Introducing Defeasibility into OWL Ontologies

Giovanni Casini[1,2,4], Thomas Meyer[3,4], Kody Moodley[4,5],
Uli Sattler[6], and Ivan Varzinczak[4,7]

[1] University of Luxembourg, Luxembourg
[2] Department of Philosophy, University of Pretoria, South Africa
[3] Department of Computer Science, University of Cape Town, South Africa
[4] Centre for Artificial Intelligence Research, CSIR Meraka, South Africa
[5] School of Mathematics, Statistics, and Computer Science, University of KwaZulu-Natal, South Africa
[6] University of Manchester, United Kingdom
[7] Universidade Federal do Rio de Janeiro, Brazil

Abstract. In recent years, various approaches have been developed for representing and reasoning with exceptions in OWL. The price one pays for such capabilities, in terms of practical performance, is an important factor that is yet to be quantified comprehensively. A major barrier is the lack of naturally occurring ontologies with defeasible features - the ideal candidates for evaluation. Such data is unavailable due to absence of tool support for representing defeasible features. In the past, defeasible reasoning implementations have favoured automated generation of defeasible ontologies. While this suffices as a preliminary approach, we posit that a method somewhere in between these two would yield more meaningful results. In this work, we describe a systematic approach to modify real-world OWL ontologies to include defeasible features, and we apply this to the Manchester OWL Repository to generate defeasible ontologies for evaluating our reasoner DIP (Defeasible-Inference Platform). The results of this evaluation are provided together with some insights into where the performance bottle-necks lie for this kind of reasoning. We found that reasoning was feasible on the whole, with surprisingly few bottle-necks in our evaluation.

1 Introduction

Reasoning with *exceptions* has been a major topic in AI since the 80s. Classical *monotonic* formalisms such as OWL, assume that represented knowledge is infallible and do not admit exceptions; such systems generally cannot accommodate the addition of new information which contradicts what is known. For example, if a monotonic system is told that "*Students do not pay taxes*" then, upon encountering an exception (a student who works), it will still conclude that this student is exempt from taxes [10].

Defeasible reasoning is concerned with the development of formalisms which are able to represent and reason with defeasible (non-strict) facts: "Typically, *students do not pay taxes*" is the defeasible counterpart of "*Students do not pay taxes*".

Key approaches for defeasible reasoning in KR formalisms have been through adaptations and combinations of the following systems: Circumscription [4], Default Logic [22], Negation as failure [15], Probabilistic logic [17] and Preferential reasoning [6, 8, 10].

The theoretical foundation of our work is a Description Logic (DL) [1] adaptation of the preferential reasoning approach by Lehmann et al. [16]. DLs form the logical underpinning of OWL and so our approach is applicable in this setting as well.

The motivation for focusing on the preferential approach is that it derives intuitive inferences using procedures that reduce to classical OWL reasoning. This gives the advantage of being able to use "off-the-shelf" OWL reasoners such as FaCT++ (`owl.man.ac.uk/factplusplus`) and HermiT (`hermit-reasoner.com`), to perform defeasible inference. In particular, we have implemented a defeasible entailment regime called *Rational Closure* (RC) in our reasoner DIP (Defeasible-Inference Platform) [21]. This implementation is a variant of the one by Casini and Straccia [8].

However, there is a lack of insight into the expected practical performance of implementations such as DIP. A major barrier is the lack of tools for representing defeasibility in OWL, which in turn leads to the absence of naturally occurring data using defeasible features - the ideal candidates for testing performance. Currently, the majority of datasets for testing defeasible extensions of OWL, are automatically generated with the only mature attempt at a standardisation being LoDEN (`loden.fisica.unina.it`).

Our main goal in this paper is to take the next step from completely synthetic data, to a systematic approach for introducing defeasible features into naturally occurring ontologies that do not contain such features. We apply this approach to construct a dataset for evaluating our RC implementation in DIP. First, we introduce $\mathcal{ALC}$, the DL of choice for our implementation and a defeasible notion of subsumption that we introduce into the logic. We then give a concise description of RC for this logic and sketch the algorithms for computing the construction. Section 3 is the heart of the paper, here we detail a procedure for introducing *defeasible subsumption* into real-world ontologies and apply it to the Manchester OWL Repository to generate data for evaluating the performance of RC. We present the results and compare these with related work. Finally, we conclude by mentioning future work to be undertaken in the area.

2 Preliminaries

2.1 Description Logics

DLs are decidable fragments of first-order logic with a variety of applications, notably the formalisation of ontologies. They are very popular since they represent the logical underpinning of the Web Ontology Language (`w3.org/TR/owl-features`). In this paper we focus on $\mathcal{ALC}$, a representative member of the family of DLs, although our algorithms are applicable to a wide class of DLs, in particular $\mathcal{SHIQ}$ [14] .

Let $\mathsf{N}_{\mathscr{C}} = \{A_1, A_2, \ldots\}$ (resp. $\mathsf{N}_{\mathscr{R}} = \{r_1, r_2, \ldots\}$) be a finite set of *class names* (resp. *role names*) s.t. ($\mathsf{N}_{\mathscr{C}} \cap \mathsf{N}_{\mathscr{R}} = \emptyset$). The language, $\mathcal{L}$, of complex classes is:

$$C ::= A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists r.C \mid \forall r.C \mid \bot \mid \top$$

$\mathcal{ALC}$ has a standard set-theoretic semantics defined in the provided reference [1].

A classical DL ontology consists of a TBox $\mathcal{T}$ (and optionally an ABox $\mathcal{A}$), $\mathcal{T}$ contains the terminology describing the domain of discourse, i.e., $\mathcal{T}$ is a finite set of *inclusion axioms* $C \sqsubseteq D$; such an axiom is read as "C is subsumed by D", that is, every

individual that falls under the class C, falls also under the class D. $\mathcal{A}$ is a finite set of *instance axioms* (called assertions) of the form $C(a)$ or $R(a, b)$, where the former represents that a is an instance of the concept C, and the latter that a is related to b via the role R. $\mathcal{ALC}$ has a classical monotonic relation of entailment, and we use $\models$ to indicate this standard entailment relation, i.e., $\mathcal{T} \cup \mathcal{A} \models \alpha$ indicates that all the interpretations satisfying all the axioms contained in $\mathcal{T}$ and $\mathcal{A}$ also satisfy the axiom α. There are efficient tools for deciding $\mathcal{ALC}$ entailment. More details on DLs ($\mathcal{ALC}$ in particular [1]) and the relationship between OWL and DL [9] can be found in the provided references.

2.2 An algorithm to compute Rational Closure in $\mathcal{ALC}$

RC has a series of desirable properties from a formal perspective: the consequence relation has a solid logical foundation, is characterised by a set of structural properties that should be satisfied by any nonmonotonic formalism [5, 16], and its computation can be reduced to classical monotonic decision steps.

The applicability of RC to $\mathcal{ALC}$ is predicated on the ability to represent defeasible information. To model such information, we introduce a type of inclusion, i.e., a *defeasible inclusion* $C \tilde{\sqsubseteq} D$, which is read as "*Typically* an instance of C is also an instance of D", that is, if we know that an object x is in the set referred to by C, we can conclude that x is in the set referred to by D, unless we have knowledge to the contrary. For the semantics of such axioms, we refer the reader to the work by Britz et al. [5, 6].

