

Defeasible Description Logics

Guido Governatori

School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
guido@itee.uq.edu.au

Abstract. We propose to extend description logic with defeasible rules, and to use the inferential mechanism of defeasible logic to reason with description logic constructors.

1 Introduction

This paper examines the addition of intelligent properties to the implementation of and reasoning on ontologies, where an open world is assumed. The motivation of this work lies in a current inability to reason intuitively on ontologies with incomplete or inconsistent information. Very often it is not possible to have complete information or complete knowledge of a domain that we wish to reason on. And herein lays the problem for the logics currently employed for reasoning on knowledge bases. This problem lies in the fact that most ontologies are represented using a language based on First order logic which is inappropriate for reasoning on partial, incomplete and inconsistent knowledge. For this reason an attempt has been made, by using a language which has proven successful in the field of ontologies and combining its principles with properties of a more flexible logic, to solve in part the motivating problem driving this work. The proposed method for solving this is to integrate two established forms of logic, Description logic [8] and Defeasible logic [4,10,24]. Description logics though imparting strong and conclusive reasoning mechanisms, lack the flexibility of Defeasible logics non-monotonic reasoning mechanisms, which add flexibility to knowledge bases that have partial knowledge. The project involves specifically the addition of defeasible rules to a description logic knowledge base to achieve a flexible and decidable language on which reasoning can occur.

Though adding non-monotonicity to description logics is not entirely a new concept [19,21,7,25,26,6,28], in this paper we take a unique perspective on the problem domain and the method by which this problem could be solved.

In [6,7,25,28,26] Description logic is extended with default with and without priorities; moreover [25,26] discuss the notion of defeasible subsumption.

Another approach, discussed in [19], involves the study of the intersection between two formal logics, Description Logic Programs and Description Horn Logic, with the aim of adding a large amount of expressiveness to description logic. In [21] an extension to Description logic *SHOQ* is attempted via the

addition of a preference order similar to the superiority relation in Defeasible Logic. This paper discusses the applicability of the “introduction of preferred models”, in this case the preference being to “select axioms that defeat as few conclusions as possible . . . or the least preferred axioms”.

Finally some combinations of Defeasible logic and Description logic have been proposed. [1,5,29] propose to combine the two logics by adding a layer of rules (from Defeasible Logic) on top of ontologies in Description Logic. The language is partitioned in two disjoint classes, literals and dl-literals. In this approach dl-literals corresponds to the concepts defined in Description Logic and can appear only in the antecedents of rule, while normal literals not subject to such restriction. Thus dl-literal exhibit a monotonic behaviour while normal literals are non-monotonic. A similar approach is adopted in [13] for the integration of Description Logic knowledge bases and Logic Programs.

The paper is organised as follows: in Section 2 and 3 we introduce the two basic logic to be integrated, namely Description Logic and Defeasible Logic; then in Section 4 we discuss how the two formalisms can be combined to produce a Defeasible Description Logic, and in Section 5 we illustrate some of the added features of the new logic with the help of an example. Finally Section 6 presents a short discussion of the results and hints for future work.

2 Description Logic

Description logics are monotonic formalisms based upon first-order logic [9]. Essentially Description logics are comprised of atomic concepts and atomic roles. Atomic concepts are related to particular entities within the knowledge base, an example of this is that ‘Man’ may be a concept, with perhaps BILL as an instance of the concept. Atomic roles, however, are used to express binary relationships between individuals, that is wherever we may have an instance of BILL being a parent, therefore we can conclude that BILL is can be associated with the role ‘hasChild’, for example ‘hasChild(BILL, PETER)’, where hasChild is the role that binds the father, BILL, to his son, PETER. A Description logic is characterised by a set of constructors that facilitate building complex concepts and roles. Concepts are interpreted as sets of objects and roles are seen as (binary) relations between objects in the domain.

Description logic allows for the representation of concept conjunctions, concept disjunctions, and concept negations [9].

Concept conjunction relates to the ability to join one or more concepts to define a complex concept or at least its properties or characteristics. Concept conjunction can be illustrated by the following example

$$\text{Father} \sqsubseteq \text{Man} \sqcap \text{Parent}$$

Here we can see that the instances in the set of men who also appear as instances in the set of parents can be defined, as subsuming the set of instances of the concept father, which is to say all instances of the concept **Father** will also be instances in the set of concepts **Man** and **Parent**.

Concept disjunction relates to the ability to restrict the definition of a complex concept or at least its properties or characteristics, to appearing in the set of one concept or the other. Concept disjunction can be illustrated by the following example

$$\text{Person} \sqsubseteq \text{Man} \sqcup \text{Woman}$$

Here we can see that the instances in the set of person can appear in the set of man or in the set of woman. For this reason the set of instances of man is a subset of the set of people and furthermore the set of woman is a subset of person.

