Defeasible Justification for KLM-Style Logic

Victoria Chama[1,2], Steve Wang[1,2], Thomas Meyer[1,2,3] and Giovanni Casini[3,1,2]

[1]University of Cape Town, South Africa
[2]CAIR, South Africa
[3]CNR-ISTI, Italy

Abstract

Research in description logics (DLs) and formal ontologies has dedicated quite an effort to the investigation of the notion of explanation for DL reasoning, for example relying on the notion of *justification* [1]. There has also been some effort dedicated to the definition of defeasible reasoning for DLs that, contrary to the classical monotonic reasoning, is appropriate for dealing with incomplete/uncertain information. In the present paper, we extend the notion of justification to the framework of defeasible reasoning for DLs; specifically, we consider *rational closure* [2], an entailment relation that is of particular importance in the area of defeasible reasoning. Here we present the main theoretical results for the DL $\mathcal{ALC}$, and an implementation of our solution, at the moment developed for propositional logic.

Keywords

Defeasible entailment, rational closure, justification-based explanations

1. Introduction

Knowledge Representation and Reasoning (KRR) is an essential aspect of Artificial Intelligence (AI). Logics such as Propositional Logic (PL) and Description Logic (DL) provide a way to formulate and represent knowledge in a structured manner. Reasoning with such formalised knowledge representations allows us to deduce new information from explicit knowledge. In monotonic (classical) reasoning adding new knowledge cannot cause the retraction of previously drawn deductions. This characteristic of monotonic reasoning is not appropriate for reasoning with rules and generalisations that admit exceptions.

Consider the classic 'penguin example': "penguins are birds", "penguins cannot fly" and "birds can fly". Monotonic reasoning concludes that there are no instances of penguins because birds can both fly and not fly. However, penguins legitimately are birds that cannot fly. Non-monotonic reasoning can overcome such restriction, in particular the approach to non-monotonicity that is known as defeasible reasoning. We use a non-monotonic extension of the DL $\mathcal{ALC}$ to perform defeasible reasoning, for which software tools already exist [3].

Conclusions produced by reasoning tools are often difficult to understand due to the volume and complexity of the statements, and there is a need for algorithms and tools that provide explanations for conclusions. In recent years, research has been done to provide explanations for conclusions in monotonic reasoning [1, 4]. Consequently, various algorithms to compute explanations and implementations of these algorithms for DL are available as plugins in Protégé [1]. However, little research has gone into providing explanations for defeasible reasoning systems. Defeasible reasoning introduces well-defined and systematic approaches to reasoning with incomplete information and rules that admit exceptions [5]. Some popular defeasible reasoning systems are based on the semantics developed by Kraus, Lehmann, and Magidor (KLM) [6]. We focus here on *rational closure*, originally defined for PL [7], but has also been lifted to the case for DLs [2, 8, 9]. In this paper we introduce an algorithm that

DL 2024: 37th International Workshop on Description Logics, June 18–21, 2024, Bergen, Norway
vchama@cs.uct.ac.za (V. Chama); wngshu003@myuct.ac.za (S. Wang); tmeyer@cs.uct.ac.za (T. Meyer); giovanni.casini@isti.cnr.it (G. Casini)
https://github.com/SteveWang7596 (S. Wang)
0000-0003-3641-4849 (V. Chama); 0000-0002-8741-2213 (S. Wang); 0000-0003-2204-6969 (T. Meyer); 0002-4267-4447 (G. Casini)

computes explanations for conclusions drawn using rational closure. The logic we use for this algorithm is a KLM extension to DL $\mathcal{ALC}$. We also present an implementation of the algorithm in PL.

The rest of the paper is structured as follows. The next section introduces the necessary preliminaries. In Section 3, we use the concepts provided in Section 2 to construct a defeasible justification algorithm. Then, we present an implementation of the defeasible justification algorithm for the propositional case in Section 4. Lastly, we conclude the paper in Section 5.

2. Preliminaries

In this section, we introduce the foundational notions and concepts required to build an algorithm that computes explanations for conclusions produced by reasoning services. After a brief introduction to the DL $\mathcal{ALC}$, we present the notion of justification defined by Horridge which serves as an explanation for classical reasoning services. Then we introduce the defeasible logic that we are going to use throughout the paper. Lastly, we provide details on the form of defeasible reasoning known as *rational closure*.

2.1. Description Logic $\mathcal{ALC}$

Description logics (DLs) are a family of logical formalisms that can represent conceptual knowledge in a structured and formally well-understood way. $\mathcal{ALC}$ is a basic formalism of DL and stands for Attributive (Concept) Language with Complements [10]. It includes concept names, concept intersection, concept union, complement, existential and universal qualifiers. This section provides the syntax and semantics of $\mathcal{ALC}$ that will be used in the rest of this paper.

Definition 1 (Syntax). *The language of $\mathcal{ALC}$ is built from a finite set of atomic concept names, A, and a finite set of atomic role names, R. Complex concepts are denoted with C and are built inductively according to the following rule:*

$$C ::= \top|\bot|a|\neg C|C \sqcap C|C \sqcup C|\exists r.C|\forall r.C$$

where $\top$ is the universal concept, $\bot$ is the bottom concept, $\neg$ is the unary negation operation, $\sqcap$ is the binary intersection operation, $\sqcup$ is the binary union operation, $\exists$ is the existential qualifier, $\forall$ is the universal qualifier, $a \in A$ and $r \in R$ [11].