We consider knowledge bases (KBs) of the form $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, where $\mathcal{T}$ is a DL TBox and $\mathcal{D}$ is known as a *defeasible TBox* (DTBox) which is a finite set of defeasible inclusions. We are not considering ABoxes here - the algorithm we introduce (based on the one by Casini and Straccia [8]), computes RC only considering classes. Given $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, it decides if $C \tilde{\sqsubseteq} D$ or $C \sqsubseteq D$ is a defeasible consequence of $\mathcal{K}$.

Example 1. Consider the KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, with $\mathcal{T} = \{\mathsf{BactMen} \sqsubseteq \mathsf{Men}, \mathsf{VirMen} \sqsubseteq \mathsf{Men}\}$ and $\mathcal{D} = \{\mathsf{Men} \tilde{\sqsubseteq} \neg\mathsf{Fatal}, \mathsf{BactMen} \tilde{\sqsubseteq} \mathsf{Fatal}\}$. $\mathcal{K}$ is about meningitis (Men), bacterial meningitis (BactMen), viral meningitis (VirMen), and their fatality (Fatal).

If all axioms in $\mathcal{K}$ were classical inclusion axioms, we derive $\mathsf{BactMen} \sqsubseteq \bot$ because the facts lead to bacterial meningitis being both fatal ($\mathsf{BactMen} \sqsubseteq \mathsf{Fatal}$) and non-fatal ($\mathsf{BactMen} \sqsubseteq \mathsf{Men}$, $\mathsf{Men} \sqsubseteq \neg\mathsf{Fatal}$). We would rather "relax" some strict facts to cater for the atypicality of BactMen (meningitis is usually not fatal, but bacterial meningitis is an exceptional type of meningitis because it usually *is* fatal).

We shall indicate the set of *materialisations* of the axioms in $\mathcal{D}$ by $\overline{\mathcal{D}}$, where the *materialisation* of an axiom $C \tilde{\sqsubseteq} D$ denotes the class expressing the same subsumption relation of the axiom (i.e., $\neg C \sqcup D$). Hence $\overline{\mathcal{D}} = \{\neg C \sqcup D \mid C \tilde{\sqsubseteq} D \in \mathcal{D}\}$.
The *classical translation* of $C \tilde{\sqsubseteq} D$ is $C \sqsubseteq D$. Similarly, for a set $\mathcal{D} = \{C_1 \tilde{\sqsubseteq} D_1$, $...,C_n \tilde{\sqsubseteq} D_n\}$, the classical translation of $\mathcal{D}$ is $\mathcal{D}' = \{C_1 \sqsubseteq D_1, ..., C_n \sqsubseteq D_n\}$.

The RC algorithm consists of a main procedure and two sub-procedures. The first sub-procedure is called Exceptional. Its aim is to determine which of the left-hand side (LHS) classes in our inclusions are *exceptional*. Intuitively, a class is exceptional in a KB if it is atypical w.r.t. one of its superclasses (*e.g.* BactMen in the example above). The exceptionality of a class can be decided using $\models$, since a class C is exceptional

in $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$ if and only if $\mathcal{T} \models \bigsqcap \overline{\mathcal{D}} \sqsubseteq \neg C$. A defeasible axiom $C \mathrel{\sqsubset\!\!\sim} D \in \mathcal{D}$ is considered exceptional if its antecedent (LHS-concept) C is exceptional. Given a finite set $\mathcal{E}$ of defeasible inclusion axioms, Procedure Exceptional gives back the subset of $\mathcal{E}$ containing the exceptional axioms.

Procedure Exceptional($\mathcal{T}, \mathcal{E}$)

Input: $\mathcal{T}, \mathcal{E} \subseteq \mathcal{D}$
Output: $\mathcal{E}' \subseteq \mathcal{E}$ such that $\mathcal{E}'$ is exceptional w.r.t. $\mathcal{E}$

$\mathcal{E}' := \emptyset$;
foreach $C \mathrel{\sqsubset\!\!\sim} D \in \mathcal{E}$ **do**
  if $\mathcal{T} \models \bigsqcap \overline{\mathcal{E}} \sqsubseteq \neg C$ **then**
    $\mathcal{E}' := \mathcal{E}' \cup \{C \mathrel{\sqsubset\!\!\sim} D\}$;
return $\mathcal{E}'$;

Since, in general, there may be "exceptions-to-exceptions" (perhaps a strain of bacterial meningitis that is usually *not* fatal), we can compute this *exceptionality ranking* by recursive application of Procedure Exceptional. I.e., we associate a value to each axiom in the KB representing its degree of exceptionality. ComputeRanking is the second sub-procedure that, using Exceptional, partitions the set $\mathcal{D}$ into $\mathcal{R} = \{\mathcal{D}_0, \mathcal{D}_1, \ldots\}$, where each set $\mathcal{D}_i$ contains the defeasible axioms having i as ranking value.

Procedure ComputeRanking($\mathcal{K}$)

Input: KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$
Output: KB $\langle \mathcal{T}^*, \mathcal{D}^* \rangle$ and the partitioning (ranking) $\mathcal{R} = \{\mathcal{D}_0, \ldots, \mathcal{D}_n\}$ for $\mathcal{D}^*$

$\mathcal{T}^* := \mathcal{T}$; $\mathcal{D}^* := \mathcal{D}$; $\mathcal{R} := \emptyset$;
repeat
  $i := 0$; $\mathcal{E}_0 := \mathcal{D}^*$;
  $\mathcal{E}_1 :=$ `Exceptional`$(\mathcal{T}^*, \mathcal{E}_0)$;
  while $\mathcal{E}_{i+1} \neq \mathcal{E}_i$ **do**
    $i := i + 1$; $\mathcal{E}_{i+1} :=$ `Exceptional`$(\mathcal{T}^*, \mathcal{E}_i)$;
  $\mathcal{D}^*_\infty := \mathcal{E}_i$; $\mathcal{T}^* := \mathcal{T}^* \cup \{C \sqsubseteq D \mid C \mathrel{\sqsubset\!\!\sim} D \in \mathcal{D}^*_\infty\}$; $\mathcal{D}^* := \mathcal{D}^* \setminus \mathcal{D}^*_\infty$;
until $\mathcal{D}^*_\infty = \emptyset$;
for $j = 1$ *to* i **do**
  $\mathcal{D}_{j-1} := \mathcal{E}_{j-1} \setminus \mathcal{E}_j$; $\mathcal{R} := \mathcal{R} \cup \{\mathcal{D}_{j-1}\}$;
return $\langle \mathcal{T}^*, \mathcal{D}^* \rangle, \mathcal{R}$;

ComputeRanking receives KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$ as input and outputs an *equivalent* KB $\mathcal{K}^* = \langle \mathcal{T}^*, \mathcal{D}^* \rangle$ (*i.e.*, satisfied in the same interpretations as $\mathcal{K}$ [5]), but in which *implicit* strict knowledge contained in the DTBox has been moved to the TBox, and all the information in the DTBox has been ranked w.r.t. its exceptionality. We call axioms which are implicitly strict and yet concealed in the DTBox, *totally exceptional* axioms. Consider the KBs: $\mathcal{K}_1 = \{C \mathrel{\sqsubset\!\!\sim} D, C \mathrel{\sqsubset\!\!\sim} \neg D\}$ and $\mathcal{K}_2 = \{E \mathrel{\sqsubset\!\!\sim} D, C \sqsubseteq E, C \mathrel{\sqsubset\!\!\sim} \neg D\}$. C is unsatisfiable w.r.t. $\mathcal{K}'_1$ and $\mathcal{K}'_2$. In $\mathcal{K}'_1$, the unsatisfiability does not indicate an

exception, whereas in $\mathcal{K}'_2$, it does. The former case is a knowledge engineering problem, indicating logical incoherence in defeasible KBs, while the latter is not considered such.

