Defeasible Reasoning Approach to Resolve Conflict in Inconsistent Ontologies

by

Viet Hoai TO

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Mitsuru Ikeda

School of Knowledge Science
Japan Advanced Institute of Science and Technology

September, 2014

Abstract

With the emergence of new knowledge sources such as machinery knowledge and organizational knowledge, knowledge creation process has shifted to a new level of co-creation process where multiple participants share and utilize other's knowledge to create newly collaborative knowledge. One issue of knowledge co-creation is the heterogeneity and conflict among knowledge from various sources. Ontologies are the solution for heterogeneous problem since they can explicitly represent the knowledge within each participant to make the knowledge understandable and manipulatable by the others. They have been used widely in many areas relating to Knowledge Science such as knowledge managements, service science, bioinformatics, e-healthcare, e-learning, among others. There are various specification for ontology representation among which description logics (DLs) is chosen as the recommendation in Semantic Web environment due to their good tradeoff between expressive power and computational efficiency. DL ontology represents the world by concepts, which correspond to groups of objects, and roles, which correspond to the relationship among concepts. A DL ontology contains logical axioms, which are understandable by a computer and make it is possible to automatically handle an ontology.

This research aims to accelerate and accurate knowledge co-creation process by studying and extending ontology abilities to handle heterogeneous and conflict knowledge shared among collaborators. We construct an expressive ontology for School of Knowledge Science, *KS ontology*, and propose an extension of description logic (DL) ontology, named *defeasible DL*, which can handle conflict knowledge in inconsistent ontologies. The KS ontology is an explicit representation of educational program of School of Knowledge Science which allows students to exchange and create their own knowledge about studying plan with the assistant of computer. DL axioms are used to express educational program so that computer programs can process the ontology to provide assistant services. The application of KS ontology demonstrates the role of DL ontology to support knowledge sharing and co-creation process in real world situation.

Upon building ontology for real-world application, we recognize the problem of conflict knowledge. Conflict appears naturally because of the incompleteness of participant's knowing in a knowledge co-creation environment. For example, the advances of machine learning and data processing techniques can allow computer to generate the knowledge that have not been discovered by human being. Conflict knowledge can make an ontology inconsistent and infer meaningless conclusion. Reasoning with inconsistent ontologies is necessary and has attracted much attention in recent years. Those researches extend ontology reasoning ability by integrating non-monotonic reasoning mechanisms or providing uncertainty representation to handle conflict knowledge in ontology.

We propose defeasible DL, an extended representation of DL ontology, which contains defeasible axiom and priority relation. Defeasible axiom provides a mean to represent knowledge that can be retracted when there is contradict evidence and priority relation is used to assign the preference among axioms to resolve conflict among defeasible knowledge. Based on the principle of defeasible reasoning, we propose a reasoning framework for defeasible DL ontology which utilizes a selection function to select axioms and create a consistent sub-theory of original ontology and perform reasoning. The selection function is skeptical because it looks ahead to see whether there is any contradict conclusion that can be inferred and only select an axiom if all contradict conclusions are eliminated. Compared with other related works, our approach has simplicity and flexibility, expressive power, and computational efficiency as the advantages. Therefore, this research can help to accurate the conflict issue in knowledge co-creation application. Due to the simplicity, our approach also has the limitation of not exploiting the semantics of elements in the knowledge. This limitation can be solved by integrating our reasoning approach with more complicated uncertainty representation, which is the future perspective of our research.

Keywords: ontology, defeasible reasoning, description logics, knowledge representation and reasoning, conflict knowledge

Acknowledgments

I wish to express my sincere gratitude to my major advisor Professor Mitsuru Ikeda of Japan Advanced Institute of Science and Technology for his constant encouragement and kind guidance without which this dissertation would not have materialized. Under his supervision, I have had a very good opportunity to develop my attitude, ability and knowledge as a researcher.

I would like to express the deepest appreciation to Associate Professor Le Hoai Bac of HCMC University of Science,Vietnam National University for his precious support and encouragement during my off-campus research term for the minor research topic.

I would like to express my appreciation to Professor Riichiro Mizoguchi and Associate Professor Huynh Nam Van for their fruitful comments and discussion during my research.

I would like to thank my dissertation examiners, Professor Yoshiteru Nakamori, Associate Professor Le Hoai Bac, Associate Professor Huynh Nam Van and Associate Professor Takaya Yuizono, for their interest in my work and for making their time available for me.

Thanks to all members in the Ikeda Laboratory for their friendliness and constructive comments during discussions. They have made my doctoral life less stressful and more memorable.

I am grateful to my family for supporting my in every moment. Special thanks to my wife who shows an endless sympathy and sacrifice whenever I encounter difficulty.

Last but not least, I would like to thank the FIVE-JAIST dual program between JAIST and University of Science, VNU-HCMC and Vietnamese Ministry of Education, for the financial support during my study.

Contents

List of Figures

Chapter 1

Introduction

1.1 Ontology and Knowledge Science

For along time, knowledge creation process has been solely associated with human because it is one of the most intellectual processes. However, modern society witnesses the emergence of new sources of knowledge: organizational knowledge and machinery knowledge. *Organizational knowledge* is created through the interaction among agents in a complex system or organization. Typical research into collaborative knowledge is Nonaka and Takeuchi's [87] which introduces how knowledge is organizationally created in enterprises and produces successful products and technologies. This work establishes the foundation of research into what knowledge is and how knowledge is created and utilized, which are the main objectives of *Knowledge Science (KS). Machinery knowledge* is greatly motivated by two recent factors of information science: deep learning technique and the 'big data'. *Deep learning* is machine learning technique that allows computer to learn knowledge from data with very little human assistant. Meanwhile, *'Big data'* refers to any amount of data that is too large and complex to be processed using on-hand database management tools or traditional data processing applications. With the appearance of these factors, it is expected that machines can generate knowledge beyond human insight.

Knowledge creation scheme has shifted to a new level, the *knowledge co-creation* scheme where knowledge is created during the mutually interaction among individual knowledge, organizational knowledge and machinery knowledge, as shown in Figure 1.1. Knowledge co-creation consists of sharing, utilizing, and synthesizing knowledge from different sources to create newly collaborative knowledge. One of the problems of knowledge co-creation is heterogeneity among knowledge from different sources. For example, the knowledge learned by the computer can be distinct or even conflict with knowledge that have been known by the human. *This research aims to contribute to knowledge science by addressing the problem of conflict knowledge to accelerate and accurate the knowledge sharing and co-creation process.*

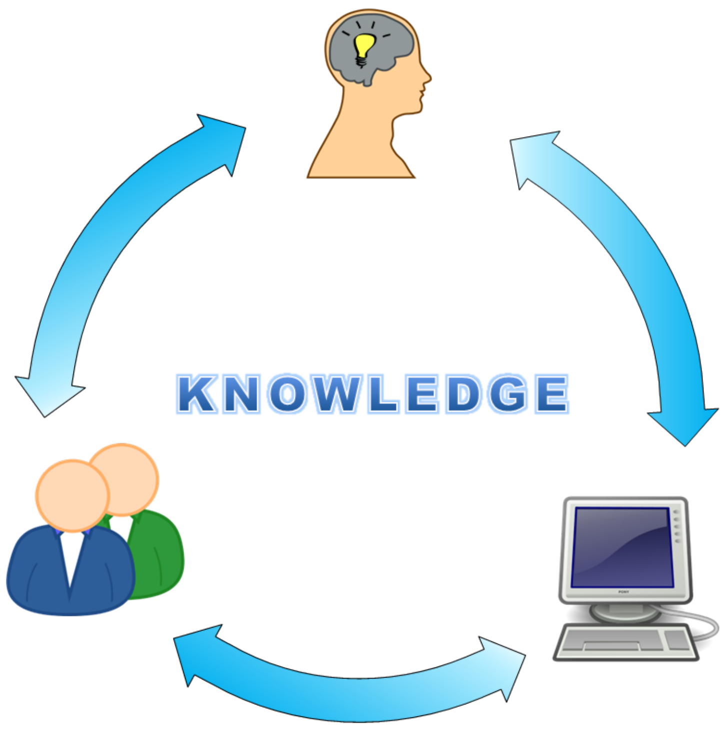


Figure 1.1: Knowledge Co-Creation in Modern Society where Knowledge can be created by sharing, utilizing, and synthesizing from Individual, Organizational and Machinery Knowledge

Ontology is the answer for heterogeneous problem when sharing and integrating knowledge from different sources. The term 'ontology' originally refers to the subfield of philosophy, which deals with the nature of existence. This branch of metaphysics tries to identify and describe the things that actually exist. The world ontology has recently been adopted by computer science to formally describe a domain of discourse and allow knowledge sharing within a certain domain or among various domains. Definitions of ontology have evolved over the years and also varies by perspectives. The most accepted definition of ontology is given by Studer et al. [111], which extends Gruber's definition [46], which states that "an ontology is an explicit and formal specification of a conceptualization". Therefore, ontology formally represents knowledge about a world or a domain to allow to share and incorporate mutual understanding among collaborators in knowledge sharing and co-creation as shown in Figure 1.2.

Another definition that is suitable to perspective of this research is Guarino and Giaretta's definition which considers ontology as a logical theory which gives an explicit, partial account of a conceptualization [47]. From computer science aspect, an ontology consists of a shared understanding of a domain of interest and this understanding is represented is a formal way so that a machine can handle it. Usually an ontology will contain a set of terms together with relationships between them. The terms typically denote classes of objects and relationships include the most often used *is-a* hierarchy and other

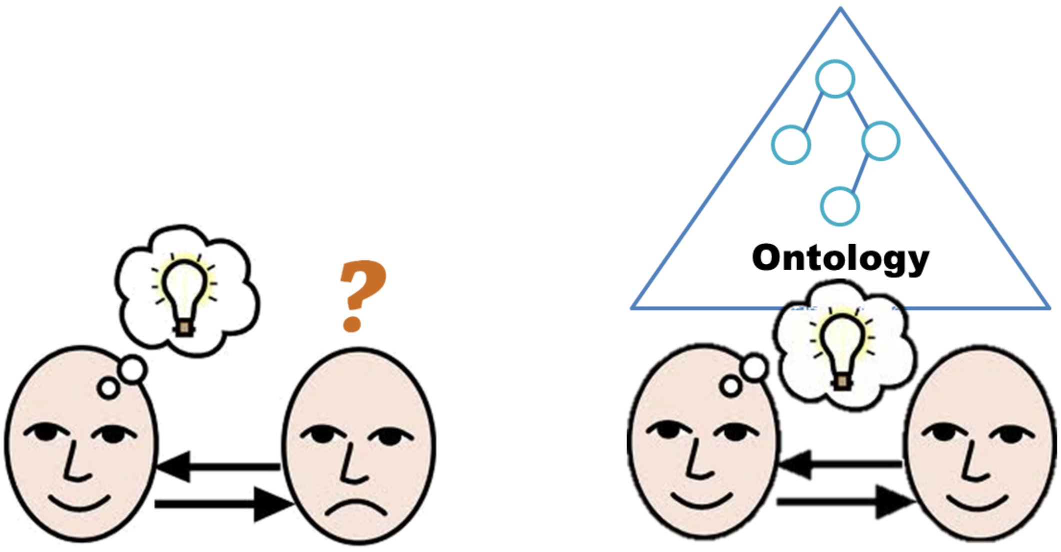


Figure 1.2: Ontology for Knowledge Sharing and Co-Creation

domain-specific associations.

Construction of ontologies is a major concern in *Knowledge Engineering (KE)*. Although KE is originally an artificial intelligent (AI) discipline to integrate knowledge into computer systems in order to solve complex problems normally requiring a high level of human expertise, many KE systems today have a deep involvement of human experts and organization in many phases of modeling and utilizing the knowledge in the domain. Therefore, KE systems exhibit both art and engineering features. KE involves various activities from diverse disciplines such as cognitive science [112], socio-cognitive engineering [107], ontological engineering [39], and service science.

1.2 Ontology and Semantic Web

The coordination of ontology and Semantic Web creates an ideal tool for many researches and applications in knowledge science even though they are initially developed separately. Ontology working group adopts semantic web elements to construct contemporary ontology languages which allow us to describe domain models and the current trend in the Semantic Web community is to utilize ontologies for expressing the meaning of information in the Web. Figure 1.3 shows some typical examples of research and applications of the duet in various topics of researches and applications in Knowledge Science.

Ontologies had been created before the Semantic Web concept was introduced, firstly using AI modeling techniques based on frames and first-order logic (FOL). Cyc ontology [72], one of the first reputable ontologies, expresses a large knowledge base of commonsense knowledge using CycL, a formal language based on FOL. Ontolingua [33] is a represen-

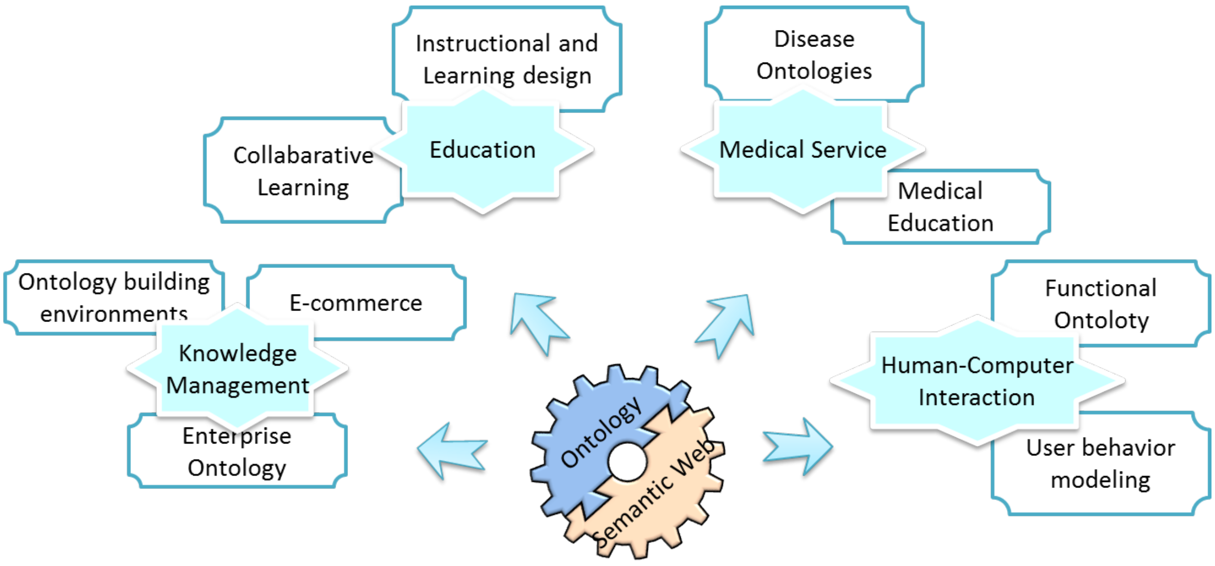


Figure 1.3: Ontology and Semantic Web for Knowledge Engineering

tation language based on KIF (Knowledge Interchange Format) [36], a language for the interchange of knowledge between heterogeneous software systems, with the addition of a frame ontology. The traditional representation formalism like FOL has two drawbacks: it can be confusing even for many software developers and computational cost of practical system is too expensive. With the rapid development of Semantic Web techniques in the last few years, knowledge representation languages based on description logics (DL) [11], a subset of FOL with sufficient expressive power and rational computational expense, like OIL [34], DAML-ONT [81], DAML+OIL [21], OWL [24], and OWL 2 [23, 109], have been emerging in the context of Semantic Web and OWL 2 becomes the recommendation of the W3C Web Ontology Working Group for ontology language.

The Semantic Web, which was presented by Tim Berners-Lee and extensively developed from 2001 [15, 7, 54], is a vision for transforming the World Wide Web into a semantically rich web of information to enable a distributedly massive intelligence in Internet environment. The World Wide Web currently consists mostly of linked hypertext documents, which are readable for people but are understandable for computers because they are written in the natural language. Instead, computer programs can only treat texts as streams of words, which are not connect with their meaning. The Semantic Web, just as the Internet and the World Wide Web, is decentralized and open for everyone to add content. Moreover, people publish content in various forms such as text, images, sound and video. The Web also spans across languages and cultures. This makes it very hard to capture the meaning of each piece of information made public on the Web by individuals around the world. It is necessary for the Semantic Web to utilize a formal representation

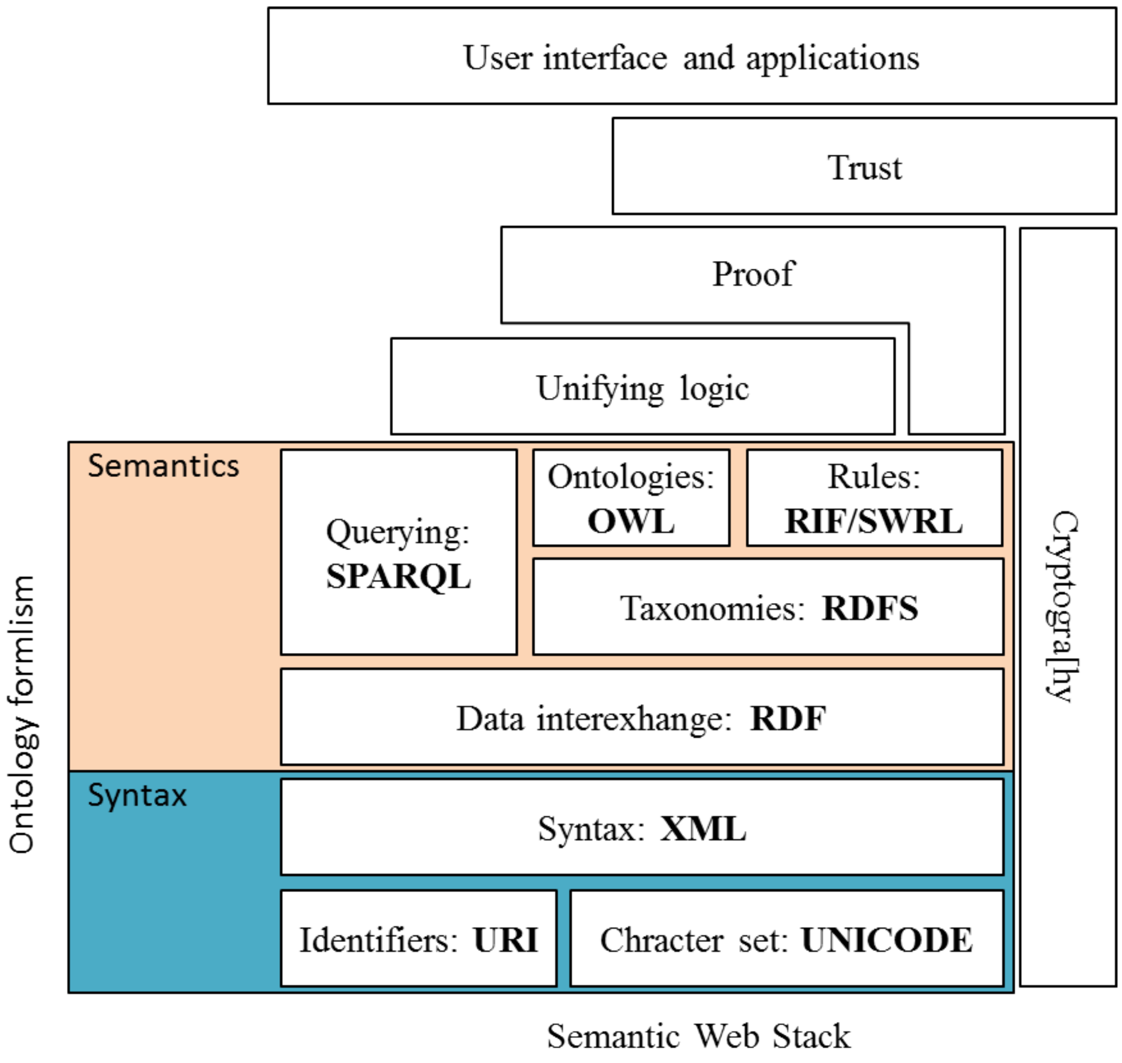


Figure 1.4: Ontology and Semantic Web Stack

such as ontology to present information that is understandable and manipulatable by computers.

As discussed above, ontology and Semantic Web are mutually interacting to promote researches and applications in each field and other areas of knowledge engineering and knowledge science. Figure 1.4, which is modified from the schema first proposed by Tim Berners-Lee[1], clarifies the interaction between them by showing the use of Semantic Web layers in current ontology formalism.

The white part of the figure presents the main layers the Semantic Web Stack and the color part presents two main component of an ontology formalism. The Semantic Web utilizes the same underlying technologies as the current World Wide Web. Uniform Resource Identifier (URIs) to identify web resources, Unicode as the standard character encoding and XML as a universal data format are already the dominant standards. These

[1]`http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html`

two layers provide syntactical units to build ontology.

The next three layers provide building blocks to represent meaningful information in the ontology. The contemporary ontology languages are primarily using RDF (Resource Description Framework) [105] as the underlying data model. RDF is a very general data model for describing resources and relationships between them. The RDF data model is built with statements in the form of object-attribute-value triples. These triples also allow efficient store of the data model on a distributed environment like Internet because it does not need a unified schema like in the case of relational database [1]. The primary syntax of RDF is based on XML [35]. However, other syntaxes could be used to express the RDF data model. RDF Schema [19] specializes RDF for describing properties and classes of RDF resources. It provides the means to express hierarchies of these classes and properties. RDF Schema explicitly defines the "is subclass of" relationship, which does not exist in RDF. OWL (Web Ontology Language) [109] is semantically and syntactically a further specialization of RDF. It provides an expressive language for describing relationships between classes and individuals.

Apart from representation, *formal semantics* is needed to precisely define the meaning of knowledge of ontology. The semantics of ontologies is usually based on a logic formalism, such as FOL or DL. Formal semantics is needed to make each statement unambiguous. An ontology language has to support efficient reasoning, which is enabled by the formal semantics of the language. Reasoning support is required among others in order to check the inconsistency of a knowledge base and to answer queries. For example, the OWL ontology language is based on description logic and thus existing DL reasoners can be employed. A sufficient *expressive power* is also required to describe as many aspects of a domain as possible. Every ontology language will have its limitation as to the knowledge it can express. However, expressivity is always increased at the cost of computational complexity reasoning.

1.3 Conflict Knowledge and Inconsistent Ontologies

Basic elements of ontology include individual, concept, role, and axiom. These elements correspond to a particular entity in a domain, a set of entities, a relationship among individuals or concepts, and logical statement to explicitly represent known truth about the domain, respectively. Conflict knowledge can be encountered in all levels of element. The conflict in individual, concept and role often occurs when integrate multiple ontologies due to the homonyms or synonyms of vocabulary to express the terms, for example, a 'plant' can be both a tree or a manufactory. The reason for this problem is the different convention of naming elements among organizations. It brings back the heterogeneous problem to ontology and can be addressed by syntactical approaches to match elements

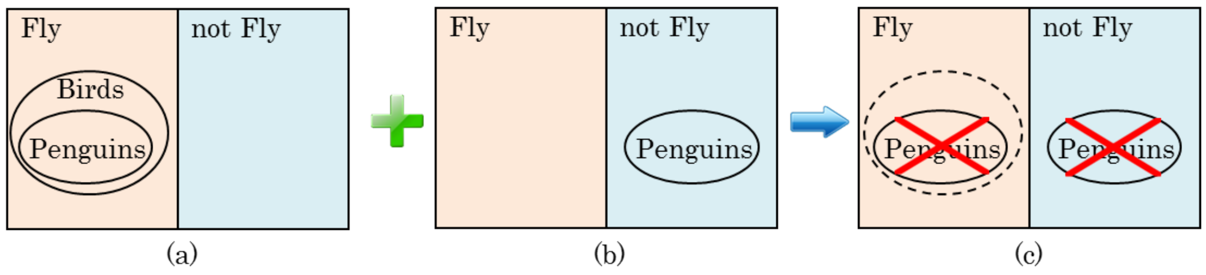


Figure 1.5: A demontration of Inconsistent Ontologies

among ontologies [32, 86]. Hyvönen [57] argues that we should prevent instead of solve heterogeneous problem by proposing a set of principles for ontology development.

On the other hand, conflict knowledge at axiom, or logical, level make an ontology inconsistent and possible to infer invalid knowledge. Let's consider the following example. A bird ontology can contain axioms saying that 'birds fly' and 'penguins are birds'. From those axioms, implicit knowledge such as 'penguins fly' can be deduced by ontology reasoning process. This example is illustrated in Figure 1.5(a). Assume that bird ontology has another axiom saying that penguin does not fly, as depicted in Figure 1.5(b). Then concept 'penguins' is unsatisfiable or inconsistent because it belongs to two mutually disjoint concepts 'fly' and 'not fly' concurrently, as shown in Figure 1.5(c). This situation is not acceptable according to the law of non-contradiction in classical logic, the underlying logic of current ontology formalism.