We use $\mathcal{L}$ to denote the language of all $\mathcal{ALC}$ concepts. The semantics of $\mathcal{L}$ is given by the notion of *interpretation*.

Definition 2 (Semantics). *An interpretation is a structure $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set called the domain, and $\cdot^{\mathcal{I}}$ is an interpretation function. For every $c \in C$, $c^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$ and for every $r \in R$, $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For some complex concepts C and $D \in \mathcal{L}$, the interpretation function $\cdot^{\mathcal{I}}$ extends to interpret all complex concepts of $\mathcal{L}$ in the following way:*

$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}}/C^{\mathcal{I}}, (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}},$$
$$(\forall r.C)^{\mathcal{I}} = \{x | y \in C^{\mathcal{I}} \text{ for every } y \text{ s.t } (x, y) \in r^{\mathcal{I}}\}, (\exists r.C)^{\mathcal{I}} = \{x | (x, y) \in r^{\mathcal{I}} \text{ for some } y \in C^{\mathcal{I}}\}$$

Given $C, D \in \mathcal{L}$, $C \sqsubseteq D$ is called a subsumption statement (or a statement). Concept equivalence, denoted $C \equiv D$, is the abbreviation for $C \sqsubseteq D$ and $D \sqsubseteq C$. An $\mathcal{ALC}$ TBox $\mathcal{T}$ is a finite set of statements. We refer to such statements in $\mathcal{T}$ as axioms. We use $\alpha, \beta, ...$ to denote statements in $\mathcal{T}$. An interpretation $\mathcal{I}$ satisfies a statement, $C \sqsubseteq D$, denoted $\mathcal{I} \Vdash C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of TBox $\mathcal{T}$ if $\mathcal{I}$ satisfies every statement in $\mathcal{T}$. Given a TBox $\mathcal{T}$ and a statement α, $\mathcal{T}$ entails α, denoted $\mathcal{T} \models \alpha$, if and only if for every interpretation $\mathcal{I}$ such that $\mathcal{I} \Vdash \mathcal{T}$, it also holds that $\mathcal{I} \Vdash \alpha$. In this paper, we are not interested in the ABox $\mathcal{A}$ which is a finite set of concept assertions (instantiation of concepts) and role assertions (instantiation of roles).

2.2. Justifications

Given an entailment, $\mathcal{T} \models \alpha$, an explanation for $\mathcal{T} \models \alpha$ is a subset of $\mathcal{T}$ such that it entails α. Intuitively, we want to identify the axioms in the $\mathcal{T}$ that contributed towards the entailment. There are several approaches to computing explanations for entailments [1, 12]. In this paper, we base the explanation for entailment on the notion of *justification* defined by Horridge [1]. Horridge defines justification for an entailment as the minimal subsets of the knowledge base that entails the query.

Definition 3 (Justification). *Let $\mathcal{T}$ be a TBox and α be a statement. $\mathcal{J}$ is a justification for $\mathcal{T} \models \alpha$ if $\mathcal{J} \subseteq \mathcal{T}$, $\mathcal{J} \models \alpha$ and, for all $\mathcal{J}' \subset \mathcal{J}$, it holds that $\mathcal{J}' \not\models \alpha$.*

Example 1. *Let us consider the penguin example that we mentioned in the introduction. We can formalise it in a classical TBox, $\mathcal{T}$, about birds, penguins and a special penguin that has flying capabilities.*

$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{Bird} \sqsubseteq \mathsf{Fly}, \\ \mathsf{Bird} \sqsubseteq \mathsf{Wings}, \\ \mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly}, \\ \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}, \\ \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Penguin}, \\ \mathsf{Robin} \sqsubseteq \mathsf{Bird} \end{array} \right\}$$

Such a TBox entails *that special penguins fly ($\mathcal{T} \models \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}$). We can identify two* justifications *for such a conclusion, that is, two minimal subsets of $\mathcal{T}$ that allow the derivation of* $\mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}$*:*

$$\mathcal{J}_1 = \{\mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}\}$$
$$\mathcal{J}_2 = \{\mathsf{SpecialPenguin} \sqsubseteq \mathsf{Penguin}, \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \mathsf{Bird} \sqsubseteq \mathsf{Fly}\}$$

$\mathcal{J}_1$ is a valid justification because the query, $\mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}$, is an axiom in $\mathcal{T}$, and $\mathcal{J}_2$ is also valid according to Definition 3.

Horridge presents an implementation for computing justifications in Protégé [1, Algorithm 4.1]. In what follows we will refer to it as Algorithm ComputeAllJustifications (Algorithm 4.1), that, given a TBox $\mathcal{T}$ and a statement $C \sqsubseteq D$ entailed by $\mathcal{T}$, returns all associated justifications.

2.3. Defeasible Reasoning

In this section, we discuss how defeasible reasoning has been implemented in DLs. Defeasible logic enables us to reason with rules and generalisations that model typical scenarios that admit exceptions. For example, birds typically fly but penguins are atypical birds that do not fly.

There is existing work for extending description logics with defeasible reasoning [2, 3, 8, 9, 13, 14, 15, 16, 17, 18]. As mentioned in the introduction our work is based on the so-called KLM approach to defeasible reasoning [6], and in particular on a specific entailment relation, *rational closure* [7]. We focus on rational closure since it is a particularly significant entailment relation in defeasible reasoning: other relevant entailment relations are based on it [19, 20, 21], it has already been redefined for description logics [2, 8, 9, 22], moreover it can be implemented just using a relatively simple procedure on top of some classical DL reasoner [3]. We introduce our explanations for defeasible reasoning using a well-known extension of the DL $\mathcal{ALC}$ to model defeasible reasoning [2]. Such an extension allows for an additional type of subsumption statement: '$C \sqsubseteq\!\!\!\!\sim D$' is a defeasible statement which is read "*Typically*, C's are D's, as opposed to "all C's are D's" for the classical subsumptions '$C \sqsubseteq D$'.