Sub-procedure Exceptional reaches a fixed point $\mathcal{E}_i = \mathcal{E}_i + 1$ ($\mathcal{E}_i$ is possibly empty) during Procedure ComputeRanking. If $\mathcal{E}_i$ is not empty, then it represents a set of totally exceptional axioms, and we assign ∞ as the ranking value to each of these axioms (indicated by $\mathcal{D}^*_\infty$). We move such information to the TBox (that is, if $C \,\tilde{\sqsubseteq}\, D$ is in $\mathcal{D}^*_\infty$, we eliminate it from $\mathcal{D}$ and add $C \sqsubseteq \bot$ to $\mathcal{T}^*$). We repeat the procedure (Lines 2-8) until *all* implicit strict facts in $\mathcal{D}$ are moved to the TBox. Consider the example:

Example 2. Consider $\mathcal{K} = \langle \mathcal{T}, \mathcal{D} \rangle$, with $\mathcal{T} = \{E \sqsubseteq D\}$ and $\mathcal{D} = \{F \,\tilde{\sqsubseteq}\, \exists r.C, C \,\tilde{\sqsubseteq}\, \neg D, C \,\tilde{\sqsubseteq}\, E\}$. Applying Procedure ComputeRanking, we obtain that C is exceptional, i.e., $\mathcal{E}_2 = \mathcal{E}_1 = \{C \,\tilde{\sqsubseteq}\, \neg D, C \,\tilde{\sqsubseteq}\, E\}$. This means that $\mathcal{D}^*_\infty = \{C \,\tilde{\sqsubseteq}\, \neg D, C \,\tilde{\sqsubseteq}\, E\}$, and therefore, $\mathcal{T}^* = \{E \sqsubseteq D, C \sqsubseteq \bot\}$ and $\mathcal{D}^* = \{F \,\tilde{\sqsubseteq}\, \exists r.C\}$. Repeating Lines 2-8 of the procedure we get $\mathcal{E}_1 = \mathcal{E}_0 = \{F \,\tilde{\sqsubseteq}\, \exists r.C\}$, hence $\mathcal{D}^*_\infty = \{F \,\tilde{\sqsubseteq}\, \exists r.C\}$ and we end up with a TBox $\mathcal{T}^* = \{E \sqsubseteq D, C \sqsubseteq \bot, F \sqsubseteq \bot\}$ and an empty DTBox.

Once the procedure has identified $\langle \mathcal{T}^*, \mathcal{D}^* \rangle$, it ranks the remainder axioms in the DTBox (if any): an axiom has ranking value i if i is the highest label for which it turns out to be exceptional. The result is a partition of $\mathcal{D}$ into $\mathcal{R} = \{\mathcal{D}_0, \ldots, \mathcal{D}_n\}$.

Example 3. Consider $\mathcal{K}$ in Example 1. ComputeRanking takes $\mathcal{K}$ as input, executes Lines 2 - 8 to obtain the sequence $\mathcal{E}_0 = \mathcal{D}$, $\mathcal{E}_1 = \{\mathsf{BactMen} \,\tilde{\sqsubseteq}\, \mathsf{Fatal}\}$, $\mathcal{E}_2 = \emptyset$ and the TBox $\mathcal{T}^* = \mathcal{T}$. Finally, applying Lines 9 - 10, we obtain the partition $\mathcal{D}^*$ of $\mathcal{D}_0 = \{\mathsf{Men} \,\tilde{\sqsubseteq}\, \neg\mathsf{Fatal}\}$, $\mathcal{D}_1 = \{\mathsf{BactMen} \,\tilde{\sqsubseteq}\, \mathsf{Fatal}\}$.

Given our computed ranking, we can ask queries of the form $C \,\tilde{\sqsubseteq}\, D$. Note that if we are confronted with a strict query (classical inclusion axiom $C \sqsubseteq D$), one can determine if it is in the RC of the KB by checking if it is classically entailed by the strict facts alone in $\mathcal{K}$ (that is, $\mathcal{T}^*$). This was implemented as an optimisation.

Procedure RationalClosure($\mathcal{K}$,δ)

Input: KB $\mathcal{K} = \langle \mathcal{T}^*, \mathcal{D}^* \rangle$ with no implicit strict facts, $\mathcal{E}_0, \ldots, \mathcal{E}_n$, query $\delta = C \,\tilde{\sqsubseteq}\, D$.
Output: `true` iff $C \,\tilde{\sqsubseteq}\, D$ is in the RC of $\mathcal{K}$

1 $i := 0$;
2 **while** $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_i} \sqcap C \sqsubseteq \bot$ *and* $i \leq n$ **do**
3 $\quad i := i + 1$;
4 **if** $i \leq n$ **then**
5 $\quad$ **return** $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_i} \sqcap C \sqsubseteq D$;
6 **else**
7 $\quad$ **return** $\mathcal{T}^* \models C \sqsubseteq D$;

However, for simplicity, Algorithm RationalClosure considers only the case in which the query is a defeasible inclusion axiom. The algorithm takes the ranking $\mathcal{R}$ and query $C \,\tilde{\sqsubseteq}\, D$ as input, determines which portion of $\mathcal{R}$ is compatible with the class C, i.e., which portion of defeasible knowledge does not imply that C is exceptional, starting from the most normal situations up to increasing levels of exceptionality. By example:

Example 4. Consider the ranking $\mathcal{R}$ in Example 3 and the query $\mathsf{VirMen} \mathrel{\tilde{\sqsubset}} \neg\mathsf{Fatal}$. The RC algorithm checks if $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_0} \sqsubseteq \neg\mathsf{VirMen}$, which is not the case. Hence, we have to check if $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_0} \sqcap \mathsf{VirMen} \sqsubseteq \neg\mathsf{Fatal}$, which is true. However, if our query is $\mathsf{BactMen} \mathrel{\tilde{\sqsubset}} \neg\mathsf{Fatal}$, we obtain a different result: since $\mathcal{T}^* \models \bigsqcap \overline{\mathcal{E}_0} \sqsubseteq \neg\mathsf{BactMen}$ but $\mathcal{T}^* \not\models \bigsqcap \overline{\mathcal{E}_1} \sqsubseteq \neg\mathsf{BactMen}$, $\mathsf{BactMen}$ is an exceptional class of level 1, compatible with $\mathcal{D}_1$, and we have to move one level higher in the ranking eliminating the facts in $\mathcal{D}_0$ from consideration. It turns out that $\mathcal{T}^* \not\models \bigsqcap \overline{\mathcal{E}_1} \sqcap \mathsf{BactMen} \sqsubseteq \neg\mathsf{Fatal}$, and that is the right conclusion since we have $\mathsf{BactMen} \mathrel{\tilde{\sqsubset}} \mathsf{Fatal}$ in our KB.

