:

- $\mathcal{I} \Vdash \mathcal{T}$ i.e. $\mathcal{I}$ is a model of $\mathcal{T}$ if and only if $\mathcal{I} \Vdash \mathcal{C} \sqsubseteq \mathcal{D}$ for every $\mathcal{C} \sqsubseteq \mathcal{D} \in \mathcal{T}$

- $\mathcal{I} \Vdash \mathcal{A}$ i.e. $\mathcal{I}$ is a model of $\mathcal{A}$ if and only if:

  - $\mathcal{I} \Vdash \mathcal{C}(a)$ for every $\mathcal{C}(a) \in \mathcal{A}$

Course
Full-Time Student $\sqsubseteq$ Student
Part-Time Student $\equiv \neg$Full-Time Student
Teaching Assistant $\equiv \exists$ doesAdmin.Course
Teaching Assistant $\sqsubseteq$ Full-Time Student
Lecturer $\sqsubseteq$ Staff
Lecturer $\equiv \exists$ teachesCourse.Course
PostgradSupervisor $\sqsubseteq \exists$ supervises.Student
PostgradSupervisor $\sqsubseteq$ Lecturer

Table 2.2: An example of a TBox for a domain where the topic is university

Course(Theoretical Physics)
Teaching Assistant(Szilard), Student(Oppenheimer),
PostgradSupervisor(Feynman)
Lecturer(Einstein), teachesCourse(Theoretical Physics, Einstein)
supervises(Oppenheimer, Feynman)

Table 2.3: An example of a ABox for a domain where the topic is university

- $\mathcal{I} \Vdash r(a, b)$ for every $r(a, b) \in \mathcal{A}$

As with propositional logic, certain interpretations can also be models:

Definition 2.2.17 (Model)**.** Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, an interpretation $\mathcal{I}$ is a model of the knowledge base if it is both a model of $\mathcal{T}$ and $\mathcal{A}$.

The interpretation must satisfy all assertions in the knowledge base's ABox and all subsumption statements in its TBox.

It is also possible to define entailment for description logic in a very analogous way to propositional logic. The key differences between propositional and description logic is that we have a partitioned knowledge base for the latter. However, the essence of the idea remains the same - given some form of explicit information, what new information can we conclude? For some given TBox $\mathcal{T}$, we are trying to determine what other subsumptions will follow. Similarly, given an ABox $\mathcal{A}$, what other assertions can be calculated? And finally, given a DL knowledge base, $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, what statements follow from it [16]?

In a similar vein to propositional logic, a statement α is entailed by an ABox/TBox/DL knowledge base if α holds in every model of the ABox/TBox/DL KB [33]. More formally:

Definition 2.2.18 (Entailment)**.** Given some knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and α, a TBox statement, entailment is defined as the following:

- $\mathcal{T}$ entails α ($\mathcal{T} \models \alpha$) if and only if for every $\mathcal{I}$, if $\mathcal{I} \Vdash \mathcal{T}$ then $\mathcal{I} \Vdash \alpha$
- $\mathcal{A}$ entails α ($\mathcal{A} \models \alpha$) if and only if for every $\mathcal{I}$, if $\mathcal{I} \Vdash \mathcal{A}$ then $\mathcal{I} \Vdash \alpha$
- $\mathcal{K}$ entails α ($\mathcal{K} \models \alpha$) if and only if for every $\mathcal{I}$, if $\mathcal{I} \Vdash \mathcal{K}$ then $\mathcal{I} \Vdash \alpha$

For example, if we let our knowledge base be tables 2.2 and 2.3, then the following entailment holds: $\mathcal{K} \models \texttt{Person} \sqcap \neg\texttt{Part-Time Student(Szilard)}$.

2.3 Classical Justifications

A justification is, in the context of propositional and description logic, a minimal subset of a knowledge base that permits an entailment to hold. The subset consists of axioms pulled from the knowledge base, and is referred to as *minimal* due to the fact that if one or more axioms are removed from it, the remaining axioms would be insufficient for the entailment to hold.

Definition 2.3.1. (Justification) Given a knowledge base $\mathcal{K}$ and some formula α with $\mathcal{K} \models \alpha$. $\mathcal{J}$ is said to be a justification for α in $\mathcal{K}$ if $\mathcal{J} \subseteq \mathcal{K}$, $\mathcal{J} \models \alpha$ and for all $\mathcal{J}' \subset \mathcal{J}$, it is the case that $\mathcal{J}' \not\models \alpha$.

As an informal example, consider these three statements:

1. Birds are a type of animal which can fly
2. Birds have wings
3. A pigeon is a bird

If we consider a query such as "does a pigeon have wings?", then (2) and (3) could be used as a justification for stating that pigeons indeed have wings. Though adding (1) wouldn't be incorrect, the information itself has no bearing on whether or not we can conclude that pigeons have wings.

Justifications for the classical case have been studied at depth, particularly for description logics, and how they are computed does differ between logics [26]. In this section, we provide an overview of some of the key aspects regarding the literature on classical justifications and provide further reading for out of scope minutiae, particularly if the results are linked to characteristics of description logic. Where necessary, we also translate some of the concepts and results into a propositional context. Justifications have proven to be a very desirable choice for explanations. In the classical case they are relatively straightforward to understand, can usually be presented in the form of natural language, have an obvious relationship with the knowledge base used to derive them and there are a number of existing algorithms which can be used to compute them [26].

2.3.1 A Brief History

The underlying idea of explanations can be traced back to Reiter's *Theory of Model Diagnosis* [49, 25] (MBD). One of its core ideas is that faults in a system occur as a result of bad interactions between different set of components within said system. The term *diagnosis* was co-opted from the medical field, where it refers to the process and aftermath of using a set of symptoms to identify a medical problem. In this context, the slightly abstracted definition of a diagnosis is a minimal subset of components from a broken system that would repair the system if they were replaced. *Model* refers to the idea that one take the expected/predicted and observed behaviour of a system and use both to create a system model. Disparities between expected and observed behaviour demonstrates a system fault. Such model-based diagnosis strategies have applications across a variety of different fields, such as spreadsheet debugging or diagnosing faulty gates in circuits [26].

The Relationship Between Model-Based Diagnosis and Justification-Based Explanation

There are some very direct parallels between the notions of justifications and model-based diagnosis. The knowledge base is functionally the same as the system, axioms are the components, an entailment corresponds to a system fault, a justification is essentially a minimal conflict set and a diagnosis is a repair of the knowledge base. The algorithms studied in model-based diagnosis are concerned with computing diagnoses. Given the direct relationship

between diagnosis and KB repair, it is possible to appropriate these algorithms for entailment computation. This allows us to find all justifications for a given entailment [26].

2.3.2 Current literature

Justifications as we study them now rose to prominence out of the need for explanation capabilities for reasoning systems, particularly those that used description logic. It has long been argued that explanatory facilities are one of the most crucial components of KR systems, particularly in terms of its usability [40]. Initially, it was thought that an explanation was in essence a proof or part thereof and the general consensus was that for entailments, there needed to be a strong alignment between explanation systems and the reasoning system. However, there was a paradigm shift in this thinking during the early 2000s, wherein the idea of what an explanation should comprise of, changed [26]. There were two influential papers which jump-started this. The first was written by Schlobach and Cornet [51] - the work was concerned with the diagnosis and subsequent repair of ontologies that contained unsatisfiable concepts, and the result was Minimal Unsatisfiable Preserving Sub-TBoxes (MUPS). This is a variant of the notion of a typical justification, whereby we now construct minimal subsets that allow for an entailment which holds for the unsatisfiability of a particular concept. In other words, these are justifications for classes which are unsatisfiable. The other, equally as important, paper was written by Parsia and Kalyanpur [46]. The focus of their work was on developing tools and techniques for debugging and repairing OWL ontologies. From this work, they would go on to produce more publications and they were the researchers that initially identified the value of using justifications as a tool for explanations and ontology repair.

There are many active areas of research regarding justifications, such as computation, granularity and understanding [26]. The utility of justifications for use in real-world applications is contingent on having algorithms which are both practical and can perform efficiently for realistic inputs. Much of previous research, such as the works of Horridge[26] Kalyanpur [28], Suntisrivaraporn et al. [58] and Meyer et al. [42] (to name a few) and ongoing research is concerned with robustly testing these justification-finding algorithms across different logics and providing convincing evidence for the practicality of these algorithms. *Justification granularity* focuses on the axioms found in justifications, as axioms dictate the level at which justifications operate. These

axioms may themselves be composed of nested and possibly also complex concepts. In terms of justifications, this implies that one or more parts of an axiom may be considered redundant for any particular entailment. The convolutedness of such axioms may lead to usability problems, that is, users may have trouble understanding or repairing the knowledge base. Research in this area focuses, for example, on the different notions of justifications, such as precise/fine-grained, their computation, as well as on how these can be used in the repair of ontologies [26]. Finally, *justification-understanding* is concerned with how people interpret and understand justifications in whatever form they are presented to them. There is some evidence to suggest that sometimes justifications can be difficult to understand, if not impossible. Removing all the layers of abstractions, justifications boil down to taking some premises and performing an operation which allows you to draw a specific conclusion. Sometimes, this process fails to effectively communicate the non-obvious reasoning steps which allow it to bridge the gap between the premises and conclusion. Thus, there is a need for research into justification understanding, more specifically, what kind of mechanisms can assist users in coping with unintuitive and complex justifications.

2.3.3 Justification Computation

In the context of description logics, there are two axes which are used to classify the algorithms used to compute justifications. Single-all-axis is one of them - this is when the algorithm computes either a single justification or all justifications for an entailment. The other one is the reasoner-coupling axis, which deals with whether the algorithm is of the black-box or glass-box variety. A block-box algorithm is simply one where the user cannot view the inner mechanisms of the algorithm and glass-box, as the name implies, is one where the user can see all the inner workings [26].

1. **Single-All axis** Practically it is necessary to differentiate between the single justification and all justification algorithms for two reasons:(a) there is often a dependency between these two types of algorithms, namely that computing all justifications might utilise computing one justification as a subroutine and (b) showing a single justification can prove extremely useful for non-automated ontology debugging and repair. A single justification may reveal the exact source of a problem

and provide the user with a deeper understanding of the problem and an idea for the accompanying solution [26].

2. **Reasoner-Coupling axis** An algorithm for computing a diagnosis can be classified as either a black-box or glass box, a notion introduced by Parsia et al. [46]. For glass-box algorithms, the justifications are derived from the internals of the reasoner - in other words they are the direct product of the reasoning process used to compute the diagnosis. This is not the case for black-box algorithms - they instead use reasoning to calculate if an entailment holds given a specific set of axioms.

With these two axes, this means there are three combinations for computing all justifications for a given entailment:

1. All-black-box with a single-black-box subroutine
2. All-black-box with a single-glass-box subroutine
3. All-glass-box with a single-glass-box subroutine

Black box algorithms have the advantage of being robust and relatively simple to implement - this is due to the fact that the algorithm relies on the availability of a reasoner for the logic being considered. Additionally, these algorithms are compatible for any monotonic logic with decidable entailment checking. Some researchers argue, however, that black-box algorithms may not be practical or efficient as they have the potential to produce a large search space [57].

Justifications being computed "for free" as a result of the reasoning process is regarded as one of the advantages of glass-box algorithms. However, their disadvantages is that, particularly in comparison to black-box algorithms, the implementation is a lot more challenging, as an existing reasoner must be modified to support glass-box algorithms. Doing so is a non-trivial task [26].

The above listed three possible combination algorithms were defined for description logic in [26] and were translated into propositional logic by Wang [61].

Many DL reasoners utilise tableaux-based algorithms, which can be used for many reasoning problems, such as subsumption and satisfiability [5].

Satisifiablility in the case of description logic is essentially - given a concept, is it satisfiable with regards to the knowledge base. Subsumption is concerned with whether or not one concept is a sub-concept of another [3]. These algorithms can be used in conjunction with axiom-pinpointing algorithms for justification finding. Note that an involved discussion on tableaux algorithms and axiom-pinpointing does not fall within the scope of this dissertation, and we refer the reader to the work of Baader [5], as well as that of Baader and Peñaloza [4] for further detail. Axiom-pinpointing is the process of computing the minimal subsets which entail a statement for a given knowledge base and was developed out of a need to help users understand why a particular consequence holds. These subsets can also be used to find the maximal subsets which don't entail a statement. In particular, Baader and Peñaloza developed a general approach for extending a tableaux-based algorithms to a pinpointing algorithm [4]. The significance of this is that for many of the previous tableaux-based algorithms, their use case was restricted to particular DLs. However, the algorithm from Baader and Peñaloza is highly generalisable and thus is applicable to many types of tableaux-based algorithms from DLs as well as other types of reasoning processes, such as finding maximally satisfiable subsets from sets of propositional formulae.

Chapter 3

Non-monotonic Logic

The monotonicity property is one of the biggest limitations of classical logic. We use an example to illustrate this. Here, we present a small knowledge base about birds:

1. Birds are animals that can fly
2. Animals that can fly are unable to swim

From these two facts, anyone could reasonably conclude that birds cannot swim. Of course, this then poses problems when we discover that there are indeed birds that can both fly and swim - such as ducks. We add these facts to our knowledge base:

1. Birds are animals that can fly
2. Animals that can fly are unable to swim
3. Ducks are a type of bird
4. Ducks can fly and swim

From the above, using classical logic we would have to conclude that ducks do not exist. Adding another statement such as "Donald is a duck" would then lead to the knowledge base becoming inconsistent. This is because we have already established that birds are flying creatures which cannot swim and ducks (being birds which can swim) invalidates this inference. This very issue is caused by the monotonicity property mentioned earlier - the inability

to retract inferences that were made earlier when new information is added to the knowledge base that subsequently causes undesirable conclusions or inconsistencies. Of course, one might be tempted to state that we could simply change how the information is modelled. Maybe we could say that *typically* animals that can fly are unable to swim. However, there is no way of modelling this with any monotonic logic. The inability to express typicality within the language consequently implies that such logics cannot to used to model human knowledge which is fraught with exceptions. Explicit handling of exceptions is also unfeasible, given that the nature of the information could change and the amount of information itself could increase. Devising formulae for the model which can account for these changes would be an ongoing, never-ending process.

An argument can also be made that we should simply not allow contradictory information to be added to the knowledge base - but this is unrealistic - knowledge should not be expected to be perfect or entirely non-contradictory before we can use it. Of course, one could perform lengthy checks on the entire knowledge base before allowing information to be added to ensure that no contradictions occur. This is what belief revision does with the revision operator. Belief revision and defeasible reasoning do have an underlying link [19], though for the context of this discussion, especially concerning justifications, we are choosing to focus on KLM-defeasible reasoning as a solution (rather than belief revision).

One proposed solution to this problem is non-monotonic logics. For the curious reader, section 3 from McCarthy [39] catalogues some problems and applications for non-monotonic logics (which are out of scope for this dissertation).

3.1 Preferential Approach

The preferential approach provides the basis for the non-monotonic reasoning used in this dissertation. Initially devised by Shoham [52, 53], the proposal was that the semantics of a logic should be augmented with a preference relation over all interpretations. Then, in order for a formula to be satisfiable, it needs to be satisfied by the set of all its preferred models. This set is composed of all the models of the formula that are minimal with respect to the preference relation. This had implications for entailment, which became preferential entailment - for some formulae, say α, β, we state that $\alpha \mid\!\approx \beta$

i.e. "α preferentially entails β" if and only if all preferred models of α also satisfy β. Additionally, Gabbay provided a basis for a congruent proof-theoretic system for preferential reasoning [21]. Preferential reasoning is stated to be the core of non-monotonic reasoning [27] as it provides the semantic foundations for many non-monotonic logics.

Kraus, Lehmann and Magidor (KLM) [32, 35, 34] expanded on all of these ideas, culminating in the development of the KLM framework. This framework was chosen for this dissertation for the following reasons:

1. It has a model theory that is based on preferential semantics (introduced above)
2. It has a proof theory that provides a set of postulates
3. Finally, the reasoning algorithms for the preferential approach are computationally comparable to the performance of classical reasoning algorithms

Model theory is concerned with the meaning of statements in the language, as well as how truth is interpreted in that language. Logic is used in this context to describe properties of objects and establish when one or more statements are true of objects. In model theory, the concept of truth for an object must be meticulously and precisely defined, and the intended interpretation also has consequences of the meaning of truth. One reasonable assumption one could make use of under these circumstances is that the truth of a statement can be found by examining the truth of its parts [22], at least for the case of propositional logic. This idea of a recursive definition for truth that is based on the syntactical structure of the statement, can be ascribed to Tarski [17].

Proof theory involves using a logic to fulfill the role of a deduction system, otherwise known as a proof system. A deduction system consists of axioms - which is a set of facts, and deduction rules, both of which are used to calculate which facts can be established using these two features. The goal is to determine which proofs can be constructed (if any) and consequently there is little to no concern with the meaning of the statements themselves, but rather the focus is centered on how they can be mechanically manipulated. Due to this characteristic, this paradigm is one that easily lends itself to implementation with a computer [22].

Kraus, Lehmann and Magidor asserted that typicality should be able to be explicitly stated in a logic. This, for example, could look like "a bird

typically flies". As mentioned earlier, this cannot be represented using the syntax of classical logic. Therefore, Lehmann and Magidor [35] and KLM [32] developed an extension to propositional logic which was able to achieve this.

Firstly, we introduce the idea of a preferential consequence relation, $\mid\!\sim$, for example, $\alpha \mid\!\sim \beta$ is read as "from α we conclude β, unless conflicting information is presented". A preferential consequence relation is one which satisfies the KLM postulates (introduced below). Though it should be noted that initially in the literature, the preferential consequence relation operator, twiddle, $\mid\!\sim$, was a meta-level concept, however at some stage its use changed to that of an object-level connective. Both of these views are possible, however in this dissertation $\mid\!\sim$ is viewed as a connective.

Formulae of the form $\alpha \mid\!\sim \beta$ are now read as "α defeasibly implies β", as $\mid\!\sim$ now serves as the defeasible analogue to $\rightarrow$. Note that unlike other connectives in propositional logic, $\mid\!\sim$ cannot be nested, so formulae of the form $\alpha \mid\!\sim (\beta \mid\!\sim \gamma \wedge \sigma)$ would not be permitted.

We also introduce the notion of the KLM postulates:

1. (LLE) Left logical equivalence: $\dfrac{\top \models \alpha \leftrightarrow \beta,\ \alpha \mid\!\sim \gamma}{\beta \mid\!\sim \gamma}$
2. (RW) Right weakening: $\dfrac{\top \models \alpha \rightarrow \beta,\ \gamma \mid\!\sim \alpha}{\gamma \mid\!\sim \beta}$
3. (Ref) Reflexivity: $\alpha \mid\!\sim \alpha$
4. And: $\dfrac{\alpha \mid\!\sim \beta,\ \alpha \mid\!\sim \gamma}{\alpha \mid\!\sim \beta \wedge \gamma}$
5. Or: $\dfrac{\alpha \mid\!\sim \gamma,\ \beta \mid\!\sim \gamma}{\alpha \vee \beta \mid\!\sim \gamma}$
6. (CM) Cautious Monotonicity: $\dfrac{\alpha \mid\!\sim \gamma,\ \alpha \mid\!\sim \beta}{\alpha \wedge \beta \mid\!\sim \gamma}$

There is some nuance to the above postulates - they can be interpreted in two ways:

1. They are properties of preferential consequence relations - this is a more axiomatic approach

2. They act as rules of inference, each providing a pattern of reasoning for defeasible information- a more proof-theoretic/procedural approach

For the purposes of this and later sections, we are electing to use option (1) above, as the introduction of rational monotonicity in later sections breaks the procedural paradigm.