Definition 4. *(Defeasible Inclusion) For $C, D \in \mathcal{L}$, a defeasible inclusion is a statement of the form $C \sqsubseteq\!\!\!\!\sim D$.*

A defeasible statement $C \sqsubseteq\!\!\!\!\sim D$ can be read as "*usually*, an instance of the class C is also an instance of the class D". A statement like "birds typically fly", formally represented by $\mathsf{Bird} \sqsubseteq\!\!\!\!\sim \mathsf{Fly}$ in defeasible $\mathcal{ALC}$, indicates that if something is a bird then it is reasonable to infer that it flies, if we are not informed of the contrary. ' $\sqsubseteq\!\!\!\!\sim$ ' is meant to be the defeasible counterpart of the classical subsumption $\sqsubseteq$ [2]. We define a DBox $\mathcal{D}$ as a finite set of defeasible statements, and a defeasible knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$ as consisting of a TBox and DBox. We refer to statements stated in $\mathcal{K}$ as axioms. The statements that have a defeasible form are those that can have exceptions to the statements: since most birds fly, but there are exceptions such as penguins and ostriches, we use $\mathsf{Bird} \sqsubseteq\!\!\!\!\sim \mathsf{Fly}$ instead of $\mathsf{Bird} \sqsubseteq \mathsf{Fly}$.

Example 2. *Consider the following defeasible version of TBox $\mathcal{T}$ shown in Example 1.*

$$\mathcal{T} = \left\{ \begin{array}{c} \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ \mathsf{Robin} \sqsubseteq \mathsf{Bird} \\ \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Penguin} \end{array} \right\}$$

$$\mathcal{D} = \left\{ \begin{array}{c} \mathsf{Bird} \sqsubseteq\!\!\!\!\sim \mathsf{Fly}, \\ \mathsf{Bird} \sqsubseteq\!\!\!\!\sim \mathsf{Wings}, \\ \mathsf{Penguin} \sqsubseteq\!\!\!\!\sim \neg\mathsf{Fly} \\ \mathsf{SpecialPenguin} \sqsubseteq\!\!\!\!\sim \mathsf{Fly} \end{array} \right\}$$

2.3.1. Rational Closure

As mentioned, we focus our attention on *rational closure*, a specific form of defeasible entailment. For the full description, the semantics, and the motivation for the use of rational closure in $\mathcal{ALC}$, the reader is directed to the work of Britz et al. [2], while here we focus only on the description of the algorithmic procedure to decide rational closure. In essence, given a knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$, the algorithm produces a ranking of the defeasible axioms in $\mathcal{D}$, all of which are then converted to classical axioms, from which defeasible entailment (rational closure) can then be computed by a series of classical entailment checks. The ranking of axioms is based on their *exceptionality*. Intuitively, in case of potential conflicts exceptionality arranges axioms in the knowledge base according to specificity: if we have conflictual information, (e.g., birds fly, penguins are birds, and penguins do not fly), the axioms that are considered less specific than others will be discarded first (e.g., 'birds fly' is discarded in favour of 'penguins do not fly' since information about penguins is more specific than information about birds in general).

In [2, Sect. 5.2] the procedure Exceptional is aimed at deciding which defeasible axioms are more exceptional w.r.t. a knowledge base. Exceptionality is based on the specificity among classes in case we have a conflict: for example 'penguins usually do not fly' ($\mathsf{Penguin} \sqsubseteq\!\!\!\!\sim \neg\mathsf{Fly}$) is more exceptional than 'birds usually fly' ($\mathsf{Bird} \sqsubseteq\!\!\!\!\sim \mathsf{Fly}$), since there is a potential conflict and the antecedent of the former (Penguin) is a subclass (more specific) than the antecedent of the latter (Bird). We do not provide the procedure Exceptional here, referring the reader to [2, Sect. 5.2]. On top of procedure Exceptional we build Algorithm 1, that takes as input a knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$ and returns a ranking of the axioms in $\mathcal{D}$ according to exceptionality: from rank 0, with $\mathcal{E}_0$ containing all the defeasible axioms in $\mathcal{D}$, to some rank n, with $\mathcal{E}_n$, containing the most exceptional axioms in $\mathcal{D}$. Algorithm 1 is taken from [23, Sect. 5.2]. We again refer the reader to this paper for proper explanations and analysis. Here we present the algorithm and go through an example to give a sense of how the algorithm works.

Example 3. *For the defeasible knowledge base $\mathcal{K}$ from Example 2,* ComputeRanking *will be executed as follows.*[1]

1. *The algorithm takes as input $\mathcal{K}$ and creates a hierarchy of defeasible axioms w.r.t their exceptionality. First it considers the entire set $\mathcal{D}$, associating it with $\mathcal{E}_0$ (line 6):*

[1]Note that the algorithm also introduces a knowledge base $\mathcal{K}^* = \mathcal{T}^* \cup \mathcal{D}^*$. $\mathcal{K}^*$ is a reformulation of the input $\mathcal{K}$ that is needed for knowledge bases with a particular form, which is not the case for our example. Here the reader can just assume that $\mathcal{K}^* = \mathcal{K}$ and, again, we refer to [2] for an explanation.