According to the principle of explosion (or *ex contradictione quodlibet*) of underlying classical, anything can be inferred from contradictory premises. For example, we can infer that 'penguins are human' from an unsatisfiable concept such as penguins in above example. There are two main ways to deal with inconsistency in ontologies: one is to diagnose and repair it when it is encountered; another is to avoid the inconsistency and to apply a non-standard inference relation to obtain meaningful answers [38]. Although common ontology editors such as Protégé provide function to identify the potential axioms that can cause conflict, performing repairing ontologies can be very difficult. For example, we do not have enough rights to modify remote ontology or even expert ontology engineers also find it confusing to work out the underlying errors [114].

The latter approach to scope with inconsistency by utilizing a non-standard inference relation. This approach has attracted considerable interest in recent years because inconsistency in unavoidable due to the nature of knowledge, as indicated in many studies. A famous example of inconsistency from theoretical aspect is Gödel's first incompleteness theorem which states that for any consistent axiomatic theory of arithmetic, which can be recognized to be sound, there will be an arithmetic truth not provable in it, but which can be established as true by intuitively correct reasoning. The appearance of inconsistency many practical knowledge-based applications like knowledge management is also certain. There are two common cases that causes the inconsistency in practical applications. The first case is when the ontology contains some default knowledge such as bird ontology in above example. Default knowledge is the knowledge that is generally true but false in some exception. The second case is when we migrate the knowledge from another resource or integrate multiple sources of knowledge. Those knowledge sources can be locally consistent but contain conflict knowledge with the others.

This research aims to solve the problem of inconsistent ontologies by providing more tolerant reasoning ability for ontology. Using proposed reasoning approach, meaningless answers is removed while meaningful answers are retained. For example, if the ontology includes axiom 'birds have wings', the conclusion 'penguins have wings' is obtained. Due to the necessary of non-monotonic reasoning, a great amount of research has been devoted to propose conflict representation and reasoning approaches. Typical works of non-monotonic logics include paraconsistent logic [93], default logic [69], among others. Those works overcomes the principle of explosion or avoid inconsistency by extending the representation of classical propositional logic. Those approaches have been integrated with ontology reasoning in various related researches.

In this research, we propose an extended representation of DL ontology, named as *defeasible DL ontology*, to express conflict knowledge and *defeasible reasoning* approach to process this knowledge. The extended representation is based on the integration of defeasible logics into DL ontology formalism. Among non-monotonic reasoning approaches, defeasible logics [88, 89] standout for their deeply theoretical foundation and efficiently practical applications. They have been studied deeply in term of proof theory [4], semantics [42, 78] and its relation to logic programming [5, 6]. The use of defeasible logics and related approaches in various application domains has been advocated, including the modeling of regulations and business rules [45], modeling of contracts [98, 41], legal reasoning [44], agent negotiations [29], and agent modeling [43]. Defeasible logics are also introduced in semantic web [13, 3]. Those applications are supported by efficient implementation systems [3, 67]. Defeasible inference in propositional form of logic can be performed in linear time as proved in [78].

Our proposed extension adds two features to DL ontology: defeasible axiom and priority relation. *Defeasible axiom* provides a mean to represent knowledge that can be retracted when there is contradict evidence. In defeasible DL ontology, defeasible axiom is used to represent the meaning such as 'most birds fly', which is depicted in Figure 1.6(a). In this figure, we also see the inference result when integrating new defeasible axiom with classical axiom 'penguins are birds'. The conclusion is 'most penguins fly' and can be defeated by the classical axiom 'penguins do not fly' in Figure 1.6(b). As a

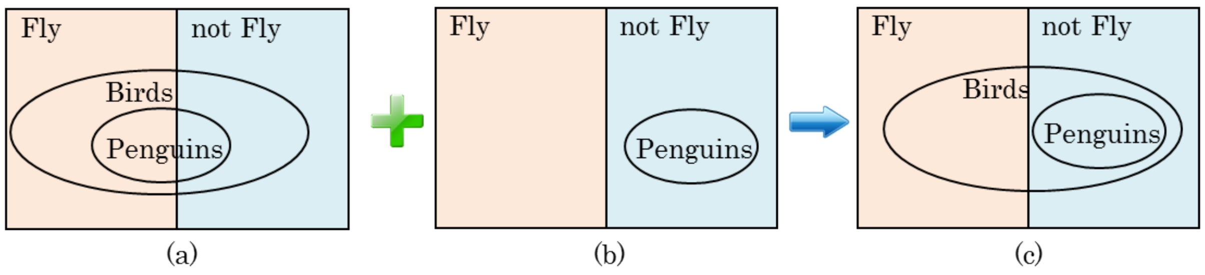


Figure 1.6: Defeasible axiom to resolve inconsistency

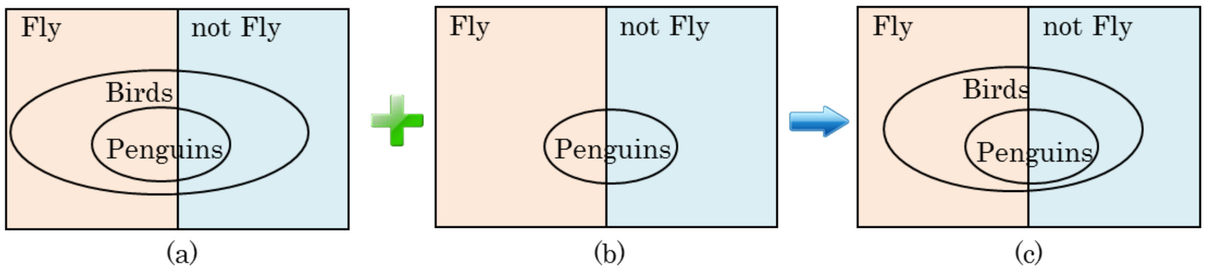


Figure 1.7: Conflict among defeasible axioms

consequence, ‘penguins’ is satisfiable in defeasible DL ontology, as shown in Figure 1.6(c).

Defeasible reasoning is skeptical and *priority relation* is used to assign the preference among axioms to resolve conflict among defeasible knowledge. Defeasible reasoning is skeptical because it rejects both conclusions if two conflict conclusions can be inferred. To demonstrate this principle, assume that we replace the classical axiom in Figure 1.6(b) by defeasible axiom ‘most penguins do not fly’ in Figure 1.7(b) (since we may expect that some gene modified penguins or super penguins can fly). The reasoning result of new ontology is illustrated in Figure 1.7(c). We have a half-half situation where ‘most penguins fly’ and ‘most penguins do not fly’ occur simultaneously. Without preferential information, defeasible reasoning removes both conclusions. We can assign priority relation among conflict axioms to make defeasible reasoning generate desired result. For example, specifying that ‘most penguins do not fly’ is stronger than ‘most birds fly’ makes defeasible reasoning infer the former as a conclusion and discard conclusion ‘most penguins fly’.

1.4 Research Contributions

Figure 1.8 illustrates the roadmap to knowledge science in our research that aims to harmonize the conflict knowledge to accurate and alleviate the knowledge sharing and co-creation process. The blue paths show our research contributions while the gray ones show alternative approaches for encountered problems. We first examine the ability of contemporary ontology specification for knowledge sharing and co-creation by creating a comprehensive ontology about educational program of School of Knowledge Science of JAIST that helps the students to understand the knowledge in educational guideline and build their own knowledge about studying plan. The constructed model shows that current ontology language is sufficient and efficient enough to handle complicated situations in practical knowledge sharing and co-creation application.

Building and utilizing the ontology in an practical knowledge sharing application also reveals conflict knowledge which causes the inconsistencies in ontology. In this research, we propose a defeasible reasoning approach to handle inconsistent ontologies. The proposed approach is applied on the ontology layer of a DL ontology so that it can inherit the semantic expressiveness of DL ontology and computational efficiency of defeasible reasoning. The green paths present our future works to scope with contextual knowledge, i.e. the knowledge that has the meaning depending on the context or viewpoint, which is the more general case than conflict knowledge. We believe that kind of knowledge is a appropriate form to encode knowledge in more complicated scenario in knowledge sharing and co-creation.

Compared with other related works, our approach has simplicity and flexibility, expressive power, and computational efficiency as the advantages. Therefore, this research can help to expand the applicability of ontology and knowledge representation in knowledge sharing and co-creation applications. A short comparison is given as follows and more detail discussion can be found in chapter of related works .

Owning to its simplicity, flexibility and efficiency, defeasible proof theory has been introduced to DLs by other research such as in [40, 2, 91]. However, defeasible logic relies on a proof-based semantics, which is not fully compatible with model-based semantics of description logic. This limitation prevents the application of defeasible reasoning on the ontology reasoning. Current work on defeasible description logic [40, 91] converts DL axioms into propositional rules and perform reasoning with logic programming techniques. This approach shows some restrictions on the expressiveness and efficiency. This approach relies on the semantics of propositional logic, which is much “weaker” than those of current DLs such as proposed in [23], causing much difficulty in extending the semantics of proposed defeasible DL. Transformation from a DL ontology to logic program increase the size of theory and thus significantly affect the reasoning complexity. A similar work

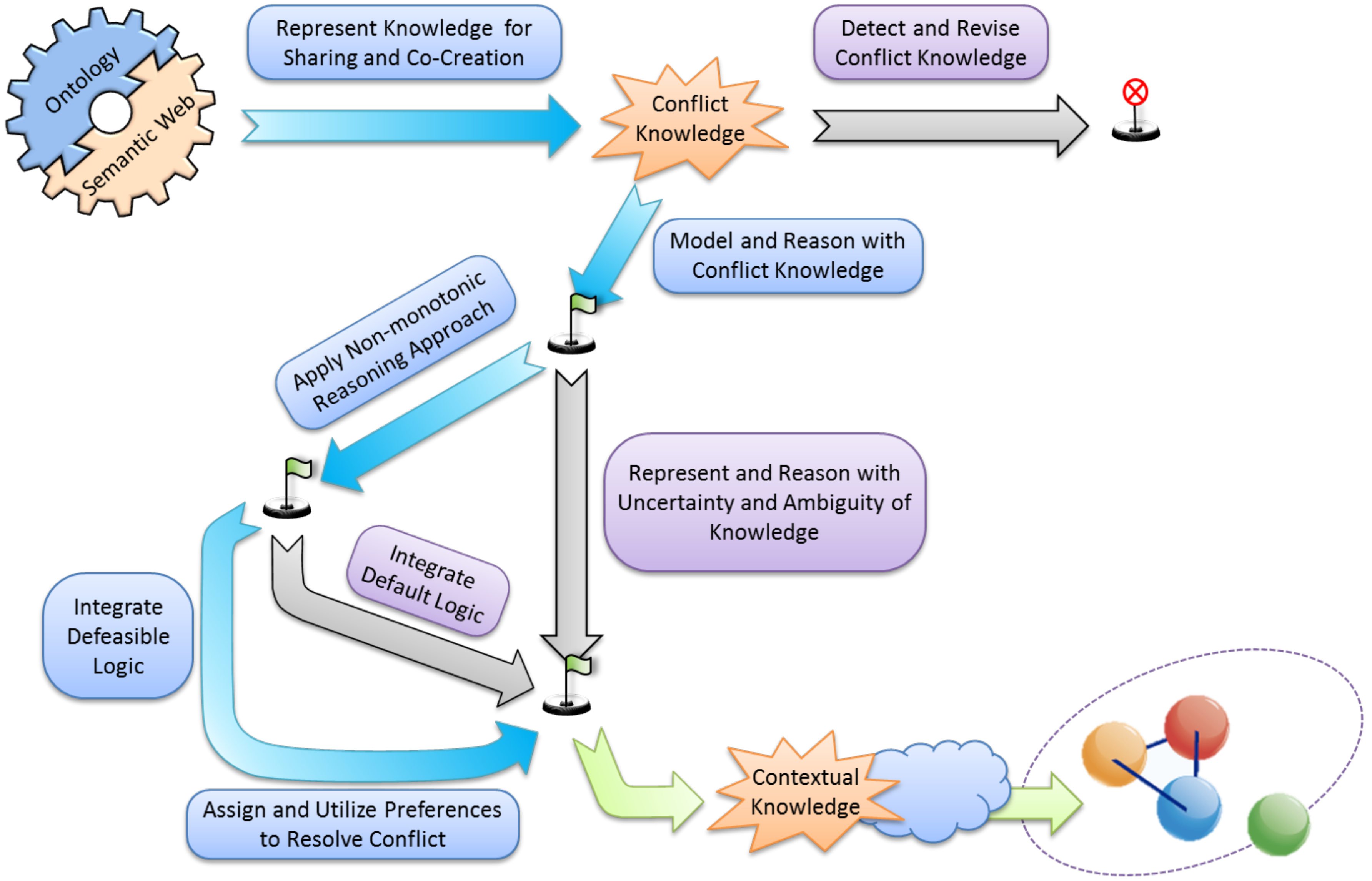

Ontology
Semantic Web
Represent Knowledge for Sharing and Co-Creation
Conflict Knowledge
Detect and Revise Conflict Knowledge
Model and Reason with Conflict Knowledge
Apply Non-monotonic Reasoning Approach
Represent and Reason with Uncertainty and Ambiguity of Knowledge
Integrate Default Logic
Integrate Defeasible Logic
Assign and Utilize Preferences to Resolve Conflict
Contextual Knowledge

Figure 1.8: A Road Map to Knowledge Science

is introduced in [38] that translates a DL ontology into defeasible logic program DeLP and perform argumentative reasoning by an justification abstract machine. However, this approach lack of expressiveness power of DL ontology and computing argumentation in logic program is also known to be untraceable. Variants of paraconsistent logic are also integrated with DL ontology in [61]. Zhang et al. [117] introduces an extension of tableau algorithm for $\mathcal{ALC}$ to handle paraconsistent and nonmonotonic reasoning in ontology. Those works have the similar drawback due to the lack of expressiveness of extended languages.

Another approach to handle inconsistency is modeling the uncertainty and incompleteness of knowledge. Some typical works of uncertainty representation and reasoning in ontology which can also be applied to handle conflict knowledge are probabilistic DL [74], possibilistic DL [97, 95], Bayesian network in OWL [27], Fuzzy OWL [18], and representation of belief network in OWL [31]. Those researches provide flexible frameworks to scope with inconsistent ontologies and have powerful expressiveness. However, those works make the ontology language more complicated with new concepts of uncertainty. It is also difficult for ordinary ontology modeler to determine the value of uncertainty in practice.

The contributions of this research are as follows:

- We demonstrate the applicability of present ontology formalisms in knowledge sharing and co-creation that can fulfill the complicated requirements in real-world application.
- We propose defeasible DL ontology, an extension of DL ontology which has the ability to represent and reason with conflict knowledge.
- Based on defeasible proof theory, we introduce a reasoning approach for defeasible DL ontology and a formal interpretation of defeasible semantics.
- We introduce heuristics to automatically determine the preferences among knowledge to resolve conflict to facilitate knowledge sharing.
- We implement a prototype of defeasible DL as a Protégé plugins to demonstrate and evaluate the applicability of the proposed approach.

The remaining of this dissertation is organized as follows. Chapter 2 introduce constructed ontology for educational program of School of Knowledge Science as well as formal foundation of description logic ontology. Chapter 3 summarizes and discusses about relevant knowledge and features about defeasible logic which is integrated into our framework to handle conflict in inconsistent ontologies. The formal definition of proposed framework is presented in detail in Chapter 4 and the implementation and evaluation in Chapter

5. Chapter 6 discusses about alternative approaches to model and reason with conflict knowledge in ontologies. Chapter 7 concludes this dissertation and considers future works for our research.

Chapter 2

Description Logics Ontology

2.1 Introduction to Description Logics

Description Logics (DLs) [66] are a family of knowledge representation (KR) that provides the underlying semantics for the current recommended ontology language, OWL. Although developed from other KR formalisms such as semantics web and frames, DLs differentiate from their predecessors by their formal, logic-based semantics. This distinguished feature allows DLs to focus on their reasoning ability, which can infer implicitly represented knowledge from the knowledge that is explicitly contained in the knowledge base [23].

As indicated in their name, DLs describe the important notion of domain by the concept *descriptions*. The building blocks of these expressions are *atomic concepts*, which denotes a set of individuals, and *atomic roles*, which denotes binary relationships between individuals. Complex descriptions are constructed from atomic ones inductively with concept constructors. The set of allowed constructors depends on the language of a particular DL system but the following constructors are supported by most of DLs including *concept conjunctions*, *concept disjunctions* and *concept negations*.

Assume we want to construct a knowledge-based system of educational program of Doctoral Course in School of Knowledge Science. Then the set of atomic concepts of the domain may include $\mathsf{Professor}$, $\mathsf{Student}$, and Course which relate to the ordinary definition of those entities. It also consists of other concepts that express more specific notions such as 'student of School of Knowledge Science', $\mathsf{KSStudent}$, or 'doctoral student', $\mathsf{DoctorStudent}$. The set of atomic roles contains $\mathsf{supervises}$, which denotes the relationship that a professor is the main supervisor of a student, and passes, which denotes the relationship that a student passes a lecture.

By using concept constructions introduced above, we can define complex concepts such as 'KS doctoral student' by $\mathsf{KSStudent} \sqcap \mathsf{DoctorStudent}$ with concept conjunction, 'a doctor or master student' by $\mathsf{DoctorStudent} \sqcup \mathsf{MasterStudent}$ with concept disjunction, and 'not doctoral student' by $\neg\mathsf{DoctorStudent}$ with concept negation.

Concept descriptions are used to build two types of logical statements, or axioms, in DL knowledge: terminological and assertional statements. Terminological statements describe the properties of concepts and roles, and relationships between thems. The general form of terminological statements is *general concept inclusion (GCI)* axiom, which has the form $C \sqsubseteq D$ where C and D are concept descriptions and it is read as "C is subsumed by D". For example, the statement that "all students of School of Knowledge Science are JAIST students and JAIST students are student" is represented by the following axioms

$$\begin{aligned}&\mathsf{KSStudent} \sqsubseteq \mathsf{JaistStudent},\\ &\mathsf{JaistStudent} \sqsubseteq \mathsf{Student}.\end{aligned}$$

Another form of terminological axiom is *equality*, denoted as $C \equiv D$, which is equivalent to $C \sqsubseteq D$ and $D \sqsubseteq C$. If C is an atomic concept, this axiom is called a *definition*. For example, the statement "As a graduate school, a regular student of JAIST is either a doctor or a master student" is represented by two axioms

$$\begin{aligned}&\mathsf{JaistStudent} \sqsubseteq \mathsf{DoctorStudent} \sqcup \mathsf{MasterStudent},\\ &\mathsf{DoctorStudent} \equiv \neg\mathsf{MasterStudent}.\end{aligned}$$

The former states that a regular JAIST student can be one of two types, a master or a doctoral student, and the latter (in the context of JAIST) states that these two types are mutually disconnected, which implies an individual can not belongs to both concepts concurrently.

Assertional statements describe a concrete situation by stating properties of individuals or relationship among individuals. For example, the assertions

$$\mathsf{KSProfessor(YAMADA)}, \quad \mathsf{supervises(YAMADA, SUZUKI)}, \quad \mathsf{passes(SUZUKI, K612)}$$

state that Yamada is a Professor of School of Knowledge Science, that he is the main supervisor of Suzuki and that Suzuki passes the course K612.

DL statements are typically separated into two parts, *terminological part TBox* and *assertional part ABox*. For more expressive ontology, *role box RBox* is added to describe the properties of roles or relationship among roles. TBox corresponds to the schema in the database setting while ABox to the data. But in contrast to the database setting, ontology inference takes into consideration both terminological statements and the assertional statements to provide reasoning services. These services will be discussed thoroughly in Section 2.4.

This sections so far has introduced basic syntax of DLs and we will discuss about an important notion in the domain of educational program in School of Knowledge Science, Courses, before go into detail of syntax and semantics of DLs. Courses in School of Knowledge Science are considered with respect to two aspects, level aspect and area aspect. According to the area aspect, courses are categorized into 3 areas as *Social Knowledge* (A), *Knowledge Media* (B), and *Systems Knowledge* (C). Besides, some courses such as K228 *Introduction to Knowledge Science I* can be categorized as any of those three areas. Thus, we consider these courses as *Multi-Area Courses* and create a new concept for them. The formulation of this categorization is as follows[1]:

$$\mathsf{KSCourse} \equiv \mathsf{SocialCourse} \sqcup \mathsf{MediaCourse} \sqcup \mathsf{SystemCourse} \sqcup \mathsf{MultiAreaCourse}$$

With respect to the level aspect, courses are classified as *Introductory Lectures, Basic Lectures, Intermediate Lectures* and *Advanced Lectures and Seminars.* The classification corresponds to the level of students: generally, the three former lectures are for master students and the last ones for doctoral students. Among those lectures are *English (Technical Communication)* courses that is considered in a different way. Therefore, a new concept for these lectures is necessary for representing and processing these knowledge. We define the concept $\mathsf{KSLecture}$, which is indeed equivalent to $\mathsf{KSCourse}$, to describe this partition as below

$$\mathsf{KSLecture} \equiv \mathsf{IntroLecture} \sqcup \mathsf{BasicLecture} \sqcup \mathsf{InterLecture} \sqcup \mathsf{AdvanLecture} \sqcup \mathsf{EnglishLecture}$$

$$\mathsf{KSCourse} \equiv \mathsf{KSLecture}$$

2.2 Syntax and Semantics of Description Logics

2.2.1 Syntax

A member of DLs family is distinguished and named by the constructors it provides. The simplest and much studied expressive DL is $\mathcal{ALC}$ (Attributive language with Complement) which provides concept conjunction, concept disjunction, concept negation, universal restriction, and existential restriction. Two notable extensions of $\mathcal{ALC}$ DL are $\mathcal{SHOIN}$, which is the basic of OWL ontology language, and $\mathcal{SROIQ}$, which is the basic of OWL 2 ontology language. $\mathcal{SHOIN}$ DL extends $\mathcal{ALC}$ DL with simple hierarchical roles, nominals, inverse roles and (unqualified) number restrictions. $\mathcal{SROIQ}$ DL in turn extends $\mathcal{SHOIN}$ with a full role box RBox and qualified number restrictions. [62]

Because the requirements of KS Ontology is limited as discussed later, we assume an empty RBox in this research but the results of proposed approach can be easily extend

[1]For the sake of brevity, the prefix KS in concept name is omitted where it does not cause any ambiguity.

to Rbox in general. Therefore, we consider $\mathcal{SHOIQ}$, a DL that is more expressive than $\mathcal{SHOIN}$ but less expressive than $\mathcal{SROIQ}$. A concept description in $\mathcal{SHOIQ}$ is defined by the following syntactic rule:

$$\begin{array}{rcll}
C, D & \longrightarrow & A \,| & \text{(atomic concept)} \\
 & & \top \,| & \text{(top concept)} \\
 & & \bot \,| & \text{(bottom concept)} \\
 & & \neg C \,| & \text{(concept negation)} \\
 & & \{o\} \,| & \text{(nominals)} \\
 & & C \sqcap D \,| & \text{(concept conjunction)} \\
 & & C \sqcup D \,| & \text{(concept disjunction)} \\
 & & \exists R.C \,| & \text{(existential restriction)} \\
 & & \forall R.C \,| & \text{(universal restriction)} \\
 & & \geq nR.C \,| & \text{(qualified at-least restriction)} \\
 & & \leq nR.C & \text{(qualified at-most restriction)}
\end{array}$$

where A is a atomic concept name, o an individual, C and D concepts, $\top$ abbreviation for $C \sqcup \neg C$, $\bot$ abbreviation for $C \sqcap \neg C$, R a atomic role or its inverse, and n a non-negative integer. The meanings of the constructors and the letters associated with them are briefly explained and demonstrated as follows.

Universal restriction (or *value restriction*) allows us to pose range restriction of roles, it requires every individual that participates in a relationship must be a particular concept. For example, we may state that a student must be supervised by professor by the statement

$$\mathsf{Student} \sqsubseteq \forall \mathsf{supervisedBy.Professor}.$$

Existential restriction expresses that a relationship can be satisfied by at least one member of the concept of discourse. For example, the requirement that a student must pass at least one course of Advance Lectures and Seminars to graduate can be expressed by the statement

$$\mathsf{GraduateStudent} \sqsubseteq \exists \mathsf{passes.AdvanLecture}.$$

Universal and existential restrictions can be used to set the domain and range of role such as specified by the following axioms

$$\begin{array}{ll}
\exists \mathsf{supervises}.\top \sqsubseteq \mathsf{Professor} & \text{(domain restriction)} \\
\top \sqsubseteq \forall \mathsf{supervises.Student} & \text{(range restriction)}
\end{array}$$

These axioms state that $\mathsf{Professor}$ is the domain and $\mathsf{Student}$ is the range of $\mathsf{supervises}$.

The letter $\mathcal{S}$ is the abbreviation of $\mathcal{ALC}$ with *transitive role*, which can represents role like $\mathsf{locatedIn}$ which states that the location of a concept is within the other. Transitive role allows automatic implication such that if JAIST is located in Ishikawa and Ishikawa is located in Japan, JAIST is located in Japan.

Simple hierarchical role (indicated by the letter $\mathcal{H}$) allows role hierarchical statements such as

$$\mathsf{transfers} \sqsubseteq \mathsf{passes}$$

to describe that if a student transfer a valid course from his master program, he is considered to pass this course.