KLM semantics is based upon preferential semantics - the idea that an agent may prefer one interpretation over another, with typicality used as the criterion for this choice. More specifically, they may choose a more typical interpretation over another. For defeasible reasoning, this translates to preferred interpretations being those that are more typical, referred to as *preferential interpretations*.

In order to create this preference ranking, we define *states* - a meta-level object that maps to a classical interpretation. Multiple states can map to the same interpretation, and there can thus be an infinite number of states for a given interpretation.

Definition 3.1.1 (Preferential interpretation)**.** A preferential interpretation $\mathcal{P}$ is a triple $\langle S, l, \prec \rangle$. S denotes the possibly infinite set of states, $l : S \mapsto \mathcal{U}$ is a function which maps the states to interpretations and $\prec$ is a strict partial order on S

For every formula in the language and some preferential interpretation, we want to consider the set of all states for the preferential interpretation where the interpretation associated with each state satisfies the formula. This is similar to the notion of a model for classical PL. More formally:

For every $\alpha \in \mathcal{L}$ and some $\mathcal{P}$, we define $[\![\alpha]\!]^{\mathcal{P}} := \{s \mid s \in S, S \in \mathcal{P}, l(s) \Vdash \alpha\}$

Minimality can also be defined for preferential interpretations:

Definition 3.1.2. For any $\mathcal{P} = \langle S, l, \prec \rangle$, a state is minimal in $\mathcal{P}$ if and only if there is no $s' \in S$ such that $s' \prec s$. $\min_{\prec}(\mathcal{P})$ denotes the set of all such s in some $\mathcal{P}$

Preferential interpretations can have the properties of smoothness and well-foundedness, more precisely:

Definition 3.1.3 (Well-Founded)**.** A preferential interpretation $\mathcal{P}$ is well-founded if and only if $\langle S, \prec \rangle$ is well-founded. For $\langle S, \prec \rangle$ to be well-founded, it must be the case that for any non-empty $S' \subseteq S$ there is a $s \in S'$ that is minimal.

In other words, there must be a subset of S which contains a minimal state. Smoothness is a property of the partial order and is essentially a weakening of well-foundedness, in that its only requirement is that all formulae must have a minimal state which satisfies it.

Definition 3.1.4 (Smooth)**.** A partial order $\prec$ for a given preferential interpretation $\mathcal{P}$ is smooth on condition that for any formula in the language $\alpha \in \mathcal{L}$, it is the case that $[\![\alpha]\!]^{\mathcal{P}}$ has a minimal state s.

If and only if the set of states S in a preferential interpretation is finite in size, can $\mathcal{P}$ then be classified as finite. Both well-founded preferential interpretations and finite interpretations satisfy smoothness.

If a consequence relation satisfies the KLM postulates, then it is said to be a preferential consequence relation. There is also a link between preferential interpretations and preferential consequence relations [32, 35, 27], in that consequence relations can be constructed from preferential interpretations using the following:

Definition 3.1.5. For a preferential interpretation $\mathcal{P} = \langle S, l, \prec \rangle$ and $\alpha, \beta \in \mathcal{L}$, we state that $\mathcal{P}$ defining $\mid\!\sim_{\mathcal{P}}$ (a preferential consequence relation) such that $\alpha \mid\!\sim_{\mathcal{P}} \beta$, is contingent upon $s \Vdash \beta$ for s minimal in $[\![\alpha]\!]^{\mathcal{P}}$.

The above results in the representation theorem [32]:

Theorem 3.1.1. $\mid\!\sim$ *defines a preferential consequence relation if and only if it is a consequence relation defined by a preferential interpretation. In a finite language, every preferential consequence relation is defined by a finite preferential interpretation.*

Ranked interpretations

Ranked interpretations are a subset of preferential interpretations which have a partial order that is modular. A modular partial order is one which satisfies the following properties:

Lemma 3.1.2. *If $\prec$ is a strict partial order on a set $\mathcal{G}$, these next conditions are equivalent:*

1. *for any $x, y, z \in \mathcal{G}$, if $x \not\prec y$, $y \not\prec x$ and $z \prec x$, then $z \prec y$*
2. *for any $x, y, z \in \mathcal{G}$, if $x \prec y$, then either $z \prec y$ or $x \prec z$*
3. *there is a totally ordered set $\Omega, <$ (a strict order) and a ranking function $r : \mathcal{G} \mapsto \Omega$ such that $s \prec t$ if and only if $r(s) < r(t)$*

The vital difference here between the modular and non-modular partial orders is that two incomparable states are placed into the same tier. Therefore in the context of ranked interpretations, given two states, either one is preferred or they are otherwise allocated the same rank; whereas with preferential interpretations there might be multiples ways that states are incomparable. Furthermore, ranked interpretations do not allow for duplicate states, which renders the use of states redundant for ranked interpretations. This in turn enables them to be directly defined as a preference ordering on a set of interpretations - in essence a ranked interpretation acts as a function from the set of interpretations to a totally ordered set [27].

Definition 3.1.6 (Ranked Interpretation)**.** A ranked interpretation is a function $\mathcal{R} : \mathcal{U} \mapsto \mathbb{N} \cup \{\infty\}$ satisfying this convexity condition: for every $i \in \mathbb{N}$, if there exists some $u \in \mathcal{U}$ such that $\mathcal{R}(u) = i$, then there must be a $v \in \mathcal{U}$ such that $\mathcal{R}(v) = j$ with $0 \leq j < i$

Ranked interpretations inherit all the properties of preferential interpretations. This has the following result:

Theorem 3.1.3. *Just as preferential interpretations defines a preferential consequence relation, so too do ranked interpretations - they define a specific type of preferential consequence relation, namely a rational consequence relation [35]*

A rational consequence relation has the following definition:

Definition 3.1.7 (Rational Consequence Relation)**.** A rational consequence relation is a type of preferential consequence relation, $\mid\!\sim$ that satisfies the KLM postulates: LLE, RW, Ref, And, Or, CM and an additional one, rational monotonicity:

$$\text{Rational monotonicity (RM): } \frac{\alpha \mid\!\sim \gamma,\ \alpha \not\mid\!\sim \neg\beta}{\alpha \wedge \beta \mid\!\sim \gamma}$$

3.1.1 Preferential Entailment

For defeasible information, one now needs to redefine how defeasible inferences can be drawn. This starts with defining a new language, $\mathcal{L}_p := \mathcal{L} \cup \{\alpha \mid\sim \beta \mid \alpha, \beta \in \mathcal{L}\}$. This new language will be used for the rest of this dissertation. Given the above, we now provide a formal definition for a defeasible implication:

Definition 3.1.8 (Defeasible Implication)**.** Given $\alpha, \beta \in \mathcal{L}$, a statement of the form $\alpha \mid\sim \beta \in \mathcal{L}_p$ is known as a defeasible implication

Satisfaction for a defeasible implication and a preferential interpretation is as follows:

Definition 3.1.9. Given some preferential interpretation $\mathcal{P}$ and $\alpha \mid\sim \beta$, a defeasible implication, $\mathcal{P} \Vdash \alpha \mid\sim \beta$ if and only if for every s minimal in $[\![\alpha]\!]^{\mathcal{P}}$, $s \Vdash \beta$.

If $\mathcal{P} \Vdash \alpha \mid\sim \beta$ then $\mathcal{P}$ is a model of $\alpha \mid\sim \beta$. We extend the above to sets of defeasible implications:

Definition 3.1.10. Given a defeasible knowledge base, $\mathcal{K}$ and a preferential interpretation $\mathcal{P}$, $\mathcal{P} \Vdash \mathcal{K}$ iff for all $\alpha \mid\sim \beta \in \mathcal{K}$, $\mathcal{P} \Vdash \alpha \mid\sim \beta$.

$\mathcal{P}$ is a model of $\mathcal{K}$ if $\mathcal{P} \Vdash \mathcal{K}$.

It is also possible to express a classical propositional logic formula as a defeasible implication:

Definition 3.1.11. $\mathcal{P} \Vdash \alpha \in \mathcal{L}$, read as "a preferential interpretation satisfies a classical formula from the language", if and only if for all states $s \in \mathcal{P}$ then $s \Vdash \alpha$

The above definition grants us the ability to define classical formulas as defeasible implications:

Corollary 3.1.3.1. *We can express any formula $\alpha \in \mathcal{L}$ as a defeasible implication $\neg\alpha \mid\sim \bot$. For any preferential interpretation $\mathcal{P}, \mathcal{P} \Vdash \alpha$ if and only if $\mathcal{P} \Vdash \neg\alpha \mid\sim \bot$.*

Essentially, a classical formula is satisfied in a preferential interpretation if it is satisfied by every state in said preferential interpretation. By definition, $[\![\bot]\!] = \emptyset$ and $\mathcal{P} \Vdash \neg\alpha \mid\sim \bot$ if and only if $[\![\neg\alpha]\!]^{\mathcal{P}}$. As a result of this, there

are no minimal states where α is false and consequently α is true in the preferential interpretation.

We also define a defeasible knowledge base $\mathcal{K}$ to be a finite set of defeasible implications. It is possible to define a knowledge base of infinite size but we do not consider that condition for this context.

Defeasible entailment will be denoted with $\mid\!\approx$ as opposed to $\models$ and preferential entailment is shown with a P subscript i.e. $\mid\!\approx_P$. Preferential entailment is an entailment relation that uses preferential semantics:

Definition 3.1.12. Given some defeasible knowledge base $\mathcal{K}$ and defeasible implication, we state that $\mathcal{K} \mid\!\approx_P \alpha \mid\!\sim \beta$ if and only if for every preferential model $\mathcal{P}$ of $\mathcal{K}$, $\mathcal{P} \Vdash \alpha \mid\!\sim \beta$.

The above essentially parallels classical entailment semantics, though modified to account for preferential semantics. In other words, a query is preferentially entailed by the knowledge base if it is satisfied by every preferential model of the knowledge base. The KLM postulates (excluding rational monotonicity) defined in previous sections can be rewritten for the preferential approach - see below. Note that since $\mid\!\sim$ has become an object-level connective, $\mid\!\approx$ has now become a meta-level notion of consequence.

- Left logical equivalence (LLE): $\frac{\mathcal{K} \mid\!\approx \alpha \leftrightarrow \beta, \mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \mid\!\approx \beta \mid\!\sim \gamma}$
- Right weakening (RW): $\frac{\mathcal{K} \mid\!\approx \alpha \rightarrow \beta, \mathcal{K} \mid\!\approx \gamma \mid\!\sim \alpha}{\mathcal{K} \mid\!\approx \gamma \mid\!\sim \beta}$
- Reflexivity (Ref): $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \alpha$
- And: $\frac{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta, \mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma}{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta \wedge \gamma}$
- Or: $\frac{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma, \mathcal{K} \mid\!\approx \beta \mid\!\sim \gamma}{\mathcal{K} \mid\!\approx \alpha \vee \beta \mid\!\sim \gamma}$
- Cautious Monotonicity (CM): $\frac{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma, \mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta}{\mathcal{K} \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma}$

A defeasible entailment relation $\mid\!\approx$ is said to LM-rational if it adheres to these postulates and rational monotonicity [27]:

- Rational Monotonicity (RM): $\frac{\mathcal{K} \mid\!\approx \alpha \mid\!\sim \gamma,\ \mathcal{K} \not\mid\!\approx \alpha \mid\!\sim \neg \beta}{\mathcal{K} \mid\!\approx \alpha \wedge \beta \mid\!\sim \gamma}$

3.2 Non-monotonic reasoning

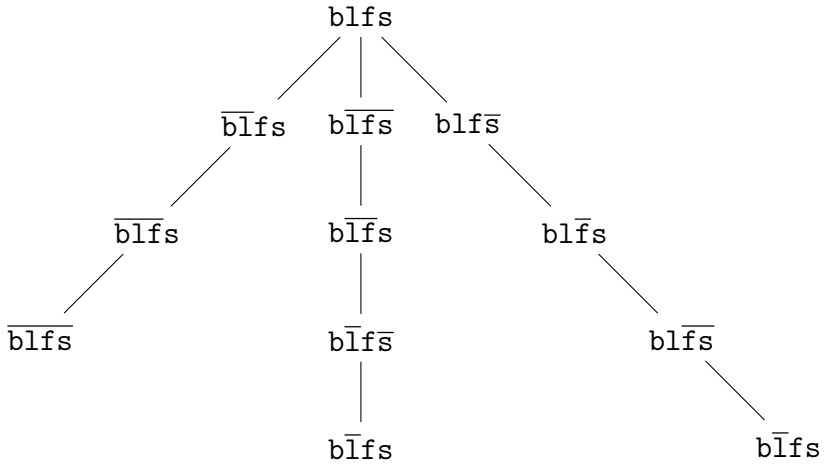
Preferential entailment for defeasible knowledge bases is still monotonic and is thus still insufficient for use in defeasible reasoning. Proofs of this can be found in [35].

To provide an example of how the above works, consider the knowledge base shown in table 3.1 and query $\mathcal{K} \mid\!\!\approx_P l \mid\sim s$ from [27].

boat $\mid\sim$ floats $(b \mid\sim f)$
boat $\mid\sim$ sailors $(b \mid\sim s)$
$\neg$ (leaky $\rightarrow$ boat) $\mid\sim \bot$ $(\neg(l \rightarrow b) \mid\sim \bot)$
leaky $\mid\sim \neg$ floats $(l \mid\sim \neg f)$

Table 3.1: Simple defeasible knowledge base about boats

We also define a preferential interpretation $\mathcal{P}^{\mathcal{K}}$. We depict this visually - each node represents a state such that is $s \prec s'$ then s is positioned below s' and these nodes are connected by an edge. Each state is labelled $l(s)$, which connects the valuation and the state.



We can now test if this preferential interpretation models the knowledge base. For that to be true, each statement in the knowledge base must be satisfied in $\mathcal{P}^{\mathcal{K}}$. The process is as follows:

1. We start with $b \mid\sim f$, which is a defeasible implication and thus check only the minimal b interpretations. For this example, it is b̄lfs. This statement also models f and so $\mathcal{P}^{\mathcal{K}} \Vdash b \mid\sim f$.

2. Next is $b \mid\sim s$ - once again only the minimal b interpretation needs to be checked and so we test the same interpretation as in the previous step. $\mathtt{b\bar{l}fs} \Vdash s$ and thus $\mathcal{P}^{\mathcal{K}} \Vdash b \mid\sim s$.

3. For the third statement, $\neg(l \rightarrow b) \mid\sim \bot$, we convert it to it's classical logic equivalent, $l \rightarrow b$. This statement needs to hold in all interpretations of $\mathcal{P}^{\mathcal{K}}$ rather than only the minimal ones. The following interpretations satisfy satisfy `l`: `blfs`, $\mathtt{blf\bar{s}}$, $\mathtt{bl\bar{f}s}$ and $\mathtt{bl\overline{fs}}$. All four of these interpretations are also models of b and hence $\mathcal{P}^{\mathcal{K}} \Vdash \neg(l \rightarrow b) \mid\sim \bot$.

4. Finally, we have `leaky` $\mid\sim \neg$ `floats` - in order for this to be satisfied by the interpretation, all the minimal interpretations which satisfy `leaky` must not satisfy `floats`. The minimal interpretation for `leaky` in this instance is $\mathtt{bl\overline{fs}}$, and since we have $\mathtt{\bar{f}}$, the consequence is that $\mathcal{P}^{\mathcal{K}} \Vdash l \mid\sim \neg f$

The above allows us to conclude that the preferential interpretation is a model of $\mathcal{K}$ i.e. $\mathcal{P}^{\mathcal{K}} \Vdash \mathcal{K}$. Now we can test if $\mathcal{P}^{\mathcal{K}}$ satisfies the original query: $\mathcal{K} \mid\approx_P l \mid\sim s$. The minimal interpretation for `leaky` is $\mathtt{bl\overline{fs}}$ - however, we have $\mathtt{\bar{s}}$, and so this minimal interpretation is not a model of `sailors`. Hence we conclude $\mathcal{P}^{\mathcal{K}} \nVdash \mathcal{K} \mid\approx_P l \mid\sim s$.

This example demonstrates a very important property of preferential entailment - that is, exceptional sub-classes do not always inherit properties from the original class. Note that in previous sections, it was mentioned that a knowledge base preferentially entails a defeasible implication if and only if every preferential model of the knowledge base is also a model of the defeasible implication. The above serves as a counterexample to this, with the conclusion being that $\mathcal{K} \mid\approx_P$ **leaky** $\mid\sim$ **sailors**.

The above result illustrates the need for a different entailment relation in order to define non-monotonic entailment using preferential semantics. The key entailment relation which is used for this is rational closure - it is the non-monotonic core of defeasible entailment and is the most conservative form of reasoning [15]. Highly detailed explanations of the varying entailment relations is beyond the means of this dissertation, though references for further reading will be provided.

Some scholars argue that preferential entailment is actually adequate for reasoning with defeasible knowledge bases. Furthermore, they suggest that rational monotonicity may lead to unwanted inferences being drawn (when used as a rule of inference) [23]. Their argument is based in description logic,

however these can be reworked for the context of propositional logic. An in-depth analysis of this school of thought is however, beyond the means of this dissertation.

3.2.1 Minimal Ranked Entailment

Minimal ranked entailment provides the semantics for a non-monotonic entailment relation. Models for a defeasible knowledge base can (a) be ranked and (b) have a partial order defined over them, such that the more preferred ranked interpretations, will be "pushed" further down the rankings. This partially ordered set has a minimal element [24] (essentially a ranked interpretation) that represents a specific pattern of reasoning and will align precisely with that of the preferential entailment relation. This is referred to as "minimal ranked entailment" (MRE) and its non-monotonicity makes it highly desirable as a basis for entailment relations in defeasible reasoning [27].

3.2.2 Rational Closure

Rational closure (RC) is a form of non-monotonic entailment and provides a syntactic definition for MRE, and vice-versa, MRE is the semantic definition of RC. Given that the focus of this dissertation is on how justifications are *computed* for KLM-style defeasible reasoning, we are electing not to provide a lengthy discussion on minimal ranked entailment (i.e. the semantics of rational closure) - instead opting to provide a small formal definition for rational closure later on in this section. Notably, in chapter 4, the algorithms used for computing weak and strong justifications are extensions and modifications of the algorithm for rational closure. Consequently, we have decided to dedicate more time to explore the algorithms which underpin rational closure, rather than the semantics of the formalism itself.

The algorithms for rational closure uses the process of *materialisation* and *exceptionality* in order to perform entailment. In this sub-section we provide the theoretical background for some parts of the algorithm, and in the sub-section that follows we present pseudo-code and some examples. Many forms of defeasible entailment (such as lexicographic closure) can be defined by refining rational closure, which illustrates its importance in the literature.