Algorithm 1: ComputeRanking($\mathcal{K}$)

Input: $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$
Output: $\mathcal{K}^* = \mathcal{T}^* \cup \mathcal{D}^*$, an exceptionality sequence $\mathcal{E}_0, \ldots, \mathcal{E}_n$ and a partitioning $R = \{\mathcal{D}_0, \ldots, \mathcal{D}_n\}$ for $\mathcal{D}^*$

1 $\mathcal{T}^* \leftarrow \mathcal{T}$
2 $\mathcal{D}^* \leftarrow \mathcal{D}$
3 $R \leftarrow \emptyset$
4 **repeat**
5 $\quad i \leftarrow 0$
6 $\quad \mathcal{E}_0 \leftarrow \mathcal{D}^*$
7 $\quad \mathcal{E}_1 \leftarrow$ Exceptional$(\mathcal{T}^*, \mathcal{E}_0)$
8 $\quad$ **while** $\mathcal{E}_{i+1} \neq \mathcal{E}_i$ **do**
9 $\quad\quad i \leftarrow i + 1$
10 $\quad\quad \mathcal{E}_{i+1} \leftarrow$ Exceptional$(\mathcal{T}^*, \mathcal{E}_i)$
11 $\quad \mathcal{D}^*_\infty \leftarrow \mathcal{E}_i$
12 $\quad \mathcal{T}^* \leftarrow \mathcal{T}^* \cup \{C \sqsubseteq D \mid C \mathrel{\underset{\sim}{\sqsubset}} D \in \mathcal{D}^*_\infty\}$
13 $\quad \mathcal{D}^* \leftarrow \mathcal{D}^* \setminus \mathcal{D}^*_\infty$
14 **until** $\mathcal{D}^*_\infty = \emptyset$
15 **for** $j \leftarrow 1$ *to* i **do**
16 $\quad \mathcal{D}_{j-1} \leftarrow \mathcal{E}_{j-1} \setminus \mathcal{E}_j$
17 $\quad R \leftarrow R \cup \{\mathcal{D}_{j-1}\}$
18 $\mathcal{E} \leftarrow (\mathcal{E}_0, ..., \mathcal{E}_{i-1})$
19 **return** $\mathcal{K}^* = \mathcal{T}^* \cup \mathcal{D}^*, \mathcal{E}, R$

$$\mathcal{E}_0 = \{\mathsf{Bird} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}, \mathsf{Bird} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Wings}, \mathsf{Penguin} \mathrel{\underset{\sim}{\sqsubset}} \neg\mathsf{Fly}, \mathsf{SpecialPenguin} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}\}$$

2. *Then it applies the algorithm* Exceptional *to it (line 7).* Exceptional *identifies all the potential conflicts in $\mathcal{T}$, that in our case are two: (1) birds fly, but penguins, that are birds, do not; (2) penguins do not fly, but special penguins, that are penguins, do. From such two conflicts, the algorithm* Exceptional *identifies the most specific pieces of defeasible information that play a role in it: penguins do not fly for (1) and special penguins fly for (2). Such axioms are associated with the first level of exceptionality:*

$$\mathcal{E}_1 = \{\mathsf{Penguin} \mathrel{\underset{\sim}{\sqsubset}} \neg\mathsf{Fly}, \mathsf{SpecialPenguin} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}\}.$$

3. *Now, since $\mathcal{E}_1 \neq \mathcal{E}_0$, we apply the* Exceptional *algorithm iteratively until we reach a fixed point (lines 9-12). More in detail, we apply it to $\mathcal{T}, \mathcal{E}_1$, and* Exceptional *identifies a potential conflict corresponding to case (2) above: (2) penguins do not fly, but special penguins, that are penguins, do. Again, the algorithm identifies the most specific defeasible axioms playing a role in the potential conflicts (in this case, special penguins fly), and associates them with a new exceptionality level $\mathcal{E}_2$ (line 11):*

$$\mathcal{E}_2 = \{\mathsf{SpecialPenguin} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}\}.$$

4. *Since $\mathcal{E}_2 \neq \mathcal{E}_1$ (line 9) we again apply the* Exceptional *algorithm to $\mathcal{T}, \mathcal{E}_2$, and* Exceptional *checks that there are no more potential conflicts, and $\mathcal{E}_3 = \emptyset$. (line 11). The empty set is trivially a fixed point of our procedure. Now we have the exceptionality sequence $\{\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2\}$. The algorithm uses this sequence to assign a* ranking value *to each axiom in $\mathcal{D}$ (line 18): all the axioms that are not exceptional go into rank 0:*

$$\mathcal{D}_0 = \mathcal{E}_0 \setminus \mathcal{E}_1 = \mathsf{Bird} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}, \mathsf{Bird} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Wings}.$$

The axioms that are exceptional only w.r.t. $\mathcal{E}_0$ go into rank 1 ($\mathcal{D}_1$), and so on:

$$\mathcal{D}_1 = \mathcal{E}_1 \setminus \mathcal{E}_2 = \mathsf{Penguin} \mathrel{\underset{\sim}{\sqsubset}} \neg\mathsf{Fly}.$$
$$\mathcal{D}_2 = \mathcal{E}_2 \setminus \mathcal{E}_3 = \mathcal{E}_2 = \mathsf{SpecialPenguin} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}.$$