Nominals (indicated by the letter $\mathcal{O}$) provide us with a convenient way to define a concept by enumerating its members. The "set" (or *one-of*) constructor, i.e. $\{\}$-constructor, is used with individual names to construct the concept. For example, from requirement of course completion of School of Knowledge Science, the course of *Seminar on Knowledge Science B* contains only one course K601, which corresponds to the major research project. Thus, this concept can be expressed by $\{\mathsf{K601}\}$.

Inverse role (indicated by the letter $\mathcal{I}$) allows to use role axiom such as

$$\mathsf{supervises} \equiv \mathsf{supervisedBy}^{-1}$$

to state the $\mathsf{supervises}$ and $\mathsf{supervisedBy}$ have the reverse domain and range to each other.

Number restrictions (indicated by the letter $\mathcal{N}$) allow us to describe the number of relationship of a particular type that individuals can participate in. They are at-least restriction, written as $\geq nR$, and at-most restriction, written as $\leq nR$. For example, the restriction that a student has exactly one main supervisor can be represented by using both constructors in the statement

$$\mathsf{Student} \sqsubseteq\ \leq 1\mathsf{supervises}^{-1} \sqcap \geq 1\mathsf{supervises}^{-1}$$

which can be abbreviated as

$$\mathsf{Student} \sqsubseteq\ = 1\mathsf{supervises}^{-1}.$$

With *qualified number restrictions* (indicated by the letter $\mathcal{Q}$), we can additionally describe the type of individuals that are counted by a given number restriction. For example, the requirement that a student must pass at least 5 courses to graduate is represented by

$$\mathsf{GraduateStudent} \sqsubseteq\ \geq 5\mathsf{passes}.\mathsf{Course}.$$

With respect to the assumption of empty RBox, a $\mathcal{SHOIQ}$ ontology $\Sigma = (\mathcal{T}, \mathcal{A})$ consists of two finite and mutually disjoint sets: the TBox $\mathcal{T}$ which introduces the terminology and the ABox $\mathcal{A}$, which contains facts about particular objects in the application domain. TBox statements have the form $C \sqsubseteq D$, concept subsumption, or $C \equiv D$, concept equivalence. The ABox is referred to by a finite number of individual names and these names may be used in two types of assertional statements: *concept assertions* of the type $C(a)$ and *role assertions* of the type $R(a, b)$.

In $\mathcal{SHOIQ}$ ontology we can simulate the ABox assertions with TBox axioms by using named individuals together with the {}-constructor as follows:

$$C(a) \Leftrightarrow \{a\} \sqsubseteq C$$
$$R(a,b) \Leftrightarrow \{a\} \sqsubseteq \exists R.\{b\}$$

This simulation, which is applied in [50], is helpful to reduce all ontology reasoning services to concept checking. If the role of ABox is not necessary in the discussion, it can be omitted in upcoming sections and chapters.

2.2.2 Semantics

The description language has a model-theoretic semantics which is defined in terms of interpretation. An *interpretation* of an ontology is a tuple $\mathcal{I} = \langle \triangle^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\triangle^{\mathcal{I}}$ is a non-empty set called the *domain*, and $\cdot^{\mathcal{I}}$ is an *interpretation function* mapping each concept name A to subset $A^{\mathcal{I}}$ of $\triangle^{\mathcal{I}}$, and each role name R to binary relations $R^{\mathcal{I}}$ over $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$:

$$A^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}}, R^{\mathcal{I}} \subseteq \triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}, \top^{\mathcal{I}} = \triangle^{\mathcal{I}}, \bot^{\mathcal{I}} = \emptyset$$
$$\text{for all individual name } o, \#\{o\}^{\mathcal{I}} = 1$$

Given an interpretation $\mathcal{I} = \langle \triangle^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, $\cdot^{\mathcal{I}}$ is extended to interpret complex concepts in the following way:

$$(\neg C)^{\mathcal{I}} = \triangle^{\mathcal{I}} \setminus C^{\mathcal{I}}, \quad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \quad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$\begin{aligned}
(\exists R.C)^{\mathcal{I}} &= \{x \mid \exists y \text{ such that } (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\},\\
(\forall R.C)^{\mathcal{I}} &= \{x \mid \forall y, (x,y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\},\\
(\geq nR.C)^{\mathcal{I}} &= \{x \mid \#\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\},\\
(\leq nR.C)^{\mathcal{I}} &= \{x \mid \#\{y \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\},
\end{aligned}$$

where $\#N$ is the number of elements in N.

Given the definition of interpretation, semantics of DL is defined as follows. An interpretation $\mathcal{I}$ satisfies $C \sqsubseteq D$ or $C \equiv D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ or $C^{\mathcal{I}} = D^{\mathcal{I}}$, respectively. An interpretation $\mathcal{I}$ satisfies the assertion $C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and it satisfies $R(a,b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a model of a DL (TBox or ABox) statement φ iff it satisfies the statement, and is a model of a DL ontology Σ iff it satisfies all statements in

Σ. A concept C is *satisfiable* with respect to Σ if there exists a model of Σ such that $C^{\mathcal{I}} \neq \emptyset$. Otherwise, C is *unsatisfiable* or *inconsistent*. An ontology is inconsistent if it contains unsatisfiable concept. A DL ontology Σ entails a DL statement φ, written as $\Sigma \models \varphi$, iff every model of Σ is a model of φ.

2.3 An ontology for Educational Program in School of Knowledge Science

2.3.1 KS Ontology

A comprehensive ontology, named KS Ontology, is constructed to represent and manage knowledge about educational program in School of Knowledge Science. The overall model is illustrated in Figure 2.1 and the detail knowledge of this ontology is partially represented in Figure 2.2 and 2.4. The figures focus on representing the course and credit requirements for doctoral student, which is shown in Figure 2.3. The requirements for a doctoral student to be qualified are quite complicated. Understanding those requirements is one of the main concern of freshmen when entering the program. They often consult their seniors about this problem. However, it also costs much time for them to understand the requirements to prepare and submit the enrollment just after the entrance. This motivates us to construct this ontology to ease the burden of the newcomers. This example demonstrates the difficulty when one participant utilize the shared knowledge from other participant to create new knowledge in a knowledge co-creation process.

The application of ontological research in education is proposed by [83] and becomes one of the most fashioned and rapidly evolving in the field. Typical work of this trend is performed by [9], as well as in [25] for the more specific domain of Web-based Intelligent Systems. Recently, Dicheva [26] develops an ontology-driven web portal that provides a single network place, where researchers, students, and practitioners can find information about available research projects and successful practices in this field. Regarding to knowledge sharing application, [48] proposes a cross-language semantic model (SEMCL) for knowledge sharing, which uses semantic web technologies to provide a potential solution to the problem of ambiguity which can match knowledge descriptions in diverse languages. This model employs an ontology as an underlying layer to overcome difficulties of cross-language information retrieval. Those related works focus of knowledge retrieval that exploits the semantic relations provided by ontology to perform searching knowledge. However, they do not show the use of expressive power of ontology to derive implicit knowledge of the ontology. Our approach considers mainly expressive features of contemporary ontology language to maximize the use of reasoning ability of ontology itself.

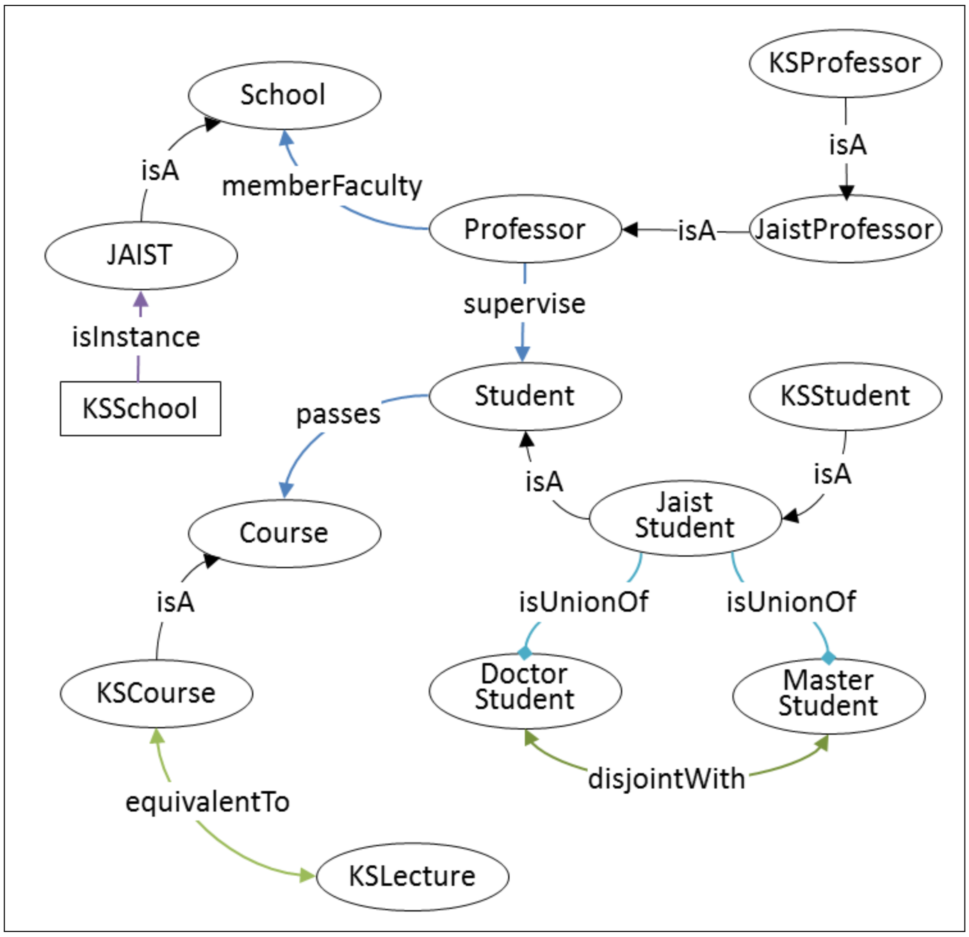


Figure 2.1: Top level concepts of the Ontology for School of Knowledge Science

The created ontology is an expressive ontology which represents knowledge in logical axioms so that the implicit knowledge can be derived through ontology reasoning process and used to create new knowledge for student by answering interest questions from the user such that:

– Is a student qualified to get a doctoral degree from School of Knowledge Science?
– Does a student's enrollment plan satisfy the course requirements?
– Given a list of course that student has passed, which course does he should take to satisfy the requirements?
– etc.

In order to answer above questions, the KS ontology defines general concepts of Professor, Student, and Courses, and the relationships among them, as shown in Figure 2.1. These general concepts are used to represent detail concepts in interest that is KSProfessor, KSStudent, KSCourse and KSLecture, among which the latter two are actually equivalent. The figure also shows the partition of regular JAIST students into solely two type of graduate students, master and doctoral students, by *isUnionOf* and *disjointWith* relationships.

The KS ontology $\Sigma_{KS} = (\mathcal{T}_{KS}, \mathcal{A}_{KS})$ consists of a TBox $\mathcal{T}_{KS}$ and an ABox $\mathcal{A}_{KS}$, which represents the detail knowledge of the domain. The TBox provides definitions of interest concepts, represented the requirements in an explicit and logical fashion. The ABox plays a role as a database which contains relevant information about interest objects in the domain and can be used to derived answers to specific questions.

2.3.2 KS Ontology TBox

Figure 2.2 presents the essential excerpt of TBox $\mathcal{T}_{KS}$. The axioms from (1) to (9) have been introduced and discussed in sections 2.1 and 2.2. The most important axiom in TBox is the definition axiom (10), which establishes the conditions of a doctoral student of School of Knowledge Science to graduate. The requirements are expressed as six terms in the right hand side of the definition. The first two terms are trivial since qualified KS doctor have to be doctoral student of School of Knowledge Science. The remaining four terms represent four conditions of the requirement respectively.

Among four remaining terms, the first two terms $\exists\mathsf{passes}.\{\mathsf{K601}\}$ and $\exists\mathsf{passes}.\{\mathsf{K602}\}$ directly correspond to the requirements 1) and 2) in Figure 2.3.

The remaining two requirements, correspond to requirements 3) and 4), are quite complicated and should be broken down. They are expressed by two concepts DrCourseCompleted and DrEssentialCompleted in axiom (10) and the corresponding requirements are represented in axioms (11) and (15), respectively.

Due to the complexity of condition 3), definition of concept DrCourseCompleted is also decomposed into 2 concepts DrAreaCompleted and DrLevelCompleted corresponding to the requirements of course area completion and course level completion.

The requirement of course area completion is represented in the main statement of the requirement 3) in the Figure 2.3 that requires a student to take course in at least 2 areas. Since there are courses in particular area and courses in multiple areas, one of following two cases can occur for the student to satisfy this requirement:

– the student passes at least 1 course in 2 of 3 areas:

$$(\exists\mathsf{passes}.\mathsf{SocialCourse} \sqcap \exists\mathsf{passes}.\mathsf{MediaCourse}) \sqcup (\exists\mathsf{passes}.\mathsf{SocialCourse} \sqcap \exists\mathsf{passes}.\mathsf{SystemCourse}) \sqcup (\exists\mathsf{passes}.\mathsf{MediaCourse} \sqcap \exists\mathsf{passes}.\mathsf{SystemCourse})$$

– or the student passes at least 1 multi-area course and 1 course in 1 of 3 areas:

$$\exists\mathsf{passes}.(\mathsf{SocialCourse} \sqcup \mathsf{MediaCourse} \sqcup \mathsf{SystemCourse}) \sqcap \exists\mathsf{passes}.\mathsf{MultiAreaCourse}$$

The union of above two condition terms forms the definition (12) for the concept DrAreaCompleted.

Terminological Box $\mathcal{T}_{KS}$:

(1) $\mathsf{KSStudent} \sqsubseteq \mathsf{JaistStudent}$

(2) $\mathsf{JaistStudent} \sqsubseteq \mathsf{Student}$

(3) $\mathsf{JaistStudent} \sqsubseteq \mathsf{DoctorStudent} \sqcup \mathsf{MasterStudent}$

(4) $\mathsf{DoctorStudent} \equiv \neg\mathsf{MasterStudent}$

(5) $\mathsf{KSCourse} \equiv \mathsf{SocialCourse} \sqcup \mathsf{MediaCourse} \sqcup \mathsf{SystemCourse} \sqcup \mathsf{MultiAreaCourse}$

(6) $\mathsf{KSLecture} \equiv \mathsf{IntroLecture} \sqcup \mathsf{BasicLecture} \sqcup \mathsf{InterLecture} \sqcup \mathsf{AdvanLecture} \sqcup \mathsf{EnglishLecture}$

(7) $\mathsf{KSCourse} \equiv \mathsf{KSLecture}$

(8) $\mathsf{Student} \sqsubseteq\ = 1 \mathsf{supervises}^{-1}$

(9) $\exists \mathsf{supervises}^{-1}.\mathsf{KSProfessor} \sqsubseteq \mathsf{KSStudent}$

(10) $\mathsf{KSQualifiedDoctor} \equiv \mathsf{KSStudent} \sqcap \mathsf{DoctorStudent} \sqcap \exists \mathsf{passes}.\{\mathsf{K601}\} \sqcap \exists \mathsf{passes}.\{\mathsf{K602}\} \sqcap \mathsf{DrCourseCompleted} \sqcap \mathsf{DrEssentialCompleted}$

(11) $\mathsf{DrCourseCompleted} \equiv \mathsf{DrLevelCompleted} \sqcap \mathsf{DrAreaCompleted}$

(12) $\mathsf{DrAreaCompleted} \equiv ((\exists \mathsf{passes}.\mathsf{SocialCourse} \sqcap \exists \mathsf{passes}.\mathsf{MediaCourse}) \sqcup (\exists \mathsf{passes}.\mathsf{SocialCourse} \sqcap \exists \mathsf{passes}.\mathsf{SystemCourse}) \sqcup (\exists \mathsf{passes}.\mathsf{MediaCourse} \sqcap \exists \mathsf{passes}.\mathsf{SystemCourse})) \sqcup (\exists \mathsf{passes}.(\mathsf{SocialCourse} \sqcup \mathsf{MediaCourse} \sqcup \mathsf{SystemCourse}) \sqcap \exists \mathsf{passes}.\mathsf{MultiAreaCourse})$

(13) $\mathsf{DrLevelCompleted} \equiv \exists \mathsf{passes}.\mathsf{AdvanLecture} \sqcap (\geq 5\, \mathsf{passes}.\mathsf{DrLecture} \sqcup ((\exists \mathsf{passes}.\mathsf{EnglishLecture} \sqcap \geq 4\, \mathsf{passes}.\mathsf{DrLecture})))$

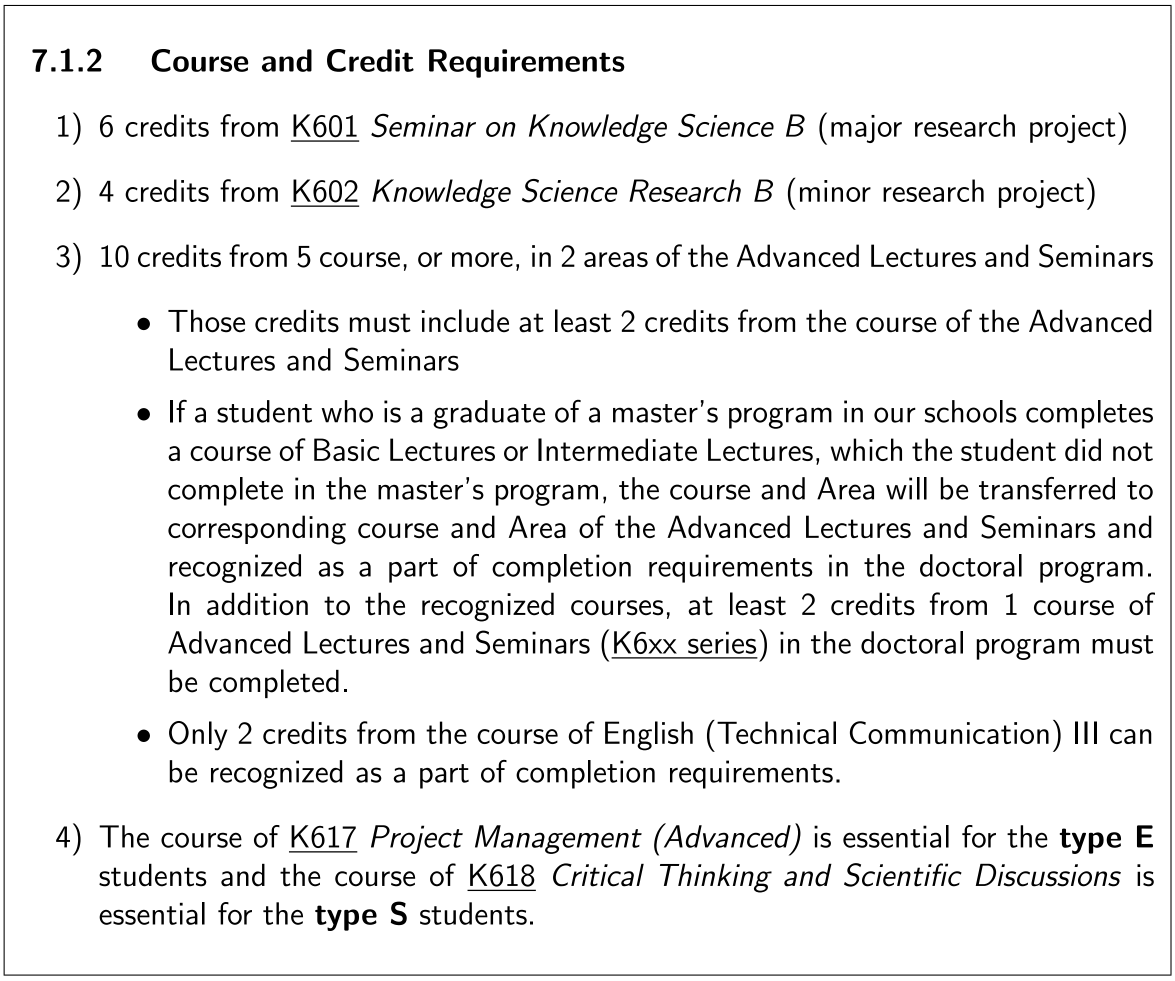
(14) $\mathsf{DrLecture} \equiv \mathsf{BasicLecture} \sqcup \mathsf{InterLecture} \sqcup \mathsf{AdvanLecture}$

(15) $\mathsf{DrEssentialCompleted} \equiv (\mathsf{TypeE} \sqcap \exists \mathsf{passes}.\{\mathsf{K617}\}) \sqcup (\mathsf{TypeS} \sqcap \exists \mathsf{passes}.\{\mathsf{K618}\})$

Figure 2.2: Terminological Box for Educational Program in Ontology for School of Knowledge Science

The requirement of course level completion, which defined in axiom (13) for the concept $\mathsf{DrLevelCompleted}$, comes from the analysis of the notes of requirement 3) as follows:

- The first note requires that the student must pass at least one course (2 credits) of Advanced Lectures and Seminars and corresponds to the term $\exists \mathsf{passes}.\mathsf{AdvanLecture}$.
- The second note implies that only courses of Basic Lectures or Intermediate Lecture or Advance Lectures and Seminars are valid for the doctoral program. We define a new concept $\mathsf{DrLecture}$ as the union of these three valid course levels use this



7.1.2 Course and Credit Requirements

1) 6 credits from K601 *Seminar on Knowledge Science B* (major research project)
2) 4 credits from K602 *Knowledge Science Research B* (minor research project)
3) 10 credits from 5 course, or more, in 2 areas of the Advanced Lectures and Seminars
   - Those credits must include at least 2 credits from the course of the Advanced Lectures and Seminars
   - If a student who is a graduate of a master's program in our schools completes a course of Basic Lectures or Intermediate Lectures, which the student did not complete in the master's program, the course and Area will be transferred to corresponding course and Area of the Advanced Lectures and Seminars and recognized as a part of completion requirements in the doctoral program. In addition to the recognized courses, at least 2 credits from 1 course of Advanced Lectures and Seminars (K6xx series) in the doctoral program must be completed.
   - Only 2 credits from the course of English (Technical Communication) III can be recognized as a part of completion requirements.
4) The course of K617 *Project Management (Advanced)* is essential for the **type E** students and the course of K618 *Critical Thinking and Scientific Discussions* is essential for the **type S** students.

Figure 2.3: Course and Credit Requirements for Doctoral Program in School of Knowledge Science from JAIST Degree Completion Guide 2010-2011

definition as the shorthand in the following requirement. The definition of created concept is represented by

$$\mathsf{DrLecture} \equiv \mathsf{BasicLecture} \sqcup \mathsf{InterLecture} \sqcup \mathsf{AdvanLecture}.$$

- The last note, together with the main statement of the requirement, indicates two cases that a student satisfies the requirements of course level:
  - he passes at least 5 courses of doctoral lectures:

$$\geq 5\, \mathsf{passes}.\mathsf{DrLecture},$$

  - or he passes one English course and at least 4 courses of doctoral lectures:

$$(\exists \mathsf{passes}.\mathsf{EnglishLecture} \sqcap \geq 4\, \mathsf{passes}.\mathsf{DrLecture}).$$

The conjunction of description in the first and third notes forms definition for the concept DrLevelCompleted.

For the essential requirement, the mapping between condition terms of axiom (15) for concept DrEssentialCompleted and the requirement 4) in Figure 2.3 is straightforward: one student satisfies this requirement if

– he is a type E student and pass the course of K617:

$$\mathsf{TypeE} \sqcap \exists \mathsf{passes}.\{\mathsf{K617}\},$$

– or he is a type S student and pass the course of K618:

$$\mathsf{TypeS} \sqcap \exists \mathsf{passes}.\{\mathsf{K618}\}.$$

2.3.3 KS Ontology ABox

Figure 2.4 shows important assertions in the ABox relating to three students in interest, namely Robert, Suzuki, and Nguyen. The upper part shows the assertions relating to the courses of School of Knowledge Science. This part corresponds to the core knowledge of educational program and is built in the implementation phase of the ontology together with the TBox. The lower part describes a specific circumstance in interest and is used to verify the correctness of the ontology. Those assertions can be inputted by the user when using this ontology.

In the particular situation described by this figure, all three students are KS students among who Robert and Nguyen are explicitly represented as KS students while Suzuki is implicitly via the statements that Yamada is the main supervisor of Suzuki and Yamada is a professor of School of Knowledge Science. The latter case is used to verify the representation ability the ontology. Suzuki and Nguyen are doctoral students while Suzuki a master student and all of them are type S students. All those students have pass 5 courses as shown in the figure and Robert also finished his major and minor researches, K601 and K602 respectively. Together with knowledge about courses and lectures, this ABox allows to infer the status of each student as given in the following section.