Defeasible implications also have a material counterpart, otherwise known as the materialisation:

Definition 3.2.1 (Materialisation)**.** A defeasible implication $\alpha \mathrel{|\!\sim} \beta$ has the material counterpart $\alpha \rightarrow \beta$. Given a defeasible knowledge base $\mathcal{K}$, the materialised knowledge base $\overrightarrow{\mathcal{K}}$, is the set containing the materialised counterpart for every defeasible implication found in $\mathcal{K}$.

For any defeasible knowledge base, $\mathcal{K}$, materialisation is used to determine which classical formulae are satisfied by the most preferred interpretations of $\mathcal{K}$ [35], per this lemma:

Lemma 3.2.1. *Given some defeasible knowledge base, $\mathcal{K}$ and some formula $\alpha \in \mathcal{L}$, $\overrightarrow{\mathcal{K}} \models \alpha$ holds if and only if $\mathcal{K} \mathrel{|\!\approx}_P \top \mathrel{|\!\sim} \alpha$*

The above lemma provides a crucial result that can be used to define *exceptionality*. We can define how exceptional a formula is relative to a knowledge base:

Definition 3.2.2 (Exceptional)**.** Given a knowledge base $\mathcal{K}$ and α, a propositional formula, α is exceptional for $\mathcal{K}$ if and only if $\mathcal{K} \mathrel{|\!\approx}_P \top \mathrel{|\!\sim} \neg\alpha$

In other words, if a formula's negation is typical in every preferential interpretation that satisfies the knowledge base, then that formula is said to be exceptional. In this context, preferential entailment is used to delimit levels of specificity in a knowledge base. Using exceptionality as a basis, we can then define an ε *function* which returns a subset of the knowledge base that contains all defeasible statements which have antecedents found to be exceptional in the original knowledge base:

Definition 3.2.3 (ε function)**.** $\varepsilon(\mathcal{K}) = \{\alpha \mathrel{|\!\sim} \beta \in \mathcal{K} \mid \mathcal{K} \mathrel{|\!\approx}_P \top \mathrel{|\!\sim} \neg\alpha\}$

The ε function is used to define the exceptionality sequence for a knowledge base, where each successive knowledge base contains statements that are more specific i.e. less defeasible than the ones that preceded it:

Definition 3.2.4 (Exceptionality sequence)**.** The exceptionality sequence is an iterative sequence of knowledge bases, $\mathcal{E}_0^{\mathcal{K}}, \mathcal{E}_1^{K} ... \mathcal{E}_n^{\mathcal{K}}$ where $\mathcal{E}_0^{\mathcal{K}} = \mathcal{K}$ and $\mathcal{E}_{i+1}^{\mathcal{K}} = \varepsilon(\mathcal{E}_i^{\mathcal{K}})$ for $1 \leq i \leq n$.

The last knowledge base, $\mathcal{E}_n^{\mathcal{K}}$, is the smallest i such that $\varepsilon(\mathcal{E}_i^{\mathcal{K}}) = \mathcal{E}_{i+1}^{\mathcal{K}}$.

The final knowledge base in the sequence may be labelled with ∞ rather than n, i.e. $\mathcal{E}_\infty^{\mathcal{K}}$ and formulae that fall into $\mathcal{E}_\infty^{\mathcal{K}}$ are essentially equivalent to classical formulae. Note also that since we defined $\mathcal{K}$ to be finite, n must exist. Furthermore, it is also possible that $\mathcal{E}_\infty^{\mathcal{K}} = \emptyset$ in the case where the knowledge base does not contain any statements which are not retractable.

Using a similar intuition to the $\mathcal{E}$ subsets defined above, we define a ranking of statements in $\mathcal{K}$, the defeasible knowledge base: $\mathcal{R}_0^{\mathcal{K}}, ..., \mathcal{R}_\infty^{\mathcal{K}}$, whereby the ranks are disjoint from each other.

Definition 3.2.5 (Defeasible Statement Ranking)**.** For some knowledge base $\mathcal{K}$, $\mathcal{R}_i^{\mathcal{K}} = \mathcal{E}_i^{\mathcal{K}} \setminus \mathcal{E}_{i+1}^{\mathcal{K}}$ for $0 \leq i \leq n-1$ and $\mathcal{R}_\infty^{\mathcal{K}} = \mathcal{E}_\infty^{\mathcal{K}}$

Following on from this, we can now define a formula's *base rank* for its antecedent, i.e. $br_{\mathcal{K}(\alpha)}$:

Definition 3.2.6 (Base Rank)**.** $br_{\mathcal{K}(\alpha)}$ is the smallest r such that α is not exceptional within $\mathcal{E}_r^{\mathcal{K}}$. It is defined thus: $br_{\mathcal{K}(\alpha)} := min\{r \mid \mathcal{E}_r^{\mathcal{K}} \not\approx \top \mathrel{|\!\sim} \neg\alpha\}$

If there is no r where $\mathcal{E}_r^{\mathcal{K}} \not\approx \top \mathrel{|\!\sim} \neg\alpha$, then the base rank of the antecedent is said to be infinity i.e. $br_{\mathcal{K}}(\alpha) = \infty$.

The intuition here is that a statement with a lower rank is *more defeasible* than that of a statement that ranks above it, and the base rank of the defeasible implication is the base rank of the antecedent. For example, $br_{\mathcal{K}}(\alpha \mathrel{|\!\sim} \beta) \equiv br_{\mathcal{K}}(\alpha)$.

Using the above concept of base rank, one can define rational closure as the following: [24]:

Definition 3.2.7. Given $\mathcal{K}$ as a defeasible knowledge base and $\alpha \mathrel{|\!\sim} \beta$ as a defeasible implication, $\alpha \mathrel{|\!\sim} \beta$ falls into the rational closure of $\mathcal{K}$ i.e $\mathcal{K} \not\approx_{RC} \alpha \mathrel{|\!\sim} \beta$ if and only if $br_{\mathcal{K}}(\alpha) < br_{\mathcal{K}}(\alpha \wedge \neg\beta)$ or $br_{\mathcal{K}}(\alpha) = \infty$.

Algorithm

Previous sections detailed that rational closure can be defined semantically and then also in terms of base ranks. In this section, we present an algorithm for computing the rational closure of a defeasible knowledge base. It nominally consists of 2 parts: `BaseRank` (proposed by Freund) [20] and `RationalClosure`.

First, we define BaseRank [27], an algorithm that takes $\mathcal{K}$ as input computes the base rank for all explicit formulae in $\mathcal{K}$ and maps these formulae to $\mathbb{N} \cup \{\infty\}$

Each formula is assigned the lowest rank possible whilst still respecting exceptionality. Initially, $\mathcal{K}$ undergoes the materialisation process, producing $\overrightarrow{\mathcal{K}}$, which is the knowledge base that most of the algorithm operates on. `BaseRank` outputs a tuple of sets of the material counterparts for defeasible implications found in $\mathcal{K}$. This algorithm ensures that no "gaps" occur between ranks, as this is one of the requirements for a well-defined base rank.

Algorithm 1 BaseRank

Input: A knowledge base $\mathcal{K}$
Output: An ordered tuple $(R_0, ..., R_{n-1}, R_\infty, n)$

```
1: i := 0;
2: E_0 := K→;
3: while E_{i-1} ≠ E_i do
4:     E_{i+1} := {α → β ∈ E_i | E_i ⊨ ¬α};
5:     R_i := E_i \ E_{i+1};
6:     i := i + 1;
7: R_∞ := E_{i-1};
8: if E_{i-1} = ∅ then
9:     n := i − 1;
10: else
11:     n := i;
12: return (R_0, ..., R_{n-1}, R_∞, n)
```

The second part, RationalClosure, also accepts a defeasible knowledge base as input ($\mathcal{K}$), as well as some query, say $\alpha \mathrel{|\!\sim} \beta$. If $\mathcal{K}$ entails the query, then the algorithm returns true, and false otherwise. Simply, the algorithm works as follows:

1. Check if $\overrightarrow{\mathcal{K}} \models \neg\alpha$, i.e. if the entire materialised knowledge base (classically) entails the negated antecedent of the query

2. If yes, then the implication is that the antecedent is exceptional (and so by extension the query itself).

3. The statements of the lowest numbered rank is removed, which is rank 0. Step 1 is repeated until we find that the antecedent is no longer exceptional i.e. produce some subset of the original knowledge base OR we run out of ranks to remove i.e. only statements within the infinite rank remain (as noted earlier, these are non-retractable).

   (a) If a suitable subset has been found, then check if that entails the materialised query and output that result - either True or False

Algorithm 2 RationalClosure

Input: A knowledge base $\mathcal{K}$, and a defeasible implication $\alpha \mathrel{|\!\sim} \beta$
Output: **true**, if $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$, and **false** otherwise

1: $(R_0, ..., R_{n-1}, R_\infty, n) :=$ BaseRank($\mathcal{K}$);
2: $i := 0$
3: $R := \bigcup_{i=0}^{j<n} R_j$;
4: **while** $R_\infty \cup R \models \neg\alpha$ and $R \neq \emptyset$ **do**
5: $\quad R := R \setminus R_i$;
6: $\quad i := i + 1$;
7: **return** $R_\infty \cup R \models \alpha \rightarrow \beta$;

Given some defeasible knowledge base $\mathcal{K}$ and some query, $\alpha \mathrel{|\!\sim} \beta$, Freund [20] has shown that the above algorithm will return true if and only $\alpha \mathrel{|\!\sim} \beta$ is in the rational closure of $\mathcal{K}$. It is also comparable computationally to classical entailment, as it utilises a classical entailment checker, which means the number of entailment checks being performed for this algorithm is a polynomial function of the knowledge base's size [27]. It is also worth mentioning that most algorithms for the various defeasible entailment relations mentioned in this dissertation (and their accompanying justification-deriving algorithms) will return either `True` or `False` unless explicitly otherwise state.

Worked Example

Example 3.2.1 (Rational Closure)**.** Consider the defeasible knowledge base from Morris et al.[45], given in table 3.2. Note that for statements which appear to be classical e.g. `adults` $\rightarrow$ `people`, these are actually defeasible implications of the form $\neg\alpha \mathrel{|\!\sim} \bot$ where α represents a formula or formulae.

`adults` $\rightarrow$ `people` $(a \rightarrow p)$
`students` $\rightarrow$ `people` $(s \rightarrow p)$
`people` $\mid\sim$ `watch movies` $(p \mid\sim m)$
`people` $\mid\sim$ `adults` $(p \mid\sim a)$
`people` $\mid\sim$ `pay taxes` $(p \mid\sim t)$
`students` $\mid\sim \neg$ `taxes` $(s \mid\sim \neg t)$

Table 3.2: Simple defeasible knowledge base about people

∞	$a \rightarrow p, s \rightarrow p$
1	$s \mid\sim \neg t$
0	$p \mid\sim m, p \mid\sim a, p \mid\sim t$

Table 3.3: Materialised base ranking derived from the knowledge base shown in 3.2

For typographic compactness and readability we are using this representation here and for the remaining sections of this dissertation.

Now, consider a simple query such as "do students typically not pay taxes", or $s \mid\sim \neg t$? Naturally, for trivial examples such as this, where one can easily and quickly scan the knowledge base, the answer is obvious. But here, we are aiming to demonstrate a concept which can be expanded to far more complex knowledge bases where it is very possible (and probable) that most of its users may not be domain experts.

First, we convert every statement in the knowledge base to its materialised counterpart and perform the `BaseRank` step of the algorithm, which results in the following materialised ranking shown in table 3.3:

Then, we test if the entire knowledge base entails the negated antecedent of the query i.e. $\overrightarrow{\mathcal{K}} \models \neg s$. Since it does, we remove R_0 and perform another check. In this situation, the answer is no. From here, we now consider only $R_1 \bigcup R_\infty$ and since we can no longer conclude $\neg s$, we stop and perform a classical entailment check: $R_1 \bigcup R_\infty \models s \rightarrow \neg t$. For this example, the algorithm would return `True`.

bird $\mid\sim$ flies
bird $\mid\sim$ wings
robin $\rightarrow$ bird
penguin $\rightarrow$ bird
penguin $\mid\sim \neg$ flies

Table 3.4: Simple defeasible knowledge base (B)

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

Table 3.5: Base Ranking for simple knowledge base (B)

3.2.3 Lexicographic Closure

Of all the KLM defeasible entailment formalisms, rational closure is the most conservative. It is adequate enough for many applications, however, here are some of the associated problems with using this conservative approach:

- It can lead to entire ranks of statements being removed, even if some of said statements have no relation to those that "disprove" the antecedent.
- It may be the case that of the statements that do disprove the antecedent, only one or two might need to be discarded.

To illustrate this, consider the following knowledge base (table 3.4 with accompanying ranking (table 3.5).

For the query $p \rightarrow f$, in the context of rational closure, we would remove all of R_0 before checking the entailment. However, the statement $b \mid\sim w$ has no effect on whether or not we can conclude $p \mid\sim \neg f$ and it would thus be unnecessary to remove it.

The solution to the above requires a more fine-grained approach to statement removal, which is where lexicographic closure comes into play. Proposed by Lehmann [34], lexicographic closure uses Reiter's [48] concept of default reasoning. Since it can be defined by a ranked interpretation (i.e. has a semantic definition), it is therefore also LM-rational.

Algorithm

The algorithm for lexicographic closure is a refinement of the one for rational closure - either the ranking or removal process is altered. For the purposes of this and later sections, we will only consider the latter. For lexicographic closure, we are trying to ensure that only the exact statement or statements which cause a conflict are removed.

At a high level, the method involves two main steps:

1. When given a query, we first rank the statements using the base ranking algorithm.

2. We then construct all possible subsets of the worst ranked statements and test to see which causes an inconsistency, removing the minimal subset such that there are no longer any inconsistencies.

More specifically, for step (2) above, given a rank R_i, we consider all fixed-size subsets of the rank and combine each of them with the rest of the ranks that follow R_i, i.e. $R_i + 1, ..., R_\infty$. We take n to refer to the number of statements that a rank contains and x to be the number of statements we want to discard, with x starting at 1. We then calculate all subsets of size $n - x$, and these represent all the possible configurations to remove x statements from rank $R_i 1$.

Upon considering all of these "sub-knowledge bases", if we can disprove the antecedent in all of them, then it is evident that that statements required to disprove the antecedent are still contained within them - which prompts the need for more statements to be removed. Incrementing x by 1 for every iteration, this process will be repeated until there is some x where the sub-knowledge base does not get disproved or $x = n$. For the the latter, the entire rank is removed and the entire process then gets repeated with the next level.

Combined Formula Approach

Before we introduce the algorithm, there are two observations which need to be taken into consideration:

1. $\mathcal{K} \cup \{\alpha \vee \beta\} \models \gamma$ iff $\mathcal{K} \cup \{\alpha\} \models \gamma$ and $\mathcal{K} \cup \{\beta\} \models \gamma$

2. $\mathcal{K} \cup \{\alpha \wedge \beta\} \models \gamma$ iff $\mathcal{K} \cup \{\alpha, \beta\} \models \gamma$

Point (1) essentially states that differing formulae can be combined using the *or* connection, and in doing so, one can compute precisely the statements which are entailed by all the knowledge bases. Point (2) holds in the context of classical entailment and shows that by combining statements within the knowledge base using the *and* connective, multiple statements can be treated as though they were a single statement.

These two properties inform how the sub-knowledge bases created in step 7 of the algorithm, where we do the construction of all possible subsets of the worst-ranked statements and then combine then, can be combined and checked:

- For a single subset of size y from rank R_i, let $S = \alpha_1 \wedge \alpha_2 \wedge ... \wedge \alpha_y$
- When taking into account all possible subsets of size y from rank R_i i.e. $S_1, S_2, ..., S_y$, take the disjunction of these subsets $S_1 \vee S_2 \vee ... \vee S_z$

Bearing the above in mind, we now present the algorithm for Lexicographic Closure.

Algorithm 3 Lexicographic Closure

Input: A knowledge base $\mathcal{K}$ and query $\alpha \mid\sim \beta$

Output: if $\mathcal{K} \mid\approx_{LC} \alpha \mid\sim \beta$ return **true**, otherwise return **false**

1: $(R_0, ..., R_{n-1}, R_\infty, n) := BaseRank(\mathcal{K})$
2: i := 0;
3: $R := \bigcup_{j=0}^{j<n} R_j$;
4: **while** $R_\infty \cup R \models \neg\alpha$ and $R \neq \emptyset$ **do**
5: $\quad R := R \setminus R_i$;
6: $\quad m := |R_i| - 1$;
7: $\quad R_{i,m} := \bigvee_{X \in Subsets(R_i,m)} \bigwedge_{x \in X} x$;
8: $\quad$ **while** $R_\infty \cup R \cup \{R_{i,m}\} \models \neg\alpha$ and $m > 0$ **do**
9: $\quad\quad m := m - 1$;
10: $\quad\quad R_{i,m} := \bigvee_{X \in Subsets(R_i,m)} \bigwedge_{x \in X} x$;
11: $\quad R := R \cup \{R_{i,m}\}$
12: $\quad i := i + 1$;
13: **return** $R_\infty \cup R \models \alpha \rightarrow \beta$

The Lexicographic Closure algorithm demonstrates that essentially what we are doing is performing classical entailment checking on a series of sub-

knowledge bases. The final entailment holds if and only if the classical entailment of the materialised version of the defeasible query holds in each of the sub-knowledge bases.

Worked Example

Example 3.2.2 (Computing Lexicographic Closure)**.** Given the knowledge base in table 3.6[1] and the accompanying table of statement rankings produced when the `BaseRank` algorithm is applied, consider the query: $p \mathrel{|\!\sim} f$?

`bird` $\mathrel{|\!\sim}$ `claws` $(b \mathrel{|\!\sim} c)$
`bird` $\mathrel{|\!\sim}$ `hollow bones` $(b \mathrel{|\!\sim} h)$
`bird` $\mathrel{|\!\sim}$ `feathers` $(b \mathrel{|\!\sim} f)$
`penguin` $\mathrel{|\!\sim} \neg$`(claws` $\wedge$ `hollow bones)` $(p \mathrel{|\!\sim} \neg(c \wedge h))$
`penguin` $\rightarrow$ `bird` $(p \rightarrow b)$

Table 3.6: Simple defeasible knowledge base about bird morphology

∞	$p \rightarrow b$
1	$p \mathrel{|\!\sim} \neg(c \wedge h)$
0	$b \mathrel{|\!\sim} f$, $b \mathrel{|\!\sim} c$, $b \mathrel{|\!\sim} h$

Table 3.7: Base Ranking for simple knowledge base shown in table 3.6

First, we consider whether we can conclude $\neg p$. Since this is the case, we can then consider all ways of removing statements from R_0. This produces a set of sub-knowledge bases, which we label as $R_{0:1}, R_{0:2}, R_{0:3}$, where 0 represents the rank and the : *number* represents the specific sub-knowledge base:

0	$b \mathrel{|\!\sim} f$, $b \mathrel{|\!\sim} c$	0	$b \mathrel{|\!\sim} c$, $b \mathrel{|\!\sim} h$	0	$b \mathrel{|\!\sim} f$, $b \mathrel{|\!\sim} h$

Table 3.8: Sub-knowledge bases for example 3.2.2

[1]For the statements in the knowledge base: penguins have solid, hard bones, whereas most other types of birds have hollow, air-filled bones. See [41] for more information

It is at this point we would like to remind the reader that R_∞ and R_1 are included in each of these sub-knowledge bases. From here, we would then once again perform a classical entailment check, with each of the sub-knowledge bases; more explicitly, we check: $R_{0:1} \vee R_{0:2} \vee R_{0:3} \bigcup R_1 \bigcup R_\infty \models \neg p$? This does not hold, and so the statement removal phase of the algorithm stops and we perform a final classical entailment check: $R_{0:1} \vee R_{0:2} \vee R_{0:3} \bigcup R_1 \bigcup R_\infty \models p \rightarrow f$? This does hold, and so the algorithm returns `True`.