The correctness of Algorithm RationalClosure follows from the procedure by Casini and Staccia [8] as it is a reformulation thereof. The computational complexity of the entire procedure is the same as that of the underlying monotonic entailment relation $\models$, i.e., it is an EXPTIME-complete problem ([5] and [8, Corollary 2]). Moreover, note that the defined procedures can be applied to DLs more expressive than $\mathcal{ALC}$, still preserving the computational complexity of the decision problem w.r.t. the underlying monotonic entailment relation, and, is still sound and complete for logics up to $\mathcal{SHIQ}$. Using a more expressive DL than $\mathcal{ALC}$, the defeasible information will still be represented only by defeasible inclusion axioms $C \mathrel{\tilde{\sqsubset}} D$, while the strict information different from $\mathcal{ALC}$ inclusion axioms (role inclusion axioms, role transitivity, etc.) must be considered as background knowledge at each step of the decision procedure.

3 Performance Evaluation

An important question about RC is: how much do we pay for the additional expressivity, in terms of practical reasoning performance? We have shown that the worst case computational complexity of RC in $\mathcal{ALC}$ is not higher than reasoning with classical $\mathcal{ALC}$. This is good news, but does not guarantee good performance in practice. As illustrated in our algorithms (Section 2.2), we have to perform some additional computation over and above the classical decision checks. In general, we perform multiple classical entailment checks to answer a single defeasible entailment question. The question is how much more work are we doing. We aim to investigate this in order to provide evidence of the feasibility of adding defeasible features to ontologies.

In terms of data, the norm until now has been the use of automatically generated ontologies with defeasible features (the most notable attempt at a benchmark of synthetic defeasible ontologies is LoDEN). Indeed, we have also used synthetic data in the past as a preliminary indicator of performance [7]. Naturally, there are obvious shortcomings with such an approach, such as possible biases in the ontology generation methodology. However, there is no question of finding *representative* data because there are virtually no naturally occurring ontologies with intended defeasible features.

We instead choose a middle-ground approach, taking advantage of the rich set of (classical) OWL ontologies on the Web in various repositories and corpora. The basic idea is to modify selected subsumptions in these ontologies to be *defeasible*, thereby making them useful as data to evaluate our defeasible reasoner. Of course, this has to be done with care to generate cases which are challenging for the reasoner. For example, we need to ensure that there are cases where there are multiple ranks in the ranking of

the ontology (see Procedure ComputeRanking). Our method is described in Section 3.2, together with a discussion about its strengths and weaknesses. Now, we describe the curation process used for sampling our initial set of *unmodified* OWL ontologies.

3.1 Non-defeasible Dataset

For our initial data, we sample some classical OWL ontologies which we can later pass through our procedure for the introduction of defeasible features. The natural choice is to select the same data that is traditionally used to evaluate the performance of existing *classical* OWL reasoners. However, even in such a setting, there is no precise concensus on what data to use. The result is that data is generally curated manually by choosing "well-known" ontologies and corpora from which to sample, or arbitrarily selecting from the variety of respectable corpora on the web.

Choice of corpora: While there are bona fide ontology benchmarks available such as LUBM [11] and its extensions, it was pointed out that there are shortcomings in manual selection of ontologies and ontology corpora for evaluation [18]. In particular, the main limitation with such selections is that they lack sufficient *variety*. Thus the results of evaluations can be heavily biased towards the selected benchmarks. The Manchester OWL Repository [19] is an effort to address this issue. The Repository is a framework for sharing ontology datasets for OWL empirical research. The current version of the repository contains three core datasets, namely versions of NCBO Bioportal (`bioportal.bioontology.org`), The Oxford Ontology Library or OOL (`cs.ox.ac.uk/isg/ontologies`) and MOWLCorp [18]. While Bioportal and OOL are established corpora actively used in reasoner evaluations, MOWLCorp is a recent gathering of ontologies through sophisticated web crawls and filtration techniques [18].

We obtain a recent snapshot of the Manchester OWL Repository as the base dataset for our evaluation. There are 344, 793 and 20,996 ontologies in the Bioportal, OOL and MOWLCorp corpora respectively.

Filtration Process and Choice of OWL Reasoner: For loading and analysing ontologies of our dataset, we use the popular and well-supported Java-based OWL API [13]. The API contains parsers for a wide variety of different syntaxes of ontologies such as RDF, Turtle and OBO. As we have shown in Section 2.2, our algorithms are built upon classical entailment checks. Thus, we would need to select an existing OWL 2 DL reasoning implementation to perform these classical entailment checks from within our defeasible reasoner. While running our evaluation with multiple reasoners would have been interesting, such an investigation is not necessary to ascertain the price we pay for reasoning with defeasible (in addition to classical) subsumption. We chose to utilise a single OWL 2 DL reasoner for our evaluation. In particular, we would ideally like to use the fastest and most robust implementation.

Consulting the latest results of the OWL Reasoner Evaluation Workshop (`dl.kr.org/ore2014/results.html`), we identified the top three OWL 2 DL reasoners for the standard reasoning tasks of: *classification*, *consistency checking* and *satisfiability testing* (in terms of performance and robustness). Robustness was measured as the number of ontologies that were successfully processed in the allotted time. The top reasoners were Konclude (`derivo.de/produkte/konclude.html`), HermiT, MORe (`cs.ox.ac.uk/isg/tools/MORe`), Chainsaw (`chainsaw.sourceforge.net`), FaCT++ and

TrOWL (`trowl.org`). As we shall see in Section 3.2, we require to check incoherence of ontologies before introducing defeasible subsumptions into them. Modern OWL reasoners are optimised for classification (computing the subsumption relationship between each pair of class names in the ontology), and identifying unsatisfiable class names (incoherence) is usually performed by first classifying the ontology, and then "reading" the unsatisfiable class names from the results. Thus, we chose to focus on the reasoners which performed best in OWL 2 DL classification. These were respectively, Konclude, HermiT and MORe. Konclude, unfortunately, does not yet support the OWL API. Therefore, our choice was to select the next best reasoner - HermiT.

Given our choice of tools for manipulating and reasoning with the ontologies in our dataset, we filtered out the ontologies that could be loaded and parsed by the OWL API (each within an allotted 40 minutes). The resulting ontologies were then tested to determine if they were classifiable by HermiT within an additional 40 minutes each. Ontologies which did not pass this test were removed from the data. In order to remove some of the cases which are very likely to be easy for our reasoner, we elected to remove ontologies with less than 100 logical axioms (ignoring annotations and other axioms carrying meta-information). This is justifiable because ontology size is proven to be an overwhelmingly dominant factor in reasoning performance [24]. Finally, we stripped the ontologies of ABox data because our defeasible reasoner is currently purely equipped with (D)TBox entailment procedures. This leaves us with 252, 440 and 2335 ontologies in Bioportal, OOL and MOWLCorp respectively.