2.4 Ontology Reasoning

2.4.1 Ontology Reasoning Tasks

A DL system is emphasized by its reasoning or inference ability which plays the central role of an artificial intelligence system. Figure 2.5 shows the architecture of knowledge

Assertion Box $\mathcal{A}_{KS}$:

EnglishLecture(K218)	BasicLecture(K211)	IntroLecture(K116)
BasicLecture(K228)	BasicLecture(K213)	BasicLecture(K217)
InterLecture(K417)	BasicLecture(K229)	InterLecture(K414)
AdvanLecture(K613)	AdvanLecture(C612)	InterLecture(416)
AdvanLecture(K618)	AdvanLecture(K612)	AdvanLecture(616)
MultiAreaCourse(K218)	SocialCourse(K211)	SystemCourse(K116)
MultiAreaCourse(K228)	MultiAreaCourse(K213)	SystemCourse(K217)
MediaCourse(K417)	MultiAreaCourse(K229)	SystemCourse(K414)
MultiAreaCourse(K613)	MediaCourse(K612)	SystemCourse(K416)
MultiAreaCourse(K618)	MediaCourse(C612)	SystemCourse(K616)
	⋮	
KSProfessor(YAMADA)		
KSStudent(ROBERT)	supervises(YAMADA,SUZUKI)	KSStudent(NGUYEN)
DoctorStudent(ROBERT)	MasterStudent(SUZUKI)	DoctorStudent(NGUYEN)
TypeS(ROBERT)	TypeS(SUZUKI)	TypeS(NGUYEN)
passes(ROBERT,K218)	passes(SUZUKI,K211)	passes(NGUYEN,K116)
passes(ROBERT,K228)	passes(SUZUKI,K213)	passes(NGUYEN,K217)
passes(ROBERT,K417)	passes(SUZUKI,K229)	passes(NGUYEN,K414)
passes(ROBERT,K613)	passes(SUZUKI,C612)	passes(NGUYEN,K416)
passes(ROBERT,K618)	passes(SUZUKI,K612)	passes(NGUYEN,K616)
passes(ROBERT,K601)	passes(ROBERT,K602)	

Figure 2.4: A part of Assertion Box which relates to knowledge about Students and Courses for Educational Program in School of Knowledge Science

representation and usage using ontology-based knowledge system. The user constructs and updates the knowledge base by stating his expertizing knowledge using DL language and stores it in TBox and ABox of the ontology. Reasoning engine accesses this knowledge and performs inference to deduce the implicitly represented knowledge and returns result to the user as requested. Mid-tier programs or applications can be added to facilitate the user to interact with the system by proving features such as knowledge interpretation, which expresses the knowledge in a language that is familiar to human, or user friendly interface.

Typical reasoning services for DL ontology include:

- *Concept subsumption* checking determines the inclusion relationship among concepts: given two concept C and D, whether D is more general than C or C is subsumed by D, denoted as $C \sqsubseteq D$, that is all instance of C is necessarily instances of D. For example, from the statements that KS students are JAIST students and JAIST students are students, it is possible to infer that KS students are students, i.e. $\mathsf{KSStudent} \sqsubseteq \mathsf{Student}$.

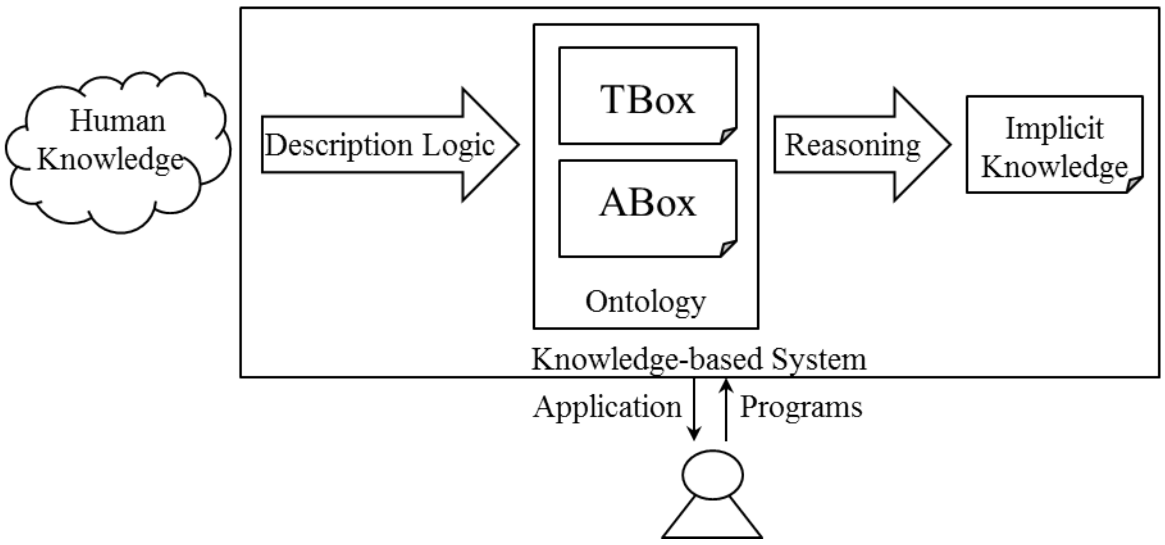


Figure 2.5: Knowledge Representation Systems based on Description Logics

- *Concept satisfiability* checking determines whether a concept C is *satisfiable* or *consistent*, or literally speaking, whether it is *not* subsumed by an empty concept, denoted as $C \not\sqsubseteq \bot$. This is the main ontology reasoning service since every reasoning tasks con be reduce to concept satisfiability checking. For example, if we want to check whether a KS student is a student, we can determine the satisfiability of its contradiction, $\mathsf{KSStudent} \sqcap \neg \mathsf{Student}$. The *unsatisfiability* of the negated concept implies a positive answer and vice versa.

- *Instance* checking determines whether an individual i is an instance of concept C. For example, from the fact that Yamada is the main supervisor of Suzuki and Yamada is a Professor of School of Knowledge Science, it is possible to conclude that Yamada is a student of School of Knowledge Science, $\mathsf{KSStudent(SUZUKI)}$, according to the axiom (9) in Figure 2.2.

- *Consistency* checking determines whether a knowledge base (consisting of a set of assertions and a set of terminological axioms) is non-contradictory. For example, if we add $\neg\mathsf{KSStudent(SUZUKI)}$ to the ABox $\mathcal{A}_{KS}$, which conflicts with the above inferred assertion $\mathsf{KSStudent(SUZUKI)}$, the knowledge base becomes inconsistent.

Another enhanced service is ontology classification, that is to discover all concept pairs $\langle C, D\rangle$ such that C is subsumed by D and all individual-concept pairs $\langle i, C\rangle$ such that i is a instance of C. Since the cost subsumption checking is computationally high, various optimization techniques have been introduced which utilize explicit hierarchical information of concepts and individuals to minimize the number of subsumption checking when classifying concepts and instances [37, 64, 110, 71]. Ontology classification is provided by modern ontology editors as a default function for ontology reasoning which automatically performs all above reasoning services in one single command.

Figure 2.6 shows the important results when performing inference on KS Ontology to determine the status for every student in the ABox. They are mainly instance assertions which shows the status of individuals in discourse. ROBERT is classified as a qualified KS doctor since he has satisfied all requirements that is stated in axiom (10). SUZUKI is classified as a KS student as expected and even being a master students, he considered to satisfies doctoral course requirements because all the courses he has passed belongs doctoral courses and cover at least 2 areas. Finally, NGUYEN does not satisfies any requirements and thus is not assigned to any corresponding concepts.

Inference results:

JaistStudent(ROBERT)	Student(ROBERT)
KSQualifiedDoctor(ROBERT)	DrCourseCompleted(ROBERT)
DrEssentialCompleted(ROBERT)	DrLevelCompleted(ROBERT)
DrAreaCompleted(ROBERT)	
JaistStudent(SUZUKI)	Student(SUZUKI)
KSStudent(SUZUKI)	DrCourseCompleted(SUZUKI)
DrEssentialCompleted(SUZUKI)	DrLevelCompleted(SUZUKI)
JaistStudent(NGUYEN)	Student(NGUYEN)

Figure 2.6: Reasoning results for individuals of KS Ontology

2.4.2 Tableau Reasoning Algorithms

Most ontology reasoners today utilize tableau reasoning algorithms, which initially presented by [104] for $\mathcal{ALC}$ DL and extensively developed for more expressive DLs such as in [12, 55, 85, 65, 63], to check concept satisfiability.

Tableau algorithms prove the satisfiability of a concept C by constructing a *model*, an interpretation $\mathcal{I}$ in which $D^{\mathcal{I}}$ is not empty. A *tableau* is a directed graph which represents such a model, with nodes correspond to individuals (elements of $\triangle^{\mathcal{I}}$) and edges correspond to relationships between individuals (elements of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$). Each node x is labeled with a set of concepts ($\mathcal{L}(x) = \{C_1, \ldots, C_n\}$), and each edge $\langle x, y\rangle$ is labeled with a role ($\mathcal{L}(\langle x, y\rangle) = R$). When a concept C is in the label of a node $x(C \in \mathcal{L}(x))$, it represents a model in which the individual corresponding with x is in the interpretation of C. When an edge $\langle x, y\rangle$ is labeled $R(\mathcal{L}\langle x, y\rangle) = R)$, it represents a model in which the tuple corresponding with $\langle x, y\rangle$ is in the interpretation of R. A node y is called an R-successor of a node x if there is an edge $\langle x, y\rangle$ labeled R, x is called the predecessor of y if y is an R-successor of x, and x is called an ancestor of y if x is the predecessor of y or there exists some node z such that z is the predecessor of y and x is an ancestor of z.

$\sqcap$-rule:	if 1.	$C_1 \sqcap C_2 \in \mathcal{L}(x)$, x is not blocked, and
	2.	$\{C_1, C_2\} \not\subseteq \mathcal{L}(x)$
	then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_1, C_2\}$	
$\sqcup$-rule:	if 1.	$C_1 \sqcup C_2 \in \mathcal{L}(x)$, x is not blocked, and
	2.	$\{C_1, C_2\} \cap \mathcal{L}(x) = \emptyset$
	then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$ for some $C \in \{C_1, C_2\}$	
$\exists$-rule:	if 1.	$\exists R.C \in \mathcal{L}(x)$, x is not blocked, and
	2.	x has no R-successor y with $C \in \mathcal{L}(x)$
	then create a new node y with $\mathcal{L}(\langle x, y \rangle) = \{r\}$ and $\mathcal{L}(y) = \{C\}$	
$\forall$-rule:	if 1.	$\forall R.C \in \mathcal{L}(x)$, x is not blocked, and
	2.	there is an R-successor y of x with $C \in \mathcal{L}(y)$
	then $\mathcal{L}(y) = \mathcal{L}(y) \cup \{C\}$	
$\sqsubseteq$-rule:	if 1.	$C_1 \sqsubseteq C_2 \in \mathcal{L}(x)$, x is not blocked, and
	2.	$C_2 \sqcup \neg C_1 \notin \mathcal{L}(x)$
	then set $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C_2 \cup \dot{\neg} C_2\}$	

Figure 2.7: The tableau expansion rules for $\mathcal{ALC}$

The algorithm initializes a tree to contain a single node x (called the *root* node) with $\mathcal{L}(x) = \{C\}$, and then expands the tree by applying rules that either extend node labels or add new leaf nodes. A set of *expansion rules* for the $\mathcal{ALC}$ DL is shown in Figure 2.7 which cited from [11], where C and D are concepts, and R is a role. For example, if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, and either $C_1 \notin \mathcal{L}(x)$ or $C_2 \notin \mathcal{L}(x)$, then the $\sqcap$-rule adds both C_1 and C_2 to $\mathcal{L}(x)$; if $\exists r.C \in \mathcal{L}(x)$, and x does not yet have an r-successor with C in its label, then the $\exists$-rule generates a new r-successor node y of x with $\mathcal{L}(y) = \{C\}$. Blocking prevents application of expansion rules when the construction becomes repetitive; i.e., when it is obvious that the sub-tree rooted in some node x will be "similar" to the sub-tree rooted in some predecessor y of x. The algorithm stops if it encounters a *clash*: there exists $\{C, \neg C\} \subseteq \mathcal{L}(x)$ for some node x and some concept name C. A node x is *blocked* if there is an ancestor of y of x such that $\mathcal{L}(x) \subseteq \mathcal{L}(y)$, or if there is an ancestor z of x such that z is blocked. The tree is fully expanded when none of the expansion rules can be applied. If a fully expanded and clash-free tree can be found, then the algorithm returns $satisfiable$; otherwise it returns $unsatisfiable$.

Among expansion rules, $\sqcup$-rule is different from the other rules in that it is $non-$ *deterministic*: if $C_1 \sqcup C_2 \in \mathcal{L}(x)$ and neither $C_1 \in \mathcal{L}(x)$ nor $C_2 \in \mathcal{L}(x)$, then it adds *either* C_1 *or* C_2 to $\mathcal{L}(x)$, the algorithm may be necessary to explore all possible choices of rule applications and it cause the highly computational complexity. Adding other rules to handle other constructors in more expressive DL may also increase the complexity. A great amount of research effort has been paid to optimize expansion rules and strategy to select expansion rules to make reasoning algorithm practical in real-world applications.

2.4.3 Open World Assumption

Ontology applies the so called Open World Assumption (OWA), which is different from the Close World Assumption (CWA) in relational database and logic programming. In database and logic programing settings, a fact is assumed to be false if it cannot be proved from the existing data. Meanwhile, ontology reasoning leave the value of non-existing fact as undetermined. For example, without assertion $\mathsf{supervises}(\mathsf{YAMADA}, \mathsf{ROBERT})$, a database engine can conclude that Professor Yamada does not supervise Robert while ontology reasoning make the conclusion unknown.

The OWA assumption has a significant affect the results of the reasoning process. Let consider the following example to clarify the affect. From the reasoning results of KS Ontology in previous subsection, SUZUKI and NGUYEN are *not* classified as KS qualified doctor because of their unsatisfiability of the requirements. Assume we add assertions $\mathsf{KSQualifiedDoctor}(\mathsf{SUZUKI})$ and $\mathsf{KSQualifiedDoctor}(\mathsf{NGUYEN})$ to the ABox. We may expect that adding these assertions will cause contradiction with current assertions in the ABox and the system can detect the inconsistency. Adding assertion $\mathsf{KSQualifiedDoctor}(\mathsf{SUZUKI})$ indeed causes inconsistency because it implies $\mathsf{DoctorStudent}$ (SUZUKI) which explicitly conflicts with assertion $\mathsf{MasterStudent}(\mathsf{SUZUKI})$ by the definition $\mathsf{DoctorStudent} \equiv \neg\mathsf{MasterStudent}$ in the TBox. However, adding assertion $\mathsf{KSQualifiedDoctor}(\mathsf{NGUYEN})$ does not cause inconsistency even the assertions about courses that Nguyen has passed does not satisfy the requirements. Regarding to OWA, ontology reasoning assume that there may be courses that Nguyen has passed but not provided to the ABox yet, and thus, accept the added assertion.

The reason for OWA is that it is appropriate for the Semantic Web setting which is characterized by its distributively huge-scale and dynamically changing properties. Thus, it is very often that the knowledge cannot be collected completely before performing reasoning and thus assuming what is not known to be false cannot be used.

2.5 Motivating Examples for Inconsistent Ontology

We introduce two motivating examples for inconsistent ontologies in this section. These examples will be used to explain and demonstrate our approach in following chapter.

Example 2.1 (Subsumption checking)**.** This is the formalism of Penguin example which is introduced in Section 1.3, a well-known example of default knowledge. Assume two ontologies Bird and Penguin which represent that birds typically fly and have wings while penguins are birds that do not fly. Those ontologies are denoted as follows [2,3]:

$$\Sigma_{Bird} = \left\{ \begin{array}{ll} BF: & \mathsf{Bird} \sqsubseteq \mathsf{Fly}, \\ BW: & \mathsf{Bird} \sqsubseteq \mathsf{HasWings} \end{array} \right\}$$

$$\Sigma_{Penguin} = \left\{ \begin{array}{ll} PB: & \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ PF: & \mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly} \end{array} \right\}$$

This example demonstrates the logically conflict issue when integrating ontologies. Each of above ontology is locally consistent. However, when Penguin ontology refers to Bird ontology through import operation, concept Penguin becomes unsatisfiable because it is subsumed by two mutually disjoint concepts, Fly, according to axiom BF, and ¬Fly, according to axiom PF.

Example 2.2 (Instance checking)**.** This example extends the KS Ontology with exceptional knowledge of regulation about students. According to the regulation, all adult people who aged 20 or over in Japan are obliged to enroll in the National Pension system. However, students whose income is usually less than a certain level are not required to pay the monthly contributions. The interpretation of this knowledge is as follows:

$$\Sigma_{Student} = \left\{ \begin{array}{ll} AP: & \mathsf{Adult} \sqsubseteq \mathsf{PaysPension}, \\ SP: & \mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension} \end{array} \right\}$$

Assume we instantiate the ontology with a particular individual Suzuki that is a student and an adult, denoted as

$$\begin{array}{ll} SA: & \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Adult}, \\ SS: & \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Student} \end{array}$$

Adding those axioms to the ontology will cause the inconsistency because we cannot create a model with the individual SUZUKI since in one hand, Suzuki has to pay pension contributions as stated in axiom AP, but in another hand, Suzuki does not need to pay as stated in axiom SP.

Due to the classical property of underlying logic, DL ontologies are unable to scope with inconsistencies as shown in above examples. In detail, there are two kinds of inconsistency which are introduced: concept inconsistency in the former example and model inconsistency in the latter. For the latter case, it is impossible to construct a model for the ontology. Therefore, no reasoning service can be provided for this ontology. In the

[2]According to the last note in Subsection 2.2.1 which states that ABox assertions can be simulated by TBox axioms, we assume hereby that an ontology contains only terminological statements.

[3]We also add the axiom label at the beginning of each axiom to allow convenient process afterwards.

former case, for every possible model of the ontology, concept Penguin must be equivalent to an empty concept. Then meaningless conclusions such as “Penguins are human” are inferred from unsatisfiable concepts. Both issues of classical DL ontology can be handled by providing more tolerant reasoning ability for the ontology. This research considers an extension of classical ontology which combine defeasible logic reasoning approach with DL ontology. The following chapter thoroughly discusses various aspect of defeasible logic and defeasible reasoning to establish the foundation for our approach.

Chapter 3

Defeasible logic

3.1 Introduction to Defeasible Logic

Defeasible Logic is an approach to nonmonotonic reasoning which has been developed by Nute [88, 89] over several years with a particular concern about computational efficiency and ease of implementation. Defeasible inference in propositional form of logic can be performed in linear time as proved in [78]. Efficient implementations of defeasible inference such as in [102, 77, 67] can answer queries from a system with 100,000's of defeasible rules or even calculate all conclusions. Apart from efficiency, defeasible logic is flexible enough to deal with several intuitions of non-monotonic reasoning, and it has been applied to modeling of regulations and business rules [45], modeling of contracts [98, 41], legal reasoning [44], agent negotiations [29], and agent modeling [43].

As being developed based on the idea of logic programming without negation as failure, basic elements of a defeasible theory include facts and rules. Informally, a defeasible theory contains five different kinds of knowledge: facts, strict rules, defeasible rules, defeaters, and a superiority relation.

Facts are indisputable statements, for example, "Suzuki is a student" which might be logically expressed as $\mathsf{Student}(\mathsf{SUZUKI})$.

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 "JAIST students are students" which is written formally as:

$$\mathsf{JaistStudent}(x) \rightarrow \mathsf{Student}(x)$$

Defeasible rules are rules that can be defeated by contrary evidence. An example of such a rule is "adult peoples need to pay pension"; written formally:

$$\mathsf{Adult}(x) \Rightarrow \mathsf{PaysPension}(x).$$

The rule states that if we know that someone is an adult people, then we may conclude that he/she is required to pay pension contributions, *unless there is other evidence suggesting that he/she does not need to pay.*

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 "students might not work". Formally:

$$\mathsf{Student}(x) \rightsquigarrow \neg\mathsf{Works}(x).$$

The main point is that the information that somebody is a student is not sufficient evidence to conclude that he/she might not work. In other words, we do not wish to conclude $\neg$Works if Student, we simply want to prevent a conclusion Works 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 two defeasible rules

$$r : \mathsf{Adult}(x) \Rightarrow \mathsf{PaysPension}(x)$$
$$r' : \mathsf{Student}(x) \Rightarrow \neg\mathsf{PaysPension}(x)$$

which contradict each other. If an instance is represented as both an adult and a student, no conclusive decision can be made about whether a he/she pays the pension or not. But if we introduce a superiority relation $>$ which $r' > r$, then we can indeed conclude that he does not need to pay the pension.

Notice that the superiority relation is required to be acyclic. In the above example, it makes no sense to have both $r > r'$ and $r' > r$. Moreover, priorities are *local* in the following sense: two rules are considered to be competing with one another only if they have complementary heads. Thus, since the superiority relation is used to resolve conflicts among competing rules, it is only used to compare rules with complementary heads; the information $r > r'$ for rules r, r' without complementary heads may be part of superiority relation, but has no effect on the proof theory.

3.2 Formal Definitions

For the formal definitions below, we consider only essentially *propositional rules* without loss of generality. Rules containing free variables are interpreted as the set of their variable-free instances. For a propositional literal q, $\sim q$ 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 rule $r : A(r) \hookrightarrow C(r)$ consists of its unique *label* r, its *antecedent* $A(r)$ which is a finite set of literals ($A(r)$ may be omitted if it is the empty set), an arrow $\hookrightarrow$ (which maybe one of three specific arrows for three kinds of rule), and its *head* (or *consequence*) $C(r)$ which is a literal. In writing rules we omit set notation for antecedents, and sometimes we omit the label when it is not relevant for the context. There are three kinds of rules, each represented by a different arrow. Strict rules use $\rightarrow$, defeasible rules use $\Rightarrow$, and defeaters use $\leadsto$.

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 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.

A defeasible theory $D = (F, R, >)$ consists of a set of facts F, a set rules R, and a set of superiority relation $>$ among rules. D is called *well-formed* if and only if $>$ is acyclic and $>$ is only defined on rules with complementary heads. Defeasible logic has a proof-based semantics. Provability of defeasible logic is based on the concept of a *derivation* (or *proof*) which is a finite sequence $P = (P(1), \ldots, P(n))$ of tagged literals. A *tagged literal* consists of a sign ($+$ denotes provability, $-$ finite failure to prove), a tag and a literal. There are two tags signify the "strength" of the conclusions they are attached to, and correspond to different classes of derivations: Δ denotes definite provability based on monotonic proofs, and *df* defeasible provability based on non-monotonic proofs. The interpretation of the proof tags is as follows:

$+\Delta q$: there is a definite derivation of q using only strict rules and facts;

$-\Delta q$: it is not possible to obtain a definite derivation of q;

$+df q$: there is a defeasible (non-monotonic) derivation of q;

$-df q$: it is not possible to obtain a defeasible (non-monotonic) derivation of q.

3.3 Defeasible Proof Theory

In above interpretation, definite provability, or monotonic proof, follows classical derivation of facts and strict rules. Furthermore, defeasible logic introduces and exploits definite non-provability as the strong negation of definite provability. The conditions of definite provability are expressed as follows:

$+\Delta$) We may append $P(i+1) = +\Delta q$ if either

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

$-\Delta$) We may append $P(i+1) = -\Delta q$ if

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

where $R_s[q]$ is the set of strict rules which have q as their conclusions, and $A(r)$ is the antecedent of r.

The reason for considering definite non-provability is that defeasible provability, or non-monotonic proof, is skeptical. It prevents conflict conclusions by detecting the conflict in advance and one conclusion is defeasibly derived if all of its conflict derivations are eliminated. In other words, a conclusion is tagged defeasible provability only when its conflict conclusion is unprovable or defeated. For the sake of doing so, defeasible reasoning needs to manage definitely unprovable tags ($-\Delta$) as well as defeasibly unprovable ones ($-df$).

In general, the conditions to establish a *non-monotonic proof* have three phases, which has a similar structure to arguments.

(1) In the first phase we have to put forward an argument for the conclusion we want to prove;

(2) in the second phase (the attack phase in argumentation terms) we have to consider all possible arguments against the thesis (counter-arguments);

(3) in the third and last phase we rebut the counter-arguments. To rebut an argument we have to two options:
   - (a) we can show that the argument is not founded, i.e., some of the premises do not hold, or
   - (b) we can defeat the argument by providing a stronger counterargument.

The followings represent the framework of proof theory for defeasible tags which showing the usage of those tags to determine defeasible provability and non-provability and realizing the three-phase structure as discussed above:

$+df$) We may append $P(i+1) = +df\, q$ if either

- 1) $+\Delta q \in P[1..i]$; or
- 2) The following three conditions all hold
  - 2.1) $-\Delta \sim q \in P[1..i]$, and
  - 2.2) there is an ***applicable*** strict or defeasible rule for q, and
  - 2.3) every rule for $\sim q$ is either
    - 2.3.1) ***discarded*** or
    - 2.3.2) ***defeated***.

$-df$) We may append $P(i+1) = -df\, q$ if

- 1) $-\Delta q \in P[1..i]$, and
- 2) either
  - 2.1) $+\Delta \sim q \in P[1..i]$, or
  - 2.2) every strict or defeasible rule for q is ***discarded*** or

2.3) there is rule for ∼q such that

2.3.1) the rule is ***applicable*** and

2.3.2) the rule is not ***defeasible***.