Concept negation is the construct that allows complex concepts to be defined with the negation of another concept. It is useful in situations where we want to define concepts that are disjoint to another concept as seen in the following example

$$\text{Woman} \sqsubseteq \neg \text{Man} \sqcap \text{Person}$$

In this example, the concept man is negated to denote all instances in the domain which do not appear in the set of man unioned with the concept person appears as the subset of the concept woman. Therefore all the instances in the set of **Woman** should appear in the set of instances for the concept **Person** but not in the set of **Man**.

Description Logic also allows value or role restriction constructs [9]. Role restriction ($\forall R.C$) is the construct that requires that all the individuals that are in a specified relationship R with the concept being described belong to the concept C . Role restriction can be illustrated using the following example

$$\forall \text{hasChild.female}$$

which returns all individuals who have only daughters and no sons.

Though description logics can offer further constructors in lower abstractions of the logic, in this paper we will focus on these four constructors. Moreover negation is applied only to atomic concepts. Therefore only the \mathcal{ALC}^- subset of description logic has been extended with defeasibility (Section 4).

The semantics of Description Logic is given in terms of an interpretation, consisting of a non empty set $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$, and is defined symbolically as

$$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$$

The interpretation function gives the extension of the concepts and roles, and it assigns to every atomic concept a subset of $\Delta^{\mathcal{I}}$ and every atomic role a binary relation in $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The semantic interpretation of the operators described

above is defined as follows:

Syntax	Semantics
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
\top	$\Delta^{\mathcal{I}}$
\perp	\emptyset
$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$\forall R.C$	$\{x \mid \forall y : (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$

Description logics knowledge bases are made up of TBoxes and ABoxes. TBoxes contain concept definitions, where concept definitions define new concepts based on preexisting concepts. An example of such a concept definition is

$$\text{Woman} \equiv \text{Person} \sqcap \text{Female}$$

The use of equivalence allows in this case for an instance of woman to be created where there is an instance of both person and female. Therefore in this way TBoxes in description logic knowledge bases provide definite and strict representations of the conditions required to create new concepts and in turn instances of those concepts. TBoxes however must contain only one definition for each unique concept name in the spirit of developing conflict free knowledge bases. Furthermore a definition cannot reference itself and as such must not cyclically reference itself.

ABoxes, conversely contain assertional information, they define specific roles or concepts and are known to change based on circumstances. An example of the type of information that can be present in an ABox includes

a concept instance such as $\text{Person}(\text{JILL})$
 a role instance such as $\text{Mother}(\text{JILL}, \text{BILL})$

From this ABox we have the assertional information that JILL is an instance of the concept type **Person**. Furthermore the role, which binds the instances of JILL and BILL via the **Mother** role, is also present and it denotes that JILL is BILL's mother. As in TBoxes, unique names must be adhered to in ABoxes and as such if there exists two instances of a concept with the same instance name, these concepts are interpreted to be equivalent.

Subsumption is the basic reasoning method of description logic. Given two concepts C and D and a knowledge base Σ , the following illustrates that D subsumes C in Σ .

$$\Sigma \models C \sqsubseteq D$$

this requires that it be proved that in Σ , that D (the subsumer) is more general than C (the subsumee), or, in other terms, that the extension of C is included in the extension of D . For example

$$\text{Woman} \sqsubseteq \text{Person}$$

means that the concept of woman is considered more specialised than the concept of person. That is the set of elements belonging to the concept **Woman** can be found in the set of elements belonging to the concept **Person**, and as such the domain of **Woman** is a subset of the domain of **Person**, formally

$$\text{Woman}^{\mathcal{I}} \subseteq \text{Person}^{\mathcal{I}}$$

An extension of the above reasoning mechanism is concept equivalence. Equivalence is the reasoning mechanism whereby two concepts are checked to see if the set of instances in one concept is the same as the set of instances for the other concept. This reasoning mechanism can be represented symbolically as

$$\Sigma \models C \equiv D$$

and an example of equivalence check in description logic may involve verifying that the set of mothers is equivalent to the set of women with children

Concept satisfiability is another reasoning mechanism employed by description logics. Concept satisfiability relates to the check that is performed to ensure that there exists a semantic interpretation of the concept that does not result in the empty set. This is represented symbolically as

$$\Sigma \not\models C \equiv \perp$$

which means that the extension of C is not equivalent to the empty set (the extension of the \perp concept).