3.2 Defeasible Dataset

In this section, we describe a systematic technique to introduce defeasible subsumptions into ontologies, thereby making them amenable to defeasible reasoning evaluation.
Methodology: Our approach hinges upon an important correspondence between *class exceptionality* (as described in Section 2.2) and classical class unsatisfiability:

Lemma 1. *If a class C is exceptional w.r.t. a defeasible KB $\langle \mathcal{T}, \mathcal{D} \rangle$ then C is unsatisfiable w.r.t. $\mathcal{T} \cup \mathcal{D}'$, where $\mathcal{D}'$ is the classical translation of $\mathcal{D}$.*

Lemma 1 states that if a class is exceptional in a defeasible ontology then it will necessarily be unsatisfiable in the classical translation of the ontology. This result is useful because we can use it to identify possible exceptional classes in classical ontologies. Taking the contrapositive of Lemma 1, we obtain the result that if a class is satisfiable w.r.t. a classical ontology then it is necessarily *not* exceptional w.r.t. any defeasible translation of the ontology. Therefore, we can eliminate ontologies from our dataset without LHS-classes of subsumptions that are unsatisfiable, because these could never become exceptional by turning classical subsumption into defeasible ones.

The following definition is a generalisation of standard incoherence to axioms with complex left hand side (LHS) concepts:

Definition 1. *A classical TBox $\mathcal{T}$ is* LHS-coherent *if each $C \sqsubseteq D \in \mathcal{T}$ is s.t. $\mathcal{T} \not\models C \sqsubseteq \bot$. $\mathcal{T}$ is* LHS-incoherent *if it is not LHS-coherent.*

Corpus	Classes			Roles			TBox size			RBox Size			Nested Classes		Conjuncts		Disjuncts	
	avg	median	max	avg	median	max	avg	median	max	avg	median	max	max	avg	max	avg	max	avg
Bioportal	17992	422	187515	50	33	152	41309	754	439208	30	17	149	28	1	10	2	5	2.1
OOL	22203	16306	89926	55	44	194	41389	32928	142996	27	9	123	136	1	68	2	16	2.5
MOWLCorp	4621	216	89926	466	13.5	16586	8719	691	137021	214	5	7749	34	1	17	2	16	2.7

Fig. 1: Ontology metrics for the LHS-incoherent cases in the dataset.

Eliminating all ontologies from our dataset that are LHS-coherent leaves us with 11, 46 and 77 ontologies in the Bioportal, OOL and MOWLCorp corpora respectively. Figure 1 provides some average properties of the ontologies in our dataset.

Thus, in total we have 134 ontologies for our performance evaluation. Now, the task is to relax some of the subsumptions of our ontologies to be defeasible. The obvious naïve approach to introducing defeasibility would be to convert *all* subsumptions to defeasible ones. Naturally, this is not likely to be the general approach of defeasible-ontology engineers in practice. The other extreme would be to develop an approach to identify the *minimal* (for some defined notion of minimality) amount of defeasibility to introduce into the ontology in order to successfully "cater for all the exceptions". The latter approach would be ideal, and we are currently investigating such an approach; however, we propose that a reasonable approximation of such an "ideal" procedure yields meaningful data for *performance* evaluation. The approach that we discuss here is in the spirit of such an approximation.

Example 5. Consider the following TBox $\mathcal{T}$ about different types of mechanics (Mech), general (GenMech), car (CarMech) and mobile (MobileMech):
$\{1.\ \mathsf{Mech} \sqsubseteq \exists \mathsf{hasWorkshop}.\top,\ 2.\ \mathsf{Mech} \sqsubseteq \exists \mathsf{hasSpecialisation}.\top,$
$3.\ \mathsf{MobileMech} \sqcup \mathsf{GenMech} \sqcup \mathsf{CarMech} \sqsubseteq \mathsf{Mech},\ 4.\ \mathsf{MobileMech} \sqsubseteq \neg\exists \mathsf{hasWorkshop}.\top,$
$5.\ \mathsf{MobileMech} \sqcap \neg\exists \mathsf{status}.\mathsf{OnStandBy} \sqsubseteq \exists \mathsf{hasWorkshop}.\top,$
$6.\ \mathsf{GenMech} \sqsubseteq \neg\exists \mathsf{hasSpecialisation}.\top,\ 7.\ \mathsf{CarMech} \sqsubseteq \exists \mathsf{hasSpecialisation}.\mathsf{Car}\}$

The classes MobileMech, GenMech and the class expression $\mathsf{MobileMech} \sqcap \neg\exists \mathsf{status}.\mathsf{OnStandBy}$ are unsatisfiable w.r.t. $\mathcal{T}$. An intuitive analysis of $\mathcal{T}$ tells us that the ontology engineer probably intended to model that mechanics *usually* have a workshop ($\mathsf{Mech} \mathrel{\tilde{\sqsubseteq}} \exists \mathsf{hasWorkshop}.\top$) and *usually* specialise in certain types of equipment that they repair ($\mathsf{Mech} \mathrel{\tilde{\sqsubseteq}} \exists \mathsf{hasSpecialisation}.\top$). This translation of Axioms 1 and 2 in Example 5, is a minimal and intuitive way to introduce defeasibility into $\mathcal{T}$, catering for exceptional types of mechanic - i.e., mobile and general mechanics.

However, we also have an exceptional type of mobile mechanic in $\mathcal{T}$ (an "exception-to-an-exception"). That is, mobile mechanics who are no longer "on standby" or "on call" ($\mathsf{MobileMech} \sqcap \neg\exists \mathsf{status}.\mathsf{OnStandBy}$). These mechanics would then be assigned a workshop for their repair tasks. To cater for such mechanics we would have to relax Axiom 4 of Example 5 as well, to express that mobile mechanics *usually* don't have a workshop ($\mathsf{MobileMech} \mathrel{\tilde{\sqsubseteq}} \neg\exists \mathsf{hasWorkshop}.\top$).

We now define a general *defeasible translation function* (DTF) for converting classical subsumptions to defeasible subsumptions in classical ontologies.

Definition 2. *(DTF) Let $\mathcal{T}$ be a set of classical subsumptions of the form $C \sqsubseteq D$, then $\mathscr{F} : \mathcal{T} \rightarrow \{C \tilde{\sqsubseteq} D \mid C \sqsubseteq D \in \mathcal{T}\} \cup \mathcal{T}$ is a* DTF *for $\mathcal{T}$.*

We also have to formalise what we mean when a particular DTF "caters for all exceptions" in the TBox. We call such a function a *safe* DTF.

Definition 3. *(safe DTF) Let $\mathcal{T}$ be a set of classical subsumptions, let $\mathscr{F}$ be a DTF for $\mathcal{T}$ and let $\mathscr{D}$ be the special DTF that translates* all *subsumptions in $\mathcal{T}$ to defeasible ones. Then, $\mathscr{F}$ is a* safe DTF *for $\mathcal{T}$ if C is totally exceptional w.r.t. $\mathscr{D}(\mathcal{T})$ if and only if C is totally exceptional w.r.t. $\mathscr{F}(\mathcal{T})$, for each $C \sqsubseteq D \in \mathcal{T}$.*

We define a safe DTF that places a small upper bound on the subset of axioms to relax using the well-known notion of *justification* [12]. A justification for an entailment α of an ontology is a minimal (w.r.t. set inclusion) subset of the ontology that entails α. If we compute the justifications for $\mathcal{T} \models \mathsf{MobileMech} \sqsubseteq \bot$ (concise reasons for MobileMech being unsatisfiable and possibly exceptional) we obtain a single justification $\{1, 3, 4\}$. Relaxing these axioms is sufficient for catering for mobile mechanics (in fact, it is only necessary to relax Axiom 1 as mentioned earlier). Similarly, we arrive at $\{2, 3, 6\}$ to cater for general mechanics and $\{4, 5\}$ for mobile mechanics no longer on call.