For the preceding example, the lexicographic closure algorithm returned `True`. However, the rational closure algorithm, being the more conservative form of reasoning [27], would return `False`. To prove this, we shall use the same knowledge base and query from example 3.2.2. We can also reuse the ranking in table 3.7, since `BaseRank` returns the same ranking regardless of the formalism under consideration

First, we test if $\overrightarrow{\mathcal{K}} \models \neg p$. Since this is true, we remove R_0 entirely. This leaves R_1 and R_∞ as the only remaining ranks. Now, we test if $R_1 \cup R_\infty \models \neg p$. Since it does not, we stop the removal process and test the query once more i.e. does $R_1 \cup R_\infty \models p \rightarrow f$? The answer is no, the entailment does not hold, with the cause being that we discarded the crucial statement, $b \nvdash f$, when R_0 was removed. This demonstrates how much less conservative lexicographic closure can be when compared to rational closure.

3.2.4 Relevant Closure

The problem with rational closure is that it can be too conservative. Therefore, Casini et al. [14] proposed a different formalism - *relevant closure*. Developed initially for description logics (though it is trivial to translate them into a propositional context as is done here), it is an adaptation of rational closure that argues that only statements in a *less specific* rank that disagree with statements in a more specific rank (in relation to a query's antecedent) should be removed. There are two forms of relevant closure: basic relevant closure (more conservative) and minimal relevant closure (less conservative compared to basic). It should also be noted that relevant closure is only defined algorithmically i.e. there is no semantic definition. And because it does not satisy some of the KLM postulates -namely *or, cautious monotonicity* and *rational monotonicity*, it is not LM-rational.

Understanding relevant closure begins by defining the notion of an ε-justification:

Definition 3.2.8 (ε-justification)**.** Given a defeasible knowledge base $\mathcal{K}$ and some formula $\alpha \in \mathcal{L}$, then this pair $(\mathcal{K}, \alpha)$ has an ε-justification, notated as $\mathcal{J}_\varepsilon$, if $\overrightarrow{\mathcal{J}_\varepsilon}$ is a classical justification for $\overrightarrow{\mathcal{K}} \models \neg\alpha$

Consider the simple knowledge base and accompanying ranking in table 3.4 from the rational closure section. An example of an ε-justification would be: $p \to b, b \mathrel{|\!\sim} f, p \mathrel{|\!\sim} \neg f$. This is because once we materialise these statements, we can conclude $\neg p$ i.e $\{p \to b, b \to f, p \to \neg f\} \models \neg p$.

Following from the above, we now elaborate on basic and minimal relevance:

Definition 3.2.9 (Basic Relevance)**.** Given a knowledge base $\mathcal{K}$ and defeasible implication $\alpha \mathrel{|\!\sim} \beta$, if $\alpha \mathrel{|\!\sim} \beta$ is an element in an ε-justification for $(\mathcal{K}, \gamma)$, then the defeasible implication is said to be basically relevant for $(\mathcal{K}, \gamma)$.

Definition 3.2.10 (Minimal Relevance)**.** Given a knowledge base $\mathcal{K}$ and query $\alpha \mathrel{|\!\sim} \beta$, the formulae that are minimally relevant to the query are:

$$\bigcup\{min_{br}\mathcal{J}_\varepsilon \mid \mathcal{J}_\varepsilon \in \mathcal{J}_\varepsilon(\mathcal{K}, \alpha)\}$$

The $min_{br}\mathcal{J}_\varepsilon$ part of the above produces the formulas in $\mathcal{J}_\varepsilon$ that have antecedents with the smallest base rank:

$$min_{br}\mathcal{J}_\varepsilon = \{\gamma \mathrel{|\!\sim} \delta \mid \gamma \mathrel{|\!\sim} \delta \in \mathcal{J}_\varepsilon, \text{and for all } \eta \mathrel{|\!\sim} \omega \in \mathcal{J}_\varepsilon, br^{\mathcal{K}}(\gamma) \leq br^{\mathcal{K}}(\eta)\}$$

Both of these relevance definitions allow for the partitioning of the knowledge base into a section that contains formulas relevant to the query and those that are not.

Definition 3.2.11 (Relevance Query Partition)**.** Given a knowledge base $\mathcal{K}$, a query such as $\alpha \mathrel{|\!\sim} \beta$ and either basic or minimal relevance, the relevance partition, denoted $< \mathcal{R}, \mathcal{R}^- >$ for $\mathcal{K}$ is given by:

$$R = \{\gamma \mathrel{|\!\sim} \delta \mid \gamma \mathrel{|\!\sim} \delta \in \mathcal{K} \text{ is relevant to } \alpha \mathrel{|\!\sim} \beta\}$$
$$R^- = \mathcal{K} \setminus R$$

The preceding definitions enable us to define relevance closure:

Definition 3.2.12 (Relevant Closure)**.** Given a knowledge base $\mathcal{K}$, a query such as $\alpha \mid\sim \beta$ and relevance partition $< \mathcal{R}, \mathcal{R}^- >$, both basic and minimal relevant closure are defined thus:

$$\mathcal{K} \mid\approx_{RelC} \alpha \mid\sim \beta \text{ if } br^{\mathcal{K}}(\alpha) = \infty \text{ or } \overrightarrow{\mathcal{E}_{\alpha}^{\mathcal{K}} \cup R^-} \models \alpha \mid\sim \beta$$

Algorithm

The algorithm for relevant closure is a derivation and modification of the RationalClosure algorithm:

Algorithm 4 RelevantClosure

Input: A knowledge base $\mathcal{K}$, a query $\alpha \mid\sim \beta$ and partition $< \mathcal{R}, \mathcal{R}^- >$
Output: **true**, if $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ for the form of relevance closure under consideration, and **false** otherwise

1: $(\mathcal{K}_0, ..., \mathcal{K}_{n-1}, \mathcal{K}_\infty, n) :=$ BaseRank($\mathcal{K}$);
2: $i := 0$;
3: $\mathcal{K} := \bigcup_{i=0}^{j<n} \mathcal{K}_j$;
4: $\mathcal{K}' := \mathcal{K}$;
5: **while** $\mathcal{K}_\infty \cup \mathcal{K} \models \neg\alpha$ and $i < n$ **do**
6: $\quad \mathcal{K}' := \mathcal{K}' \setminus (\mathcal{K}_i \cap R)$;
7: $\quad i := i + 1$;
8: **return** $\mathcal{K}_\infty \cup \mathcal{K} \models \alpha \rightarrow \beta$;

Since we now need to enumerate justifications, the computational complexity of `RelevantClosure` is higher than that of `RationalClosure`, though it is not to the point where it becomes infeasible, especially for the classical case [14, 26].

Worked Example

We now adapt an example from Morris et al. [45] in order to illustrate the relevant closure algorithm.

Example 3.2.3 (Computing Relevant Closure)**.** Given the knowledge base in table 3.9 and the accompanying table of statement rankings produced when the `BaseRank` algorithm is applied, consider the query: $p \mid\sim m$?

`adults` $\rightarrow$ `people` $(a \rightarrow p)$
`students` $\rightarrow$ `adults` $(s \rightarrow a)$
`adults` $\mid\sim$ `watch movies` $(a \mid\sim m)$
`adults` $\mid\sim$ `pay taxes` $(a \mid\sim t)$
`adults` $\mid\sim$ `have jobs` $(a \mid\sim j)$
`student` $\mid\sim$ `don't pay taxes` $(s \mid\sim \neg t)$

Table 3.9: Simple defeasible knowledge base about students

∞	$a \rightarrow p$, $s \rightarrow a$
1	$s \mid\sim \neg t$
0	$a \mid\sim m$, $a \mid\sim t$, $a \mid\sim j$

Table 3.10: Base ranking for simple knowledge base shown in table 3.9

A quick calculation shows that $br(s) = 1$, however, it is not necessary to retract all of $R_0^{\mathcal{K}}$ since $a \mid\sim j$ is irrelevant to the query for both forms of relevant closure, hence it is added to the subset under consideration, i.e.

$$\mathcal{E}_s^{\mathcal{K}} \bigcup R^- = \{s \mid\sim \neg t, a \mid\sim m, s \rightarrow a, a \mid\sim j\}$$

And from there, we can test if $\overrightarrow{\mathcal{E}_s^{\mathcal{K}} \bigcup R^-} \models s \rightarrow m$ and in this situation, $\mathcal{K} \not\approx_{BRelC} s \mid\sim m$ and $\mathcal{K} \not\approx_{NRelC} s \mid\sim m$.

3.2.5 Basic and Rational Defeasible Entailment

Some work has been done by Casini et al. [15] in order to extend the KLM framework. Notably, they argue for additional properties that entailment relations should adhere to. These take two main forms, the first is basic defeasible entailment, which requires that relations be both LM-rational and adhere to 3 additional properties - inclusion, classic preservation and classic consistency. Formally:

1. Inclusion:$\mathcal{K} \mid\!\approx \alpha \mid\sim \beta$ for every $\alpha \mid\sim \beta \in \mathcal{K}$
2. Classic Preservation: $\mathcal{K} \mid\!\approx \alpha \mid\sim \bot$ iff $\mathcal{K} \mid\!\approx_P \alpha \mid\sim \bot$
3. Classic Consistency: $\mathcal{K} \mid\!\approx \top \mid\sim \bot$ iff $\mathcal{K} \mid\!\approx_P \top \mid\sim \bot$

The second form is rational defeasible entailment - which came about because it was argued the basic defeasible entailment can be too permissive. If a basic defeasible entailment relation satisfies RC extension, then it is classed as a rational defeasible entailment relation:

- RC Extension: if $\mathcal{K} \mid\!\approx_{RC} \alpha \mid\sim \beta$ then $\mathcal{K} \mid\!\approx \alpha \mid\sim \beta$

Rational closure forms the basis of rational defeasible entailment and therefore any other form can be expressed as a refinement of it.

3.2.6 Other Forms of Defeasible Entailment

In the literature there are a number of formalisms for defeasible entailment. However, as is the case with many fields in science, they are sometimes known by other names. The key point here with all of these it that we essentially have different mechanisms for drawing non-monotonic inferences. In this dissertation, we have covered rational and lexicographic closure, both of which adhere to the criteria to be classified as both basic and rational defeasible entailment relations. There is also ranked entailment [35] and relevant closure (covered herein as well), both of which are not LM-rational.

System W is another type of formalism [31], though not explicitly labelled as defeasible reasoning, it does qualify. The exact terms used in the literature differ from what we have used in here, but the core ideas are the same: a knowledge base is essentially a set of conditionals that take the form of "If A, then usually B". Such conditional knowledge bases can have different ranking functions, otherwise referred to as ordinal conditional functions (OCFs) [55, 56]. This function, κ, assigns an implausibility rank $\kappa(\iota)$ to each interpretation ι i.e. $\kappa : \Omega \mapsto \mathbb{N}_0 \bigcup \{\infty\}$ where Ω represents the set of all interpretations. The intuition here is that an interpretation with a higher rank is more unexpected or "surprising". There is also the normalisation condition which all of these ranking functions must satisfy - there must exist an interpretation such that $\kappa^{-1}(0) \neq \emptyset$ or more plainly, there must be a maximally plausible interpretation. Some noteworthy OCFs are system Z [47], which

computes the exact same results as rational closure, and c-representations [29, 30] - system W was developed with the aim of utilising the advantages of both of these.

Chapter 4

Defeasible Justifications

In the case of explanations for defeasible reasoning, there has been some introductory work, such as that undertaken by Wang [61], Chama [16] and Everett et al.[18], but it has certainly not been explored to the same depth as classical explanation. As we move through the different types of justifications, it should become evident to the reader that there is no singular underlying definition of what a defeasible justification is. This contrasts quite strongly with justifications in the classical one - where minimality was the only requirement. In the defeasible case, what a defeasible justification "looks like" is *entirely dependent* on the formalism that is under consideration. At present, there is no underlying semantics defined for what a general defeasible justification is.

It is worth detailing why precisely one cannot simply take the processes and definitions in the classical case and naïvely translate them to the defeasible case. The three examples which follow illustrate this predicament.

Scenario 1: Incorrect Intuition for Reasoning

Example 4.0.1. We can rank the statements from simple knowledge base (A) (i.e. table 4.1 into the ranking shown in table 4.2. If we were to use the naïve definition with rational closure to justify the entailment $s \mathrel{|\!\sim} f$, there would be two resulting justification sets:

1. $\mathcal{J}_1 = \{s \mathrel{|\!\sim} f\}$
2. $\mathcal{J}_2 = \{s \mathrel{|\!\sim} p, p \mathrel{|\!\sim} b, b \mathrel{|\!\sim} f\}$

Table 4.1: Simple defeasible knowledge base (A)

bird $\mid\sim$ flies
penguin $\mid\sim$ bird
penguin $\mid\sim \neg$ flies
special penguin $\mid\sim$ penguins
special penguin $\mid\sim$ flies

∞	
2	$s \mid\sim p, s \mid\sim f$
1	$p \mid\sim b, p \mid\sim \neg f$
0	$b \mid\sim f$

Table 4.2: Base Ranking for simple knowledge base (A)

A quick deliberation on the above would show that $\mathcal{J}_2$ is asserting that special penguins fly because they are typical birds and birds typically fly, thus essentially ignoring the explicit $s \mid\sim f$ statement in $\mathcal{K}$. This way of reasoning, whilst "logically" correct, does not follow the correct intuition - we want to conclude that special penguins fly because we have *explicitly stated* that they do, NOT because they are typical birds and birds typically fly. Additionally, it also does not adhere to our reasoning formalism, since $p \mid\sim b$ and $b \mid\sim f$ are retracted due to them disagreeing with the antecedent s of the query.

bird $\mid\sim$ flies
bird $\mid\sim$ wings
robin $\rightarrow$ bird
penguin $\rightarrow$ bird
penguin $\mid\sim \neg$ flies

Table 4.3: Simple defeasible knowledge base (B)

Scenario 2: Justification computation for queries that do not hold

Standard justifications cannot account for knowledge which is not part of the justification itself, but still a part of the knowledge base, which may prevent an inference being drawn from the justification. We show a simple example of this, using the knowledge base in table 4.3, as follows:

Example 4.0.2. First, consider the query $\mathtt{p} \mid\!\sim \mathtt{f}$. One possible justification for this query is: $\mathcal{J} = \{ \mathtt{b} \mid\!\sim \mathtt{f}, \mathtt{p} \rightarrow \mathtt{b}\}$. However, there is another statement within the KB that can make this justification invalid, that is: $\mathtt{p} \mid\!\sim \neg\mathtt{f}$. This entailment does not hold in any of the forms of defeasible entailment we consider in this dissertation.

Using the naïve approach, we are able to compute justifications for entailments which do not hold, which is not useful for our purposes.

Scenario 3: Domino effect of defeasible statement removal

Due to the defeasible nature of the information in defeasible knowledge bases, it is possible that the removal of one statement prevents an inference from being drawn, and the removal of more statements allows for the inference to be redrawn. We illustrate this predicament using the knowledge base and ranking from scenario 1, tables 4.1 and 4.2 and query $s \mid\!\sim f$.

With the above we can conclude $s \mid\!\sim f$. However, if we remove $s \mid\!\sim f$, then that entailment no longer holds. If we go one step further and remove $p \mid\!\sim \neg f$, we can once again conclude $s \mid\!\sim f$.

Given the above problems, it is evident that non-monotonicity adds several layers of complexity to defeasible explanations. Some work has been done which aims to address the difficulties presented by the above: weak and strong justifications.

In this chapter, we will cover the algorithms used for computing weak justifications for these cases of rational, lexicographic and relevant closure. Furthermore, we will examine the algorithm for strong justifications for rational closure. As we investigate more, it will become apparent that computing weak justifications is a far simpler process than computing strong justifications (for the case of defeasible reasoning).

4.1 Weak justifications

4.1.1 Rational Closure

Developed for description logic, Chama [16] modified the rational closure algorithm in order to propose a different notion of defeasible explanation, limited to the case of rational closure. The process is as follows: as with the original rational closure algorithm, one should first remove more general statements. We then materialise the remaining statements and attempt to compute classical justifications for the query (which has also been materialised). There is no assumption that a justification exists for a given query, and in cases where one cannot be computed, such as if the query is not entailed by the knowledge base, we simply return an empty set.

Henceforth, these will be labelled as *weak justifications* - a term coined by [18]. To notationally differentiate between the types of justifications and the formalisms under which they fall, we will use the superscript to show the defeasible justification system under consideration, namely rational (RC), lexicographic (LC) or relevant closure (RelC), and use the subscript to detail whether we are using strong or weak justifications. Thus, a weak justification under rational closure is notated as such: $\mathcal{J}_W^{RC}$. In much of the literature, a justification is annotated with a J [18, 61, 16], though the exact form may differ and we will follow that convention using $\mathcal{J}$. We also explicitly number justifications from 1 to n in cases where there is more than one justification for a given entailment, and hence (for example) the first weak justification we compute for a specific entailment under rational closure would be labelled as $\mathcal{J}_W^{RC}1$.

Algorithm

For KLM propositional logic, there are two definitions for weak justifications for rational closure, namely a procedural and more mathematical one [18].

Definition 4.1.1 (Procedural definition for RC weak justifications)**.** Given a defeasible knowledge base $\mathcal{K}$ and an entailment $\mathcal{K} \mid\!\approx_{RC} \alpha \mid\sim \beta$, we can state that $\mathcal{J}_W^{RC}$ is a weak justification for the entailment if it is an element of the set of justifications returned by the WeakJustifyRC algorithm.

Note that for the weak justification algorithms for rational AND relevant closure (introduced in 4.1.3), there is an underlying assumption that

`ComputeAllJustifications` (as described in Horridge [26]) is an algorithm that takes a classical knowledge base and entailment query $\alpha \in \mathcal{L}$ as input and outputs a set of the justifications for $\mathcal{K} \models \alpha$. If the entailment does not hold then it outputs $\emptyset$ instead.