3 Defeasible Logic

Defeasible Logic has been developed by Nute [24] over several years with a particular concern about computational efficiency (indeed, its efficiency is linear cf. [22]) and ease of implementation (nowadays several implementations exist [11,23] and some of them can deal with theories consisting of over 100,000 propositional rules [23]). In [3] it was shown that Defeasible logic is flexible enough to deal with several intuitions of non-monotonic reasoning, and it has been applied to legal reasoning [2,16], automated negotiation [14,12], contracts [27], business rules [20], and multi-agent systems [18,17,15,16].

It is not possible in this short paper to give a complete formal description of the logic. However, we hope to give enough information to make the discussion intelligible. We refer the reader to [24,10,4] for more thorough treatments. As usual with non-monotonic reasoning, we have to specify 1) how to represent a knowledge base and 2) the inference mechanism.

We begin by presenting the basic ingredients of Defeasible Logic. A defeasible theory contains five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation. We consider only essentially propositional rules. Rules containing free variables are interpreted as the set of their variable-free instances.

Facts are indisputable statements, for example, “Guido is a lecturer”. In the logic, this might be expressed as $\text{Lecturer}(\text{GUIDO})$.

Strict rules are rules in the classical sense: whenever the premises are indisputable (e.g., facts) then so is the conclusion. An example of a strict rule is “Lecturers are faculty member”. Written formally:

$$\text{Lecturer}(x) \rightarrow \text{FacultyMember}(x).$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is “people giving lectures are faculty members”; written formally:

$$\text{GivesLectures}(x) \Rightarrow \text{FacultyMember}(x).$$

The idea is that if we know that someone gives a lecture, then we may conclude that he/she is a faculty member, *unless there is other evidence suggesting that it may not be a faculty member*.

Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. In other words, they are used to defeat some defeasible rules by producing evidence to the contrary. An example is “tutors might not be faculty members”. Formally:

$$\text{Tutor}(x) \rightsquigarrow \neg \text{FacultyMember}(x).$$

The main point is that the information that somebody is a tutor is not sufficient evidence to conclude that he/she is not a faculty member. It is only evidence that the tutor *may* not be a faculty member. In other words, we do not wish to conclude $\neg \text{FacultyMember}$ if Tutor , we simply want to prevent a conclusion FacultyMember in absence of further information.

The *superiority relation* among rules is used to define priorities among rules, that is, where one rule may override the conclusion of another rule. For example, given the defeasible rules

$$\begin{aligned} r &: \text{GivesLectures}(x) \Rightarrow \text{FacultyMember}(x) \\ r' &: \text{GuestLecturer}(x) \Rightarrow \neg \text{FacultyMember}(x) \end{aligned}$$

which contradict one another, no conclusive decision can be made about whether a guest lecturer is a faculty member. But if we introduce a superiority relation $>$ with $r' > r$, then we can indeed conclude that the guest lecturer cannot be a faculty member. The superiority relation is required to be acyclic. It turns out that we only need to define the superiority relation over rules with contradictory conclusions.

A rule r consists of its *antecedent* (or *body*) $A(r)$ ($A(r)$ may be omitted if it is the empty set) which is a finite set of literals, an arrow, and its *consequent* (or *head*) $C(r)$ which is a literal. Given a set R of rules, we denote the set of all strict rules in R by R_s , the set of strict and defeasible rules in R by R_{sd} , the set of defeasible rules in R by R_d , and the set of defeaters in R by R_{dft} . $R[q]$ denotes the set of rules in R with consequent q . If q is a literal, $\sim q$ denotes the

complementary literal (if q is a positive literal p then $\sim q$ is $\neg p$; and if q is $\neg p$, then $\sim q$ is p).

A *defeasible theory* D is a triple $(F, R, >)$ where F is a finite set of facts, R a finite set of rules, and $>$ a superiority relation on R .

A *conclusion* of D is a tagged literal and can have one of the following four forms:

- $+\Delta q$, which is intended to mean that q is definitely provable in D (i.e., using only facts and strict rules).
- $-\Delta q$, which is intended to mean that we have proved that q is not definitely provable in D .
- $+\partial q$, which is intended to mean that q is defeasibly provable in D .
- $-\partial q$ which is intended to mean that we have proved that q is not defeasibly provable in D .

Provability is based on the concept of a *derivation* (or proof) in $D = (F, R, >)$. A derivation is a finite sequence $P = (P(1), \dots, P(n))$ of tagged literals satisfying four conditions (which correspond to inference rules for each of the four kinds of conclusion). $P(1..i)$ denotes the initial part of the sequence P of length i

- $+\Delta$: If $P(i+1) = +\Delta q$ then
 - (1) $q \in F$ or
 - (2) $\exists r \in R_s[q] \forall a \in A(r) : +\Delta a \in P(1..i)$.
- $-\Delta$: If $P(i+1) = -\Delta q$ then
 - (1) $q \notin F$ and
 - (2) $\forall r \in R_s[q] \exists a \in A(r) : -\Delta a \in P(1..i)$.