The basic idea is thus to take the union of the justifications for the unsatisfiable LHS-classes and relax these axioms to defeasible ones. We obtain that $\{1, 2, 3, 4, 5, 6\}$ should be relaxed in Example 5, which is admittedly a large proportion of our TBox. However, as we discover in Section 3.4, the proportion is much smaller in practice on larger real-world ontologies. However, while computing all justifications has been shown to be feasible in general on real-world ontologies, *black-box* (reasoner-independent) procedures are known to be exponential in the worst case [12]. To avoid this potential computational blowup, we obtain a small upper bound of the union of justifications by extracting a *star locality based module* [23] for the ontology in question, w.r.t. the set of unsatisfiable LHS-classes. A module of an ontology w.r.t. a signature (set of terms from the ontology) is a small subset of the ontology that preserves the meaning of the terms in the signature. We specifically choose star locality based modules because of two key properties: (i) they preserve all justifications in the ontology for all entailments (or axioms) that can be constructed with the given signature (*depleting* property [23, Section 3]), and (ii) they are smaller in size relative to other modules which have the depleting property. The pseudocode of our procedure is given in Algorithm 1.

Algorithm 1: *relaxSubsumption*

Input: LHS-incoherent TBox $\mathcal{T}, \mathcal{C} = \{C \mid (C \sqsubseteq D \in \mathcal{T} \text{ for some } D) \wedge (\mathcal{T} \models C \sqsubseteq \bot)\}$
Output: Defeasible ontology $\langle \mathcal{T}, \mathcal{D} \rangle$

$\mathcal{T} := \emptyset$; $\mathcal{D} := \emptyset$; $\mathcal{M} := extractStarModule(\mathcal{O}, sig(\mathcal{C}))$; $\mathcal{T} := \mathcal{O} \backslash \mathcal{M}$;
foreach $X \sqsubseteq Y \in \mathcal{M}$ **do**
    $\mathcal{D} := \mathcal{D} \cup \{X \tilde{\sqsubseteq} Y\}$;
return $\langle \mathcal{T}, \mathcal{D} \rangle$;

Theorem 1. *(safety of our DTF) Let $\mathscr{F}$ be the DTF defined by Algorithm 1 and let $\mathcal{T}$ be a set of classical subsumptions, then $\mathscr{F}$ is a safe DTF for $\mathcal{T}$.*

Discussion: There are two conflicting issues with the procedure we have presented for introducing defeasibility into OWL ontologies: (i) minimality of modification to the original ontology and (ii) the representative quality of the resulting defeasible ontology as something that might be built by a ontology engineer. While (i) and (ii) would be the ultimate goal for a methodology automating the introduction of defeasible features into OWL ontologies, our approach does not yet meet such desiderata. It is clear that the minimal axioms to relax in Example 5 would be $\{1, 2, 4\}$, yet we relax $\{3, 5, 6\}$ as well. On a related note, relaxing $\{1, 2, 3, 4, 5, 6\}$ does not capture a "natural" defeasible translation. For instance, it does not make sense, intuitively, to relax $\mathsf{MobileMech} \sqsubseteq \mathsf{Mech}$ (all mobile mechanics are mechanics) to $\mathsf{MobileMech} \mathrel{\tilde{\sqsubseteq}} \mathsf{Mech}$ (*typical* mobile mechanics are mechanics). Such constraints should ideally remain strict.

Furthermore, a critical observation is that incoherence in classical ontologies may be caused by erroneous modelling. In ontology development tools, large emphasis has been placed on debugging incoherence by making modifications to the ontology to remove the "unwanted" entailments such as $C \sqsubseteq \bot$. This is likely to have prevented many developers publishing incoherent ontologies.

Given the above main shortcomings of our approach, we do not argue that our approach is the ideal methodology. Rather, we hope that it serves as a stepping stone from purely synthetic approaches to investigate and develop more suitable methodologies.

Hypotheses: Our general predictions for the evaluation are that (i) the ranking computation will be dramatically more performance intensive than testing entailment, (ii) entailment testing will be feasible for on-demand use and (iii) the number of incoherent LHS-classes (and the number of defeasible subsumptions) will affect the performance significantly, (iv) we anticipate the occurrences of totally exceptional LHS-classes to be rare and minimal, (v) since these cases also require recursive execution of the ranking procedure, we also anticipate such cases to be significantly harder for reasoning and (vi) in terms of the ranking of the defeasible subsumptions in the ontologies, we expect there to be not more than 2 levels of exceptionality (or 3 ranks in total). I.e., we expect exceptions-to-exceptions in the data, although we anticipate very few of these cases. We expect the majority of cases to have either no exceptions or 1 level of exception (2 ranks in total). Of course, we also predict a general trend of the higher the number of (logical) axioms in the ontology, the longer to compute inferences.

3.3 Experiment Setup

Our setup, methodologies and design choices for the experimental evaluation can be summarised as follows:

Data summary: The input data for our experiments are 134 LHS-incoherent ontologies (curated as described in Section 3.1) from the Manchester OWL Repository. The ontologies are divided across three corpora: 11 , 46 and 77 in Bioportal, OOL and MOWL-Corp respectively. The average ratio of defeasible to strict axioms in each ontology is 8%, the median being 1.5%, the minimum ratio being 0.01% and the maximum being 98%. The DL expressivity distribution of the data ranges from variants of $\mathcal{ALC}$ all the way up to $\mathcal{SROIQ}$. There are 35 DL variants in total represented in the data.

Additionally, we generated a set of entailment queries (defeasible subsumptions of the form $C \mathrel{\tilde{\sqsubseteq}} D$ and strict subsumptions of the form $C \sqsubseteq D$) for each ontology to

present to our defeasible reasoner. For the C's, we focus on all LHS-incoherent classes in the ontology. The motivation is two-fold: (i) if we instead focus on C's that are satisfiable, we would not require execution of Lines 2-3 of Algorithm RationalClosure in Section 2.2 because C could never be exceptional (see Lemma 1). Thus, we focus on incoherent C's since these are the only ones which could possibly be exceptional and result in harder cases for reasoning. Instead of generating such incoherent C's, we use the existing LHS-concepts that are unsatisfiable in the ontology as a preliminary strategy. Admittedly, generating incoherent C's might also be interesting for future evaluation.

For the D's we first take the $\bot$-syntactic locality module for the ontology w.r.t. to the signature of C (including $\bot$), and then take all nested class expressions present in the axioms of this module. The reason being that we want to preserve entailments over the signature of C in the module. We collect all LHS-incoherent classes C from the ontology and then collect all class expressions in the $\bot$-module for C to be the consequents D. We then test if $C \sqsubseteq\!\!\!\sim D$ (and $C \sqsubseteq D$) is in the RC of the defeasible ontology. All our data is available for download in ZIP format (`cair.za.net/ontologies`).