Algorithm 5 WeakJustifyRC

Input: A defeasible knowledge base $\mathcal{K}$ and a query $\alpha \mathrel{|\!\sim} \beta$
Output: A set of weak justifications for $\mathcal{K} \mathrel{|\!\approx}_{RC} \alpha \mathrel{|\!\sim} \beta$

1: $(R_0, ..., R_{n-1}, R_\infty$,n):= BaseRank($\mathcal{K}$)
2: i := 0
3: $R := \bigcup_{i=0}^{i<n} R_i$
4: **while** $R_\infty \cup R \models \neg\alpha$ and $R \neq \emptyset$ **do**
5: $\quad R := R \setminus R_i$
6: $\quad i := i + 1$
7: **return** $\{\{\gamma \mathrel{|\!\sim} \delta \mid \gamma \rightarrow \delta \in \mathcal{J}\} \mid \mathcal{J} \in$ ComputeAllJustifications$(R \cup R_\infty, \alpha \rightarrow \beta)\}$

The alternative to the above is as follows:

Definition 4.1.2 (Weak Justification for Rational Closure)**.** Given a knowledge base $\mathcal{J}$ and an entailment $\mathcal{K} \mathrel{|\!\approx}_{RC} \alpha \mathrel{|\!\sim} \beta$, we can state that $\mathcal{J}$ is a weak justification for the entailment if $\overrightarrow{\mathcal{J}}$ is a classical justification for $\overrightarrow{\varepsilon_\alpha^\mathcal{K}} \models \alpha \rightarrow \beta$.

$\varepsilon_\alpha^\mathcal{K}$ is syntactic shorthand for $\varepsilon_\alpha^\mathcal{K} = \varepsilon_r^\mathcal{K}$ where $r = br_\mathcal{K}(\alpha)$.

Example 4.1.1. Consider knowledge base (A) (table 4.1) and its corresponding base ranking (table 4.2). For the query $s \mathrel{|\!\sim} f$ we then compute the exceptionality set for s: $\varepsilon_s^\mathcal{K} = \{s \mathrel{|\!\sim} p, s \mathrel{|\!\sim} f\}$.
This set is then materialised to produce $\overrightarrow{\varepsilon_s^\mathcal{K}} \models s \rightarrow f$, a set of classical justifications. The result for this example is a unique weak justification for the query.

From the above example, the connection between defeasible entailment for rational closure and weak justification should be evident. One caveat to take note of is that, since rational closure is inferentially weaker compared to other defeasible entailments (such as lexicographic closure or relevant closure), it might have a more limited utility. Though again, this may depend on the use case.

Properties of Weak Justifications

Everett et al. [18] provide an intuitive characterisation for weak justifications. First, they consider the KLM postulates and how defeasible justification applies to them, in the case of rational closure, lexicographic closure and relevant closure.

We present the results for rational closure here, and refer the reader to Everett et al.[18], who provide proofs and further reading for the other aforementioned formalisms.

We begin with the concept of a deciding knowledge base:

Definition 4.1.3 (Deciding Defeasible Knowledge Base)**.** For some defeasible knowledge base $\mathcal{D} \subseteq \mathcal{K}$ is deciding for an entailment $\mathcal{K} \mid\!\approx \alpha \mid\!\sim \beta$ if $\mathcal{D} \subseteq \mathcal{E}^{\mathcal{K}}_{\alpha}$ and $\overrightarrow{\mathcal{D}} \models \alpha \rightarrow \beta$

From the above and using definition 4.1.1, we can therefore state that for a given entailment, all weak justifications are deciding and that any deciding knowledge base is always a superset of a weak justification. Following from this, there are two properties:

Definition 4.1.4. Given some entailment $\mathcal{K} \mid\!\approx \alpha \rightarrow \ \beta$ and $\mathcal{D}$ (a deciding knowledge base for the entailment) and $br^{\mathcal{K}}(\alpha) \neq \infty$, then $\mathcal{D} \mid\!\approx \alpha \mid\!\sim \beta$

Definition 4.1.5. Given some entailment $\mathcal{K} \mid\!\approx \alpha \rightarrow \beta$ which has some $\mathcal{D}$ as its deciding knowledge base and $\mathcal{D} \mid\!\approx \alpha \mid\!\sim \beta$, then $\mathcal{J}_W(\mathcal{D}, \alpha \mid\!\sim \beta) \subseteq \mathcal{J}_W(\mathcal{K}, \alpha \mid\!\sim \beta)$, where $\mathcal{J}_W$ refers to a set of weak justifications

One consideration for the above is that it is only applicable for the case when $br^{\mathcal{K}}(\alpha) \neq \infty$ where α is the query's antecedent.

With the aforementioned in mind, one can now formalise these new postulates for rational closure "style" weak justifications.

Definition 4.1.6 (KLM Postulates for Weak Justification for Rational Closure)**.** For any knowledge bases $\mathcal{K}, \mathcal{J}^{RC}_{W}1, \mathcal{J}^{RC}_{W}2$:

- Left Logical Equivalence: if $\mathcal{J}^{RC}_{W}1 \in \mathcal{J}_W(\mathcal{K}, \alpha \leftrightarrow \beta)$ and $\mathcal{J}^{RC}_{W}2 \in \mathcal{J}_W(K, \alpha \mid\!\sim \gamma)$, then $\mathcal{J}^{RC}_{W}1 \cup \mathcal{J}^{RC}_{W}2$ is deciding for $\mathcal{K} \mid\!\approx \beta \mid\!\sim \gamma$
- Right Weakening: if $\mathcal{J}^{RC}_{W}1 \in \mathcal{J}_W(\mathcal{K}, \alpha \rightarrow \beta)$ and $\mathcal{J}^{RC}_{W}2 \in \mathcal{J}_W(K, \gamma \mid\!\sim \alpha)$, then $\mathcal{J}^{RC}_{W}1 \cup \mathcal{J}^{RC}_{W}2$ is deciding for $\mathcal{K} \mid\!\approx \gamma \mid\!\sim \beta$

- And: if $\mathcal{J}_W^{RC}1 \in \mathcal{J}_W(\mathcal{K}, \alpha \mid\sim \beta)$ and $\mathcal{J}_W^{RC}2 \in \mathcal{J}_W(K, \alpha \mid\sim \gamma)$, then $\mathcal{J}_W^{RC}1 \cup \mathcal{J}_W^{RC}2$ is deciding for $\mathcal{K} \mid\approx \alpha \mid\sim \beta \wedge \gamma$
- Or: if $\mathcal{J}_W^{RC}1 \in \mathcal{J}_W(\mathcal{K}, \alpha \mid\sim \gamma)$ and $\mathcal{J}_W^{RC}2 \in \mathcal{J}_W(K, \beta \mid\sim \gamma)$, then $\mathcal{J}_W^{RC}1 \cup \mathcal{J}_W^{RC}2$ is deciding for $\mathcal{K} \mid\approx \alpha \vee \beta \mid\sim \gamma$

For any knowledge base $\mathcal{K}$ and any propositional formula, say α:

- Reflexivity: $\mathcal{J}_W(\mathcal{K}, \alpha \mid\sim \alpha) = \{\emptyset\}$

For any knowledge base $\mathcal{K}$:

- Cautious Monotonicity: if $\mathcal{K} \mid\approx \alpha \mid\sim \gamma$ and $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ then every $\mathcal{J} \in \mathcal{J}_W(\mathcal{K}, \alpha \mid\sim \gamma)$ is deciding for $\mathcal{K} \mid\approx \alpha \wedge \beta \mid\sim \gamma$
- Rational Monotonicity: if $\mathcal{K} \mid\approx \alpha \mid\sim \gamma$ and $\mathcal{K} \not\mid\approx \alpha \mid\sim \neg\beta$ then every $\mathcal{J} \in \mathcal{J}_W(\mathcal{K}, \alpha \mid\sim \gamma)$ is deciding for $\mathcal{K} \mid\approx \alpha \wedge \beta \mid\sim \gamma$

These postulates present a strong case that weak justification is a weakening of classical justification in a way that strongly resembles how defeasible entailment is a weakening of classical entailment. The aim of them is to demonstrate the similarities between the classical justifications for propositional logic and weak justifications for defeasible propositional logic. They prove that weak justifications are sound enough for use as a formalism for justifications for specifically KLM-style defeasible entailment.

4.1.2 Lexicographic Closure

When using rational or lexicographic closure to determine if a defeasible statement is entailed by a defeasible knowledge base, we are basically doing classical entailment checking on all the leftover ranks once the algorithm performs the rank removal step (for rational closure) or statement removal step (lexicographic). Bearing this in mind, it therefore follows that classical justifications must exist for the (classical) entailment in each of the sub-knowledge bases. These are what will be used to compute the defeasible justifications.

In the previous section on lexicographic closure, the notion of a "combined formula approach" was used. However, in the context of weak justifications, the concepts of subsets and sub-knowledge bases are more suitable as it allows for precise pinpointing of the formulas needed for an entailment to hold and for working with formulas individually.

Algorithm

The algorithm for computing weak justifications for lexicographic closure, WeakJustificationsLexicographicClosure uses a sub-procedure (LexicographicClosureForJustifications), which is based on a modification of the original lexicographic closure algorithm, presented in the previous chapter. The major difference is that we take the sub-knowledge bases which are used for computing the final entailment (which have been derived as per the original LC algorithm) and *then* compute all classical justifications for the materialised query for *each* of these sub-knowledge bases.

For this algorithm, we introduce a new variable which represents the size of the subset used for the final entailment computation. The output for LexicographicClosureForJustifications will therefore be a combination of the result for the final entailment computation and an ordered tuple (i, m), with i denoting the lowest complete rank used for the computation. All of these elements combined will be used to reconstruct the final sub-knowledge bases.

Algorithm 6 LexicographicClosureForJustifications

Input: A knowledge base $\mathcal{K}$ and query $\alpha \mid\sim \beta$

Output: if $\mathcal{K} \not\approx_{LC} \alpha \mid\sim \beta$ return **true** and an ordered tuple (i,m) showing the rank and subset size, otherwise return **false**

1: $(R_0, ..., R_{n-1}, T_\infty, n) := BaseRank(\mathcal{K})$
2: i := 0;
3: m := 0;
4: $R := \bigcup_{j=0}^{j<n} R_j$;
5: **while** $R_\infty \cup R \models \neg\alpha$ **and** $R \neq \emptyset$ **do**
6: $\quad R := R \setminus R_i$;
7: $\quad m := |R_i| - 1$;
8: $\quad R_{i,m} := \bigvee_{X \in Subsets(R_i,m)} \bigwedge_{x \in X} x$;
9: $\quad$ **while** $R_\infty \cup R \cup \{R_{i,m}\}$ **and** $m > 0$ **do**
10: $\quad\quad m := m - 1$;
11: $\quad\quad R_{i,m} := \bigvee_{X \in Subsets(R_i,m)} \bigwedge_{x \in X} x$;
12: $\quad R := R \cup \{R_{i,m}\}$
13: $\quad i := i + 1$;
14: **return** $R_\infty \cup R \models \alpha \rightarrow \beta, (i, m)$

Example 4.1.2 (`LexicographicClosureForJustifications`)**.** Suppose you are given the knowledge base in table 4.4[1], the accompanying base ranking for it in table4.5 and a query, say $p \mid\sim fs$ (do penguins have feathers?):

`bird` $\mid\sim$ `feathers` ($b \mid\sim fs$)
`bird` $\mid\sim$ `four toes` ($b \mid\sim ft$)
`bird` $\mid\sim$ `digitigrade` ($b \mid\sim d$)
`penguin` $\mid\sim$ `is not both a digitigrade and has four toes` ($p \mid\sim \neg(d \wedge ft)$)
`penguin` $\rightarrow$ `bird` ($p \rightarrow b$)

Table 4.4: Simple defeasible knowledge base about bird morphology

∞	$p \rightarrow b$
1	$p \mid\sim \neg(d \wedge ft)$
0	$b \mid\sim fs$, $b \mid\sim ft$, $b \mid\sim d$

Table 4.5: Base ranking for simple knowledge base shown in table 4.4

We provide a step-wise explanation of what happens when the above knowledge base is input into the LexicographicClosureForJustifications algorithm:

1. Lines 1-5: Both i and m are initialised to 0 and $R = R_0 \cup R_1$. We then check if $R_\infty \cup R_0 \cup R_1 \models \neg p$, which is true.

2. Lines 6-8: $R = R \setminus R_0$, thus $R = R_1$ and $m = |R_0| - 1 = 3 - 1 = 2$. Following that, we calculate $R_{i,m}$ by considering all the ways to remove 1 formula from R_0 and then combine with the rest of the knowledge base. Therefore, $R_{i,m} = \{(b \mid\sim fs, b \mid\sim ft) \vee (b \mid\sim ft, b \mid\sim d) \vee (b \mid\sim fs, b \mid\sim d)\}$.

3. Lines 9-12: We perform another check - $R_\infty \cup R \cup \{R_{i,m}\}$ and $m > 0$. In other words, does $R_\infty \cup R_1 \cup R_{i,m} \models \neg p$? Since the answer is no, we skip lines 10 to 11.

[1] "Digitigrade" is used in the knowledge base and is used to describe when an animal walks on the toes rather than their entire foot.

4. Lines 13-16: $R = R_1 \cup \{R_{i,m}\}$ and $i = 1$. We then check again if $R_\infty \cup R_0 \cup R \models \neg p$. This then changes to false and the while loop exits. The algorithm then returns $R_\infty \cup R \models p \rightarrow w = \texttt{True}, (1, 2)$

It is vital to note that since we are working with sub-knowledge bases, it would not be sensible to present only one set of statements as the final defeasible justification. If the justifications derived from each knowledge base was simply combined, then we run the risk of losing information on the structure of sub-knowledge bases. This in turn could result in the algorithm only providing partial reasoning as to why the given final entailment holds. If one justification is only present in one of the sub-knowledge bases, then there is a chance this justification could be lost when the various sets of justifications are computed and combined.

Therefore, it is imperative that the algorithm provides a structure that contains a "memory" which can be used to show exactly which statement in a specific sub-knowledge base allows an entailment to hold. This is achieved by returning a tuple of justifications where element i in the tuple is a justification from sub-knowledge base i for the entailment.

When $m = 0$, then there is a single knowledge base and hence we will only compute a set of 1-tuples. In other words, we are basically computing a set of single justifications from the remaining ranks - which is the same set that the rational closure weak justification algorithm would have computed.

Now, we can present the entire algorithm for computing weak justifications for lexicographic closure: WeakJustificationsLexicographicClosure.

The caveat for this approach is that it is limited to the case where the justifications are in the subset rank level. Otherwise, should they all be located in higher ranks, this may lead to the tuple having the same justification for each element. The implication of this is that we will have the same description level as that of a single set. In order to mitigate this pitfall, we first use the combined formula approach together with the rational closure weak justification finding algorithm to determine if the formulas in the subset rank form part of the final entailment, and then use the WeakJustificationsLexicographicClosure algorithm.

Worked Example

Example 4.1.3. Using once again the knowledge base in table 4.4, its accompanying ranking and the same query ($p \mid\sim fs$).

Algorithm 7 WeakJustificationsLexicographicClosure

Input: A knowledge base $\mathcal{K}$ and query $\alpha \mid\sim \beta$

Output: Set of all justifications for $\mathcal{K} \mid\approx_{LC} \alpha \mid\sim \beta$

1: $(R_0, ..., R_{n-1}, T_\infty, n) := BaseRank(\mathcal{K})$
2: (i,m) := $LexicographicClosureForJustifications(\mathcal{K}, \alpha \mid\sim \beta)$
3: $R := \bigcup_{j=0}^{j<n} R_j$;
4: $\mathcal{J} := \emptyset$
5: **if** $m > 0$ **then**
6: subsets := $Subsets(R_{i-1}, m)$;
7: $x := 0$
8: **for** S in *subsets* **do**
9: $\mathcal{F} := S \bigcup R \bigcup R_\infty$
10: $\mathcal{J}_{Wi} := ComputeAllJustifications(\mathcal{F}, \alpha \mid\sim \beta)$
11: $x := x + 1$
12: $\mathcal{J} := \mathcal{J}_0 \times \mathcal{J}_1 \times ... \times \mathcal{J}_{x-1}$
13: **else**
14: $\mathcal{J}_W := ComputeAllJustifications(R \cup R_\infty)$
15: **return** $\mathcal{J}$

0	$b \mid\sim d$, $b \mid\sim ft$
1	$p \mid\sim \neg(d \wedge ft)$
∞	$p \rightarrow b$

0	$b \mid\sim d$, $b \mid\sim fs$
1	$p \mid\sim \neg(d \wedge ft)$
∞	$p \rightarrow b$

0	$b \mid\sim ft$, $b \mid\sim fs$
1	$p \mid\sim \neg(d \wedge ft)$
∞	$p \rightarrow b$

Table 4.6: Sub-knowledge bases for example 3.2.2

LexicographicClosureForJustifications yields the a set of sub-knowledge bases, shown in table 4.6.

We then compute the justifications for the entailment in each of the sub-knowledge bases:

1. $\mathcal{J}_W^{LC}0 = \{\{b \mid\sim d, p \mid\sim \neg(d \wedge ft), b \mid\sim ft, p \rightarrow b\}\}$
2. $\mathcal{J}_W^{LC}1 = \{\{p \rightarrow b, b \mid\sim fs\}\}$
3. $\mathcal{J}_W^{LC}2 = \{\{p \rightarrow b, b \mid\sim fs\}\}$

We then take the Cartesian product of all these sets, which produces a single defeasible justification:

$J_W^{LC} = (\{b \mid\sim d, p \mid\sim \neg(d \wedge ft), b \mid\sim ft, p \rightarrow b\}, \{p \rightarrow b, b \mid\sim fs\}, \{p \rightarrow b, b \mid\sim fs\})$

The above example illustrates a rather important observation - for some of the sub-knowledge bases constructed, it might be possible to conclude $\neg\alpha$ in some of them. Namely, $\mathcal{J}_W^{LC}0$ in the example entails $\neg p$. Consequently, some of these disproving justifications could be included in the justifications for the final entailment. This leads to us no longer having the ability to reason about α. However, it does provide information on the structure of the sub-knowledge base.

4.1.3 Relevant Closure

The notion of weak justification for rational closure has also been extended to relevant closure[18]. The relevance partition is still maintained, with the part not relevant to the query added to the materialised exceptionality sequence in order to compute the classical justification (similar to what is done for rational closure). First, we provide a formal definition of a weak justification for the case of relevant closure:

Definition 4.1.7 (Weak justifications for relevant closure)**.** Given some knowledge base $\mathcal{J}$ and query $\alpha \mid\sim \beta$, we state that $\mathcal{J}$ is a weak justification for (both forms of) a relevant closure entailment $\mathcal{K}_{BRelC}\alpha \mid\sim \beta$ or $\mathcal{K}_{MBRelC}\alpha \mid\sim \beta$ if $\overrightarrow{J}$ is a classical justification for

$$\overrightarrow{\mathcal{E}_\alpha^{\mathcal{K}} \bigcup R^-} \models \alpha \rightarrow \beta$$

(R, R^-) is a relevance partition of $\mathcal{K}$ for $\alpha \mid\sim \beta$ and is constructed using whatever form of relevance is under consideration.

Algorithm

The algorithm for computing weak justifications for relevant closure is a combination of `RelevantClosure` and the `RationalClosure`, with some modifications. Proofs of soundness and completeness for the algorithm can once again be found in Everett et al. [18].