For defeasible provability, a monotonic proof is also non-monotonic proof according to condition $+df.1$. In the second condition for a literal q to be defeasibly provable or not defeasibly provable, rules are categorized as following situations. A rule is *applicable* (condition $+df.2.2$ and $-df.2.3.1$) if all elements of its antecedent are definitely or defeasibly provable. On the other hand, a rule is *discarded* (condition $+df.2.3.1$ and $-df.2.2$) if one element of its antecedents is not provable. For the condition 2.3.2, a rule is *defeated* ($+df.2.3.2$) if there exists an applicable rule that is superior to it while it is *defeasible* ($-df.2.3.2$) if there a non-discarded rule that is stronger than it.

Notice that strict rules can be used in two different ways, depending on the degrees of provability of their antecedents. If their bodies are proved definitely then their head is proved definitely, then strict rules are used as in classical logic, regardless of any reasoning chains with the opposite conclusion. But strict rules can also be used to show defeasible provability, given that some other literals are known to be defeasibly provable. In this case, strict rules are used exactly like defeasible rules. For example, a strict rule may have its body proved defeasibly, yet it may not fire because there is a rule with the opposite conclusion that is not weaker. Furthermore, strict rules are not automatically superior to defeasible rules. In [78] strict rules are duplicated as defeasible rules, and definite reasoning use the strict rules (as always), while defeasible reasoning may use only defeasible rules. This duplication and separation of rules does not modify the consequences. When there is a duplicate defeasible rule for every strict rule in a defeasible theory, we say that the theory has *duplicated strict rules.*

A desired characteristic of non-monotonic logics is that it can avoid inconsistent conclusions from the theory. In defeasible logic, this is assured by the *coherent* property which states that a well-formed theory cannot conclude that both a proposition p and its negation are defeasibly true unless they are both definitely true. Or equivalently speaking, a defeasible theory is consistent if its strict part is consistent. This property was first proved in [16] and extensively analyzed in [4]. This property suggests that an extension of DL ontology based on defeasible logic is sufficient to handle to inconsistency with a proper treatment for the conflicting axioms. This problem will be discussed in detail in the following chapter.

3.4 Variants of Defeasible Logics

There are four variations of defeasible logics, which correspond to four different pairs of inference rules, that have been introduced in literature [79]. The variants utilize different strategies when resolving conflict proofs and handling inference after encountering an ambiguity, a situation where there are proofs for two contract conclusions and there is no superiority among them. According to the manner to resolve conflicts, we have team defeat and non-team defeat strategies. According to the method to handling inference after encountering an ambiguity, we have ambiguity blocking and ambiguity propagation strategies.

The four logics are labelled as $\partial, \delta, \partial^*$, and δ^* and produce conclusions of the form (respectively) $+\partial q, -\partial q, +\delta q, -\delta q$, etc. The meanings of those labels are as follows:

- ∂: team defeat with ambiguity blocking;
- ∂^*: non-team defeat with ambiguity blocking;
- δ: team defeat with ambiguity propagation; and
- δ^*: non-team defeat with ambiguity propagation.

3.4.1 Team Defeat vs Non Team Defeat

Figure 3.1 highlights the difference between team defeat and non team defeat with ambiguity blocking defeasible logics. As we can see in the conditions ($+\partial.2$) and ($+\partial.2.2.3$), $+\partial$ relies on a team consisting of r and all the rules t that are needed to overcome the competing rules s. In order for a rule to be applied, every competing rule can be defeated by any of supporting rules. In comparison, $+\partial^*$ relies on a single rule to overcome all competing rules. Thus $+\partial$ employs team defeat while $+\partial^*$ relies on a single rule overcoming all opposition. The same analysis is applied in defeasible logics with ambiguity propagation, δ and δ^*.

For example, consider the following defeasible theory D on whether animals are mammals [4].

$$\begin{array}{rrcl}
r_1 : & monotreme & \Rightarrow & mammal \\
r_2 : & hasFur & \Rightarrow & mammal \\
r_3 : & laysEggs & \Rightarrow & \neg mammal \\
r_4 : & hasBill & \Rightarrow & \neg mammal \\
 & r_1 > r_3 & & \\
 & r_2 > r_4 & &
\end{array}$$

For a platyrus, we have the facts: $monotreme, hasFur, laysEggs$, and $hasBill$, then those four rules are all applicable. The rules r_3 and r_4 for $\neg mammal$ are over-ruled by, respectively, r_1 and r_2. Consequently, under inference with team defeat (∂ and δ), we conclude $+\partial mammal$ and $+\delta mammal$. Under inference without team defeat (∂^* and δ^*), there is no rule that overrules all opposing rules. Consequently we cannot make any positive conclusion; we conclude $-\partial^* mammal$ and $-\partial^* \neg mammals$, and similarly for δ^*.

Team Defeat

$+\partial$) $+\partial q \in P(i+1)$ iff either

1) $+\Delta q \in P(1..i)$; or

$\boxed{\text{2) } \exists r \in R_{sd}[q] \text{ such that}}$

2.1) $\forall a \in A(r), +\partial a \in P(1..i)$ 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

$\boxed{\begin{array}{l}\text{2.3.2) } \exists t \in R_{sd}[q] \text{ such that} \\ \quad \forall a \in A(t), +\partial a \in P(1..i) \text{ and } t > s\end{array}}$

$-\partial$) $-\partial q \in P(i+1)$ iff

1) $-\Delta q \in P(1..i)$ and

$\boxed{\text{2) } \forall r \in R_{sd}[q] \text{ either}}$

2.1) $\exists a \in A(r), -\partial a \in P(1..i)$; or

2.2) $+\Delta \sim q \in P(1..i)$; or

2.3) $\exists s \in R[\sim q]$ such that

2.3.1) $\forall a \in A(s), +\partial a \in P(1..i)$ and

$\boxed{\begin{array}{l}\text{2.3.2) } \forall t \in R_{sd}[q] \text{ either} \\ \quad \exists a \in A(t), -\partial a \in P(1..i) \text{ or } t \not> s\end{array}}$

Non-Team Defeat

$+\partial^*$) $+\partial^* q \in P(i+1)$ iff either

1) $+\Delta q \in P(1..i)$; or

$\boxed{\text{2) } \exists r \in R_{sd}[q] \text{ such that}}$

2.1) $\forall a \in A(r), +\partial^* a \in P(1..i)$ and

2.2) $-\Delta \sim q \in P(i..1)$ 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

$\boxed{\text{2.3.2) } r > s}$

$-\partial^*$) $-\partial^* q \in P(i+1)$ iff

1) $-\Delta q \in P(1..i)$ and

$\boxed{\text{2) } \forall r \in R_{sd}[q] \text{ either}}$

2.1) $\exists a \in A(r), -\partial^* a \in P(1..i)$; or

2.2) $+\Delta \sim q \in P(1..i)$; or

2.3) $\exists s \in R[\sim q]$ such that

2.3.1) $\forall a \in A(s), +\partial^* a \in P(1..i)$ and

$\boxed{\text{2.3.2) not } r > s}$

Figure 3.1: Comparison between Proof Theories of Team Defeat and Non-Team Defeat with Ambiguity Blocking

3.4.2 Ambiguity Blocking vs Ambiguity Propagating

Both defeasible logics discussed in the previous subsection are ambiguity blocking. Intuitively, a literal is *ambiguous* if there is a (monotonic) chain of reasoning that supports a conclusion that p is true, another that supports that $\neg p$ is true, and the superiority relation does not resolve this conflict. In ambiguity blocking inference, such literal is not used in any inference afterwards. Meanwhile, ambiguity propagation allows the use of this literal to prevent some possible conclusion.

The difference between ambiguity blocking and ambiguity propagation is appealing in the case of court debate where different principles of regulation application can be applied. Let consider the following example in a legal case to demonstrate the difference. This example consists of two rules stating that evidence A suggests that the defendant in is not responsible while a second piece of evidence B indicates that he/she is responsible; moreover the sources are equally reliable. According to the underlying legal system a defendant is presumed innocent (i.e., not guilty) unless responsibility has been proved (without any reasonable doubt). The scenario is encoded in the following defeasible theory:

$$r_1 : evidence_A \Rightarrow \neg responsible$$
$$r_2 : evidence_B \Rightarrow responsible, \qquad r_3 : responsible \Rightarrow guilty,$$
$$r_4 : \Rightarrow \neg guilty.$$

Given both $evidence_A$ and $evidence_B$, the literal *responsible* is ambiguous. There are two applicable rules ($r1$ and $r2$) with the same strength, each supporting the negation of the other. As a consequence r_3 is not applicable, and so there is no applicable rule supporting the *guilty* verdict. This behavior is called ambiguity blocking, since the ambiguity of *guilty* has been blocked by the failure to prove *responsible*. In contrast, ambiguity propagation describes a behavior where ambiguity of a literal is propagated to dependent literals. If we propagate ambiguity then the literals *guilty* and $\neg guilty$ are ambiguous; thus an undisputed conclusion cannot be drawn. On the other hand, if we assume an ambiguity blocking stance, the literal $\neg guilty$ is not ambiguous and a verdict can be reached. If we extend the theory with the rule $r_5 : \neg guilty \Rightarrow compensation$, saying that in case of a not guilty verdict the defendant is entitled to compensation, then, despite the existence of equally strong pieces of evidence, he/she is entitled to compensation, while this is not the case in the ambiguity propagating case.

The example and the discussion above show that both ambiguity blocking and ambiguity propagation can be used in the same application domain. In legal reasoning, typically, civil law takes an ambiguity blocking stance, while criminal law more often opts for an ambiguity propagation point of view. In addition there are situations where both types of reasoning are required at once. For example, in legal proceedings, often, the different parties involved have different burden of proof [92], and these types call for different ways to lead to the conclusions. Typical, in criminal litigation, a plaintiff, to win a case, has to prove it without any benefit of doubt, while a defendant is only required to produce an exception to the accusation. This means that the claim of the plaintiff must be proved using ambiguity propagation reasoning, while the claim of the defendant by ambiguity propagation [76]. In addition the conditions to justify plaintiff claims are more stringent than those for the defendant.

The inference rules δ and δ^* require auxiliary tags and inference rules, denoted by σ and σ^*, respectively. The complexity of those inference rules are beyond of this research and interested readers can refer to [17, 68, 80] for more detail interpretation and discussion. In this research, we rely on team defeat with ambiguity blocking defeasible logic.

3.5 Defeasible Reasoning Algorithm

3.5.1 Basic Defeasible Logic

The research on correctness and efficiency of defeasible reasoning algorithm is performed on *basic defeasible logic* which is a subset of defeasible logic that has no superiority statements and no defeaters. A general defeasible theory can be transformed into basic theory by simulating those elements by defeasible rules. Figure 3.2 shows a simple example of reduction to basic defeasible theory. This example theory consists of 2 strict rules, 3 defeasible rules and 1 superiority relationship between two defeasible rules. The example employs a simple transformation which alters each defeasible rule by two corresponding defeasible rules with an auxiliary literal. Those auxiliary literals are used to simulate superiority relationships between defeasible rules. This simulation is performed by introducing a defeasible rule which takes the auxiliary literal corresponding to the superior rule as the antecedent and the negation auxiliary literal corresponding to the inferior rule as the consequence such as the last rule in the corresponding basic theory. In the basic theory, if *penguin* is provable, its corresponding auxiliary r_2 is also provable. With the simulated rule, r_1, and thus, fly, becomes unprovable.

	Defeasible Theory	**Basic Theory**
	$\rightarrow penguin$	$\rightarrow penguin$
	$penguin \rightarrow bird$	$penguin \rightarrow bird$
$r_1:$	$bird \Rightarrow fly$	$bird \Rightarrow r_1$
		$r_1 \Rightarrow fly$
$r_2:$	$penguin \Rightarrow \neg fly$	$penguin \Rightarrow r_2$
		$r_2 \Rightarrow \neg fly$
$r_3:$	$bird \Rightarrow hasWings$	$bird \Rightarrow r_3$
		$r_3 \Rightarrow hasWings$
	$r_2 > r_1$	$r_2 \Rightarrow \neg r_1$

Figure 3.2: Simple Example of Reduction to Basic Defeasible Theory

A full treatment of the transformation, including proofs of correctness and other properties, is presented in [4]. The transformation can increase the size of the theory by at most a factor of 12. Furthermore, the time taken to produce the transformed theory is linear in the size of the input theory. Consequently, the implementation of full defeasible logic by first transforming the input theory to a theory without defeaters and superiority statements, and the applying an algorithm like Algorithm 1 to the transformed theory provides a linear implementation of defeasible logic.

For basic defeasible logic, the inference rules are simplified by introducing two auxiliary tags ($+\sigma$ and $-\sigma$), namely *tentative* conclusions, as follows:

$+\sigma$: We may append $P(i+1) = +\sigma q$ if

$\exists r \in R_{sd}[q]\, \forall a \in A(r) : +\partial a \in P(1..i)$

$-\sigma$: We may append $P(i+1) = -\sigma q$ if

$\forall r \in R_{sd}[q]\, \exists a \in A(r) : -\partial a \in P(1..i)$

With the addition of auxiliary tags and corresponding inference rules, the inference rules for ∂ are reduced to:

$+\partial$: We may append $P(i+1) = +\partial q$ if either

$+\Delta q \in P(1..i)$ or

$\{+\sigma q, -\Delta \sim q, -\sigma \sim q\} \subseteq P(1..i)$

$-\partial$: We may append $P(i+1) = -\partial q$ if

$-\Delta q \in P(1..i)$ and

$\{-\sigma q, +\Delta \sim q, +\sigma \sim q\} \cap P(1..i) \neq \emptyset$

3.5.2 Efficient Implementation

The algorithm to compute all conclusions of a defeasible theory is shown in Algorithm 1. This algorithm is based on a modification of forward chaining for propositional logic with rules to label conclusions from the theory. The algorithm first transforms the input theory into basic theory and duplicates all strict rules as defeasible ones. A set of definitely (and defeasibly) provable literals S is initialized from the known facts. Those provable literals iteratively "fire" a set of rules whose bodies contain them while the unprovable literals (which is added later) remove the corresponding rules. After modifying the rule, the second part of the iteration (line 13 to 28) generates the new definitely provable and unprovable conclusions by finding the applicability or absence of strict rules, respectively, and adding them to the conclusion set. It also records the tentatively provable and unprovable literals from the fired and removed defeasible rules, respectively. Procedure CheckInference uses the inference rules in previous subsection to check recorded literals in K and returns defeasibly provable and unprovable conclusions.

Since the second part of the algorithm is the iteration to compute defeasible tags, it is essential to optimize the calculation to achieve a linear complexity. The key to an efficient implementation of this algorithm is the data structure used to represent the rules. Figure 3.3 illustrated the data structures for the theory

$$\begin{array}{rrr} r_1 : & b, c, d \Rightarrow & a \\ r_2 : & \neg, b, d, \neg e \Rightarrow & a \\ r_3 : & d, \neg e \Rightarrow & \neg a \end{array}$$

Each rule body is represented as a doubly-linked list (horizontal arrows in Figure 3.3). Furthermore, for each literal p there are doubly-linked of the occurrences of p in the bodies of rules (diagonal arrows). For each literal p, there is a doubly-linked list of rules with

Algorithm 1: Compute all conclusions of a defeasible theory

```
Algorithm: CompAllConclusions($D$)
Input: A defeasible theory $D = (F, R, >)$
Output: Set of conclusions $C$
1  $D' = (F', R', \emptyset) \leftarrow$ Basic($D$)
2  $R \leftarrow$ DupStrict($R'$)
3  $S \leftarrow$ initialize($F'$)
4  $K \leftarrow \emptyset$
5  while $S \neq \emptyset$ do
6      choose $s \in S$ and delete $s$ from $S$
7      $C \leftarrow C \cup \{s\}$
8      switch s do
9          case $+\Delta p$: delete all occurrences of $p$ in strict rule bodies
10         case $-\Delta p$: delete all strict rules where $p$ occurs in the body
11         case $+\partial p$: delete all occurrences of $p$ in defeasible rule bodies
12         case $-\partial p$: delete all defeasible rules where $p$ occurs in the body
13     repeat
14         $S' = \emptyset$
15         while a body of a strict rule with head $h$ becomes empty do
16             $S' \leftarrow S' \cup \{+\Delta h\}$
17             $K \leftarrow K \cup \{+\Delta h, +\partial h, +\sigma h\}$
18             remove rule
19         while there are no more strict rules for a literal $h$, and $+\Delta h \notin S \cup C$ do
20             $S' \leftarrow S' \cup \{-\Delta h\}$
21             $K \leftarrow K \cup \{-\Delta h\}$
22         while a body of defeasible rule with head $h$ becomes empty do
23             $K \leftarrow K \cup \{+\sigma h\}$
24             remove rule
25         while there are no more defeasible rules for a literal $h$ do
26             $K \leftarrow K \cup \{-\sigma h\}$
27         $S' \leftarrow S' \cup$ CheckInference($K$)
28         $S \leftarrow S \cup S'$
29     until $S' \neq \emptyset$
30 return $C$
```

head p (dashed arrows). Each literal occurrence has a link to the record for the rule it occurs in (not shown in Figure 3.3).

This data structure optimizes the iteration part of defeasible reasoning algorithm and thus reduces the computational expense to linear complexity. It allows the deletion of literals and rules in time proportional to the number of literals deleted. Furthermore, we can detect in constant time whether a literal deleted was the only literal in that body, and whether a rule deleted with head h was the only rule for h. Each literal occurrence is deleted at most once, and the test for empty body is made at most once per deletion.

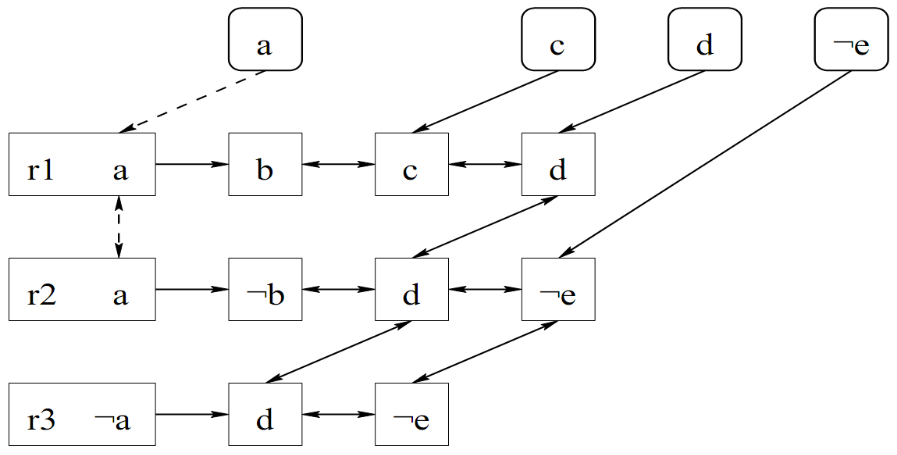


Figure 3.3: Data Structure for Rules in Defeasible Reasoning Algorithm

Similarly, each rule is deleted at most once, and the test for no more rules is made once per deletion. Thus the cost of the algorithm is $O(N)$, where N is the number of literal occurrences in D.

3.6 A Syntactical Approach for Defeasible DLs

3.6.1 Adding Defeasible Rules to DL Ontology

As indicated by the Semantic Web architecture in Figure 1.4, rules can be added on the top of ontology language to complete the knowledge representation. A natural approach to integrate defeasible reasoning to DL ontology to handle conflict knowledge is to add defeasible rules to DL ontology to represent conflict knowledge while considering TBox as set of strict rules and ABox set of facts. This syntactical approach is proposed for $\mathcal{ALC}^-$ DL in [40] and then extended for $\mathcal{ALE}$ DL in [91]. However, this syntactical approach shows some essential drawbacks because of the incompatibility between description logics and definite Horn logic (cf [8] for detail discussion about features of each logic that is not reducible to the other). To perform defeasible reasoning, a transformation is necessary to convert defeasible ontology into defeasible logic program. This transformation significantly increases the size of resulting theory and reduces the efficiency and applicability of syntactical approach of defeasible DL ontology. The following example and discussion will show the weakness of this approach.

A defeasible DL ontology is a tuple $\Sigma_{RDDL} = (\mathcal{A}, \mathcal{T}, R, >)$ (RDDL stands for rule-based defeasible DL) where $\mathcal{A}$ is the ABox, $\mathcal{T}$ is the TBox, R is a set of rules (in general, R can contain 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 in following subsection. Figure 3.4 shows

ABox:

EnglishLecture(K218)	BasicLecture(K211)	IntroLecture(K116)
MultiAreaCourse(K618)	MediaCourse(C612)	SystemCourse(K616)
KSProfessor(YAMADA)		
KSStudent(ROBERT)	supervises(YAMADA,SUZUKI)	KSStudent(NGUYEN)
passes(ROBERT,K218)	passes(ROBERT,K618)	passes(ROBERT,K601)

TBox:

(1) $\mathsf{Student} \sqsubseteq \mathsf{Adult}$

(2) $\mathsf{KSProfessor} \sqsubseteq \forall \mathsf{supervises.KSStudent}$

(3) $\forall \mathsf{passes.KSCourse} \sqsubseteq \mathsf{KSStudent}$

Rules:

(4) $\mathsf{Adult}(x) \Rightarrow \mathsf{PaysPension}(x)$

(5) $\mathsf{Student}(x) \Rightarrow \neg\mathsf{PaysPension}(x)$

>: (5) > (4)

Figure 3.4: An example of DL ontology with defeasible rules

an example of defeasible DL ontology demonstrates the example of inconsistent ontology presented in Example 2.2. In this ontology, defeasible rules are used to represent the conflict knowledge which states that an adult has to pay pension contribution (rule 4) while the student does not (rule 5) and the superiority relation states that the latter is preferred to the former. Three terminological axioms (1)–(3) are used to demonstrate the transformation to defeasible logic program.

3.6.2 Transformation to Defeasible Logic Program

Defeasible reasoning for a defeasible DL consists of those following steps:

Step 1: Transform $\mathcal{T}$ to R_S and transform $\mathcal{A}$ to F, hence, $\Sigma_{RDDL} = \langle \mathcal{T}, \mathcal{A}, R, > \rangle$ is reduced to $\Sigma = \langle F, R, > \rangle$. Given a theory Σ_{RDDL} and a conclusion φ (i.e., a query), this step is designed such that $\Sigma_{RDDL} \models \varphi$ iff $\Sigma \vdash \varphi$.

Step 2: Propositionalize Σ to Σ_{prop}, including propositional auxiliary rules for universal quantified literals ($\forall R.C(x)$) in antecedents of rules. Given a theory Σ and a conclusion φ (i.e., a query), this step guarantees that $\Sigma \vdash \varphi$ iff $\Sigma_{prop} \vdash \varphi$. It is easy to verify that this statement follows immediately from the first step statement plus nature of propositionalization.

Step 3: Apply the linear algorithm on the transformed theory.

The transformation in Step 1 is straightforward from inclusion axioms to strict rules. In particular, the inclusion axiom $\sqcap_{i=1}^{n} C_i \sqsubseteq \sqcap_{j=1}^{m} D_j$ is equivalent to the following set of strict rules (the set of rules induced by $\mathcal{T}$)

$$\begin{gathered} C_1, \ldots, C_n \rightarrow D_1 \\ \vdots \\ C_1, \ldots, C_n \rightarrow D_m \end{gathered}$$

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

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

For example, axiom (1) in above figure is transformed to $\mathsf{Student}(x) \rightarrow \mathsf{Adult}(x)$ and $\neg\mathsf{Adult}(x) \rightarrow \neg\mathsf{Student}(x)$. For axioms (2) and (3), the contraposition is not included because there are non-atomic concepts in the axioms and the negation is only allowed in atomic concepts. The universal restrictions in axioms (2) and (3) are processed differently depends on the position of the restrictions. If the universal restrictions appears in the right-hand-side of the axiom, it is recursively analyzed to move the role in the restriction to the left-hand-side and keep the concept in the restriction in RHS. For example, axiom (2) is transformed to

$$\mathsf{KSProfessor}(x), \mathsf{supervises}(x, y) \rightarrow \mathsf{KSStudent}(y).$$

The universal restriction still remain in the converted rules if it occurs in the LHS of the axiom like in axiom (3), which is transformed to

$$\forall \mathsf{passes}.\mathsf{KSCourse}(x) \rightarrow \mathsf{KSStudent}(x).$$

At this point, we need additional inference rule to deal with the remaining universal quantified concepts. However, universal quantified concepts' semantics takes into account all individuals in the DL knowledge base, in particular the ABox. Consequently, the proof conditions for universal quantified concepts will incorporate the domain of ABox $\mathcal{A}$, $\triangle_{\mathcal{A}}^{\mathcal{I}}$, in themselves as follows:

$+\Delta\forall R.C$: $+\Delta\forall R.C(a) \in P(i+1)$ iff
  $\forall b \in \triangle_{\mathcal{A}}^{\mathcal{I}}$, either (1) $-\Delta R(a, b)$ or (2) $+\Delta C(b)$
$-\Delta\forall R.C$: $-\Delta\forall R.C(a) \in P(i+1)$ iff
  $\exists b \in \triangle_{\mathcal{A}}^{\mathcal{I}}$ such that (1) $+\Delta R(a, b)$ and (2) $-\Delta C(b)$

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

$+\partial\forall R.C : +\partial\forall R.C(a) \in P(i+1)$ iff

$\forall b \in \triangle^{\mathcal{I}}_{\mathcal{A}}$ either $(1) -\partial R(a,b)$ or $(2) +\partial C(b)$

To prove a positive defeasible role restriction $+\partial\forall R.C(a)$ we have to prove that for all 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 a 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.