Now we move to Algorithm 2, that, given a knowledge base $\mathcal{K}$ and a query $C \mathrel{\underset{\sim}{\sqsubset}} D$, decides whether $\mathcal{K}$ entails $C \mathrel{\underset{\sim}{\sqsubset}} D$ under rational closure. The original algorithm is in [2, Sect. 5.2], and once again we refer the reader to this publication for proper explanations and analysis. Here we simply go through our guiding example, while pointing out the following about Algorithm 2:

- The symbol $\overline{\mathcal{E}_i}$ indicates the *materialisation* of a set of defeasible axioms: given a set $\mathcal{D} = \{C_1 \mathrel{\underset{\sim}{\sqsubset}} D_1, \ldots, C_n \mathrel{\underset{\sim}{\sqsubset}} D_n\}$, the set $\overline{\mathcal{D}} = \{\neg C_1 \sqcup D_1, \ldots, \neg C_n \sqcup D_n\}$ is the set of concepts expressing the implications that correspond to the defeasible subsumptions.
- Algorithm 2 is slightly different w.r.t. the Function RationalClosure in [2], since it returns not only the answer to the query, but also a value i that provides the exceptionality ranking of the query. This information is needed for the creation of the justifications (see Section 2.3).

Example 4. *Consider the rankings obtained in Example 3 with the query* $\mathsf{SpecialPenguin} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}$. *The rational closure algorithm starts with $\mathcal{E}_0$ (line 2), and checks whether the antecedent in the query is in a conflict w.r.t. $\mathcal{E}_0$. That is the case since (line 3):*

$$\mathcal{T} \models \bigsqcap\{\neg\mathsf{Bird} \sqcup \mathsf{Fly}, \neg\mathsf{Bird} \sqcup \mathsf{Wings}, \neg\mathsf{Penguin} \sqcup \neg\mathsf{Fly}, \neg\mathsf{SpecialPenguin} \sqcup \mathsf{Fly}\} \sqcap \mathsf{SpecialPenguin} \sqsubseteq \bot$$

The algorithms goes on checking for increasing values of $\mathcal{E}_i$ (lines 3-4), until it finds an exceptionality level s.t. SpecialPenguin *is not part of any conflict. That is the case fo $\mathcal{E}_2$, since*

$$\mathcal{T} \not\models (\neg\mathsf{SpecialPenguin} \sqcup \mathsf{Fly}) \sqcap \mathsf{SpecialPenguin} \sqsubseteq \bot$$

Now the algorithm associates the information in $\mathcal{E}_2$ with the query, and checks whether the query is implied using materialisation and classical DL reasoning (line 6). That is the case since

$$\mathcal{T} \models (\neg\mathsf{SpecialPenguin} \sqcup \mathsf{Fly}) \sqcap \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}.$$

The algorithm returns true *and the value 2 (the exceptionality level associated with the query).*

Algorithm 2: RationalClosureForJustifications($\mathcal{K}$, α)

Input: $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$ and a query $\alpha = C \mathrel{\underset{\sim}{\sqsubset}} D$.
Output: `true` if $C \mathrel{\underset{\sim}{\sqsubset}} D$ is in the rat. clos. of $\mathcal{K}$, `false` otherwise; and a rank value i

1 $(\mathcal{K}^* = \mathcal{T}^* \cup \mathcal{D}^*, \mathcal{E}_{sequence} = (\mathcal{E}_0, \ldots, \mathcal{E}_n)) \leftarrow \mathsf{ComputeRanking}(\mathcal{K})$
2 $i \leftarrow 0$
3 **while** $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_i} \sqcap C \sqsubseteq \bot$ *and* $i \leq n$ **do**
4 $\quad i \leftarrow i + 1$
5 **if** $i \leq n$ **then**
6 $\quad$ **return** $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_i} \sqcap C \sqsubseteq D$, i
7 **else**
8 $\quad$ **return** $\mathcal{T}^* \models C \sqsubseteq D$, i

3. Defeasible Justification

We now define a procedure that calculates *justifications* for defeasible entailment: ComputeDefeasibleJustifications. It is obtained by using RationalClosureForJustifications and ComputeAllJustifications* as sub-procedures. ComputeAllJustifications* is a simple modification of ComputeAllJustifications [1, Algorithm 4.1]. We will not present it here. It is sufficient to know that, given $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$ and a defeasible inclusion $C \mathrel{\underset{\sim}{\sqsubset}} D$, ComputeAllJustifications*$(\mathcal{K}, C \mathrel{\underset{\sim}{\sqsubset}} D)$ returns every minimal set $\mathcal{J} = \mathcal{T}' \cup \mathcal{D}'$ s.t. $\mathcal{J} \subseteq \mathcal{K}$ and $\mathcal{T}' \models \bigsqcap \overline{\mathcal{D}'} \sqcap C \sqsubseteq D$.

The algorithm first determines whether a particular query is in the rational closure of the knowledge base (line 1), and only if it is the case, does it proceed to calculate the justifications (lines 3-15). From RationalClosureForJustifications we know the rank i that is associated to the query (line 6), and we consider only the information in $\mathcal{E}_i$ to calculate the justifications using RationalClosureForJustifications (lines 8 or 15, depending on the value of the rank i).