The definition of Δ describes just forward chaining of strict rules. For a literal q to be definitely provable we need to find a strict rule with head q , of which all antecedents have been definitely proved previously. And to establish that q cannot be proven definitely we must establish that for every strict rule with head q there is at least one antecedent which has been shown to be non-provable.

- $+\partial$: If $P(i+1) = +\partial q$ then either
 - (1) $+\Delta q \in P(1..i)$ or
 - (2.1) $\exists r \in R_{sd}[q] \forall a \in A(r) : +\partial a \in P(1..i)$ and
 - (2.2) $-\Delta \sim q \in P(1..i)$ and
 - (2.3) $\forall s \in R[\sim q]$ either
 - (2.3.1) $\exists a \in A(s) : -\partial a \in P(1..i)$ or
 - (2.3.2) $\exists t \in R_{sd}[q]$ such that $t > s$ and $\forall a \in A(t) : +\partial a \in P(1..i)$

Let us work through this condition. To show that q is provable defeasibly we have two choices: (1) We show that q is already definitely provable; or (2) we need to argue using the defeasible part of D as well. In particular, we require that there must be a strict or defeasible rule with head q which can be applied (2.1). But now we need to consider possible “attacks”, i.e., reasoning chains in

support of $\sim q$. To be more specific: to prove q defeasibly we must show that $\sim q$ is not definitely provable (2.2). Also (2.3) we must consider the set of all rules which are not known to be inapplicable and which have head $\sim q$ (note that here we consider defeaters, too, whereas they could not be used to support the conclusion q ; this is in line with the motivation of defeaters given earlier). Essentially each such rule s attacks the conclusion q . For q to be provable, each such rule s must be counterattacked by a rule t with head q with the following properties: (i) t must be applicable at this point, and (ii) t must be stronger than s . Thus each attack on the conclusion q must be counterattacked by a stronger rule. In other words, r and the rules t form a team (for q) that defeats the rules s . In an analogous manner we can define $-\partial q$ as

$$\begin{aligned}
 &-\partial: \text{ If } P(i+1) = -\partial q \text{ then} \\
 &(1) \quad -\Delta q \in P(1..i) \text{ and} \\
 &\quad (2.1) \quad \forall r \in R_{sd}[q] \exists a \in A(r) : -\partial a \in P(1..i) \text{ or} \\
 &\quad (2.2) \quad +\Delta \sim q \in P(1..i) \text{ or} \\
 &\quad (2.3) \quad \exists s \in R[\sim q] \text{ such that} \\
 &\quad\quad (2.3.1) \quad \forall a \in A(s) : +\partial a \in P(1..i) \text{ and} \\
 &\quad\quad (2.3.2) \quad \forall t \in R_{sd}[q] \text{ either } t \not\prec s \text{ or} \\
 &\quad\quad\quad \exists a \in A(t) : -\partial a \in P(1..i)
 \end{aligned}$$

The purpose of the $-\partial$ inference rules is to establish that it is not possible to prove $+\partial$. This rule is defined in such a way that all the possibilities for proving $+\partial q$ (for example) are explored and shown to fail before $-\partial q$ can be concluded. Thus conclusions tagged with $-\partial$ are the outcome of a constructive proof that the corresponding positive conclusion cannot be obtained.

4 Defeasible Description Logics

In this section we show how to extend \mathcal{ALC}^- with defeasibility. As we saw in Section 2 a knowledge base in \mathcal{ALC}^- is a pair

$$(\mathcal{A}, \mathcal{T})$$

where \mathcal{A} , the ABox, is a set of individual assertions, and \mathcal{T} , the TBox, contains inclusion axioms or definitions of concepts. When we consider the relationships between a knowledge base in \mathcal{ALC}^- and a defeasible theory we have that the ABox corresponds to the set of facts, while the TBox corresponds to the monotonic part of the rules in a defeasible theory. Given the syntactic limitations of \mathcal{ALC}^- , we can give a more precise characterisation of the TBox in terms of strict rules (see also [5]). In particular given an inclusion axiom

$$\sqcap_{i=1}^n C_i \sqsubseteq \sqcap_{j=1}^m D_j$$

the inclusion axiom is equivalent to the following set of strict rules (the set of rules induced by \mathcal{T})

$$\begin{array}{c} C_1, \dots, C_n \rightarrow D_1 \\ \vdots \\ C_1, \dots, C_n \rightarrow D_m \end{array}$$