$-\partial\forall R.C : -\partial\forall R.C(a) \in P(i+1)$ iff

$\exists b \in \triangle^{\mathcal{I}}_{\mathcal{A}}$ such that $(1) +\partial R(a,b)$ and $(2) -\partial C(b)$

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.

In Step 2, for each universal quantified literal $\forall R.C(x)$ in an antecedent of a rule, following propositional auxiliary variables and rules are created to compute its provability:

```
for every $a_i$ in $\triangle^{\mathcal{I}}_{\mathcal{A}}$,
  $RC(a_i, a_1), \ldots, RC(a_i, a_n) \to \forall R.C(a_i)$
  for every $a_j$ in $\triangle^{\mathcal{I}}_{\mathcal{A}}$,
    $\sim R(a_i, a_j) \to RC(a_i, a_j)$
    $C(a_j) \to RC(a_i, a_j)$.
```

If C is a universal quantified literal, the process is recursively applied on C.

In the propositionalization step, each universal quantified literal in antecedent adds $O(|\triangle^{\mathcal{I}}_{\mathcal{A}}|^2)$ auxiliary rules to the theory. Thus, given d_v is the depth of nested quantifier in ontology, the size of propositionalized theory is $O(|\triangle^{\mathcal{I}}_{\mathcal{A}}|^{d_v})$. And since the logic allows roles (binary predicates), the complexity of resulting theory is $O(|\triangle^{\mathcal{I}}_{\mathcal{A}}|^{d_v+2})$, in the case we apply a linear algorithm to perform defeasible reasoning. Moreover, the quantified number restrictions require investigating combination of all auxiliary variables. This implies that extending syntactical approach for more expressive defeasible DL ontology like $\mathcal{SHOIQ}$ is untraceable in general.

Chapter 4

Defeasible DL Ontology

This chapter introduces our extension of DL ontology which includes defeasible subsumption axiom and priority relation to handle conflict knowledge in inconsistent ontology. This extension is named *defeasible DL* and presented in the first section. The following section discusses the new type of satisfiability in defeasible DL ontology, *defeasible satisfiability*, and investigates *defeasible reasoning* procedure to compute defeasible satisfiability. The third section represents the formal interpretation of defeasible satisfiability. An heuristic method to determine defeasible axioms and priority relations to facilitate the use of defeasible DL ontology is introduced in the last section.

4.1 Defeasible DL $\mathcal{DSHOIQ}$

Defeasible DL ontology employs *defeasible subsumption axiom* $C \sqsubsetsim D$ to represent the meaning that most C's are typically D's. These axioms are "weaker" than the classical ones which express that all C's are D's in the sense that $C \sqsubseteq D$ also implies $C \sqsubsetsim D$. Defeasible axioms are complemented by *superiority relations* among axioms to express preference among conflict knowledge.

Definition 1. *A $\mathcal{DSHOIQ}$ ontology is a tuple $\Sigma = \langle \mathcal{T}, \mathcal{D}, > \rangle$ where $\mathcal{T}$ is a TBox, $\mathcal{D}$ a set of defeasible axioms (DBox), and $>$ a set of superiority relations among axioms of $\mathcal{T} \cup \mathcal{D}$.*

There are two types of satisfiability for defeasible DL: classical and defeasible satisfiability. *Classical satisfiability* only considers classical axioms while defeasible satisfiability takes into account both classical axioms and defeasible axioms. Therefore, *classical consistency* and *satisfiability* with respect to a defeasible ontology Σ can be defined using the classical part of the ontology as in the following definition.

Definition 2. *Given a defeasible DL ontology Σ, a concept C is classically satisfiable with respect to Σ if and only if C is satisfiable with respect to $\mathcal{T}$. As a consequence, Σ*

is classically consistent if and only if $\mathcal{T}$ is consistent. Σ classically satisfies an axiom $C \sqsubseteq D$, denoted as $\Sigma \mathrel{|\!\approx} C \sqsubseteq D$, if and only if $\mathcal{T} \models C \sqsubseteq D$. Then, D is a classical subsumer of C and we denote Δ_C as the set of classical subsumers of C.

In above definition, we use the non-standard inference relation $\mathrel{|\!\approx}$, which is introduced in [56], for reasoning with defeasible DL ontologies. This notation indicates that we utilize a consistent sub-theory of Σ (in this case $\mathcal{T}$) to check the satisfiability of an axiom. We will employ and discuss more detail about this notation in the next subsection when studying defeasible satisfiability.

Let us demonstrate how defeasible DL can resolve inconsistentcy problem by revisitting motivating examples introduces in Section 2.5.

Example 4.1 (Penguin revisited)**.** To resolve the conflict in integrated ontology, we use a naïve approach that axioms from remote ontology is considered to be "weaker" than axioms from the local ontology. Assume we want to perform reasoning from Penguin ontology, for example to infer subsumption relations of Penguin concepts, axioms of Bird ontology are converted to defeasible ones. The integrated ontology is denoted as follow:

$$\Sigma_{Bird} = \left\langle \begin{array}{l} \left\{ \begin{array}{l} PB : \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ PF : \mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly} \end{array} \right\}, \\ \left\{ \begin{array}{l} BF : \mathsf{Bird} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{Fly}, \\ BW : \mathsf{Bird} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{HasWings} \end{array} \right\}, \\ \emptyset \end{array} \right\rangle$$

Example 4.2 (Student revisited)**.** The regulation that adults have to pay pension contribution is the general rule which can be overruled by some specific case such as a student whose income is usually less than a certain level are not required to pay the monthly contributions. The latter in turn can be reversed with more specific case such as when the student attends a research program and earn the regular salary. Therefore, we convert regulation axioms into defeasible ones and add the order set $>$ as following formal representation:

$$\Sigma_{Student} = \left\langle \begin{array}{l} \left\{ \begin{array}{l} SA : \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Adult}, \\ SS : \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Student} \end{array} \right\}, \\ \left\{ \begin{array}{l} AP : \mathsf{Adult} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{PaysPension}, \\ SP : \mathsf{Student} \mathrel{\underset{\sim}{\sqsubset}} \neg\mathsf{PaysPension} \end{array} \right\}, \\ \{SP > AP\} \end{array} \right\rangle$$

In above examples, adding defeasibility to terminological axioms will remove contradiction with respect to the classical parts of those ontologies. However, it is our interest that we can derive defeasible inclusion axioms such as $\mathsf{Penguin} \mathrel{\underset{\sim}{\sqsubset}} \mathsf{HasWings}$ from Σ_{Bird}.

Moreover, we also want to employs the superior relation in the inference, i.e. allow MP to defeat TP and derive $\{\mathsf{SUZUKI}\} \precsim \neg\mathsf{PaysPension}$ from $\Sigma_{Student}$.

4.2 Reasoning for Defeasible DL Ontology

4.2.1 Defeasible Satisfiability

We study defeasible satisfiability by considering reasoning tasks for defeasible ontology. Two basic reasoning tasks for defeasible ontology that extend from corresponding classical reasoning services are *defeasible instance checking* and *defeasible subsumption checking.* Both tasks can be represented as answering the query $\Sigma \mid\!\approx \varphi$ where $\varphi = \{a\} \precsim C$ (instance checking) or $\varphi = C \precsim D$ (subsumption checking).

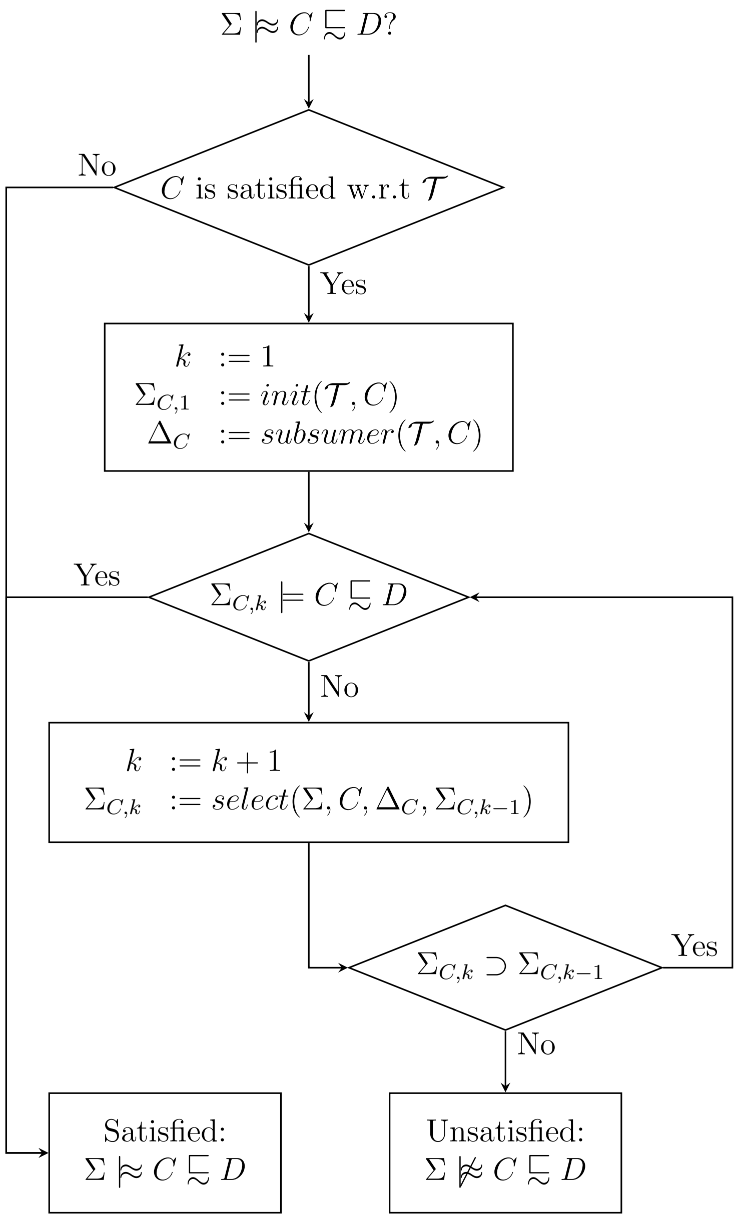
Huang et al. introduces a general framework to answer those queries from an inconsistent ontologies by constructing a consistent subtheory in [56]. This framework utilizes a *selection function s* to partition the axioms in the ontology. Then, the reasoning framework will add those partitions iteratively to construct subtheories of the ontology and check the answer of the query by using those subtheories. The iteration stops when the answer is achieved or no more consistent subtheory is constructed. Since the function linearly adds axiom to extend subtheories, this frame work is named linear extension strategy.

We modify Huang's framework to utilize a *skeptical selection function* for defeasible reasoning, as illustrated in Figure 4.1. There are two important differences in this modification compared to the original work. Firstly, inconsistency checking only needs to be performed at the first step of defeasible reasoning. The skeptical selection function ensures that additional axioms to a consistent subtheory do not make the subtheory inconsistent. In this step, if the condition of the query is not classically satisfied, the query is also satisfied according to the property of classical reasoning.

The second difference relates to *selection function select*. In our framework, subtheory at the step k is computed by using the subtheory of step $k-1$. In order to preserve the consistency, the skeptical selection function needs to use the reasoning results of the previous step to prevent any conflict in advance. And due to the separation of defeasible and classical axioms, the selection process at the initial step is also dedicated as *init* function. The following two subsections explain *init* and *select* functions in detail.

4.2.2 Initial Selection Function

The selection function *init* finds the subtheory of classical part of the ontology that is *justified* with respect to C. Given $\mathcal{T}$ classical part of an ontology and C an concept, this function computes all axioms of $\mathcal{T}$ which are justified with respect to C. This step



$\Sigma \models_{\approx} C \sqsubseteq_{\sim} D?$
C is satisfied w.r.t $\mathcal{T}$
No
Yes
$k := 1$
$\Sigma_{C,1} := init(\mathcal{T}, C)$
$\Delta_C := subsumer(\mathcal{T}, C)$
$\Sigma_{C,k} \models C \sqsubseteq_{\sim} D$
Yes
No
$k := k+1$
$\Sigma_{C,k} := select(\Sigma, C, \Delta_C, \Sigma_{C,k-1})$
$\Sigma_{C,k} \supset \Sigma_{C,k-1}$
Yes
No
Satisfied: $\Sigma \models_{\approx} C \sqsubseteq_{\sim} D$
Unsatisfied: $\Sigma \not\models_{\approx} C \sqsubseteq_{\sim} D$

Figure 4.1: Linear extension strategy for defeasible reasoning

corresponds to showing definite provability of defeasible logic. *Justifiability*, a "stricter" semantics for satisfiability in defeasible KB than traditional semantics, is established in Definition 3 and 4. This semantics requires every axiom in the theory to involve in the reasoning when answering a query.

Definition 3. *Let $\Sigma = \langle \mathcal{T}, \mathcal{D}, > \rangle$ be an $\mathcal{DSHOIQ}$-knowledge base and $\mathcal{I} = (\triangle^{\mathcal{I}}, \cdot^{\mathcal{I}})$ an **interpretation** of Σ. A terminological axiom $\alpha = A \sqsubseteq B \in \mathcal{T}$ or $\alpha = A \sqsubseteq_{\sim} B \in \mathcal{D}$ is **justified** w.r.t $x \in \triangle^{\mathcal{I}}$ and $\mathcal{I}$ iff $x \in A^{\mathcal{I}}$ and $x \in B^{\mathcal{I}}$. α is **justified** w.r.t to a concept expression C and an interpretation $\mathcal{I}$ if $\forall x \in \triangle^{\mathcal{I}}$ if $x \in C^{\mathcal{I}}$ then α is justified w.r.t x. If each axiom in Σ is justified w.r.t to C and $\mathcal{I}$, $\mathcal{I}$ is called an **justified model** of Σ w.r.t C.*

Definition 4. *A $\mathcal{DSHOIQ}$ ontology Σ is* ***justified*** *w.r.t a concept expression C if there exists an justified model $\mathcal{I}$ of Σ such that $C^{\mathcal{I}} \neq \emptyset$ and for all model $\mathcal{J}$ of Σ, $\mathcal{J}$ is an justified model w.r.t. C. Then, we say that Σ* ***justifies*** *a* ***defeasible subsumption*** *$C \sqsubsetsim D$, denoted as $\Sigma \models C \sqsubsetsim D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each justified model $\mathcal{I}$ of Σ.*

Example 4.3 (Penguin extended)**.** Assume we extend Penguin ontology in Example 4.1 by adding new axiom that eagles are birds.

$$\Sigma_{Bird} = \left\langle \begin{array}{l} \left\{ \begin{array}{ll} PB: & \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ PF: & \mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly} \\ EB: & \mathsf{Eagle} \sqsubseteq \mathsf{Bird} \end{array} \right\}, \\ \left\{ \begin{array}{ll} BF: & \mathsf{Bird} \sqsubsetsim \mathsf{Fly}, \\ BW: & \mathsf{Bird} \sqsubsetsim \mathsf{HasWings} \end{array} \right\}, \\ > \end{array} \right\rangle$$

Let consider the TBox $\mathcal{T}_{Bird}$. Assume two example models $\mathcal{I}_1$ and $\mathcal{I}_2$ of $\mathcal{T}_{Bird}$ such that $\triangle^{\mathcal{I}_1} = \triangle^{\mathcal{I}_2} = \{\mathsf{EMU}\}$ and interpretation functions are presented in the following table:

	Penguin	Bird	Fly	Eagle
$\mathcal{I}_1$	{EMU}	{EMU}	{EMU}	{EMU}
$\mathcal{I}_2$	{EMU}	{EMU}	{EMU}	$\emptyset$

Both $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy all three axioms in $\mathcal{T}_{Bird}$ but only $\mathcal{I}_1$ justifies all those axioms while $\mathcal{I}_2$ does not justify EB. It is easy to show that for every model $\mathcal{I}$ of $\mathcal{T}_{DB}$ such that $\mathsf{Penguin}^{\mathcal{I}} \neq \emptyset$, two axioms PB and PF is justified. Therefore, we say that PB and PF is justified with respect to Penguin but EB is not. $\mathcal{T}_{Bird}$ is not justified with respect to Penguin.

For answering defeasible query relating to Penguin concept, the result of initial step is as follows:

$$\Sigma_{Penguin,1} = \left\{ \begin{array}{ll} PB & : \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ PF & : \mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly}, \end{array} \right\}.$$

In addition, the set of definite subsumers of Penguin is:

$$\Delta_{Penguin} = \{\mathsf{Penguin}, \mathsf{Bird}, \neg\mathsf{Fly}\}.$$

Example 4.4 (Student extended)**.** We add the regulation that GRP students who have income have to pay pension. This extended ontology will be used to demonstrate defeasible reasoning later. The interpretation of extended ontology is as follows:

$$\Sigma_{Student} = \left\langle \begin{array}{l} \left\{ \begin{array}{ll} SA & : \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Adult}, \\ SS & : \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Student} \end{array} \right\}, \\ \left\{ \begin{array}{ll} AP & : \mathsf{Adult} \precsim \mathsf{PaysPension}, \\ SP & : \mathsf{Student} \precsim \neg\mathsf{PaysPension}, \\ GP & : \mathsf{GRPStudent} \precsim \mathsf{PaysPension} \end{array} \right\}, \\ \{SP > AP,\ GP > SP\} \end{array} \right\rangle.$$

Similar to analysis in Example 4.3, we can show that $\mathcal{T}_{Student}$ is justified with respect to $\{\mathsf{SUZUKI}\}$. The results of initial step for defeasibly reasoning about $\{\mathsf{SUZUKI}\}$ are $\Sigma_{Suzuki,1} = \mathcal{T}_{Student}$ and $\Delta_{Suzuki} = \{\{\mathsf{SUZUKI}\}, \mathsf{Adult}, \mathsf{Student}\}$.

4.2.3 Skeptical Selection Function

This subsection describes the main process in iteration of defeasible reasoning in the Figure 4.1, the *select* function. Given Σ an ontology, C a concept, Δ_C a set of definite subsumer of C and $\Sigma_{C,k-1}$ justified subtheory of Σ with respect to C at step $k-1$, this function chooses appropriated axioms in Σ, labeled as *justified* axioms, to construct $\Sigma_{C,k}$. The select function is named skeptical because it realizes the skeptical principle of defeasible reasoning as described in Subsection 2.3. This principle is expressed as the conditions to label concepts and axioms. Given a concept C, some concepts in Σ are labeled as *defeasible subsumer* of C or *defeasible non-subsumer* of C, corresponding to defeasible provability and defeasible non-provability, respectively. Besides, two auxiliary labels, *possible* and *tentative*, are used to help the computation. The conditions to label concepts are described below:

- For any concept description D, if $\Sigma_{C,k-1} \models C \precsim D$ then D is a *defeasible subsumer* of C. The set of defeasible subsumers of C is C_{sub}.
- The set of *possible subsumers* of C, denoted as C_{pos}, contains conclusions of all remaining axioms in Σ', where Σ' is the subset of Σ excluding axioms in $\Sigma_{C,k-1}$ and set of discarded axioms.
- The set of *tentative subsumers* of C, denoted as C_{ten}, consists of conclusions of all axioms that are applicable and not defeasible.
- The set of *defeasible non-subsumers* of C, denoted as C_{non}, consists of concept that satisfies one of following three conditions:
  (1) Concept whose complement is a defeasible subsumer of C, we denote this subset as C^1_{non}.

(2) Concept is neither definite nor possible subsumer of C, we denote this subset as C^2_{non}.

(3) Concept is not a defeasible subsumer of C and its complement is a tentative subsumer of C, we denote this subset as C^3_{non}.

The conditions to label axioms which correspond to the conditions of rules in defeasible provability explained in Section 3.3 are defined as follows:

- An axiom is *applicable* if all literals in its antecedent are defeasible subsumers of C. The set of applicable axioms is A_{app}.
- An axiom is *discarded* if one literal in its antecedent is defeasible non-subsumer of C. The set of discarded axioms is A_{dis}.
- An axiom is *defeasible* if there exists a conflicting axiom which is not discarded and stronger than it. Two axioms are conflicting if they have contradict literals as their conclusions. The set of defeasible axioms is A_{df}.
- An axiom is *defeated* if there exists a conflicting axiom which is applicable and stronger than it. The set of defeated axioms is A_{dfd}.
- An axiom is *justified* if satisfies all following conditions:

  (1) it is applicable;

  (2) its conclusion is not a defeasible non-subsumer of C;

  (3) and all of its conflicting axioms are either discarded or defeated.

  The set of defeasible axioms is A_{jus}.

The above conditions to classify concepts and axioms are the mutual interactive. For example, non-subsumer concepts cause some axioms discarded. Those axioms when withdrawn may remove some concepts from possible set and thus, add those concepts to non-subsumer set. Those new non-subsumer concepts in turn may cause other axioms discarded and the process repeats.

To solve this recursive problem, the *select* function is detailed as in Figure 4.2. Independent elements such as defeasible subsumers and applicable axioms are computed in the first step. Then an iteration starting from the second block is used to compute mutually interactive components. This iteration successively computes defeasible axioms, tentative and possible subsumers, and then defeasible non-subsumer concepts and discarded axioms. If the set of discard axioms is not empty, the procedure removes discards axiom from Σ' and continues to compute those sets again. Otherwise, the iteration terminates. Then, the sets of defeated and justified axioms are computed.

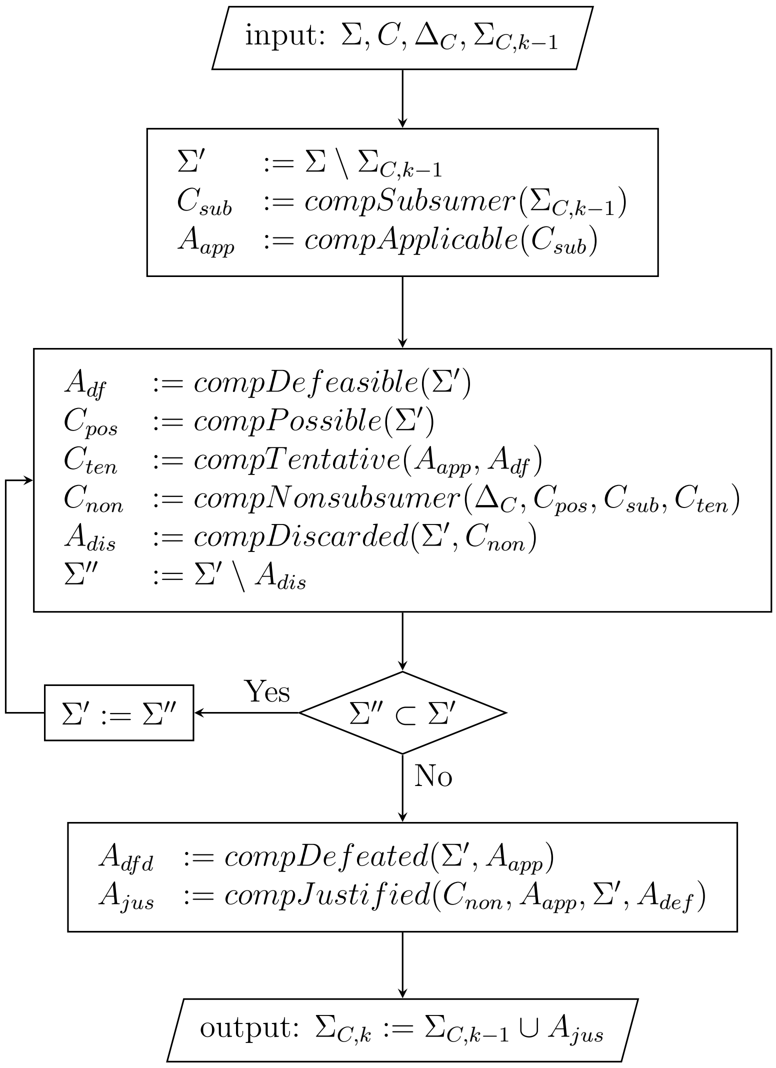

input: $\Sigma, C, \Delta_C, \Sigma_{C,k-1}$
$\Sigma' := \Sigma \setminus \Sigma_{C,k-1}$
$C_{sub} := compSubsumer(\Sigma_{C,k-1})$
$A_{app} := compApplicable(C_{sub})$
$A_{df} := compDefeasible(\Sigma')$
$C_{pos} := compPossible(\Sigma')$
$C_{ten} := compTentative(A_{app}, A_{df})$
$C_{non} := compNonsubsumer(\Delta_C, C_{pos}, C_{sub}, C_{ten})$
$A_{dis} := compDiscarded(\Sigma', C_{non})$
$\Sigma'' := \Sigma' \setminus A_{dis}$
$\Sigma' := \Sigma''$
Yes
$\Sigma'' \subset \Sigma'$
No
$A_{dfd} := compDefeated(\Sigma', A_{app})$
$A_{jus} := compJustified(C_{non}, A_{app}, \Sigma', A_{def})$
output: $\Sigma_{C,k} := \Sigma_{C,k-1} \cup A_{jus}$

Figure 4.2: Selection process of skeptical extension for defeasible reasoning

It is worth noting about the representation of an axiom. Let's consider a classical axiom in the general form as follow:

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

where C_i, D_j are *concept literals*, A or $\neg A$[1]; or *role literals*, $\exists R.C, \forall R.C, \geq nR.C$, or $\leq nR.C$ where C is in *negation normal form*.