Example 5. *Consider the defeasible knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$ from Example 2. In Example 4 we have seen that* SpecialPenguin $\mathrel{\underset{\sim}{\sqsubset}}$ Fly *is defeasibly entailed by $\mathcal{K}$. We see now how Algorithm 3 explains why* SpecialPenguin $\mathrel{\underset{\sim}{\sqsubset}}$ Fly *is defeasibly entailed by $\mathcal{K}$:*

1. *Line 1 checks whether* SpecialPenguin $\mathrel{\underset{\sim}{\sqsubset}}$ Fly *is in the rational closure of $\mathcal{K}$. From Example 4 we know that it is the case, and the algorithm moves to line 3.*
2. *We initialise a variable i to 0 (line 4) and a set $\mathcal{J}_{result}$ to an empty set (line 5). In line 6 we obtain the value $rank = 2$, since, as we have seen in Example 4,* SpecialPenguin *is associated to $\mathcal{E}_2$. Due to that, the If-statement block on lines 7-9 is skipped since $rank > 0$.*
3. *Once we obtain the ranking R from* ComputeRanking *(line 11), the while-loop in lines 12-14 removes axioms contained in $\mathcal{D}_0$ and $\mathcal{D}_1$. Therefore, we can only use only axioms in $\mathcal{D}_2$ and in the TBox $\mathcal{T}$ to compute the justifications.*
4. *On line 15 we compute all the justifications for the query* SpecialPenguin $\mathrel{\underset{\sim}{\sqsubset}}$ Fly *w.r.t $\mathcal{T} \cup \overline{\mathcal{D}_2}$. The only justification in $\mathcal{J}_{result}$ is $\mathcal{J}_1 = \{$SpecialPenguin $\mathrel{\underset{\sim}{\sqsubset}}$ Fly$\}$.*

Algorithm 3: ComputeDefeasibleJustifications($\mathcal{K}, \eta$)

Input: $\mathcal{K} = \mathcal{T} \cup \mathcal{D}$ and a query $\eta = C \mathrel{\underset{\sim}{\sqsubset}} D$.
Output: RationalClosureForJustifications($\mathcal{K}, \eta$); the set of justifications $\mathcal{J}$

```
1  if RationalClosureForJustifications(K, η) = false then
2      return RationalClosureForJustifications(K, η)
3  else
4      i ← 0
5      J_result ← ∅
6      rank ← value i from RationalClosureForJustifications(K, η)
7      if rank = 0 then
8          J_result ← ComputeAllJustifications*(K, η)
9          return RationalClosureForJustifications(K, η), J_result
10     else
11         R = {D_0, ...} ← R from ComputeRanking(K)
12         while i < rank do
13             K_new ← K_new \ D_i
14             i ← i + 1
15         J_result ← ComputeAllJustifications*(K_new, η)
16         return RationalClosureForJustifications(K, η), J_result
```

We return to Example 1 given in Section 2.2. Recall that using the knowledge base in Example 1 there are two justifications that support the query $\mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly}$: $\mathcal{J}_1 = \{\, \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Fly} \,\}$ and $\mathcal{J}_2 = \{\, \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Penguin}, \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \mathsf{Bird} \sqsubseteq \mathsf{Fly} \,\}$. Once converted the knowledge base to the defeasible formalism (Example 2), the corresponding justification sets are $\mathcal{J}_1 = \{\, \mathsf{SpecialPenguin} \mathrel{\underset{\sim}{\sqsubseteq}} \mathsf{Fly} \,\}$ and $\mathcal{J}_2 = \{\, \mathsf{SpecialPenguin} \sqsubseteq \mathsf{Penguin}, \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \mathsf{Bird} \mathrel{\underset{\sim}{\sqsubseteq}} \mathsf{Fly} \,\}$. However, our algorithm eliminates from the potential justifications all the axioms that have rank lower than 2, hence eliminating $\mathsf{Bird} \mathrel{\underset{\sim}{\sqsubseteq}} \mathsf{Fly}$, that is required for $\mathcal{J}_2$. Eventually, only justification $\mathcal{J}_1$ is eligible.

Everett et al. [24] extended the principle of justifications for both Relevant and Lexicographic Closure.

4. Implementation

In this section we present an implementation of Algorithm 3, but at the moment just for Propositional Logic (PL)[25]. The original formulation of rational closure was for PL [7], and was developed in the formal framework introduced by Kraus, Lehmann and Magidor (KLM) [6]. They use defeasible conditionals $\alpha \mathrel{|\!\sim} \beta$, where α and β are propositional formulas, and that are read as "if α, usually β". A knowledge base in this framework is a finite set of defeasible implications and propositional formulas. Since any propositional formula can be rewritten as a classical implication of the form $\alpha \to \beta$ (trivially, α can be rewritten as $\top \to \alpha$ or $\neg\alpha \to \bot$), a knowledge base in the KLM Framework can be seen as a finite set of classical implications and defeasible implications. The correspondence with our defeasible knowledge base in $\mathcal{ALC}$, which is a finite set of classical subsumption statements and defeasible subsumption statements, is quite immediate.

Example 6. *The knowledge base below is the PL version of the knowledge base presented in Example 2.*

$$\mathcal{K} = \{\mathsf{Penguin} \to \mathsf{Bird}, \mathsf{Robin} \to \mathsf{Bird}, \mathsf{SpecialPenguin} \to \mathsf{Bird}, \mathsf{Bird} \mathrel{|\!\sim} \mathsf{Fly},\\ \mathsf{Bird} \mathrel{|\!\sim} \mathsf{Wings}, \mathsf{Penguin} \mathrel{|\!\sim} \neg\mathsf{Fly}, \mathsf{SpecialPenguin} \mathrel{|\!\sim} \mathsf{Fly}\}$$

The notion of defeasible entailment used in the previous sections in this paper fully corresponds to rational closure in the propositional case, as shown by Britz et al. [23]. In other words, rational closure in the propositional case is a simplified version of rational closure in the DL $\mathcal{ALC}$ case. Similarly, the defeasible justification algorithm in the propositional case (Algorithm 4) is a simplified version of Algorithm 3. $\mathcal{K}^{\to}$ and $(\alpha \mathrel{|\!\sim} \beta)^{\to}$ are obtained substituting every instance of a defeasible conditional $\alpha \mathrel{|\!\sim} \beta$ with $\alpha \to \beta$.