In case $n = m = 1$ and C_i, D_j are atomic concepts (i.e., concepts not defined in terms of other concepts), we also have to include the contrapositive of the inclusion axiom, namely

$$\neg D_j \rightarrow \neg C_i$$

This means that we can use either structural subsumption of \mathcal{ALC}^- or strict derivability of defeasible logic to deal with the monotonic part of a defeasible description logic knowledge base. To add non-monotonicity we introduce defeasible logic rules, and defeasible logic proof theory to a knowledge base in \mathcal{ALC}^- . Hence a defeasible description logic theory is a structure

$$(\mathcal{A}, \mathcal{T}, R, >)$$

where, as before \mathcal{A} is the ABox, \mathcal{T} is the TBox, R is a set of rules (strict rules, defeasible rules and defeaters), and $>$ –the superiority relation– is a binary relation defined over the rules in R plus the strict rules induced by the inclusion axioms in \mathcal{T} , according to the construction given above.

We can now give the conditions to derive new role restrictions, but before we have to determine the domain of the theory. The domain of the theory corresponds to the Herbrand universe of the ABox of the theory, that is, the set of all individuals occurring in the assertions in \mathcal{A} ; we will use $\Delta_{\mathcal{T}}$ to denote it.

$$\begin{array}{l} +\Delta\forall R.C: \text{ If } P(i+1) = +\Delta\forall R.C(a) \text{ then} \\ \quad \forall b \in \Delta_{\mathcal{T}} \text{ either} \\ \quad (1) -\Delta R(a, b) \text{ or} \\ \quad (2) +\Delta C(b) \end{array}$$

$$\begin{array}{l} -\Delta\forall R.C: \text{ If } P(i+1) = -\Delta\forall R.C(a) \text{ then} \\ \quad \exists b \in \Delta_{\mathcal{T}} \text{ such that} \\ \quad (1) +\Delta R(a, b) \text{ and} \\ \quad (2) -\Delta C(b) \end{array}$$

Similarly the conditions to derive role restriction in a defeasible way are

$$\begin{array}{l} +\partial\forall R.C: \text{ If } P(i+1) = +\partial\forall R.C(a) \text{ then} \\ \quad \forall b \in \Delta_{\mathcal{T}} \text{ either} \\ \quad (1) -\partial R(a, b) \text{ or} \\ \quad (2) +\partial C(b) \end{array}$$

To prove a positive defeasible role restriction $-\partial\alpha R.C(a)$ we have to prove that for all the elements b in the domain of the knowledge base either we cannot prove

that b is not related via R with a , or we can show that b is an instance of the concept C .

Given the syntactic limitation of the language, it is not possible to have rules for $\neg\forall R.C$: negation is limited to atomic concept. Therefore the argument for proving a positive defeasible role restriction cannot be rebutted by another argument, but only undercut by arguments undermining the arguments used to prove the two parts of the argument for it.

$$\begin{aligned} -\partial\forall R.C: & \text{ If } P(i+1) = -\partial\forall R.C(a) \text{ then} \\ & \exists b \in \Delta_{\mathcal{T}} \text{ such that} \\ & (1) +\partial R(a, b) \text{ and} \\ & (2) -\partial C(b) \end{aligned}$$

To prove $-\partial\forall R.C(a)$ then there must exist an element b in the domain of the knowledge base such that it is defeasibly provable that b is in the role R with the concept instance a from the role restriction statement and it must be defeasibly not provable that b is an instance of the concept C .

It is immediate to verify that the condition for $-\partial\forall R.C$ corresponds to the semantic condition that evaluates $\forall R.C$ as false. Then according to the principle of strong negation advanced in [3] as tool to define derivation conditions in defeasible logic¹, we obtain the current clause for $+\partial\forall R.C$. Again it is easy to see that, intuitively, it matches the cases when $\forall R.C$ is true. Let us consider the following alternative condition

$$\begin{aligned} +\partial\forall R.C: & \text{ If } P(i+1) = +\partial\forall R.C(a) \text{ then} \\ & \forall b \in \Delta_{\mathcal{T}} \text{ if } +\partial\neg C(b) \text{ then } -\partial R(a, b) \end{aligned}$$

This definition is based on the idea that, given an interpretation of a description logic knowledge base, every element of the domain is either in the extension of a concept or in its complement. Thus if b does not satisfy C , it satisfies $\neg C$. Then a must not be related to b , if a belongs to the interpretation of $\forall R.C$. However this condition leads to (possibly) counterintuitive results. For example, given the following theory,

$$\begin{aligned} & \Rightarrow \neg C(b) \\ & \Rightarrow C(b) \\ & R(a, b) \end{aligned}$$

we derive $+\partial\forall R.C(a)$ simply because we fail to prove that $\neg C(b)$ is (defeasibly) the case, but on the other hand we do not have undisputed evidence of the truth of $C(b)$. While this may be appropriate in some interpretations, we believe that this is not the correct result in many other interpretations.