[1]According to the convention of description logics, $\neg A$ is used to represent complement of A in defeasible DL.

This axiom is equivalent to one of the following transpositions:

$$\begin{array}{lll}
\sqcap_{i=1}^{n} C_i \sqcap_{j=2}^{m} (\neg D_j) & \sqsubseteq D_1 & (1) \\
\vdots & & \\
\sqcap_{i=1}^{n} C_i \sqcap_{j=1}^{m-1} (\neg D_j) & \sqsubseteq D_m & (m) \\
\sqcap_{i=2}^{n} C_i \sqcap_{j=1}^{m} (\neg D_j) & \sqsubseteq \neg C_1 & (m+1) \\
\vdots & & \\
\sqcap_{i=1}^{n-1} C_i \sqcap_{j=1}^{m} (\neg D_j) & \sqsubseteq \neg C_n & (m+n)
\end{array}$$

A simple treatment to handle those transpositions is to replace the classical axiom by $m+n$ defeasible axioms, each axiom corresponds to one transposition. For defeasible axiom, $C \lesssim D$, it is not necessary to consider its contrapositive, i.e. $\neg D \lesssim \neg C$, as discussed in [38]. Therefore, a defeasible axiom is replaced by the transpositions (1) to (m). In the following parts of this paper, we consider Σ' as the transformation of original ontology according to this treatment. The following examples demonstrate the computational procedure of *select* function after the initial step.

Example 4.5 (Penguin reasoning)**.** We consider the selection process after initial step for extended Bird ontology in Example 4.3. The remaining ontology after removing $\Sigma_{Penguins,1}$ is

$$\Sigma' = \left\langle \begin{array}{l} \left\{ \begin{array}{ll} EB: & \mathsf{Eagle} \lesssim \mathsf{Bird}, \\ BE: & \neg\mathsf{Bird} \lesssim \neg\mathsf{Eagle}, \\ BF: & \mathsf{Bird} \lesssim \mathsf{Fly}, \\ BW: & \mathsf{Bird} \lesssim \mathsf{HasWings} \end{array} \right\}, \\ > \end{array} \right\rangle.$$

$$\Delta_{Penguin} = \{\mathsf{Penguin}, \mathsf{Bird}, \neg\mathsf{Fly}\}$$

$$\Sigma_{Penguin,1} = \left\{ \begin{array}{ll} PB & : \mathsf{Penguin} \sqsubseteq \mathsf{Bird}, \\ PF & : \mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly} \end{array} \right\}$$

The set of defeasible subsumer of Penguin inferred from $\Sigma_{Penguin,1}$ is

$C_{Sub} = compSubsumer(\Sigma_{Penguin,1}) = \{\mathsf{Penguin}, \mathsf{Bird}, \neg\mathsf{Fly}\}$.

Then the set of applicable axioms is:

$A_{app} = compApplicable(S_{sub}) = \{BF, BW\}$.

For the first iteration of computing recursive elements, the set of defeasible axioms compute from the remaining ontology is:

$A_{df} = compDefeasible(\Sigma') = \emptyset$

The set of all possible subsumers, which are the conclusions of remaining axioms, is:

$C_{pos} = compPossible(\Sigma') = \{\mathsf{Fly}, \mathsf{HasWings}, \neg\mathsf{Eagle}, \mathsf{Bird}\}$

The set of all tentative subsumers, which are the conclusions of applicable but not defeasible axioms, is:

$C_{ten} = compTentative(A_{app}, A_{def}) = \{\mathsf{Fly}, \mathsf{HasWings}\}$

The set of non-subsumer of Penguin is represented as the union of three sets corresponding to three conditions, respectively:

$$\begin{aligned}
C_{non} &= compNonsubsumer(\Delta_{Penguin}, C_{pos}, C_{sub}, C_{ten}) \\
&= C^1_{non} \cup C^2_{non} \cup C^3_{non} \text{ where} \\
C^1_{non} &= \{\neg\mathsf{Penguin}, \neg\mathsf{Bird}, \mathsf{Fly}\}, \\
C^2_{non} &= \{\neg\mathsf{Penguin}, \neg\mathsf{HasWings}, \mathsf{Eagle}, \neg\mathsf{Bird}\}, \\
C^3_{non} &= \{\neg\mathsf{HasWings}\}.
\end{aligned}$$

Then the set of discarded axioms is

$A_{dis} = compDiscarded(\Sigma', C_{non}) = \{EB, BE\}$.

Since A_{dis} is not empty, the iteration continues after removing A_{dis} from Σ'. The results of second iteration are as follows:

$$\begin{aligned}
\Sigma' &= \left\langle \begin{array}{l} \left\{ \begin{array}{ll} BF: & \mathsf{Bird} \precsim \mathsf{Fly}, \\ BW: & \mathsf{Bird} \precsim \mathsf{HasWings} \end{array} \right\}, \\ > \end{array} \right\rangle \\
A_{df} &= \emptyset, \\
C_{pos} &= \{\mathsf{Fly}, \mathsf{HasWings}\}, \\
C_{ten} &= \{\mathsf{Fly}, \mathsf{HasWings}\}, \\
C_{non} &= \{\neg\mathsf{Penguin}, \neg\mathsf{Bird}, \mathsf{Fly}\} \cup \\
&\quad \{\neg\mathsf{Penguin}, \neg\mathsf{HasWings}, \mathsf{Eagle}, \neg\mathsf{Eagle}, \neg\mathsf{Bird}\} \cup \\
&\quad \{\neg\mathsf{HasWings}\}, \\
A_{dis} &= \emptyset.
\end{aligned}$$

The iteration terminates because A_{dis} is empty. Then the sets of defeated axioms and justified axioms are

$A_{dfd} = compDefeated(\Sigma', A_{app}) = \emptyset$ and

$A_{jus} = compJustified(C_{non}, A_{app}, \Sigma', A_{def}) = \{BW : \mathsf{Bird} \precsim \mathsf{HasWings}\}$.

Axiom BF : Bird $\precsim$ Fly is not justified because Fly is a defeasible non-subsumer of Penguin.

Example 4.6 (Student reasoning)**.** For selection function after the initial step of Student ontology in Example 4.4, the remaining ontology is

$$\begin{aligned}
\Sigma' &= \left\langle \begin{array}{l} \left\{ \begin{array}{ll} AP: & \mathsf{Adult} \precsim \mathsf{PaysPension}, \\ SP: & \mathsf{Student} \precsim \neg\mathsf{PaysPension}, \\ GP: & \mathsf{GRPStudent} \precsim \mathsf{PaysPension} \end{array} \right\}, \\ \{SP > AP,\ GP > SP\} \end{array} \right\rangle \\
\Delta_{Suzuki} &= \{\{\mathsf{SUZUKI}\}, \mathsf{Adult}, \mathsf{Student}\}
\end{aligned}$$

Results of initial step of *select* function are as follows:

$C_{sub} = \{\{\mathsf{SUZUKI}\}, \mathsf{Adult}, \mathsf{Student}\}$,
$A_{app} = \{AP, SP\}$.

Results of computation in the first iteration for recursive elements are as follows:

$A_{df} = \{SP : \mathsf{Student} \precsim \neg\mathsf{PaysPension}, AP : \mathsf{Adult} \precsim \mathsf{PaysPension}\}$,
$C_{pos} = \{\mathsf{PaysPension}, \neg\mathsf{PaysPension}\}$,
$C_{ten} = \emptyset$,
$C_{non} = \{\neg\{\mathsf{SUZUKI}\}, \neg\mathsf{Adult}, \neg\mathsf{Student}, \mathsf{GRPStudent}, \neg\mathsf{GRPStudent}\}$,
$A_{dis} = \{GP\}$.

The results of computation in the second iteration are as follows:

$$\Sigma' = \left\langle \begin{array}{l} \left\{ \begin{array}{ll} AP: & \mathsf{Adult} \precsim \mathsf{PaysPension}, \\ SP: & \mathsf{Student} \precsim \neg\mathsf{PaysPension} \end{array} \right\}, \\ \{SP > AP\} \end{array} \right\rangle,$$

$A_{df} = \{AP : \mathsf{Adult} \precsim \mathsf{PaysPension}\}$,
$C_{pos} = \{\mathsf{PaysPension}, \neg\mathsf{PaysPension}\}$,
$C_{ten} = \{\neg\mathsf{PaysPension}\}$
$C_{non} = \{\neg\{\mathsf{SUZUKI}\}, \neg\mathsf{Adult}, \neg\mathsf{Student}, \mathsf{GRPStudent}, \neg\mathsf{GRPStudent}, \mathsf{PaysPension}\}$,
$A_{dis} = \emptyset$.

After the iteration, the results of remaining computation are $A_{dfd} = \{AP : \mathsf{Adult} \precsim \mathsf{PaysPension}\}$ and $A_{jus} = \{SP : \mathsf{Student} \precsim \neg\mathsf{PaysPension}\}$. Axiom SP is justified because its conflict axiom GP is defeated. Meanwhile, AP is not justified since SP is not defeated.

4.3 Semantics of Defeasible DL

Given the reasoning approach to verify defeasible query as described in previous subsections, we can define a formal semantics for defeasible DL as follows.

Definition 5. *Let Σ be a $\mathcal{DSHOIQ}$ ontology, C a concept expression, Σ_C a sub-theory of Σ that is justified with respect to C.*

1. *A concept expression D is*
   a) ***defeasible subsumer*** *with respect to C and Σ_C iff $\Sigma_C \models C \precsim D$;*
   b) ***possible subsumer*** *with respect to C and Σ_C iff $\exists \alpha \in \Sigma \setminus \Sigma_C \cup \Sigma^-$ such that $\mathsf{C}(\alpha) = D$, where Σ^- is the set of discarded axioms of Σ with respect to C and Σ_C and $\mathsf{C}(\alpha)$ is conclusion of α;*
   c) ***tentative subsumer*** *with respect to C and Σ_C iff $\exists \alpha \in \Sigma \setminus \Sigma_C \cup \Sigma^-$ such that $\mathsf{C}(\alpha) = D$ and α is applicable and not defeasible with respect to C and Σ_C;*
   d) ***defeasible non-subsumer*** *with respect to C and Σ_C iff ($\neg D$ is a defeasible subsumer with respect to C and Σ_C) or (D is neither definite subsumer nor*

possible subsumer with respect to C and Σ_C) or (D is not a defeasible subsumer and $\neg D$ is a tentative subsumer with respect to C and Σ_C).

2. *A terminological axiom $\alpha \in \Sigma \setminus \Sigma_C \cup \Sigma^-$ is*

   a) ***applicable*** *with respect to C and Σ_C iff $\forall D$ such that $\mathsf{A}(\alpha) \sqsubseteq D, D$ is a defeasible subsumer with respect to C and Σ_C where $\mathsf{A}(\alpha)$ is the antecedent of α;*

   b) ***discarded*** *with respect to C and Σ_C iff $\exists D$ such that $\mathsf{A}(\alpha) \sqsubseteq D$ and D is defeasible non-subsumer with respect to C and Σ_C;*

   c) ***defeasible*** *with respect to C and Σ_C iff $\exists \beta \in \Sigma \setminus \Sigma_C \cup \Sigma^-$ such that $\mathsf{C}(\beta) = \neg \mathsf{C}(\alpha)$ and β is not discarded with respect to C and Σ_C and $\beta > \alpha$;*

   d) ***defeated*** *with respect to C and Σ_C iff $\exists \beta \in \Sigma \setminus \Sigma_C \cup \Sigma^-$ such that $\mathsf{C}(\beta) = \neg \mathsf{C}(\alpha)$ and β is applicable with respect to C and Σ_C and $\beta > \alpha$;*

   e) ***justified*** *with respect to C and Σ_C iff (α is applicable) and ($\mathsf{C}(\alpha)$ is not a defeasible non-subsumer) and ($\forall \beta \in \Sigma \setminus \Sigma_C \cup \Sigma^-$ such that $\mathsf{C}(\beta) = \neg \mathsf{C}(\alpha)$, β is defeated).*

Definition 6. *Let Σ be a $\mathcal{DSHOIQ}$ ontology, C a concept expression. Subtheories of Σ with respect to C are defined as follows:*

$$\begin{aligned} \Sigma_{C,1} &= \{\alpha |\ \alpha \in \mathcal{T} \cup \{C \sqsubseteq C\} \text{ and } \alpha \text{ is justified with respect to } C\} \\ \Sigma_{C,n+1} &= \Sigma_{C,n} \cup \{\alpha \mid \alpha \in \Sigma \text{ and } \alpha \text{ is justified with respect to } C \text{ and } \Sigma_{C,n}\} \end{aligned}$$

$\Sigma_{C,M}$ is the ***maximal subtheory*** *with respect to C of Σ if $\Sigma_{C,M} = \Sigma_{C,M+1}$.*

Given a concept C, subtheories of Σ with respect to C form a *strictly partial order*:

$$\Sigma_{C,1} \subset \Sigma_{C,2} \subset \cdots \subset \Sigma_{C,M} \subseteq \Sigma.$$

Finally, defeasible satisfiability of a defeasible subsumption is given in Definition 7. An important property of a defeasible theory is that it is coherent in the terms that it do not derive contradict conclusions if its classical part is consistent, which is introduced in Theorem 8.

Definition 7. *A defeasible $\mathcal{DSHOIQ}$ ontology Σ* ***satisfies*** *a defeasible subsumption $C \underset{\sim}{\sqsubset} D$, denoted as $\Sigma \mathrel{|}\!\!\approx C \underset{\sim}{\sqsubset} D$ if $\Sigma_{C,M} \models C \underset{\sim}{\sqsubset} D$ where $\Sigma_{C,M}$ is the maximal applied subtheory of Σ with respect to C.*

Theorem 8. *If Σ is consistent and $\Sigma \mathrel{|}\!\!\approx C \underset{\sim}{\sqsubset} D$ then $\Sigma \not\mathrel{|}\!\!\approx C \underset{\sim}{\sqsubset} \neg D$.*

Proof by contradiction. Assume that $\Sigma \approx\!\!\!\!\!\models C \precsim D$ and $\Sigma \not\approx\!\!\!\!\!\models C \precsim \neg D$ there there exists m and n such that $\Sigma_{C,n} \models C \precsim D, n = 1$ or $\Sigma_{C,n-1} \not\models C \precsim D$ and $\Sigma_{C,m} \models C \precsim \neg D, m = 1$ or $\Sigma_{C,m-1} \models C \precsim \neg D$. Because Σ is consistent, which implies $\Sigma_{C,1}$ can not derives $C \sqsubseteq D$ and $C \sqsubseteq \neg D$ concurrently, m and n cannot be 1 concurrently. Without loss of generality, we can assume that $n \leq m$.
Case $1 \leq n < m$. Then $\Sigma_{C,m-1} \models C \precsim D$ and $\exists \alpha \in \Sigma_{C,m} \setminus \Sigma_{C,m-1}, \mathsf{C}(\alpha) = \neg D$ and α is justified with respect to C and $\Sigma_{C,m-1}$. This contradicts with the second condition of condition 2.e) in Definition 5 because D is a defeasible subsumer of C.
Case $1 < n = m$. Then $\exists \alpha \in \Sigma_{C,n} \setminus \Sigma_{C,n-1}$ and $\beta \in \Sigma_{C,m} \setminus \Sigma_{C,m-1}$ such that $\mathsf{C}(\alpha) = D$ and $\mathsf{C}(\beta) = \neg D$ and α, β are justified with respect to C and $\Sigma_{C,n-1}$. This contradicts with the third condition of condition 2.e) in Definition 5 because both conflict axioms are not defeated. Therefore, none of them can be justified. □

4.3.1 Complexity of Defeasible Reasoning

As discussed in Subsection 4.2, defeasible reasoning is performed by a linear strategy, expressed as a single loop in Figure 4.1, which employs ontology reasoner as the basis for checking consistency. However, in each iteration, the algorithm needs to use ontology reasoner to compute subsumers of concept iteratively. Therefore, the algorithm complexity is quadric according to ontology reasoning. Given n is the size of ontology and $\mathcal{C}$ the complexity of ontology reasoning, computational complexity of defeasible reasoning is $O(n^2\mathcal{C})$. Theoretically, the consistency problem of a $\mathcal{SHOIQ}$ ontology is NEXPTIME-complete [113], but for expressive ontologies that have been developed so far, the reasoning services can performed in practically acceptable [54]. Therefore, defeasible reasoning approach has a rational complexity for inference with practical ontologies.

4.4 Determine Defeasible Axioms and Priority Relations in Inconsistent Ontologies

Even defeasible DL provides an efficient way to deal with inconsistent ontologies, it may be difficult for ordinary ontology modelers to aware where to modify the ontology. We aim to facilitate the use of defeasible reasoning by providing automatic function to assign defeasibility and priority relationships among problem axioms. The defeasibility and priority relationships in examples used in previous sections are determined by the semantic meanings of the concepts and axioms and required human or expert knowledge. However, the following approach uses only syntactical relationship among axioms to give a quick solution for inconsistency. Ontology modelers then can modify the suggestion based on their own preference.

4.4.1 Detecting Inconsistency in Ontology

It is easy to verify that defeasible reasoning is not different from classical reasoning for entailment which is not related to the inconsistency. Therefore, we need to find the minimal set of axioms that cause the inconsistent, which defined as minimal inconsistent set as in following definition.

Definition 9 (Minimal Inconsistent Set (MIS))**.** *Given an inconsistent ontology Σ, a subset $\mathcal{M} \subseteq \Sigma$ is a* minimal inconsistent set (MIS) *if there exists a concept C such that $\mathcal{M} \models C \sqsubseteq \bot$ and for all $\mathcal{M}' \subsetneq \mathcal{M}, \mathcal{M}' \not\models C \sqsubseteq \bot$.*

Single MIS for inconsistent ontology can be generated by a “divide and conquer” method presented in Algorithm 2, which is modified from algorithm in [53]. The algorithm receives a set of restriction S, which is initially empty, and the ontology as inputs. It divides the ontology into to parts S_L and S_R and looks for inconsistency in each part. If inconsistency is found in one part, the algorithm recursively looks through this part for the minimal set of axioms. If inconsistency is not found, the algorithm computes the sub-part of S_L which causes the inconsistency w.r.t. $S \cup S_R$, denoted S'_L, and then compute the corresponding sub-part of S_R which causes the inconsistency w.r.t. $S \cup S'_L$, denoted S'_R. The combination of S'_L and S'_R is one MIS.

Algorithm 2: Compute one MIS for inconsistent ontology

```
Function 2: ComputeSingleMIS($\Sigma$)
1  return ComputeSingleMIS($\emptyset, \Sigma$)
Function 2R: ComputeSingleMIS($S, F$)
1  if $|F| = 1$ then
2      return $F$
3  $S_L, S_R \leftarrow$ split($F$)
4  if IsInconsistent($S \cup S_L$) then
5      return ComputeSingleMIS($S, S_L$)
6  if IsInconsistent($S \cup S_R$) then
7      return ComputeSingleMIS($S, S_R$)
8  $S'_L \leftarrow$ ComputeSingleMIS($S \cup S_R, S_L$)
9  $S'_R \leftarrow$ ComputeSingleMIS($S \cup S'_L, S_R$)
10 return $S'_L \cup S'_R$
```

Reconsider the Student ontology in previous examples in its classical form as below, Figure 4.3 demonstrates the calculation of one MIS for this ontology.

$$\Sigma_{Student} = \left\{ \begin{array}{l} \mathsf{Adult} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension} \\ \mathsf{GRPStudent} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \mathsf{Adult} \\ \mathsf{GRPStudent} \sqsubseteq \mathsf{Student} \end{array} \right\}$$

Given the procedure to compute one single MIS, Algorithm 3 generates all MIS by the following idea. It first find any MIS, then remove each of the axioms in the MIS individually, and find a new MIS according to the new ontology. This process is exhaustively performed to generate all MIS. An example to compute all MIS for Student ontology is shown in Figure 4.4. The algorithm presented here is a brief version of algorithm to compute justifications in ontology reasoning. For approaches that aim to debug the ontology such as in [60, 53], the algorithm for computing all MIS as well as the repair adopts Reiter's hitting set tree algorithm from Reiter's general theory diagnosis [100].

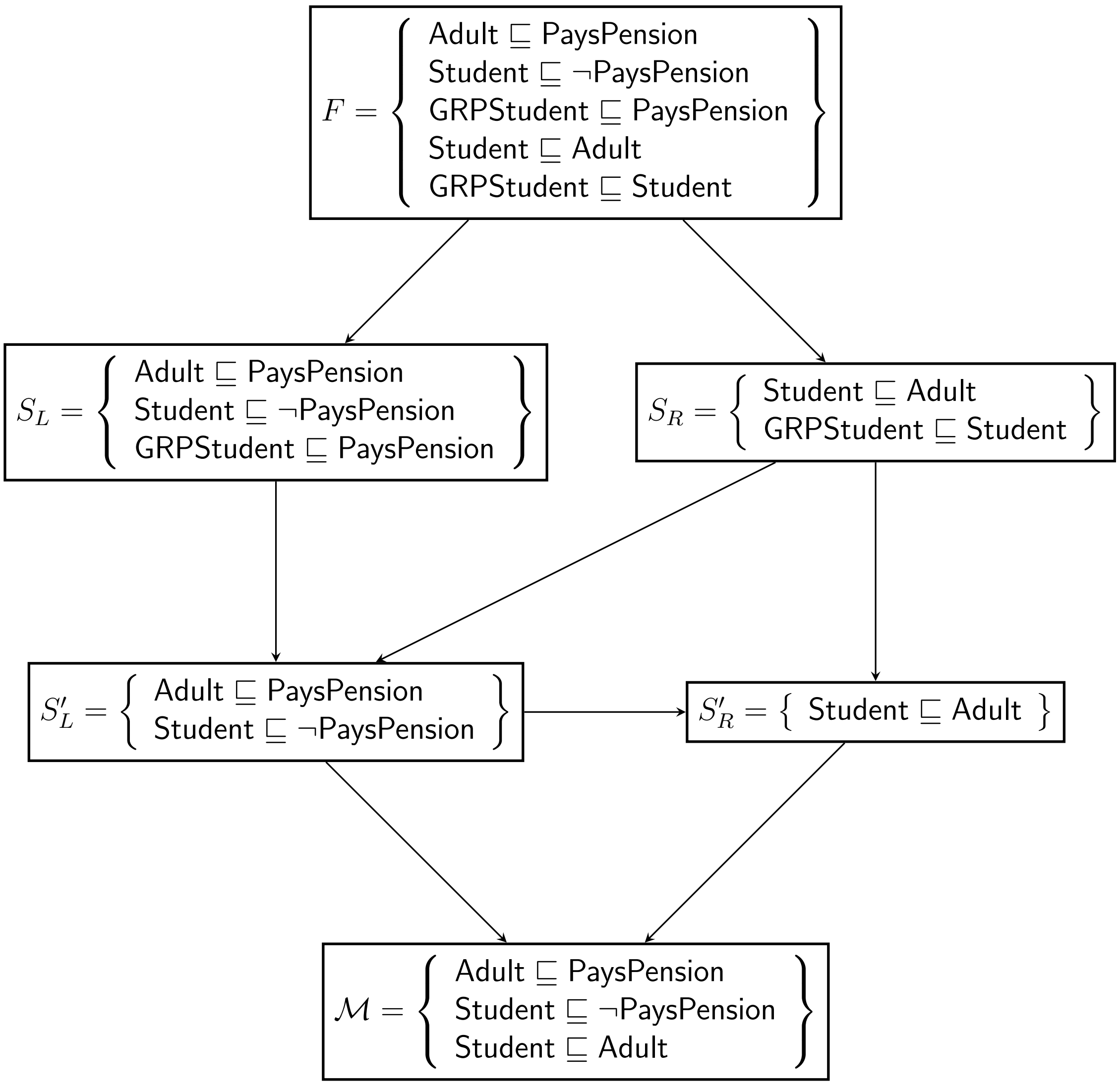
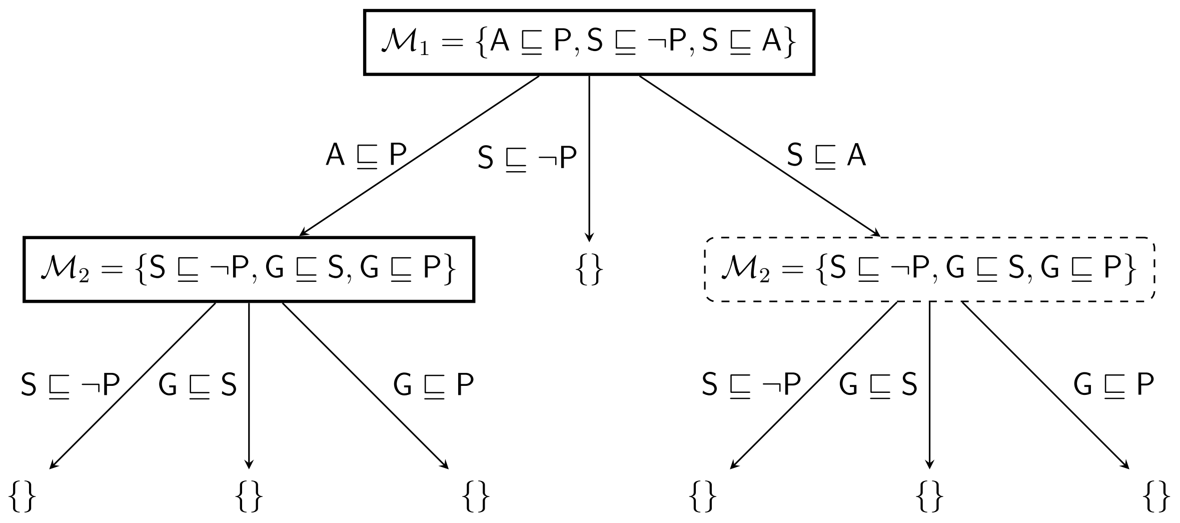