Algorithm 8 WeakJustifyRelevantClosure

Input: A knowledge base $\mathcal{K}$, a query $\alpha \mid\sim \beta$ and M, which is the value of $R^{-}(\mathcal{K}, \alpha)$

Output: The set of weak justifications for $\mathcal{K} \mid\approx_{R} elC\alpha \mid\sim \beta$ for the form of relevance closure under consideration

1: $(R_0, ..., R_{n-1}, R_{\infty}$,n):= BaseRank($\mathcal{K}$)
2: i := 0
3: $R := \bigcup_{i=0}^{j<n} R_J$
4: **while** $R_{\infty} \cup R \models \neg\alpha$ and $R \neq \emptyset$ **do**
5: $\quad R := R \setminus R_i$
6: $\quad i := i + 1$
7: **return** $\{\{\gamma \mid\sim \delta \mid \gamma \rightarrow \delta \in \mathcal{J}\} \mid \mathcal{J} \in$ ComputeAllJustifications$(R \cup M, \alpha \rightarrow \beta)\}$

Worked Example

Example 4.1.4. Given the knowledge base in table 4.7 and the accompanying table of statement rankings produced when the `BaseRank` algorithm is applied, consider the query: $pgu \mid\sim w$?

Here, we will try and compute some weak justifications for the query using the minimal relevance formalism.

Since $br_{\mathcal{K}}(pgu) = 1$, it thus follows that:

$$\mathcal{E}^{\mathcal{K}}_{pgu} = \{p \mid\sim \neg f, p \rightarrow b, pg \rightarrow p\}$$

As with relevant closure, we do not retract all of the statements in rank 0 - we retain $b \mid\sim w$ since it does not conflict with any of the more specific statements in rank 1.

Therefore, the knowledge base where we attempt to derive some justifications looks like the following:

$$\mathcal{E}^{\mathcal{K}}_{pgu} \bigcup R^{-} = \{p \mid\sim f, p \rightarrow b, pg \rightarrow p\, b \mid\sim w\}$$

Applying definition 4.1.7 mentioned earlier enables to compute a singular unique weak justification for the query $\mathcal{K} \mid\approx_{MRelC} pgu \mid\sim w$:

$$\mathcal{J}^{RelC}_{W1} = \{pgu \rightarrow p, p \rightarrow b, b \mid\sim w\}$$

bird $\mid\sim$ flies ($b \rightarrow f$)
penguin $\mid\sim$ bird ($p \mid\sim b$)
penguin $\mid\sim \neg$ flies ($p \mid\sim \neg f$)
pingu $\rightarrow$ penguin ($pgu \rightarrow p$)
bird $\mid\sim$ wings ($b \mid\sim w$)

Table 4.7: Simple defeasible knowledge base about types of penguins

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

Table 4.8: Base ranking for knowledge base in table 4.7

The above example demonstrates one of the most important mechanisms of computing weak justifications for relevant closure - formulae discovered to be irrelevant to the query remain part of the reduced knowledge base when it is materialised, in a similar manner to when entailment is tested for relevant closure.

4.2 Strong Justifications

Developed entirely divorced from the notion of weak justifications, Brewka and Ulbricht [11] proposed the notion of strong explanations - a different approach to defeasible explanation. Their solution is based upon earlier work which was concerned with inconsistency, namely *strong inconsistency* [10] and is achieved by strengthening the standard concept of inconsistency for non-monotic logics by adding a constraint on the classical explanation: information from within the knowledge base, but outside of the justification(s), cannot render the entailment of the justification false. Given that justifications are one type of expression of an explanation [26], we refer to justifications which have this criterion of strong inconsistency as *strong justifications*.

In their paper, Brewka and Ulbricht considered this concept in the context of logic programs and answer set semantics, however, their results are highly generalisable and can trivially be translated to the KLM defeasible propositional logic context:

bird $\mid\sim$ flies
ostrich $\mid\sim$ bird
ostrich $\mid\sim \neg$ flies
special ostrich $\mid\sim$ bird
special ostrich $\mid\sim$ flies

Table 4.9: Simple knowledge base about ostriches

Definition 4.2.1 (Strong Justification)**.** A knowledge base $\mathcal{J}_S$ is considered to be strong justification for some entailment, say $\mathcal{K} \mid\approx \alpha \mid\sim \beta$ if $\mathcal{J}_S$ is a minimal subset of $\mathcal{K}$, $\mathcal{J}_S \subseteq \mathcal{K}$ and these two conditions hold:

1. $\mathcal{J}_S \mid\approx \alpha \mid\sim \beta$
2. There is no superset $\mathcal{J}'_S \supset \mathcal{J}_S$ where $\mathcal{J}'_S \subseteq \mathcal{K}$ such that $\mathcal{J}'_S \not\mid\approx \alpha \mid\sim \beta$

A subset is minimal if it has no proper subset that satisfies these conditions.

Note that for the above there is no restriction to any particular defeasible entailment formalism. The logic agnostic nature of this approach - in the sense that it can be used for both monotonic and non-monotonic logics - is perhaps its greatest advantage.

It may be tempting to try to assume that weak and strong justifications inherently relate to each other, given that fact that in the final step of their computation, weak justifications use classical justification techniques and strong justifications are merely an extension to classical justifications. Therefore, one might presuppose that both strong and weak justifications refer to the same fundamental concept, or that if a set of statements constitutes a weak justification, that it also might be considered a strong justification.

However, *there is no inherent relation* between them. We will use an example to illustrate this:

Example 4.2.1. Consider knowledge base 4.9 with accompanying query s $\mid\sim$ f. In this context, there are two justifications for the query (these constitute the set of weak justifications) notated as $\mathcal{J}_W$:

1. $\mathcal{J}_W 1 = \{\text{s} \mid\sim \text{f}\}$
2. $\mathcal{J}_W 2 = \{\text{s} \mid\sim \text{o}, \text{o} \mid\sim \text{b}, \text{b} \mid\sim \text{f}\}$

Unfortunately, $\mathcal{J}_W 2$ does not qualify as a strong justification since adding $\{\text{o} \mid\sim \neg\text{f}\}$ causes the entailment to fail. Conversely, $\mathcal{J}_W 1$ *does* fit the criteria for a strong justification. Though we have not yet introduced an algorithm for computing strong justifications, this example is trivial enough that one can manually calculate and notice that there is no superset of $\mathcal{J}_W 1$ i.e. $\mathcal{J}'$ such that $\mathcal{J}' \not\approx_{RC} s \mid\sim f$. This example demonstrates that not every weak justification automatically qualifies as a strong justification.

In the example above, the issue that emerges is that information can be added to $\mathcal{J}_i$ (where $i \geq 0$) that allows for the antecedent (say α) of the entailment statement to be disproved - in this case, o - whilst also not disproving all the other antecedents of statements within $\mathcal{J}_W i$. The result of this is that certain statements within $\mathcal{J}_W i$ will get "mixed into" lower ranks, where they can still disprove the antecedent, α. These statements are then removed during the algorithm for computing weak justification, and this removal results in the final entailment computation returning `False`.

4.2.1 Rational Closure

Despite the above, strong justifications can still be defined as an extension of weak justifications, whilst also maintaining the minimality constraint required for justifications. Limited to the case of rational closure, this involves taking a weak justification, $\mathcal{J}_W^{RC} i$, and extending it to create a set $\mathcal{J}_S^{RC} i$, such that regardless of what statements are added to the set from the original knowledge base, a partition is preserved between the statements - on one side, there is a set of ranked statements that can disprove the antecedent. On the other side there is a set of ranked statements where the antecedent cannot be disproved, and the weak justification falls into these latter ranks.

This idea can be encapsulated by means of a diagram, shown in figure 4.1

The algorithm for computing strong justifications for rational closure consists of multiple sub-process. First, we must partition the rankings of the original knowledge base into those that contain formulas which entail $\neg\alpha$ and those that do not. This is accomplished by computing all the justifications for $\neg\alpha$ in $\mathcal{K}$, denoted as $\mathcal{J}_{\neg\alpha}$.

Thus, let $\mathcal{K}'$ be a subset of $\mathcal{K}$ and let the rank of $\mathcal{K}'$ rank that of the lowest ranked formula within $\mathcal{K}'$ relative to the ranking of $\mathcal{K}$. This allows us to classify the ranking of a justification according to the minimum rank which would have to be removed to ensure that a justification does not hold

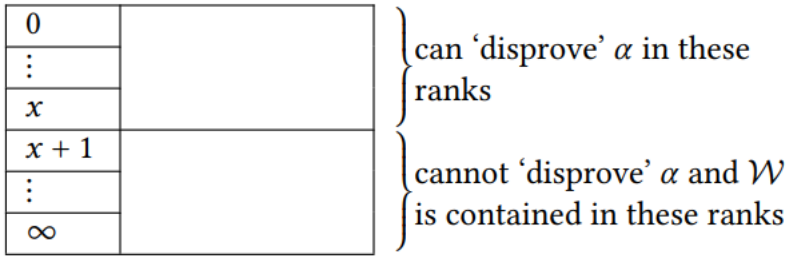


Figure 4.1: Visual depiction of the partition needed for creating strong justifications for rational closure. $\mathcal{W}$ refers to a weak justification

in $\mathcal{K}$. To demonstrate: for example, if we have the ranking shown in table 4.8 and let $\mathcal{K}' = \{b \mid\sim f, p \mid\sim \neg f\}$, then the ranking of $\mathcal{K}' = 0$ since the rank of $\{b \mid\sim f\} = 0$

This algorithm is constrained to the case of only deriving strong justifications where all justifications have a finite rank i.e. $\mathcal{J}_{\neg\alpha}$ and $\mathcal{J}_{\mathcal{W}}$ have $R_i \neq R_\infty$.

In essence, the above constraint allows us to construct sequence of subsets, notated with $(K_0, K_1, ..., K_x)$, where:

1. $br_{K_i}(\gamma) = i$
2. For all $K' \subset K_i, br_{K'}(\gamma) < br_{K_i}(\gamma)$

These sequences are constructed for $\mathcal{J}_W i$, for all $\mathcal{J}_{\neg\alpha}$ with $\gamma = \alpha$ for $\mathcal{J}_{\neg\alpha}$ and $\gamma = \alpha \wedge \neg\beta$ for $\mathcal{J}_W i$. These are derived by selecting either $\mathcal{J}_W i$ or $\mathcal{J}_{\neg\alpha}$ to be the initial subset (whichever is more suitable). Each successive set, K_i will contain K_{i-1} i.e. the preceding set, be minimal and adhere to the two criteria mentioned above. Furthermore, each K_i will also disprove all the antecedents of the formulas in K_{i-1}.

In a more abstract sense, these sequences show how to minimally move the $\mathcal{J}_{\neg\alpha}$ and $\mathcal{J}_W i$ up the rank, with each subsequent step allowing the justifications to move up by one rank. Whilst we must consider all the variations of raising $\mathcal{J}_{\neg\alpha}$ up through the ranks, for the case of $\mathcal{J}_W i$, we need only be concerned with the variations in which $\mathcal{J}_W i$ has a higher rank than that of the highest ranked $\mathcal{J}_{\neg\alpha}$ justification in $\mathcal{K}$ for the final set in the successive K sequences. Let us present a modified example from [18] to demonstrate why this requirement is necessary:

Table 4.10: Simple defeasible knowledge base about penguins

birds $\rightarrow$ animal ($b \rightarrow a$)
chicken $\rightarrow$ bird ($c \rightarrow b$)
wild chicken $\rightarrow$ chicken ($wc \rightarrow c$)
animals $\mid\sim \neg$ wings ($a \mid\sim \neg$ w)
digitgrade $\mid\sim$ bird ($d \mid\sim b$)
digitgrade $\mid\sim \neg$flies($d \mid\sim \neg f$)
bird $\mid\sim$ wings ($b \mid\sim w$)
bird $\mid\sim$ flies ($b \mid\sim f$)
bird $\mid\sim$ hollow bones ($b \mid\sim h$)
chicken $\mid\sim \neg$ flies ($c \mid\sim \neg f$)
wild chicken $\mid\sim$ flies ($\mid\sim$)

∞	$b \rightarrow a$, $wc \rightarrow c$, $c \rightarrow b$
3	$wc \mid\sim f$
2	$c \mid\sim \neg f$, $d \rightarrow b$, $d \mid\sim \neg f$
1	$b \mid\sim f$, $b \mid\sim w$, $b \mid\sim h$
0	$a \mid\sim \neg w$

Table 4.11: Base ranking for knowledge base in table 4.10

Example 4.2.2. Consider the knowledge base in table 4.10 and the accompanying table of statement rankings produced when the `BaseRank` algorithm is applied.

For a simple query such as $wc \wedge d \mid\sim f$, we can calculate a weak justification i.e. $\mathcal{J}_W^{RC} = \{wc \mid\sim f\}$. We compute $\mathcal{J}_{\neg wc} = \{b \mid\sim f, d \mid\sim \neg f\}$. There are two possible ways to "push" these sets up through the rankings, as seen in table 4.12.

It is not possible to add statements from the ranking of table (a) to the weak justification $\mathcal{J}_W^{RC}$ to enable us to "push" $\mathcal{J}_W^{RC}$ above $\mathcal{J}_{\neg\alpha}$ in the other ranking (b). This is because the formulas in table (a) constrain the ranking of $\mathcal{J}_W^{RC}$ to a maximum of 1, which has the same ranking as $\mathcal{J}_{\neg\alpha}$.

Therefore to ensure we are only considering the correct ways of "pushing" the weak justification up the ranks, we use the idea of $\mathcal{E}$ subsets from rational closure. We select only statements from $\mathcal{E}_i$, which guarantees that

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

(a) Pushing up $\mathcal{J}_W^{RC}$

∞	$b \rightarrow a$
2	$d \rightarrow b$, $d \mid\sim \neg f$
1	$b \mid\sim f$, $b \mid\sim w$
0	$a \mid\sim \neg w$

(b) Pushing up $\mathcal{J}_{\neg wc}$

Table 4.12: Different ways of pushing justifications up the rankings

the formulae will be brought up to at least rank i. For the set $\mathcal{E}_n$, n is the rank of the highest $\mathcal{J}_{\neg\alpha}$ justification in $\mathcal{K}$. Initially, we will thus choose formula which simultaneously $\models \neg\alpha$ in $\mathcal{E}_n$ from $\mathcal{E}_n$. From there, we then work backwards through these $\mathcal{E}$ sets until we obtain a sequence where the rank of $\mathcal{J}_W$ in the final set is greater than n. As we move through each $\mathcal{E}$ subset, we are selecting the minimum number of formulae from $\mathcal{E}_i$ which can disprove the formulae added from $\mathcal{E}_i$ until we've constructed a sequence where the the rank of $\mathcal{J}_W$ in the last set of the sequence is greater than n. This provides us with the minimum number of statements required to push $\mathcal{J}_W$ above any $\mathcal{J}_{\neg\alpha}$ in $\mathcal{K}$.

Algorithm

This process has been formalised in the `ComputeSubsetSequences` algorithm, which uses `Sequences` as a sub-process, accompanying proofs done by Morris in [18].

Note that `ComputeSubsetSequences` will return at least one sequence of sets if and only if the base rank of the formula are not equal to ∞.

In example 4.2.3, we demonstrate how `ComputeSubsetSequences` calculates a sequence for a given weak justification:

Example 4.2.3. Consider the ranking of statements in table 4.13 and entailment $(sp \wedge 4t) \mid\sim f$ the weak justification $\mathcal{J}_W^{RC} = \{sp \mid\sim f\}$. The highest rank for $\mathcal{J}_{\neg\alpha}$ where $\alpha = (sp \wedge 4t)$ is 2. Thus, the weak justification must have a rank of 3. The following K subsets are computed: $K_0 = \emptyset$

Algorithm 9 ComputeSubsetSequences

Input: A knowledge base $\mathcal{K}$, a formula α, the rank $n \leq br_{\mathcal{K}}(\alpha)$ needed for α in the final subset of a sequence

Output: Set of all sequences $(K_1, K_2, ..., K_n)$ where α has $br_{\mathcal{K}}(\alpha) = i$, and each K_i is minimal

$i := 0$;
$\mathcal{E}_0 = \overrightarrow{\mathcal{K}}$;
while $\mathcal{E}_{i-1} \neq \mathcal{E}_i$ **do**
    $\mathcal{E}_{i+1} := \{\alpha \rightarrow \beta \in \mathcal{E}_i \mid \mathcal{E}_i \models \neg\alpha\}$;
    $i := i + 1$;
$\mathcal{E}_\infty := \mathcal{E}_{i-1}$;
sequences := Sequences$((\mathcal{E}_0, ..., \mathcal{E}_\infty), \alpha, (\emptyset), n, 1)$;
return sequences;

Algorithm 10 Sequences

Input: A knowledge base $\mathcal{K} = (\mathcal{E}_0, \mathcal{E}_1, ...\mathcal{E}_\infty)$, a formula α, the current sequence of subsets $(K_0, ..., K_{i-1})$

Output: A set of sequences of length n

if $i > n$ **then**
    return $\{(K_0, ..., K_{i-1})\}$;
$\mathcal{A} := \neg\alpha \wedge (\bigwedge_{\beta \in antecedents(K_{i-1})} \neg\beta)$;
$\mathcal{S}$:= MinimalExtension$(\mathcal{A}, \mathcal{E}_{n-1}, K_{i-1})$;
if $\mathcal{S} \neq \emptyset$ **then**
    for K_i in $\mathcal{S}$ **do**
        if $br_{K_i} = i$ and Minimise(K_i, α) is **False then**
            $\mathcal{F} = \mathcal{F} \bigcup$ Sequences$(\mathcal{K}, \alpha, (K_0, ..., K_{i-1}, K_i), n, i + 1)$;
return $\mathcal{F}$;

∞	$p \rightarrow b, sp \rightarrow p, b \rightarrow a$
3	$sp \mid\sim f$
2	$p \mid\sim \neg f, 4t \mid\sim \neg f, 4t \mid\sim b, 4t \mid\sim \neg j$
1	$b \mid\sim w, b \mid\sim f, b \mid\sim j$
0	$a \mid\sim w$

Table 4.13: Base ranking for a set of statements

$K_1 = \{sp \mid\sim f\}$
$K_2 = K_1 \cup \{p \mid\sim \neg f, sp \rightarrow p\}$
$K_3 = K_2 \cup \{b \mid\sim f, p \rightarrow b\}$
$K_4 = K_4 \cup \{b \mid\sim w, a \mid\sim \neg w, b \rightarrow a\}$

As for `Sequences`, the `MinimalExtension`(α, A, B) sub-process determines what the minimum number of formulas are that would allow for $B \mid\sim \alpha$ to hold, using formulas from set A which are then added into set B (to ensure the entailment is true). `Minimise`(A, α) tests to see if statements can be dropped from A whilst still maintaining α's ranking.