Algorithm 4: ComputePropositionalDefeasibleJustifications($\mathcal{K}, \alpha \mathrel{|\!\sim} \beta$)

Input: Defeasible knowledge base $\mathcal{K}$ and query $\alpha \mathrel{|\!\sim} \beta$
Output: the set of justifications $\mathcal{J}$

$i \leftarrow 0$
$\mathcal{J}_{result} \leftarrow \emptyset$
$rank \leftarrow$ value i from RationalClosureForJustifications($\mathcal{K}, \alpha \mathrel{|\!\sim} \beta$)
if $rank = 0$ **then**
    $\mathcal{J} =$ ComputeAllJustifications($\mathcal{K}^{\to}, (\alpha \mathrel{|\!\sim} \beta)^{\to}$)
    return $\mathcal{J}$
$R = \{\mathcal{D}_0, \ldots\} \leftarrow R$ from ComputeRanking($\mathcal{K}$)
while $i < rank$ **do**
    $\mathcal{K}_{new} \leftarrow \mathcal{K}_{new} \setminus \mathcal{D}_i$
    $i = i + 1$
$\mathcal{J} =$ ComputeAllJustifications($\mathcal{K}^{\to}_{new}, (\alpha \mathrel{|\!\sim} \beta)^{\to}$)
return $\mathcal{J}$

Example 7. *If we run the algorithm with the knowledge base $\mathcal{K}$ from Example 6 and the query* $\mathsf{SpecialPenguin} \mid\sim \mathsf{Fly}$*, we obtain the result that corresponds to the one of Example 5, that is, the only justification is* $\{\mathsf{SpecialPenguin} \mid\sim \mathsf{Fly}\}$*, while the potential justification* $\{\mathsf{Penguin} \rightarrow \mathsf{Bird}, \mathsf{SpecialPenguin} \rightarrow \mathsf{Bird}, \mathsf{Bird} \mid\sim \mathsf{Fly}\}$ *is not considered since the rank of* $\mathsf{Bird} \mid\sim \mathsf{Fly}$ *is too low w.r.t. the query.*

Wang et al. [25] provide more detailed examples of defeasible justification in the propositional case.

The software that implements the defeasible justification algorithm for the KLM framework is coded in Java. We refer to the software tool as *KLMDefeasibleJustificationTool*. The *KLMDefeasibleJustificationTool* follows the Model View Controller (MVC) software architecture pattern and is composed of objects, a graphical user interface (GUI) and algorithm implementations. The GUI is constructed using Java Swing packages. The *KLMDefeasibleJustificationTool* takes two input parameters: a text file that consists of a KLM-style propositional knowledge base and a query string. The resulting justification(s) and statement rankings are displayed in a text area of the GUI. Two external packages are used to compute the defeasible justification: the Tweety Project [26, 27] and SAT4J SAT solver [28, 29]. The Tweety Project is a software framework that provides models and operations for a number of logics, including PL. Our implementation extends the PL models to construct models and operations required by the KLM Framework. Due to the constraints defined by the Tweety Project, we cannot denote the KLM Framework operations with conventional symbols. Instead, the negation symbol. $\neg$ is replaced with ! and the binary operations $\wedge, \vee, \rightarrow, \leftrightarrow$ and $\mid\sim$ are replaced with $\&\&, ||, =>, <=>$ and $\sim>$, respectively. Furthermore, we constructed a parser that reads strings such as "$\alpha \sim> \beta$" and produces an instance of *DefeasibleImplication*, which allows the software to perform operations such as materialisation. Classical entailment computation is needed as a sub-procedure to the defeasible justification calculations. The SAT4J SAT solver is used to compute this.

The source code of the defeasible justification tool for the KLM Framework can be found on Github [30].

5. Conclusion

The two main contributions of this paper are: (i) a defeasible justification algorithm for the DL $\mathcal{ALC}$ and (ii) an implementation of the defeasible justification algorithm in the propositional case. The obvious next step is to provide an implementation of the defeasible justification algorithm for $\mathcal{ALC}$ as a Protégé plugin. Further future work includes the definition and implementation of defeasible justification algorithms for other logics equipped with KLM-style defeasibility. One such logic is the restricted version of first-order logic with KLM-style defeasibility presented by Casini et al. [31]. Finally, here we have focused on a procedural definition of defeasible justification in the form of Algorithm 3. An investigation into a declarative definition of defeasible justification is left as future work.

Acknowledgments

This work has been partially supported by the H2020 STARWARS Project (GA No. 101086252), a type of action HORIZON TMA MSCA Staff Exchanges.

References

[1] M. Horridge, Justification based explanation in ontologies, PhD Thesis, University of Manchester, 2011.

[2] K. Britz, G. Casini, T. Meyer, K. Moodley, U. Sattler, I. Varzinczak, Principles of KLM-style Defeasible Description Logics, ACM Transactions on Computational Logic (TOCL) 22 (2020) 1 – 46. doi:10.1145/3420258.

[3] G. Casini, T. Meyer, K. Moodley, U. Sattler, I. Varzinczak, Introducing defeasibility into OWL ontologies, in: International Semantic Web Conference, Springer, 2015, pp. 409–426.