¹ The strong negation of a formula is closely related to the function that simplifies a formula by moving all negations to an innermost position in the resulting formula and replaces the positive tags with the respective negative tags and vice-versa.

² Here we have another complication with this definition, since in general $\neg C$ might not be defined in the language.

5 An Example

The following is an example of a Defeasible Description Logic knowledge base. It includes a list of concepts and the instances of those concepts as well as a list of roles and their instances. Furthermore three defeasible rules are specified each containing a role restriction constructor.

ABox

Faculty(ITEE)	Faculty(ARTS)
Faculty(LAW)	IteeCourse(INFS4201)
IteeCourse(COMP4600)	ArtsCourse(PSCY1020)
LawCourse(LAWS3010)	Student(DANIELLA)
DualDegree(ADRIAN)	Student(ROBIN)
Supervisor(GUIDO)	Supervisor(PENNY)
takes(DANIELLA, INFS4201)	takes(DANIELLA, COMP4600)
takes(ROBIN, PSCY1020)	takes(ADRIAN, COMP4600)
takes(ROBIN, COMP4600)	takes(ADRIAN, LAWS3010)
supervises(GUIDO, DANIELLA)	supervises(PENNY, ROBIN)
supervises(GUIDO, ADRIAN)	supervises(PENNY, ADRIAN)

TBox

$$\begin{aligned} \text{IteeStudent}(x) &\sqsubseteq \text{Student}(x) \\ \text{DualDegree}(x) &\sqsubseteq \text{IteeStudent}(x) \end{aligned}$$

Rules

$$\begin{aligned} \forall \text{supervises.IteeStudent}(x) &\Rightarrow \text{facultyMember}(x, \text{ITEE}) \\ \text{Student}(x), \forall \text{takes.IteeCourse}(x) &\Rightarrow \text{IteeStudent}(x) \\ \text{Student}(x), \forall \text{takes.ArtsCourse}(x) &\Rightarrow \neg \text{IteeStudent}(x) \end{aligned}$$

Finally the superiority relation is empty.

The ABox describes entries in a university database, while the TBox and the rules provide integrity constraints on possible legal records. For example the first rule says that ITEE faculty members usually can only supervise ITEE students; the second rules establish that non ITEE students have to take courses outside ITEE. Finally the last rule states that in normal circumstances students taking only arts courses are not enrolled in ITEE.

From the above Defeasible Description Logic knowledge base we can derive additional information other than that which is explicitly asserted. The method by which we can derive this information includes using a combination of subsumption and role restriction reasoning methods.

An example of the information that can be derived is, via subsumption, that the dual degree student Adrian can be classified as an ITEE student also (strict positive derivation $+\Delta$). Via the strict rule

$$\text{DualDegree}(\text{ADRIAN}) \rightarrow \text{IteeStudent}(\text{ADRIAN})$$

Daniella can be classed as an ITEE student (positive defeasible derivation $+\partial$), this is due to the fact that Daniella is a student and every course taken by Daniella is an ITEE course ($+\partial\forall\text{takes.IteeCourse}(\text{Daniella})$). To prove this we have to notice that there are no rules with head $\text{takes}(x, \text{Daniella})$, thus for every element y in the domain such that $\text{takes}(x, \text{Daniella})$ is not given in the ABox we can prove $-\partial\text{takes}(x, \text{Daniella})$. For the remaining elements of the domain, namely INFS4201 and COMP4600, we have to prove that $+\partial\text{IteeCourse}(\text{INFS4201})$ and $+\partial\text{IteeCourse}(\text{COMP4600})$. Both follow immediately since they are in the ABox, and hence are facts of the given theory.

Furthermore as Guido is a supervisor and he supervises Daniella and Adrian who are both ITEE students we can deduce via

$$\forall\text{supervises.IteeStudent}(\text{GUIDO}) \Rightarrow \text{facultyMember}(\text{GUIDO}, \text{ITEE})$$

that it is defeasibly provable that Guido is an ITEE faculty member (positive defeasible derivation $+\partial$). Deriving information in this way using a combination of role restriction and structural subsumption is, we believe, unique to Defeasible Description logic, and there would be no natural way to express this information in other current formalisms used for reasoning on ontological knowledge bases.