The above two algorithms are used only for computing the minimal ways of pushing $\mathcal{J}_W^{RC}$ up the rankings and hence we need algorithms for doing the same for the justifications for $\neg\alpha$. A similar process is used for deriving the ways of pushing up $\mathcal{J}_{\neg\alpha}$, sans the need for computing $\mathcal{E}$ subsets since there is no longer the constraint of justifications needing to be placed above a particular rank. Therefore, we can use the materialised version of the knowledge base, $\overrightarrow{\mathcal{K}}$. As for the sub-process algorithm (analogously `Sequences` for the weak justification), we now return the sequence where we can no longer minimally "disprove" the antecedents in a manner that adheres to the criterion of $\mathcal{F}$ being empty. These above processes are described in ComputeGeneralSubsetSequences and GeneralSequences.

Algorithm 11 ComputeGeneralSubsetSequences

Input: A knowledge base $\mathcal{K}$ and formula α
Output: Set of all sequences $(K_1, K_2, ..., K_n)$ where α has $br_{\mathcal{K}}(\alpha) = i$, and each K_i is minimal

1: sequences := $GeneralSequences(\overrightarrow{\mathcal{K}}, \alpha, (\emptyset), 1)$;
2: **return** sequences;

We now provide an example of how these two algorithms work:

Example 4.2.4. Using the same knowledge base and query from example 4.2.3, we can use ComputeGeneralSubsetSequences and GeneralSequences in order to calculate all the justifications for $\mathcal{J}_{\neg\alpha}$ where $\alpha = (sp \wedge 4t)$.

The aforementioned knowledge base and α are provided as input to `ComputeGeneralSubsetSequences`. The knowledge base $\mathcal{K}$ is materialised into $\overrightarrow{\mathcal{K}}$ and is fed into `GeneralSequences`, along with α, the empty set and 1, which is the index of the current subset.

Algorithm 12 GeneralSequences

Input: A materialised knowledge base, $\overrightarrow{\mathcal{K}}$, a formula α, the current sequence of subsets $(K_0, ..., K_{i-1})$, the index i of the current subset
Output: A sequence of varying lengths

1: $\mathcal{A} := \neg\alpha \wedge (\bigwedge_{\beta \in antecedents(K_{i-1})} \neg\beta)$;
2: $\mathcal{S} :=$ MinimalExtension$(\mathcal{A}, \overrightarrow{\mathcal{K}}, K_{i-1})$;
3: $\mathbf{F} := \emptyset$;
4: **if** $\mathcal{S} \neq \emptyset$ **then**
5: **for** K_i in $\mathcal{S}$ **do**
6: **if** $br_{K_i} = i$ and Minimise(K_i, α) is **False then**
7: $\mathcal{F} = \mathcal{F} \bigcup$ GeneralSequences$(\overrightarrow{\mathcal{K}}, \alpha, (K_0, ..., K_{i-1}, K_i), i + 1)$;
8: **if** $\mathcal{F} = \emptyset$ **then**
9: $\mathcal{F} = \{\{(K_0, ..., K_{i-1})\}\}$;
10: **return** $\mathcal{F}$;

In total, 9 justifications for $\neg\alpha$ are computed, with accompanying K subsets. These are:

1. $\mathcal{J}_1 = \{a \mid\sim \neg w, b \mid\sim w, 4t \mid\sim b, b \rightarrow a\}$
   (a) Sequences: $K_0^{J_1} = \emptyset, K_1^{J_1} = \mathcal{J}_1$
2. $\mathcal{J}_2 = \{b \mid\sim f, 4t \mid\sim \neg f, 4t \mid\sim b\}$
   (a) Sequences: $K_0^{J_2} = \emptyset, K_1^{J_2} = \mathcal{J}_2, K_2^{J_2} = K_1^{J_2} \cup \{b \mid\sim w, b \rightarrow a, a \mid\sim \neg w\}$
3. $\mathcal{J}_3 = \{b \mid\sim f, 4t \mid\sim \neg j, 4t \mid\sim b\}$
   (a) Sequences: $K_0^{J_3} = \emptyset, K_1^{J_3} = \mathcal{J}_3, K_2^{J_3} = K_1^{J_3} \cup \{b \mid\sim w, b \rightarrow a, a \mid\sim \neg w\}$
4. $\mathcal{J}_4 = \{sp \mid\sim f, p \mid\sim \neg f, sp \mid\sim p\}$
   (a) Sequences: $K_0^{J_4} = \emptyset, K_1^{J_4} = \mathcal{J}_4, K_2^{J_4} = K_1^{J_4} \cup \{b \mid\sim f, p \rightarrow b\}, K_3^{J_4} = K_2^{J_4} \cup \{b \mid\sim w, b \rightarrow a, a \mid\sim \neg w\}$
5. $\mathcal{J}_5 = \{b \mid\sim f, p \mid\sim \neg f, p \rightarrow b, sp \rightarrow p\}$

(a) Sequences: $K_0^{J_5} = \emptyset, K_1^{J_5} = \mathcal{J}_5, K_2^{J_5} = K_1^{J_5} \cup \{b \mid\sim w, b \rightarrow a, a \mid\sim \neg w\}$

6. $\mathcal{J}_6 = \{a \mid\sim \neg w, b \mid\sim w, p \rightarrow b, sp \rightarrow p, b \rightarrow a\}$

   (a) Sequences: $K_0^{J_6} = \emptyset, K_1^{J_6} = \mathcal{J}_6$

7. $\mathcal{J}_7 = \{4t \mid\sim \neg f, sp \mid\sim f\}$

   (a) Sequences: $K_0^{J_7} = \emptyset, K_1^{J_7} = \mathcal{J}_7, K_2^{J_7} = K_1^{J_7} \cup \{b \mid\sim f, 4t \mid\sim b, p \mid\sim \neg f, sp \rightarrow p\}, K_3^{J_7} = K_2^{J_7} \cup \{p \rightarrow b, b \mid\sim w, a \mid\sim \neg w, b \rightarrow a\}$

   (b) Sequences: $K_0^{J_7} = \emptyset, K_1^{J_7} = \mathcal{J}_7, K_2^{J_7} = K_1^{J_7} \cup \{b \mid\sim j, 4t \mid\sim b, 4t \mid\sim \neg j, sp \rightarrow p\}, K_3^{J_7} = K_2^{J_7} \cup \{p \rightarrow b, b \mid\sim w, a \mid\sim \neg w, b \rightarrow a\}$

8. $\mathcal{J}_8 = \{b \mid\sim f, 4t \mid\sim \neg f, sp \rightarrow p, p \rightarrow b\}$

   (a) Sequences: $K_0^{J_8} = \emptyset, K_1^{J_8} = \mathcal{J}_8, K_2^{J_8} = K_1^{J_8} \cup \{b \mid\sim w, b \rightarrow a, a \mid\sim \neg w\}$

9. $\mathcal{J}_9 = \{b \mid\sim j, 4t \mid\sim \neg j, sp \rightarrow p, p \rightarrow b\}$

   (a) Sequences: $K_0^{J_9} = \emptyset, K_1^{J_9} = \mathcal{J}_9, K_2^{J_9} = K_1^{J_9} \cup \{b \mid\sim w, b \rightarrow a, a \mid\sim \neg w\}$

The ability to construct these sequences allows us to now establish how weak justifications can be extended to form strong justifications. First, we must choose a sequence from a weak justification - this choice is very important, as we will use the subsets in the sequence in such a way that whenever a $\mathcal{J}_{\neg\alpha}$ justification is driven up the ranks in some subset of the knowledge base, so too will our weak justification. To achieve this, we must start by individually examining each sequence, $\mathcal{J}_{SEQ}$ for each $\mathcal{J}_{\neg\alpha}$ justification and inspect all the ways of adding the minimum number of formulae from the weak justification "pushing up" sequence where $\mathcal{J}_W^{RC}$ gets "pushed above" $\mathcal{J}_{\neg\alpha}$ in each subset in $\mathcal{J}_{SEQ}$.

We start with a set $\mathcal{S}$ which contains $\mathcal{J}_W^{RC}$, iterating through each $\mathcal{K}_i^{\mathcal{J}_{SEQ}}$ subset in $\mathcal{J}_{SEQ}$ and performing a check to determine if the antecedents of the formulae in $\mathcal{S}$ can be disproved using $\mathcal{S} \cup \mathcal{J}_{SEQ}$. If the check returns false, then we take $\mathcal{K}_{i+1}^{\mathcal{J}_W^{RC}}$ and enumerate through all possible ways of adding the minimum required number of statements to $\mathcal{S}$ such that $\mathcal{S} \cup \mathcal{K}_i^{\mathcal{J}_{SEQ}}$ "disproves" all of the antecedents of formulae previously in $\mathcal{S}$. This process is

repeated until all sets in the sequence have been constructed. The end result of this is that we have essentially derived all minimal sets of $\mathcal{S}$ that guarantee that if $\mathcal{J}_{\neg\alpha}$ has a rank of x in the sequence, than $\mathcal{J}_W^{RC}$ will have a rank of at least $x+1$. This method is shown in StrongSequences. Note however, that there are presently no soundness or completeness proofs for this algorithm.

Algorithm 13 StrongSequences

Input: The formula α, a sequence $\mathcal{N} = (S_0, S_1, ..., S_n)$ for α, a formula β, a sequence $\mathcal{M} = (J_0, J_1, ..., J_m)$ for β where $m < n$, the index of the current subset i, the current minimal entailing set $\mathcal{S}$

Output: A set of sets $\mathcal{S}$ such that $br_{\mathcal{S}\cup J_i}(\alpha) > br_{\mathcal{S}\cup J_i}(\beta)$ for all $i \leq m$ and $\mathcal{S}$ is minimal

1: **if** $i > m$ **then**
2: **return** $\{\mathcal{S}\}$;
3: $\mathcal{A} := \neg\alpha \wedge (\bigwedge_{\beta \in antecedents(K_{i-1})} \neg\beta)$;
4: $\mathcal{B} :=$ MinimalExtension$(\mathcal{A}, \mathcal{S}_{i+(n-m)}, K_{i-1})$;
5: $\mathcal{F} := \emptyset$
6: **for** x in $\mathcal{B}$ **do**
7: $S' = (x \setminus J_i) \cup \mathcal{S}$;
8: **if** $br_x(\alpha) = br_x(\beta) + 1$ and Minimise(S, α, S') is **False then**
9: $\mathcal{F} := \mathcal{F} \cup$ StrongSequences$(\mathcal{N}, \alpha, \mathcal{M}, i+1, S')$;
10: $\mathcal{S} := \mathcal{F}$;
11: **return** $\mathcal{S}$;

Once these minimal sets $\mathcal{S}$ for each sequence of the $\mathcal{J}_{\neg\alpha}$ justifications have been computed, we then need to iterate through all the variations of taking such an $\mathcal{J}_{\neg\alpha}$ set for each sequence and merging them. The smallest of these sets is considered to be the strong justification, denoted $\mathcal{J}_S^{RC}$. This method of minimising at multiple steps throughout the process ensures that the final set that gets created is minimal and thus adheres to one of the criteria to be considered a strong justification.

This entire process is demonstrated in StrongJustification.

Worked Example

Example 4.2.5. In order to bring all of the above together, consider once again the knowledge base and ranking shown in example 4.2.3, as well as the same query, and accompanying weak justification.

Algorithm 14 StrongJustification

Input: A knowledge base $\mathcal{K}$, a defeasible implication $\alpha \mid\sim \beta$ for the entailment

Output: A strong justification, $\mathcal{J}_S^{RC}$

1: $\mathcal{J} =$ ComputeGeneralSubsetSequences$(\mathcal{K}, \alpha)$;
2: $m := \max\{br_{\mathcal{K}}(j) : j \in \mathcal{J}\}$l
3: sequences := ComputeSubsetSequences$(\mathcal{K}, \alpha \wedge \neg\beta, m+1)$;
4: Choose $K' = (K_0, K_1, ..., K_{m+1})$ from sequences;
5: n := $|\mathcal{J}|$;
6: Let $S_1, S_2, ..., S_n$ be the set of minimal entailing sets associated with each sequence;
7: **for** J_i in $\mathcal{J}$ **do**
8: $\quad S_i :=$ StrongSequences$(K', \alpha \wedge \neg\beta, J_i, \beta, 0, \emptyset)$;
9: $\beta = \{s_1 \cup s_2 \cup ... \cup s_n : s_i \in S_i\}$;
10: $x = \min\{|X| : X \in \beta\}$;
11: Choose $\mathcal{J}_S^{RC}$ from $\mathcal{B}$ such that $|\mathcal{J}_S^{RC}| = x$
12: **return** $\mathcal{J}_S^{RC}$;

The "pushing up" sets for the weak justification are found in example 4.2.3.

The "pushing up" sets for $\mathcal{J}_{\neg\alpha}$ are found in example 4.2.4.

StrongSequences is then used to calculate the set of all minimum sets which adhere to the $br_{S\cup\mathcal{J}_i}(\alpha \wedge \beta) > br_{S\cup\mathcal{J}_i}(\alpha)$ criterion for each of the two sets of sequences calculated above. The two sequences for $\mathcal{J}_7$ required the largest set, with both returning $\mathcal{S} = \{sp \mid\sim f, p \mid\sim \neg f, b \mid\sim f, sp \rightarrow p, p \rightarrow b\}$. The remaining sets which we returned for the other sequences are subsets of this particular $\mathcal{S}$, and hence the minimum combination from these sets will always return $\mathcal{S}$.

Limitations of the approach

The above approach does have some limits [18], namely that:

1. Not every weak justification can be extended to create a strong one.

2. Not all strong justifications can be defined as extensions of weak justifications

∞	$w \to l$
1	$sp \mid\sim f, f \mid\sim m, f \mid\sim w, w \mid\sim x$
0	$p \mid\sim \neg f, l \mid\sim \neg x$

Table 4.14: Base ranking for a set of statements

∞	$p \to b$
1	$p \mid\sim \neg f, \neg f \mid\sim \neg w, \neg f \mid\sim r, \neg f \mid\sim s$
0	$b \mid\sim f, b \mid\sim s, r \mid\sim w$

Table 4.15: Base ranking for a set of statements

We use some examples to illustrate the above.

For point (1): given the rankings for statements (from a defeasible knowledge base) in figure 4.14, consider the query $sp \wedge f \mid\sim m \vee w$. For this entailment there are two weak justifications:

1. $\mathcal{J}_W^{RC}1 = \{sp \mid\sim f, f \mid\sim m\}$

2. $\mathcal{J}_W^{RC}2 = \{sp \mid\sim f, f \mid\sim w\}$

If we consider $\mathcal{J}_W^{RC}1$ for extension into a strong justification $\mathcal{J}_S^{RC}$, we have to ensure that $\mathcal{J}_S^{RC} \bigcup \{p \mid\sim \neg f\} \not\approx_{RC} sp \wedge f \mid\sim m \vee w$. To do so, $\{l \mid\sim \neg x, f \mid\sim w, w \mid\sim x, w \to l\}$ has to be added to $\mathcal{J}_S^{RC}$. Thus $\mathcal{J}_S^{RC} = \{sp \mid\sim f, f \mid\sim m, l \mid\sim \neg x, f \mid\sim w, w \mid\sim x, w \to l\}$.

However, we can drop $f \mid\sim m$ from $\mathcal{J}_S^{RC}$, creating $\mathcal{J}_S'^{RC}$.
Since $\mathcal{J}_S'^{RC} \not\approx sp \wedge f \mid\sim m \vee w$ for all $\mathcal{J}_S'^{RC} \subseteq \mathcal{J}_S^{RC} \subseteq \mathcal{K}$. This violates the minimality property required for a strong justification, and so $\mathcal{J}_W^{RC}1$ cannot be extended to create a strong justification.

For point (2): again, consider figure 4.15, which gives the rankings for statements from a defeasible knowledge base, and consider the query $p \mid\sim s$. This entailment has the singular weak justification $\mathcal{J}_W^{RC} = \{p \mid\sim \neg f, \neg f \mid\sim s\}$. However, we could also start with a different set for potential extension, say $\mathcal{H} = \{b \mid\sim s, p \to b\}$. This set is not a weak justification, however, it is possible to extend it into a strong justification: $\mathcal{J}_S^{RC} = \{b \mid\sim s, \neg f \mid\sim s, r \mid\sim w, \neg f \mid\sim \neg w, \neg f \mid\sim r, p \to b, \}$

Note that $\mathcal{J}_S^{RC} \not\approx_{RC} p \mid\sim s$. Now, if we add $p \mid\sim \neg f$ to $\mathcal{J}_S^{RC}$, which causes $\mathcal{H}$ to be discarded, then $\mathcal{J}_S^{RC}$ contains our initial weak justification, $\mathcal{J}_W^{RC}$ and so the entailment still holds.

4.3 Beyond weak and strong justifications: Minimal Rank Establishing Sets

The ranking of formulae is one of the fundamental steps of the KLM-style reasoning process. However, strong justifications are they are currently presented do not have this mechanism of fully capturing the ranking within the explanation itself i.e. the ranking for the final entailment. In other words, in addition to the final set of statements which allows the entailment to hold, we also would like to present some form of reasoning as to why this particular subset of statements qualifies as the most "specific".

In order to solve this problem, the notion of *Minimal Rank Establishing Sets* (MRES) was developed [44]. This was created in the context of extensions to the KLM framework [20], which was introduced in section 3.2.5.

In the framework of rational defeasible entailment, we first define a base rank preserving function, which takes all the formulae which can be derived from a set of atoms and ranks them. The knowledge base therefore becomes a specific ranked model. The particular type of model is decided by the version of rational defeasible entailment under consideration, which consequently then also defines r, the base rank preserving function. This is then used in a modified version of the rational closure algorithm, which has been generalised to enable it to compute defeasible entailment.

The key difference here is that the formulae in the ranks of this generalised algorithm do not originate from the original knowledge base and this has implications for explanations, in that we cannot simply input these generated formula into the explanation generating algorithms from previous sections and output the final justification(s). Therefore, in order to compute explanations for the statements from the original knowledge base $\mathcal{K}$, we first examine how the formulae influence the ranked model which then successively influences r, the ranking function.

Definition 4.3.1 (Minimal Rank Establishing Set)**.** Given a knowledge base $\mathcal{K}$, defeasible implication $\alpha \mid\sim \beta$ and base rank preserving function r, a *Minimal Rank Establishing Set* is a subset M of $\mathcal{K}$ such that:

1. if $r_{\mathcal{K}}(\alpha) = \infty, r_M(\alpha) = \infty$ and for all $M' \subset M, r_M(\alpha) \neq \infty$

2. if $\mathcal{K}(\alpha) \neq \infty, r_M(\alpha \wedge \neg\beta) > r_{\mathcal{K}}(\alpha)$ and for all $M' \subset M, r'_M(\alpha \wedge \neg\beta) < r_M(\alpha \wedge \neg\beta)$

The above definition allows for the explicit capturing of the ranking used in the final entailment.

4.3.1 Rational Closure

At present, MRES have only been created for the case of rational closure, thus in this context we will be using the base rank function *br* as defined in earlier chapters, with the main aim being that we are considering how MRES relates to the sequences for "pushing" the weak justification up the ranks. The algorithms used in strong justifications for rational closure, namely ComputeSubsetSequences and Sequences were modified to produce ComputeMRES and MRESSequences.