Tasks: The first task is precompiling exceptionality rankings for each ontology in the dataset. Rankings are then stored on file for later use in entailment testing. It is important to note that the computation of the ranking is considered as an offline, precompilation process for each stable version of an ontology. Such a task is not meant to be executed on-demand during defeasible entailment tests. Lemma 1 is used as an optimisation in the ranking procedure. We only need check exceptionality of $C \sqsubseteq\!\!\!\sim D$'s where C is unsatisfiable w.r.t. the classical translation of $\langle \mathcal{T}, \mathcal{D} \rangle$ (see Lines 2 to 4 of procedure Exceptional in Section 2.2).

The entailment tests are then performed on the stored rankings and results of both tasks are recorded. We recorded the average time it took to compute the rankings, and to answer entailment questions, with some additional metrics which we present in Section 3.4. For entailment tests, we made no use of any optimisation.

Equipment: The evaluation was carried out on an Intel Core i7 2.5Ghz processor running MacOSX 10.10. 8GB of memory is allocated to the Java Virtual Machine (Java version 1.6 is used). HermiT is the classical OWL 2 DL reasoning implementation.

Reproducible steps: Assuming we have already obtained our dataset (set of ontologies) and generated set of queries for each ontology (as described in the above data summary), the steps required to conduct the evaluation are as follows: (1) Compute the ranking (according to Procedure ComputeRanking) of each ontology, record the time required and store it to file (use optimisation described in Lemma 1 to avoid checking if satisfiable concepts are exceptional), (2) for each ontology, execute its set of queries on its stored ranking (using Procedure RationalClosure) and record timings.

3.4 Results and Analysis

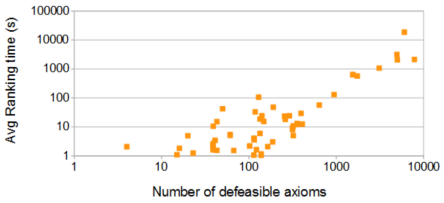
The overall results for computing the rankings and testing entailment have proven to be extremely promising. Figure 2 gives an overview of some of the more pertinent results w.r.t. the computation of rankings.

Ranking performance: Examining the ranking times in Figure 2, we notice that on average over the entire dataset, it takes 10 minutes to rank a single ontology. However, looking at the "median" column of the ranking shows the majority of rankings

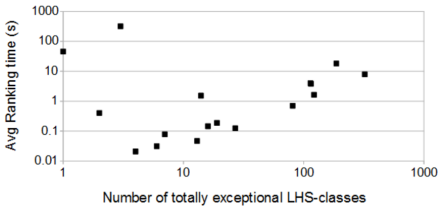
Corpus	TBox size			DTBox size			LHS-incoherent classes			Totally Exceptional classes		Number of ranks		Ranking time (s)		
	avg	median	max	avg	median	max	avg	median	max	avg	max	avg	max	avg	max	median
Bioportal	41309	754	439208	627	81	6010	477	13	4716	38	322	1.2	2	1705	18742	0.81
OOL	41389	32928	142996	415	39	7842	160	2	4018	2.7	122	1.6	2	139	3173	2.34
MOWLCorp	8719	691	137021	208	31.5	5002	76	6.5	1461	13.4	322	1.4	3	60	2070	0.16

Fig. 2: Ontology metrics and ranking computation results for the dataset.

were computed in less than a second. There are just four "outlier" ontologies which breach the 2000 second mark, while the maximum ranking time for the remainder of the data is 1000 seconds. The reason is that these four ontologies have the most number of unsatisfiable LHS classes in the data (requiring more exceptionality checks). It must be stressed that the ranking computation is concerned with stratifying only the *defeasible* axioms in the ontologies. Therefore, in general, the ranking times increase with the number of defeasible axioms (see Figure 3a). A key insight is when the number of entailment checks *and* ontology size (number of subsumptions we are checking entailment w.r.t.) is maximised, then performance will be worst for these cases.



(a) V.s. no. of defeasible axioms.



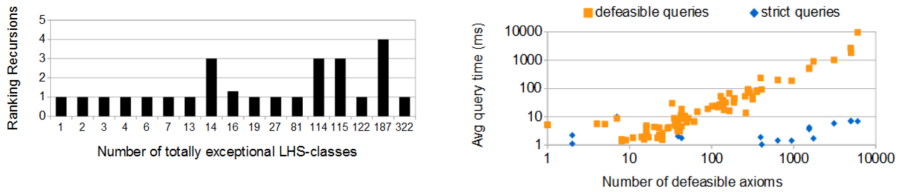
(b) V.s. totally exceptional LHS-classes.

Fig. 3: Ranking computation performance (ranking time).

The most challenging cases, in theory, for our reasoner are those with totally exceptional LHS-classes in the ontology. These cases are more intensive because we have to recursively apply the ranking procedure (see Lines 2-8 of Procedure ComputeRanking in Section 2.2), until all the totally exceptional information is added to the TBox. In our dataset of 134 ontologies, roughly 30% of them have totally exceptional LHS-classes.

Figure 3b shows that, even restricted to cases with totally exceptional LHS-classes, the ranking performance is well inside 100 seconds for the vast majority of the cases. The reason, we conjecture, is that the numbers of defeasible axioms in these ontologies stay relatively low allowing the performance to stay in check. Figure 4a illustrates the number of recursions required in Procedure ComputeRanking for these cases.

While there are some cases with 3 and 4 repetitions, the majority of cases require just one repetition. This, together with the fact the average number of defeasible axioms for these ontologies is just 127, explains the very low impact on performance that these "hard" cases have. In fact, the average number of defeasible axioms for these cases is significantly lower than that of any of the 3 general corpora in the dataset (see the table in Figure 2). We also notice that a key factor in performance is the number of LHS-



(a) Iterations required of ranking procedure.

(b) Inference performance across dataset.

Fig. 4: Recursion in ranking procedure (4a) and defeasible inference performance (4b).

classes that are unsatisfiable. Because of the optimisation represented by Lemma 1, we only need check exceptionality for these classes. Therefore the ratio of these incoherent classes to the ontology size will have a major impact on performance. Finally, we note that our hypothesis turns out to be correct concerning the number of ranks in the computed rankings. The average is 2 (single level exceptions), while there are a sprinkling of cases in which there are 3 ranks.

Entailment performance: The performance of defeasible entailment in the data is also encouraging. It seems that once the ranking of an ontology is obtained, the majority of defeasible entailment queries can be answered instantly. The average time to decide a defeasible entailment was 145ms.

The median is just 4ms highlighting that most of the entailments can be be computed almost instantly. As in the case of the ranking performance behaviour, there are a few "outlier" cases which prove much harder than in general. In one particular BioPortal ontology (the most difficult ontology in the data), it takes on average 9.6 seconds to compute a defeasible entailment. There are, however, 421,268 logical axioms in this ontology, of which 6010 are defeasible and 4716 have LHS-concepts that are classically unsatisfiable.

94% of the ontologies in our dataset take less than 200ms on average to decide defeasible entailment. It is not surprising that the prevalence of totally exceptional concepts does not significantly impact the performance of defeasible inference. This is likely because such information was moved to the TBox in the ranking step and therefore of little importance performance-wise during inference.