Furthermore given the challenge of discovering if Penny is a member of the ITEE faculty we are left with some decisions to make. Here we can see that Penny supervises two students. Adrian is the first student and we have already concluded that Adrian is an ITEE student due to the fact that he is classified as a dual degree student. The other student that Penny supervises is Robin. Robin is given as a student in the knowledge base. We can observe in the knowledge base that Robin takes one Arts course (PSCY1020) and one ITEE course (COMP4600). When we try to show that Robin is an ITEE student via the rule

$$\text{Student}(\text{ROBIN}), \forall\text{takes.IteeCourse}(\text{ROBIN}) \Rightarrow \text{IteeStudent}(\text{ROBIN})$$

we can show that $-\partial\text{IteeStudent}(\text{Robin})$. The reason for this is that we cannot show that the role restriction in this rule is defeasibly provable, in fact we can demonstrate the converse. $-\partial\text{takes.IteeCourse}(\text{Robin})$ due to the presence of the fact $\text{Arts}(\text{PSCY1020})$ and the role $\text{takes}(\text{ROBIN}, \text{PSCY1020})$, this concept and role is conducive to the conditions demonstrate the behaviour of negative role restriction.

Furthermore as Robin is defeasibly not an ITEE student we defeasibly cannot conclude that Penny is a member of the ITEE faculty. Given the rule

$$\forall\text{supervises.IteeStudent}(\text{PENNY}) \Rightarrow \text{facultyMember}(\text{PENNY}, \text{ITEE})$$

we can show that the role restriction for this rule fails and is defeasibly not provable due to the fact that we are given the information that Penny supervises

Robin and that we can derive that Robin is defeasibly not an ITEE student (this is derived in the same way we derived that Robin is not an ITEE student). As the role restriction is defeasibly not provable then we cannot defeasibly imply that Penny is a faculty member of ITEE.

6 Conclusion

In cases where we have conflicts in a knowledge base Description logic has historically proved useless at reasoning and collapses whereas Defeasible logic can be used to derive some meaningful solutions in the presence of these conflicts, via non-monotonic reasoning. Defeasible logic reasoning however is not as strong or conclusive as the reasoning in Description logic and for this reason Defeasible logic has not been considered appropriate for the reasoning on the knowledge bases of ontologies. From the inherent unsuitability of both formalisms for reasoning on partial or incomplete knowledge bases in a decidable way the aims for this paper was developed. We believe that the combination of defeasible rules with a Description logic knowledge base to derive Defeasible derivability is significant to current reasoning methods employed on ontologies.

Essentially what the addition of defeasible assertions adds to description logic is the ability to reason on knowledge bases where conflicting information is present. In this situation description logic alone will fail but through Defeasible Description Logic we are able to at least defeasibly derive some useful information. Our ability to derive these defeasible conclusions in such a decidable way means that description logic is extended via the two new strengths of derivability making reasoning on conflicting or incomplete knowledge bases conducive to deriving some useful information. Furthermore defeasible logic is extended through the reasoning mechanism of subsumption and the role restriction constructor.

Future work based on this work could include adding further description logic constructors to Defeasible Description logic and studying the strengths of derivation further. In the future this intuition could significantly be extended by adding additional constructors from Description logic at a lower level of abstraction, with a view to completely mapping the constructors in Description logic to a Defeasible Description logic. Furthermore the strengths of derivation could be studied more closely and even extended to produce new derivation types. A more in depth test of the viability of this logic could also be possible future work, carried out by implementing a knowledge base using Defeasible Description logic and doing comparative tests of the formalism against a knowledge base implemented with Description logic

References

1. Grigoris Antoniou. Nonmonotonic rule system on top of ontology layer. In I. Horrocks and J. Hendler, editors, *ISWC 2002*, number 2432 in LNCS, pages 394–398, Berlin, 2002.

2. Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. On the modeling and analysis of regulations. In *Proceedings of the Australian Conference Information Systems*, pages 20–29, 1999.
3. Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. A flexible framework for defeasible logics. In *Proc. American National Conference on Artificial Intelligence (AAAI-2000)*, pages 401–405, Menlo Park, CA, 2000. AAAI/MIT Press.
4. Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Transactions on Computational Logic*, 2(2):255–287, 2001.
5. Grigoris Antoniou and Gerd Wagner. Rules and defeasible reasoning on the semantic web. In M. Schroeder and G. Wagner, editors, *RuleML 2003*, number 2876 in LNCS, pages 111–120, Berlin, 2003.
6. F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalism. *Journal of Automated Reasoning*, 14:149–180, 1995.
7. F. Baader and B. Hollunder. Priorities on defaults with prerequisites and their application in treating specificity in terminological default logic. *Journal of Automated Reasoning*, 14:41–68, 1995.
8. Franz Baader, Diego Calvanes, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logics Handbook*. Cambridge University Press, Cambridge, 2003.
9. Franz Baader and Werner Nutt. Basic description logics. In Baader et al. [8], chapter 2, pages 43–95.
10. David Billington. Defeasible logic is stable. *Journal of Logic and Computation*, 3:370–400, 1993.
11. Michael Covington, Donald Nute, and A. Vellino. *Prolog Programming in Depth*. Prentice Hall, 1997.
12. Marlon Dumas, Guido Governatori, Arthur H. M. ter Hofstede, and Phillipa Oaks. A formal approach to negotiating agents development. *Electronic Commerce Research and Applications*, 1(2):193–207, 2002.
13. T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining answer set programming with description logics for the semantic web. In *In Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning*, pages 141–151, 2004.
14. Guido Governatori, Marlon Dumas, Arthur H.M. ter Hofstede, and Phillipa Oaks. A formal approach to protocols and strategies for (legal) negotiation. In Henry Prakken, editor, *Proceedings of the 8th International Conference on Artificial Intelligence and Law*, pages 168–177. IAAIL, ACM Press, 2001.
15. Guido Governatori and Vineet Padmanabhan. A defeasible logic of policy-based intention. In Tamás D. Gedeon and Lance Chun Che Fung, editors, *AI 2003: Advances in Artificial Intelligence*, volume 2903 of *LNAI*, pages 414–426, Berlin, 3-5 December 2003.
16. Guido Governatori and Antonino Rotolo. A computational framework for non-monotonic agency, institutionalised power and multi-agent systems. In Danièle Bourcier, editor, *Legal Knowledge and Information Systems*, volume 106 of *Frontiers in Artificial Intelligence and Applications*, pages 151–152, Amsterdam, 2003. IOS Press.
17. Guido Governatori and Antonino Rotolo. Defeasible logic: Agency and obligation. In Alessio Lomuscio and Donald Nute, editors, *Deontic Logic in Computer Science*, number 3065 in *LNAI*, pages 114–128, Berlin, 2004.

18. Guido Governatori, Antonino Rotolo, and Shazia Sadiq. A model of dynamic resource allocation in workflow systems. In Klaus-Dieter Schewe and Hugh E. Williams, editors, *Database Technology 2004*, number 27 in Conference Research and Practice of Information Technology, pages 197–206. Australian Computer Science Association, ACS, 19–21 January 2004.
19. B.N. Grosz, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *Proceedings of the twelfth international conference on World Wide Web*, pages 48–57. ACM Press, 2003.
20. B.N. Grosz, Y. Labrou, and H.Y. Chan. A declarative approach to business rules in contracts: Courteous logic programs in XML. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC-99)*. ACM Press, 1999.
21. S. Heymans and D. Vermeir. A defeasible ontology language. In R. Meersman and Z. Tari, editors, *Confederated International Conferences: CoopIS, DOA and ODBASE 2002*, number 2519 in LNCS, pages 1033–1046, Berlin, 2002. Springer-Verlag.
22. Michael Maher. Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711, November 2001.
23. Maher. M.J., A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. *International Journal of Artificial Intelligence Tools*, 10(4):483–501, 2001.
24. Donald Nute. Defeasible logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 353–395. Oxford University Press, 1987.
25. L. Padgham and T. Zhang. A terminological logic with defaults: A definition and an application. In *Proc. IJCAI'93*, pages 662–668, Los Altos, 1993. Morgan Kaufmann.
26. J. Quantz and V. Royer. A preference semantics for defaults in terminological logics. In *Proc KR'92*, pages 294–305, Los Altos, 1992. Morgan Kaufmann.
27. D.M. Reeves, B.N. Grosz, M.P. Wellman, and H.Y. Chan. Towards a declarative language for negotiating executable contracts. In *Proceedings of the AAAI-99 Workshop on Artificial Intelligence in Electronic Commerce (AIEC-99)*. AAAI Press / MIT Press, 1999.
28. U. Straccia. Default inheritance reasoning in hybrid KL-ONE-style logics. In *Proc. IJCAI'93*, pages 676–681, Los Altos, 1993. Morgan Kaufmann.
29. Kewen Wang, David Billington, Jeff Blee, and Grigoris Antoniou. Combining description logic and defeasible logic for the semantic web. In Grigoris Antoniou and Harlod Boley, editors, *RuleML 2004*, LNCS, Berlin, 2004. Springer-Verlag.