Algorithm 15 MRESSequences

Input: A knowledge base $\mathcal{K} = (E_0, E_1, ..., E_\infty)$, a formula α, the current sequence of subsets $(K_0, ..., K_{i-1})$, the rank n that is the minimum rank α can have in the final subset of the sequence and lastly the index i of the current subset

Output: A set of sequences which each have a length of least n

$\mathcal{A} := \neg\alpha \wedge (\bigwedge_{\beta \in antecedents(K_{i-1})} \neg\beta)$;
S := MinimalExtension($\mathcal{A}, E_{n-1}, K_{i-1}$)
$\mathcal{F} := \emptyset$;
if S $\neq \emptyset$ **then**
    for K_i in S **do**
        if $br_{\mathcal{K}_i(\alpha)} = i$ and Minimise(K_i) is False **then**
            $\mathcal{F} := \mathcal{F} \cup$ MRESSequences($\mathcal{K}, \alpha, (K_0, ..., K_{i-1}), n, i+1$);
if $\mathcal{F} := \emptyset$ and $i > n$ **then**
    $\mathcal{F} = \{(K_0, ..., K_{i-1})\}$;
return $\mathcal{F}$

Algorithm 16 ComputeMRES

Input: A knowledge base $\mathcal{K}$, a formula α, the minimum rank $n \leq br_{\mathcal{K}(\alpha)}$ required for α in the final subset of a sequence

Output: The set of all MREs for the entailment

1: $i := 0$;
2: $E_0 := \overrightarrow{\mathcal{K}}$;
3: **repeat**
4: $\quad E_{i+1} := \{\alpha \rightarrow \beta \in E_i \mid E_i \models \neg\alpha\}$;
5: **until** $E_{i-1} = E_i$;
6: $E_\infty := E_{i-1}$;
7: **if** $n = \infty$ **then**
8: $\quad$ MRES := ComputeAllJustifications$(E_\infty, \neg\alpha)$;
9: $\quad$ previous := $\emptyset$;
10: $\quad$ **repeat**
11: $\quad\quad$ previous := MRES;
12: $\quad\quad$ **for** S in MRES **do**
13: $\quad\quad\quad \mathcal{A} := \neg\alpha \wedge (\bigwedge_{\beta \in antecedents(S)} \neg\beta)$;
14: $\quad\quad\quad S' :=$ ComputeAllJustifications$(E_\infty, \mathcal{A})$;
15: $\quad\quad\quad$ MRES = (MRES $\setminus S) \cup S'$;
16: $\quad$ **until** MRES = previous;
17: **else**
18: $\quad$ sequences := MRESSequences$((E_0, ..., E_\infty), \alpha, (\emptyset), n, 1)$;
19: $\quad$ MRES := $\{K_i \in X : X \in$ sequences and $i \geq n\}$;
20: **return** MRES;

birds $\mid\sim$ fly $(b \mid\sim f)$
birds $\mid\sim$ eat bugs $(b \mid\sim e)$
penguins $\mid\sim \neg$ fly $(p \mid\sim \neg f)$
penguins $\mid\sim$ swim $(p \mid\sim s)$
special penguins $\mid\sim$ fly $(sp \mid\sim f)$
special penguins $\rightarrow$ penguins $(sp \rightarrow p)$
penguins $\rightarrow$ birds $(p \rightarrow b)$

Table 4.16: Knowledge base about penguins

∞	$sp \rightarrow p, p \rightarrow b$
2	$sp \mid\sim f$
1	$p \mid\sim \neg f, p \mid\sim s$
0	$b \mid\sim f, b \mid\sim e$

Table 4.17: Base ranking for statements in table 4.16

Worked Example

We now provide an example to illustrate in the most general sense, how MRESs for rational closure works.

Example 4.3.1. Consider the knowledge base shown in table 4.16, the ranking in table 4.17 and the query $p \vee sp \mid\sim s$.

We start with the antecedent $p \vee sp$, which has $br_{\mathcal{K}} = 1$. We provide this value $+1$, along with the knowledge base and the antecedent as input to the algorithm i.e. `ComputeMRES`$(\mathcal{K}, (p \vee sp) \wedge \neg s, 2)$.

The knowledge base is materialised, and `MRESSequences` computes the K sequences:
$K_0 = \emptyset$
$K_1 = \{p \mid\sim s, sp \rightarrow p\}$
$K_2 = K_1 \cup \{b \mid\sim f, p \mid\sim \neg f, p \rightarrow b\}$

Table 4.18 shows the MRES for the query taken from K_2. The information used to draw the final conclusion (shown in blue) is the most specific information one could use in order to do so.

As mentioned earlier, rational closure provides the foundation for other forms of rational defeasible entailment, and thus it may be possible to ex-

∞	$sp \rightarrow p$, $p \rightarrow b$
1	$p \mid\!\sim \neg f$, $p \mid\!\sim s$
0	$b \mid\!\sim f$

Table 4.18: MRES for the query

tend the above algorithms to these other formalisms (such as lexicographic closure).

Chapter 5

Conclusion

5.1 Summary

In this dissertation, we began by introducing propositional and description logic, highlighting the differences between their syntax and semantics. In the 90s, a need for reasoning systems to provide explanations was identified [40] and in the 2000s, work by Schlobach and Cornet [51] and Parsia and Kalyanpur [46] highlighted the ability of justifications to provide this explanatory mechanism in reasoning systems, effectively jumpstarting research on the topic. Justifications were inspired by the work of Reiter and his theory of model-based diagnosis [49]. Since then, there has been much research into justifications - mainly in terms of their computation ([26, 42, 58], granularity and understanding. The last point ties into the notion of explainable AI.

We then moved onto how classical justifications were computed in the case of description logics - that is to say, tableaux-based methods have become the popular choice [4].

The next chapter focused on defeasible reasoning. More specifically, how we need to alter the language and semantics of propositional logic in order to compute non-monotonic inferences. We started by detailing the preferential approach to reasoning, which was first suggested by Shoham [52, 53]. Additionally, we discussed KLM-style defeasible reasoning. [32, 35, 34]. Through the rest of the chapter we provided in-depth explanations for three types of defeasible reasoning formalisms, namely rational closure, lexicographic closure and relevant closure. We covered their algorithms and provided worked examples for these.

In chapter 4, the final chapter of content, we explored the current state of research in terms of propositional logic for defeasible justifications for KLM-style defeasible reasoning. We demonstrated why algorithms for computing classical justifications cannot be naïvely translated for the defeasible case. Following that, we elaborated on weak [16, 61, 18] and strong justifications [11], comparing their differences and highlighting their advantages. The algorithms for weak justifications across the different formalisms of rational, lexicographic and relevant closure are far more intuitive to understand and simpler than algorithm for strong justifications for rational closure. Furthermore, only an algorithm for strong justifications for rational closure has been defined - and not all parts of the algorithm have proofs for soundness and completeness. However, strong justifications (as they were defined in the original paper) are logic agnostic and highly generalisable and can be translated to a variety of logics. That is not to say one type of justification is better than the other, but rather that they may be suited to different contexts. We also highlighted how the formalism under consideration hugely influences what the final defeasible justification looks like. Lastly, we briefly covered minimal rank establishing sets for rational closure- which provided a different angle to the the above types of justifications. With MRES, the intention is to show why a particular set of statements can be regarded as the "most specific" when computing a justification. This is done by saving the ranking within the justification itself.

5.2 Summary of Contribution

The broad aim of this dissertation is to provide a point of reference for those interested in researching justifications. The primary objective is that it can be used as a pedagogical tool and provide a digestible introduction to classical and defeasible justifications, with a particular emphasis on propositional logic (and to a lesser extent description logic). This compilation should also provide a decent springboard for potential future research, as we've identified some of the critical gaps in the literature. We've also attempted to standardise some of the notation insofar as possible.

5.3 Future work and Open Issues

As mentioned in previous chapters, research for defeasible justifications is still relatively small. However, in terms of future work within the context of dissertation, there are a number of possibilities, ranging from more theoretical to practical.

In terms of the algorithms for weak justifications from chapter 4, only rational closure has a software implementation. This was done by Wang [61]. Thus, one could either extend Wang's work to include lexicographic and relevant closure, or create an entirely new implementation. The algorithm for weak justifications as it was defined in the original paper refined the removal process of statements. However, as mentioned in chapter 3, section 3.2.3, it is possible to instead refine the ranking of statements. Thus, research can be done on reworking the current lexicographic closure algorithm for weak justifications.

There is also scope for software implementations for the algorithms for strong justifications for rational closure or minimal rank establishing sets. This research could also involve running benchmark tests to understand how algorithms performance across various types of knowledge bases i.e. different sizes or levels of complexity.

On the more theoretical side, there is work to be done in terms of creating algorithms for strong justifications for other forms of defeasible entailment - such as lexicographic or relevant closure, or other formalisms such as system W - and other types of logics, say first-order logic or description logic. There is also the possibility of extending minimal rank establishing sets beyond rational closure to lexicographic closure or relevant closure. As we mentioned in the previous chapter, there is also presently no singular definition for a generalised defeasible justification - all present definitions are made in relation to some formalism for defeasible reasoning. Therefore, there is a need for a more formalism-agnostic way for defining a defeasible justification (or research which shows that this may not be possible).

Furthermore, justifications as they are studied now are predominantly presented in a textual form. There is scope for research into justification representation. This is linked to the concept of *justification-understanding*, which we touched upon in chapter 2. There is the possibility of presenting justifications in a more graphical form. Argumentation theory is the study of how one can use a set of premises to either support or undermine a conclusion [60], and some work has been done in this field that concerns representing

arguments using mathematical graphs i.e. sets of nodes and edges [9]. More specific to the context of this dissertation, there has been some introductory work on argument graphs for propositional logic by Besnard and Doutre [8]. This work can potentially be expanded to the context of justifications for KLM-style defeasible reasoning, wherein justifications can be represented in graph form. This also ties into explainable artificial intelligence - different representations of justifications may make certain inferences more easily understood by users of these systems or assist with debugging inconsistent knowledge bases [26].

Bibliography

[1] Avron, A. Simple consequence relations. *Information and Computation 92*, 1 (1991), 105–139.

[2] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F. *The Description Logic Handbook: Theory, Implementation and Applications*, 2 ed. Cambridge University Press, 2007.

[3] Baader, F., Horrocks, I., Lutz, C., and Sattler, U. *Introduction to description logic*. Cambridge University Press, 2017.

[4] Baader, F., and Peñaloza, R. Axiom pinpointing in general tableaux. *Journal of Logic and Computation 20*, 1 (2010), 5–34.

[5] Baader, F., and Sattler, U. An overview of tableau algorithms for description logics. *Studia Logica 69* (2001), 5–40.

[6] Belle, V. Logic meets probability: Towards explainable ai systems for uncertain worlds. In *IJCAI* (2017), pp. 5116–5120.

[7] Ben-Ari, M. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[8] Besnard, P., and Doutre, S. Checking the acceptability of a set of arguments. In *NMR* (2004), vol. 4, pp. 59–64.

[9] Besnard, P., Doutre, S., and Herzig, A. Encoding argument graphs in logic. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems: 15th International Conference, IPMU 2014, Montpellier, France, July 15-19, 2014, Proceedings, Part II 15* (2014), Springer, pp. 345–354.

[10] Brewka, G., Thimm, M., and Ulbricht, M. Strong inconsistency. *Artificial Intelligence 267* (2019), 78–117.

[11] Brewka, G., and Ulbricht, M. 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.

[12] Britz, K., Casini, G., Meyer, T., Moodley, K., Sattler, U., and Varzinczak, I. J. Theoretical foundations of defeasible description logics. *ArXiv abs/1904.07559* (2019).

[13] Calegari, R., Omicini, A., Sartor, G., et al. Argumentation and logic programming for explainable and ethical ai. In *CEUR WORKSHOP PROCEEDINGS* (2020), vol. 2742, Sun SITE Central Europe, RWTH Aachen University, pp. 55–68.

[14] Casini, G., Meyer, T., Moodley, K., and Nortjé, R. Relevant closure: A new form of defeasible reasoning for description logics. In *Logics in Artificial Intelligence: 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings 14* (2014), Springer, pp. 92–106.

[15] Casini, G., Meyer, T., and Varzinczak, I. Taking defeasible entailment beyond rational closure. In *European Conference on Logics in Artificial Intelligence* (2019), Springer, pp. 182–197.

[16] Chama, V. Explanation for defeasible entailment. Master's thesis, Faculty of Science, 2020.

[17] Etchemendy, J. Tarski on truth and logical consequence. *The Journal of Symbolic Logic 53*, 1 (1988), 51–79.

[18] Everett, L., Morris, E., and Meyer, T. Explanation for klm-style defeasible reasoning. In *Southern African Conference for Artificial Intelligence Research* (2021), Springer, pp. 192–207.

[19] Fermé, E., and Hansson, S. O. *Belief change: introduction and overview*. Springer, 2018.

[20] Freund, M. Preferential reasoning in the perspective of poole default logic. *Artificial Intelligence 98*, 1-2 (1998), 209–235.

[21] Gabbay, D. M. *Theoretical foundations for non-monotonic reasoning in expert systems*. Springer, 1985.

[22] Gallier, J. H. *Logic for computer science: foundations of automatic theorem proving*. Courier Dover Publications, 2015.

[23] Giordano, L., Gliozzi, V., Olivetti, N., and Pozzato, G. L. Preferential vs rational description logics: which one for reasoning about typicality? In *ECAI 2010*. IOS Press, 2010, pp. 1069–1070.

[24] Giordano, L., Gliozzi, V., Olivetti, N., and Pozzato, G. L. Semantic characterization of rational closure: From propositional logic to description logics. *Artificial Intelligence 226* (2015), 1–33.

[25] Greiner, R., Smith, B. A., and Wilkerson, R. W. A correction to the algorithm in reiter's theory of diagnosis. *Artificial Intelligence 41*, 1 (1989), 79–88.

[26] Horridge, M. *Justification based explanation in ontologies*. PhD thesis, University of Manchester UK, 2011.

[27] Kaliski, A. An overview of klm-style defeasible entailment. Master's thesis, University of Cape Town, 2020.

[28] Kalyanpur, A. *Debugging and repair of OWL ontologies*. University of Maryland, College Park, 2006.

[29] Kern-Isberner, G. *Conditionals in Nonmonotonic Reasoning and Belief Revision: Considering Conditionals as Agents*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003.

[30] Kern-Isberner, G. A thorough axiomatization of a principle of conditional preservation in belief revision. *Annals of Mathematics and Artificial intelligence 40* (2004), 127–164.

[31] Komo, C., and Beierle, C. Nonmonotonic inferences with qualitative conditionals based on preferred structures on worlds. In *KI 2020: Advances in Artificial Intelligence* (Cham, 2020), U. Schmid, F. Klügl, and D. Wolter, Eds., Springer International Publishing, pp. 102–115.

[32] Kraus, S., Lehmann, D., and Magidor, M. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial intelligence 44*, 1-2 (1990), 167–207.

[33] Krötzsch, M., Simancik, F., and Horrocks, I. A description logic primer. *CoRR abs/1201.4089* (2012).

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

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

[36] Levesque, H. J., and Lakemeyer, G. *The logic of knowledge bases*. MIT Press, 2001.

[37] Lyons, J. *Logical semantics*, vol. 1. Cambridge University Press, 1977, p. 138–173.

[38] McCarthy, J. Programs with common sense. In *Proceedings of the Symposium on the Mechanization of Thought Processes* (1959), London.

[39] McCarthy, J. Applications of circumscription to formalizing commonsense knowledge. *Artificial intelligence 28*, 1 (1986), 89–116.

[40] McGuinness, D. L., and Patel-Schneider, P. F. Usability issues in knowledge representation systems. In *Aaai/Iaai* (1998), pp. 608–614.

[41] Meister, W. Histological structure of the long bones of penguins. *The Anatomical Record 143*, 4 (1962), 377–387.

[42] Meyer, T., Lee, K., Booth, R., and Pan, J. Z. Finding maximally satisfiable terminologies for the description logic alc. In *AAAI* (2006), pp. 269–274.

[43] Monk, J. D. The contributions of alfred tarski to algebraic logic. *The Journal of Symbolic Logic 51*, 4 (1986), 899–906.

[44] Morris, E. Propositional defeasible explanation. `https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2021/everett_morris.zip/resources/Final_Project_Emily.pdf`, 2021.

[45] Morris, M., Ross, T., and Meyer, T. Algorithmic definitions for klm-style defeasible disjunctive datalog. *South African Computer Journal 32*, 2 (2020), 141–160.

[46] Parsia, B., Sirin, E., and Kalyanpur, A. Debugging owl ontologies. In *Proceedings of the 14th international conference on World Wide Web* (2005), pp. 633–640.

[47] Pearl, J. System z: a natural ordering of defaults with tractable applications to nonmonotonic reasoning. In *Proceedings of the 3rd Conference on Theoretical Aspects of Reasoning about Knowledge* (San Francisco, CA, USA, 1990), TARK '90, Morgan Kaufmann Publishers Inc., p. 121–135.

[48] Reiter, R. A logic for default reasoning. *Artificial intelligence 13*, 1-2 (1980), 81–132.

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

[50] Rossi, F. Building trust in artificial intelligence. *Journal of international affairs 72*, 1 (2018), 127–134.

[51] Schlobach, S., Cornet, R., et al. Non-standard reasoning services for the debugging of description logic terminologies. In *Ijcai* (2003), vol. 3, pp. 355–362.

[52] Shoham, Y. Nonmonotonic logics: Meaning and utility. In *IJCAI* (1987), vol. 10, Citeseer, pp. 388–393.

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

[54] Spackman, K. A., Campbell, K. E., and Côté, R. A. Snomed rt: a reference terminology for health care. In *Proceedings of the AMIA annual fall symposium* (1997), American Medical Informatics Association, p. 640.

[55] Spohn, W. Ordinal conditional functions: A dynamic theory of epistemic states. In *Causation in decision, belief change, and statistics: Proceedings of the Irvine Conference on Probability and Causation* (1988), Springer, pp. 105–134.

[56] Spohn, W. *The laws of belief: Ranking theory and its philosophical applications*. Oxford University Press, 2012.

[57] Stuckenschmidt, H. Debugging owl ontologies-a reality check. In *EON* (2008), vol. 359.

[58] Suntisrivaraporn, B., Qi, G., Ji, Q., and Haase, P. A modularization-based approach to finding all justifications for owl dl entailments. In *The Semantic Web: 3rd Asian Semantic Web Conference, ASWC 2008, Bangkok, Thailand, December 8-11, 2008. Proceedings. 3* (2008), Springer, pp. 1–15.

[59] Tarski, A., Corcoran, J., and Woodger, J. *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. Hackett Publishing Company, 1983.

[60] van Eemeren, F. H., Garssen, B., Krabbe, E. C. W., Henkemans, A. F. S., Verheij, B., and Wagemans, J. H. M. *Handbook of Argumentation Theory*. Springer Verlag, Dordrecht, Netherland, 2014.

[61] Wang, S., Meyer, T., and Moodley, D. Defeasible justification using the klm framework. In *Southern African Conference for Artificial Intelligence Research* (2022), Springer, pp. 187–201.