The dominant factor in performance (for both ranking compilation and query performance) remains ontology size and number of defeasible axioms (see Figure 4b). Because of the low variance in the number of ranks in ontology rankings (between 0 and 3), it is not surprising that this does not significantly impact performance, although we omit the graph illustrating this due to space constraints. It must be stated, though, that in theory the number of ranks will affect performance especially in the case where the exceptionality of the antecedent of the query is high. I.e., in such cases the number of repetitions of the while loop in Procedure RationalClosure will be higher.

4 Related Work

From a practical standpoint, the most closely related work is that of Bonatti et al. [2]. They introduce a new DL called $\mathcal{DL}^{\mathcal{N}}$ for handling exceptions by allowing or blocking

inheritance of certain properties to these exceptions. While the extra features of $\mathcal{DL}^{\mathcal{N}}$ can be built on top of any classical DL, the authors apply their evaluation to ontologies of $\mathcal{EL}$-variants [1]. They also use an underlying classical OWL reasoner ELK (`cs.ox.ac.uk/isg/tools/ELK`) which is highly optimised for such logics. In addition, they exploit incremental reasoning capabilities of ELK so unnecessary repetition of computations is not required with small changes to the ontology.

The authors use two approaches for extending the Gene Ontology (GO) (`geneontology.org`), with defeasible features. The first, is a principled injection of purely synthetic defeasible inclusions into GO, and the second is the translation of a *random* subset of classical inclusions in GO to defeasible ones. A direct comparison of their results with ours is non-sensical. (i) Their data is derived through synthetic modifications of GO which are expressed in variants of $\mathcal{EL}$ (rather than our $\mathcal{ALC}$ variants of different ontologies), (ii) they generate cases with between 5 and 25 percent of subsumptions being defeasible (whereas we utilise our DTF with no direct control of this percentage), (iii) their experiment machine is allocated 18GB of memory (whereas ours is restricted to 8GB) and (iv) they exploit the incremental reasoning of ELK which greatly increases the performance of reasoning (whereas we do not make use of such facilities). Nevertheless, for interests sake, their KB precompilation times (query time is negligable after precompilation) roughly vary between 25 and 115 seconds under these conditions.

We have, ourselves, also employed synthetic generation of data [7] in the past. All of the data in this evaluation were $\mathcal{ALC}$ ontologies and had between 150 and 5150 axioms. We varied the percentage of defeasible to strict axioms in the ontologies, in increments of 10, between 10 and 100 percent. A direct comparison with our results is also not suitable here, although our ranking times ranged between 0.5 seconds in the 10 percent case, to roughly 8 seconds in the 100 percent case. Query times, thereafter, ranged between 3 and 18 milliseconds for these respective cases.

We also know of some mature Circumscriptive [20] approaches that have been implemented [3, 4]. However, the performance results of such implementations remain unpublished to the best of our knowledge.

5 Conclusions and future work

We have presented a systematic and intuitive approach to introduce defeasible subsumption into real-world OWL ontologies. Applying this to the Manchester OWL Repository, we were able to generate test cases to evaluate the performance of Rational Closure implemented in DIP (Defeasible-Inference Platform). We report that this kind of defeasible reasoning is quite feasible on our principally generated data. While there are some mentioned limitations of our approach, we argue that the data we generate can give meaningful insight into the performance of RC for real-world ontologies. In conclusion, we hope that our approach provides a stepping stone to developing more sophisticated methodologies for introducing defeasible features into real-world ontologies, and that it will spur more investigations into the performance of defeasible reasoning in general.

Acknowledgements. Part of the work of Giovanni Casini has been supported by the Fonds National de la Recherche, Luxembourg, and cofunded by the Marie Curie Actions of the European Commission (FP7-COFUND) (AFR/9181001).

References

1. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge Univ. Press, 2003.
2. P. Bonatti, M. Faella, I. Petrova, and L. Sauro. A New Semantics for Overriding in Description Logics. *Articial Intelligence*, 2015.
3. P. Bonatti, M. Faella, and L. Sauro. Defeasible Inclusions in Low-Complexity DLs. *JAIR*, 42:719–764, 2011.
4. P. Bonatti, C. Lutz, and F. Wolter. Description Logics with Circumscription. In *Proc. of KR*, pages 400–410, 2006.
5. K. Britz, G. Casini, T. Meyer, K. Moodley, and I. J. Varzinczak. Ordered Interpretations and Entailment for Defeasible Description Logics. Technical report, CAIR, CSIR Meraka and UKZN, South Africa, 2013.
6. K. Britz, T. Meyer, and I. Varzinczak. Semantic Foundation for Preferential Description Logics. In *Proc. of AI*, pages 491–500, 2011.
7. G. Casini, T. Meyer, K. Moodley, and I. Varzinczak. Towards Practical Defeasible Reasoning for Description Logics. In *Proc. of DL*, 2013.
8. G. Casini and U. Straccia. Rational Closure for Defeasible Description Logics. In *Proc. of JELIA*, pages 77–90, 2010.
9. B. Cuenca-Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. OWL 2: The Next Step for OWL. *Web Semantics: SSAWWW*, 6(4):309–322, 2008.
10. L. Giordano, V. Gliozzi, N. Olivetti, and G. L. Pozzato. Preferential Description Logics. In *Proc. of LPAR*, pages 257–272, 2007.
11. Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. *Web Semantics: SSAWWW*, 3(2):158–182, 2005.
12. M. Horridge. *Justification Based Explanation in Ontologies*. PhD thesis, University of Manchester, 2011.
13. M. Horridge and S. Bechhofer. The OWL API: A Java API for OWL Ontologies. *Semantic Web*, 2(1):11–21, 2011.
14. I. Horrocks, U. Sattler, and S. Tobies. Practical Reasoning for Expressive Description Logics. In *Proc. of LPAR*, pages 161–180, 1999.
15. P. Ke and U. Sattler. Next Steps for Description Logics of Minimal Knowledge and Negation as Failure. In *Proc. of DL*, 2008.
16. D. Lehmann and M. Magidor. What Does a Conditional Knowledge Base Entail? *Art. Intell.*, 55(1):1–60, 1992.
17. T. Lukasiewicz. Expressive Probabilistic Description Logics. *Art. Intell.*, 172(6):852–883, 2008.
18. N. Matentzoglu, S. Bail, and B. Parsia. A Snapshot of the OWL Web. In *Proc. of ISWC*, pages 331–346. 2013.
19. N. Matentzoglu, D. Tang, B. Parsia, and U. Sattler. The Manchester OWL Repository: System Description. In *Proc. of ISWC*, pages 285–288, 2014.
20. J. McCarthy. Circumscription - A Form of Non-Monotonic Reasoning. *Art. Intell.*, 13(1–2):27–39, 1980.
21. T. Meyer, K. Moodley, and U. Sattler. DIP: A Defeasible-Inference Platform for OWL. In *Proc. of DL*, 2014.
22. R. Reiter. A Logic for Default Reasoning. *Art. Intell.*, 13(1):81–132, 1980.
23. U. Sattler, T. Schneider, and M. Zakharyaschev. Which Kind of Module Should I Extract? In *Proc. of DL*, 2009.
24. V. Sazonau, U. Sattler, and G. Brown. Predicting Performance of OWL Reasoners: Locally or Globally? In *Proc. of KR*, 2014.