[4] S. P. Bail, The justificatory structure of OWL ontologies, PhD Thesis, University of Manchester, 2013.
[5] J. L. Pollock, A theory of defeasible reasoning, International Journal of Intelligent Systems 6 (1991) 33–54.
[6] S. Kraus, D. Lehmann, M. Magidor, Nonmonotonic reasoning, preferential models and cumulative logics, Artificial intelligence 44 (1990) 167–207.
[7] D. Lehmann, M. Magidor, What does a conditional knowledge base entail?, Artificial Intelligence 55 (1992) 1–60.
[8] L. Giordano, V. Gliozzi, N. Olivetti, G. L. Pozzato, Semantic characterization of rational closure: From propositional logic to description logics, Artificial Intelligence 226 (2015) 1–33. URL: https://doi.org/10.1016/j.artint.2015.05.001. doi:10.1016/J.ARTINT.2015.05.001.
[9] P. A. Bonatti, Rational closure for all description logics, Artificial Intelligence 274 (2019) 197–223. URL: https://doi.org/10.1016/j.artint.2019.04.001. doi:10.1016/J.ARTINT.2019.04.001.
[10] M. Schmidt-Schauß, G. Smolka, Attributive concept descriptions with complements, Artificial intelligence 48 (1991) 1–26.
[11] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi, The Description Logic Handbook: Theory, Implementation and Applications, 2 ed., Cambridge university press, 2007.
[12] F. Baader, R. Penaloza, Axiom pinpointing in general tableaux, Journal of Logic and Computation 20 (2010) 5–34.
[13] K. Moodley, T. Meyer, I. J. Varzinczak, A defeasible reasoning approach for description logic ontologies, in: Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, ACM, 2012, pp. 69–78.
[14] L. Botha, T. Meyer, R. Pĕnaloza, The Probabilistic Description Logic $\mathcal{BALC}$, Theory and Practice of Logic Programming (2020). doi:10.1017/s1471068420000460.
[15] P. Ke, U. Sattler, Next Steps for Description Logics of Minimal Knowledge and Negation as Failure, Description Logics 353 (2008).
[16] L. Giordano, V. Gliozzi, A reconstruction of multipreference closure, Artificial Intelligence 290 (2021) 103398. URL: https://doi.org/10.1016/j.artint.2020.103398. doi:10.1016/J.ARTINT.2020.103398.
[17] P. A. Bonatti, C. Lutz, F. Wolter, Description Logics with Circumscription., in: KR, 2006, pp. 400–410.
[18] P. A. Bonatti, M. Faella, I. M. Petrova, L. Sauro, A new semantics for overriding in description logics, Artificial Intelligence 222 (2015) 1–48. URL: https://doi.org/10.1016/j.artint.2014.12.010. doi:10.1016/J.ARTINT.2014.12.010.
[19] D. Lehmann, Another Perspective on Default Reasoning, Annals of Mathetics and Artificial Intelligence 15 (1995) 61–82. URL: https://doi.org/10.1007/BF01535841. doi:10.1007/BF01535841.
[20] G. Casini, U. Straccia, Defeasible Inheritance-Based Description Logics, J. Artificial Intelligence Res. 48 (2013) 415–473. URL: https://doi.org/10.1613/jair.4062. doi:10.1613/JAIR.4062.
[21] G. Casini, T. Meyer, I. Varzinczak, Taking Defeasible Entailment Beyond Rational Closure, in: F. Calimeri, N. Leone, M. Manna (Eds.), Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings, volume 11468 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 182–197. URL: https://doi.org/10.1007/978-3-030-19570-0_12. doi:10.1007/978-3-030-19570-0_12.
[22] G. Casini, U. Straccia, Rational closure for defeasible description logics, in: European Workshop on Logics in Artificial Intelligence, Springer, 2010, pp. 77–90.
[23] K. Britz, G. Casini, T. Meyer, I. Varzinczak, A KLM Perspective on Defeasible Reasoning for Description Logics, in: Description Logic, Theory Combination, and All That, Springer, 2019, pp. 147–173.
[24] L. Everett, E. Morris, T. Meyer, Explanation for klm-style defeasible reasoning, in: Southern African Conference for Artificial Intelligence Research, Springer, 2021, pp. 192–207.
[25] S. Wang, T. Meyer, D. Moodley, Defeasible Justification Using the KLM Framework, in: Southern African Conference for Artificial Intelligence Research, Springer, 2022, pp. 187–201.

[26] M. Thimm, Tweety - A Comprehensive Collection of Java Libraries for Logical Aspects of Artificial Intelligence and Knowledge Representation, in: Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR'14), 2014, pp. 528–537.
[27] M. Thimm, The Tweety Project, Available online, 2014. URL: https://tweetyproject.org/.
[28] N. Eén, N. Sörensson, An extensible SAT-solver, in: International conference on theory and applications of satisfiability testing, Springer, 2003, pp. 502–518.
[29] N. Eén, N. Sörensson, SAT4J SAT Solver, 2003. URL: https://www.sat4j.org/index.php.
[30] S. Wang, GitHub - KLM Defeasible Justification Tool, Available online, 2023. URL: https://github.com/SteveWang7596/DefeasibleJustificationForPropositionalLogic.
[31] G. Casini, T. Meyer, G. Paterson-Jones, I. Varzinczak, KLM-Style Defeasibility for Restricted First-Order Logic, in: International Joint Conference on Rules and Reasoning, Springer, 2022, pp. 81–94.