$F = \left\{ \begin{array}{l} \mathsf{Adult} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension} \\ \mathsf{GRPStudent} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \mathsf{Adult} \\ \mathsf{GRPStudent} \sqsubseteq \mathsf{Student} \end{array} \right\}$
$S_L = \left\{ \begin{array}{l} \mathsf{Adult} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension} \\ \mathsf{GRPStudent} \sqsubseteq \mathsf{PaysPension} \end{array} \right\}$
$S_R = \left\{ \begin{array}{l} \mathsf{Student} \sqsubseteq \mathsf{Adult} \\ \mathsf{GRPStudent} \sqsubseteq \mathsf{Student} \end{array} \right\}$
$S'_L = \left\{ \begin{array}{l} \mathsf{Adult} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension} \end{array} \right\}$
$S'_R = \left\{ \mathsf{Student} \sqsubseteq \mathsf{Adult} \right\}$
$\mathcal{M} = \left\{ \begin{array}{l} \mathsf{Adult} \sqsubseteq \mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension} \\ \mathsf{Student} \sqsubseteq \mathsf{Adult} \end{array} \right\}$

Figure 4.3: Example of finding one MIS

Algorithm 3: Compute all MIS for inconsistent ontologies

```
**Function 3**: ComputeAllMIS($\Sigma$)
1 $S \leftarrow \emptyset$
2 ComputeAllMIS($\Sigma, S$)
3 **return** $S$
**Function 3R**: ComputeAllMIS($\Sigma, S$)
1 **if** IsConsistent($\Sigma$) **then**
2     **return**
3 $M \leftarrow$ ComputeSingleMIS($\Sigma$)
4 $S \leftarrow S \cup \{M\}$
5 **for each** $ax \in M$ **do**
6     ComputeAllMIS($\Sigma \setminus \{ax\}, S$)
```


$\mathcal{M}_1 = \{A \sqsubseteq P, S \sqsubseteq \neg P, S \sqsubseteq A\}$
$A \sqsubseteq P$
$S \sqsubseteq \neg P$
$S \sqsubseteq A$
$\mathcal{M}_2 = \{S \sqsubseteq \neg P, G \sqsubseteq S, G \sqsubseteq P\}$
{}
$\mathcal{M}_2 = \{S \sqsubseteq \neg P, G \sqsubseteq S, G \sqsubseteq P\}$
$S \sqsubseteq \neg P$
$G \sqsubseteq S$
$G \sqsubseteq P$
{}
Figure 4.4: Example of finding all MIS

4.4.2 Assigning Defeasible Axioms and Priority Relations

After retrieving all MIS from inconsistent ontologies, we apply following heuristics to assign defeasible axioms and priority relations to axioms in each MIS, which is shown in Algorithm 4. Axioms that have conflict consequences in each MIS if exists are converted to defeasible axioms. These axioms are the sources of inconsistency thus converting them to defeasible axioms ensures that defeasible reasoning will not infer conflict consequences. However, skeptical reasoning disregards all conflict consequences without preferential information about axioms.

Algorithm 4: Assign defeasible axioms and priority relations for axioms in all MISs

```
Algorithm: AssignDefeasible($S$)
Input: All minimal inconsistent sets $S$
1 for each $s \in S$ do
2     for $\phi_1 = C_1 \sqsubseteq D$ and $\phi_2 = C_2 \sqsubseteq \neg D \in s$ do
3         Convert $\phi_1$ and $\phi_2$ to defeasible axioms
4         if ($s \models C_1 \sqsubseteq \bot$ and $s \not\models C_2 \sqsubseteq \bot$) then
5             Assign $\phi_1 > \phi_2$
6         else if ($s \models C_2 \sqsubseteq \bot$ and $s \not\models C_1 \sqsubseteq \bot$) then
7             Assign $\phi_2 > \phi_1$
```

We utilize *generalized specificity* [108] which favors those axioms which are more specific or more informed as a syntax-based preference criterion to assign priority relations. In MIS $\mathcal{M}_1$ in above example, $\mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension}$ is preferred to $\mathsf{Adult} \sqsubseteq \mathsf{PaysPension}$ because the former refer to more specific concept of Student. Similarly, $\mathsf{Student} \sqcap \mathsf{HasIncome} \sqsubseteq \mathsf{PaysPension}$ is preferred to $\mathsf{Student} \sqsubseteq \neg\mathsf{PaysPension}$ because antecedent of the former has more conditions. In general, given two axiom $\phi_1 = C_1 \sqsubseteq D$ and

$\phi_2 = C_2 \sqsubseteq \neg D$, ϕ_1 is more specific or more informed than ϕ_2 if $C_1 \sqsubseteq C_2$, hence $C_1 \sqsubseteq \bot$, and vice versa. This principle is also applied in lexicographic default reasoning and logic programming with arguments as shown in chapter of related works.

Chapter 5

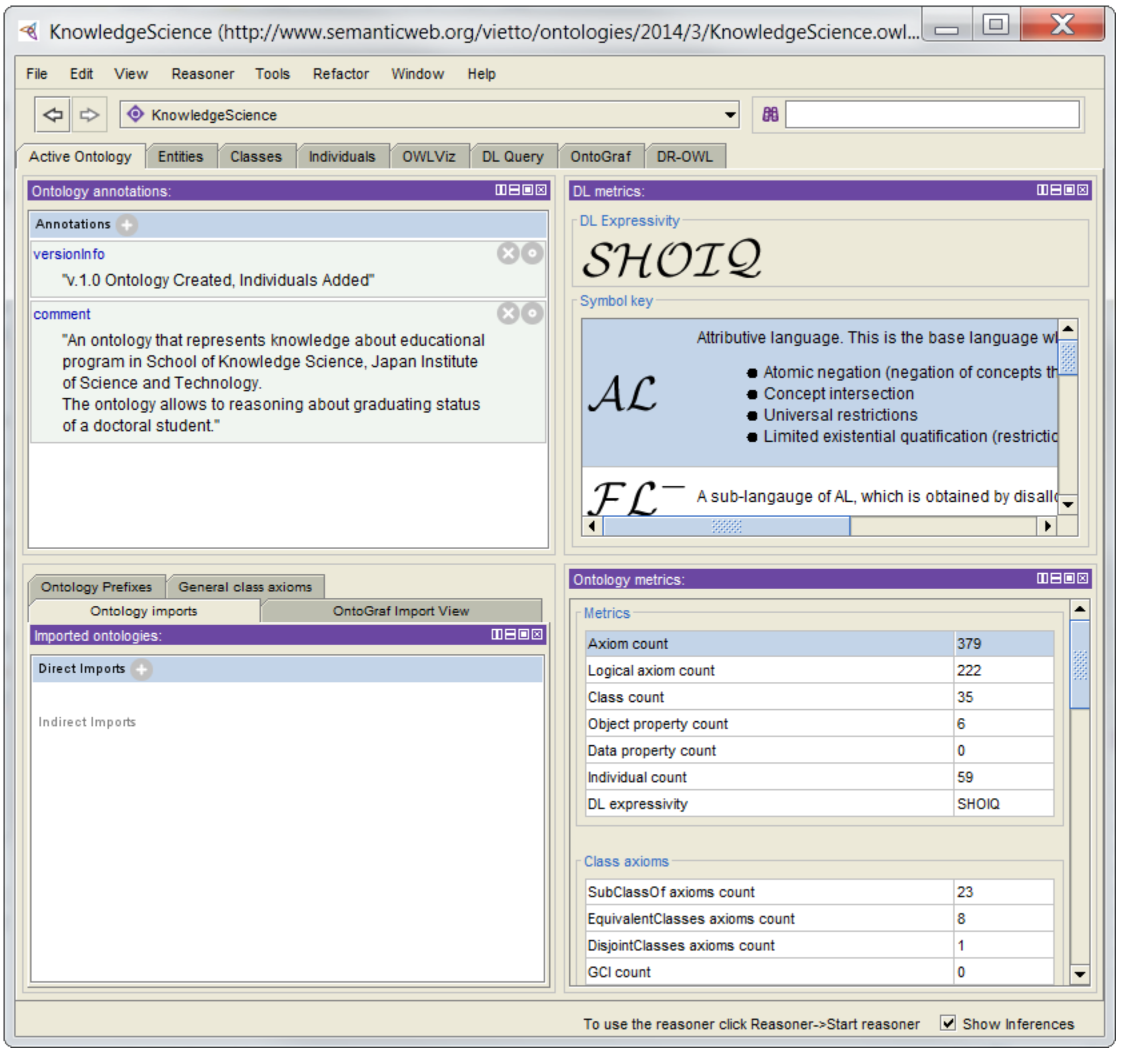
Implementation and Experiments

5.1 Protégé and Plugins

Protégé[1] is ontology development environment which is common because of its capability of handling ontologies of various formats. Protégé is developed by the Centre for Biomedical Informatics Research at Standford University. The latest versions of the editor (Protégé 4.x) is capable of handling ontologies of various formats but predominantly serves for OWL 2 ontologies. The support and alignment with the OWL 2 standard is provided by the latest version of the underlying API that Protégé uses. This is the Java-based OWL-API [51] which it depends on for executing management tasks such as creating, loading and manipulating OWL ontologies. It also provides support for integration of implemented reasoning engines (OWL) reasoners for analyzing and revealing implicit information in loaded ontologies during the modeling process. It has a plug-in friendly infrastructure, Protégé is indeed a bulk of plug-ins, which makes it ideal for extensibility. There are many Protégé plug-ins that have been developed by the Knowledge Representation community of varying types and functionality which aim to facilitate the ontology engineering process. We construct KS Ontology with Protégé 4.1 editor and develop a plug-ins for defeasible axiom representation and reasoning in this environment. In Protégé, concept is named as class and role as property.

Protégé has a tab-based interface, which opens by default to the Active Ontology tab. This tab provides an overview of the ontology, including metrics on its contents, annotations about the ontology as a whole, and other imported ontologies (if any imports exist). Figure 5.1 shows a screen shot of the overview of KS ontology. The metrics show the properties of the ontology, for example, the $\mathcal{SHOIQ}$ in DL metrics indicates that KS ontology consists of transitive and inverse roles, nominals and qualified number restriction descriptions.

[1]`http://protegewiki.stanford.edu/wiki/Main_Page`



KnowledgeScience (http://www.semanticweb.org/vietto/ontologies/2014/3/KnowledgeScience.owl...

Ontology annotations:

Annotations

versionInfo
"v.1.0 Ontology Created, Individuals Added"

comment
"An ontology that represents knowledge about educational program in School of Knowledge Science, Japan Institute of Science and Technology.
The ontology allows to reasoning about graduating status of a doctoral student."

DL metrics:

DL Expressivity

$\mathcal{SHOIQ}$

Symbol key

$\mathcal{AL}$ Attributive language. This is the base language wi
- Atomic negation (negation of concepts th
- Concept intersection
- Universal restrictions
- Limited existential quantification (restrictic

$\mathcal{FL}^{-}$ A sub-langauge of AL, which is obtained by disallo

Imported ontologies:

Direct Imports

Indirect Imports

Ontology metrics:

Metrics

Axiom count	379
Logical axiom count	222
Class count	35
Object property count	6
Data property count	0
Individual count	59
DL expressivity	SHOIQ

Class axioms

SubClassOf axioms count	23
EquivalentClasses axioms count	8
DisjointClasses axioms count	1
GCI count	0

To use the reasoner click Reasoner->Start reasoner ✓ Show Inferences

Figure 5.1: Ontology view of KS Ontology

The Entities tab is the workhorse of the ontology editor where we can explore all of the classes (or concepts), properties (or roles), and individuals in an ontology. Specific tabs such as Classes, Individuals, etc. are dedicated for corresponding particular elements in the ontology. Other tabs such as OWLViz Plugins, DL Query, etc. are external plug-ins which are packed in Protégé installation that provides extensions for ontology engineering. OWLViz Plugins enables class hierarchies in an OWL ontology to be viewed and incrementally navigated, allowing comparison of the asserted class hierarchy and the inferred class hierarchy. For example in Figure 5.2, the concept KSProfessor is highlighted as the sub-class of JaistProfessor in the inferred model of KS ontology even though it is not explicitly specified (for the purpose of verifying the inference ability of KS ontology). DL Query is another important tab that allows users to ask for detail inference results that does not shown by default reasoning service.

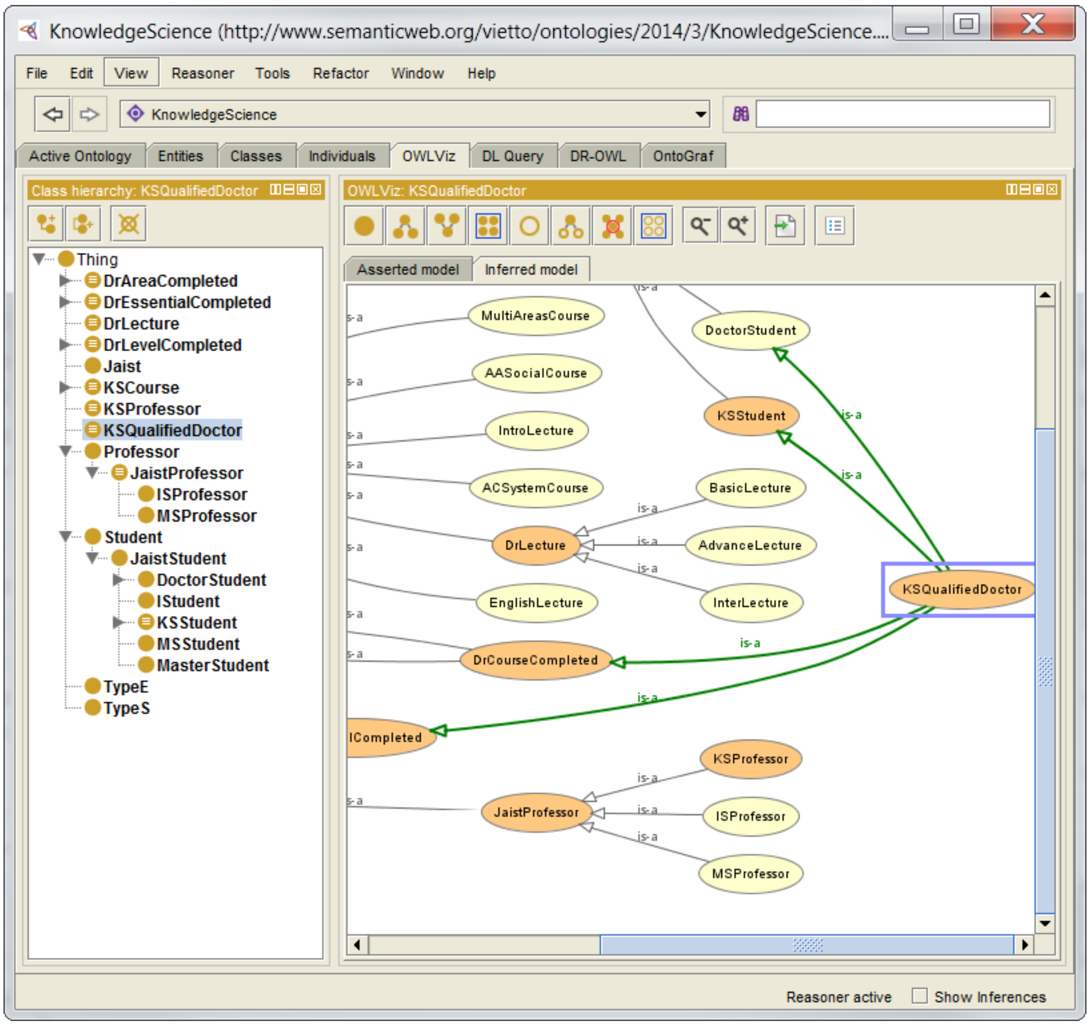


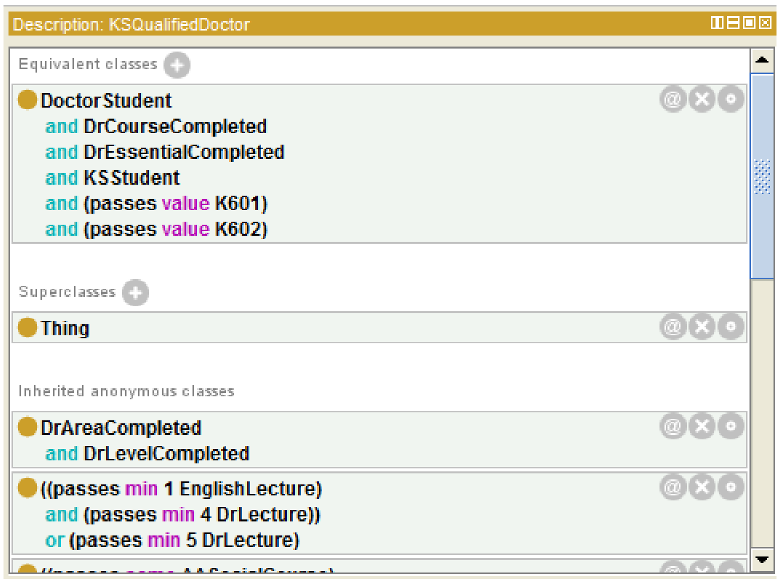
Figure 5.2: Inferred model of KS Ontology in OWLViz tab

5.2 Implementation of KS Ontology

5.2.1 Implementation of KS TBox

We implement KS TBox by adding the concept descriptions in Entities or Classes tabs. The concept description is stipulated in Description view such as in Figure 5.3. It is usual to define the equivalent classes, corresponding to equivalent axioms, and superclasses, corresponding to inclusion axioms, of a selected class. As shown in the figure, class descriptions are given in the Manchester OWL Syntax.

Manchester OWL Syntax [52] is created to produce a syntax that could be used by non-logical ontology modeler to edit class description in tools such as Protégé-OWL or Swoop. The class description syntax is shown in the Figure 5.4. The syntax replaces DL operators by the natural language keywords to make class expressions more natural to read. In addition, the syntax makes it easy to paste the plain text representation of the expression into e-mails etc. without incurring the formatting problems that can arise due


Description: KSQualifiedDoctor

Equivalent classes

DoctorStudent
 and DrCourseCompleted
 and DrEssentialCompleted
 and KSStudent
 and (passes value K601)
 and (passes value K602)

Superclasses

Thing

Inherited anonymous classes

DrAreaCompleted
 and DrLevelCompleted

((passes min 1 EnglishLecture)
 and (passes min 4 DrLecture))
 or (passes min 5 DrLecture)

Figure 5.3: Concept Descriptions for KS Ontology

to the different fonts required to represent the mathematical symbols that are used in the DL syntax.

OWL Constructor	DL Syntax	Manchester OWL	Example
intersectionOf	$C \sqcap D$	C **and** D	DoctorStudent **and** TypeS
unionOf	$C \sqcup D$	C **or** D	DoctorStudent **or** MasterStudent
complementOf	$\neg C$	**not** C	**not** DoctorStudent
oneOf	{a}⊔{b}...	{a b ...}	{K601 K602}
someValuesFrom	$\exists R.C$	R **some** C	passes **some** KSLecture
allValuesFrom	$\forall R.C$	R **only** C	supervises **only** KSStudent
minCardinality	$\geq nR.C$	R **min** n C	passes **min** 5 KSCourse
maxCardinality	$\leq nR.C$	R **max** n C	passes **max** 1 English
cardinality	$= nR.C$	R **exactly** n C	supervisedBy **exactly** 1 Professor
hasValue	$\exists R.\{a\}$	R **value** a	passes **value** {K601}

Figure 5.4: The Manchester OWL Syntax Class Constructors

The Manchester OWL Syntax encourage the minimization of the number of brackets that are used. This is achieved using operator precedence for class descriptions. The following list summarizes operator precedence – operators are shown from highest precedence to lowest precedence.

– **some, all, value, min, max, exactly**

– **not**

– **and**

– **or**

As would be expected, the syntax supports the nesting of class constructors to arbitrarily complex levels. Complex class expressions can be disambiguated by bracketing. For example, the class expression below describes the set of students who pass the level requirements for doctoral program in School of Knowledge Science:

```
(passes some AdvanceLecture) and
 (   ((passes  min  1 EnglishLecture) and (passes min 4 DrLecture))
  or (passes  min  5 DrLecture))
```

The expression has been formatted using indentation to ease readability.

Class expression that are built up using the syntax described previously can be used in tools for representing and editing items such as superclass/equivalent class expression etc. In addition to the class expression syntax, there is a full syntax for OWL entity descriptions. This means that it is possible to represent full descriptions for classes, properties and individuals in a textual manner.

5.2.2 Implementation of KS ABox

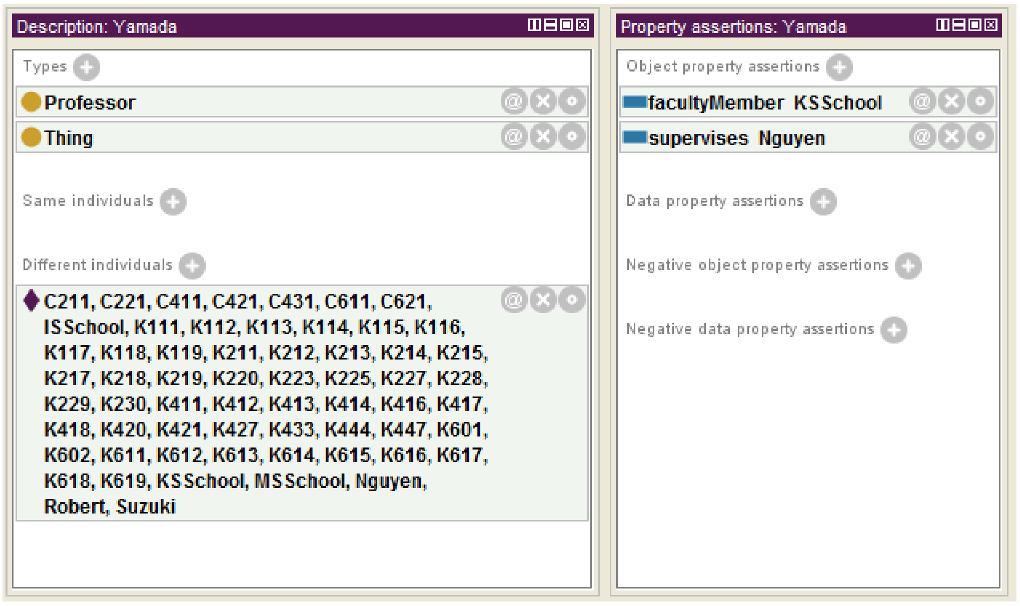
Individuals in KS Abox are created with the following three steps:

1. Create individual name and its additional information in annotation view.
2. Assign individual to some types, which are the ontology concepts. This corresponds to create concept assertion.
3. Assign property assertion(s) for the individual. Each assignment consists of a property and a value (an individual) and corresponds to a role assertion.

An example of the assertions for individual Yamada is shown in Figure 5.5. We can notice that the lower part of the left view shows the list of different individuals which covers all the individuals of the domain. The listing is necessary because the Unique Name Assumption is not applied in ontology reasoning. This implies that in an ontology, two individuals can refer to the same object in the real world. This implication makes ontology reasoning with number restrictions not work properly without clearly stating that all the individuals are different.

5.2.3 Performing Reasoning on KS Ontology

The use of KS Ontology to answer the question of interest user is carried out by calling reasoning service of Protégé. This service is provided by the built-in reasoner HermIT or another plug-in reasoner such as FaCT++. After the reasoner finishes classifying,


Description: Yamada

Types
- Professor
- Thing

Same individuals

Different individuals
- C211, C221, C411, C421, C431, C611, C621, ISSchool, K111, K112, K113, K114, K115, K116, K117, K118, K119, K211, K212, K213, K214, K215, K217, K218, K219, K220, K223, K225, K227, K228, K229, K230, K411, K412, K413, K414, K416, K417, K418, K420, K421, K427, K433, K444, K447, K601, K602, K611, K612, K613, K614, K615, K616, K617, K618, K619, KSSchool, MSSchool, Nguyen, Robert, Suzuki

Property assertions: Yamada

Object property assertions
- facultyMember KSSchool
- supervises Nguyen

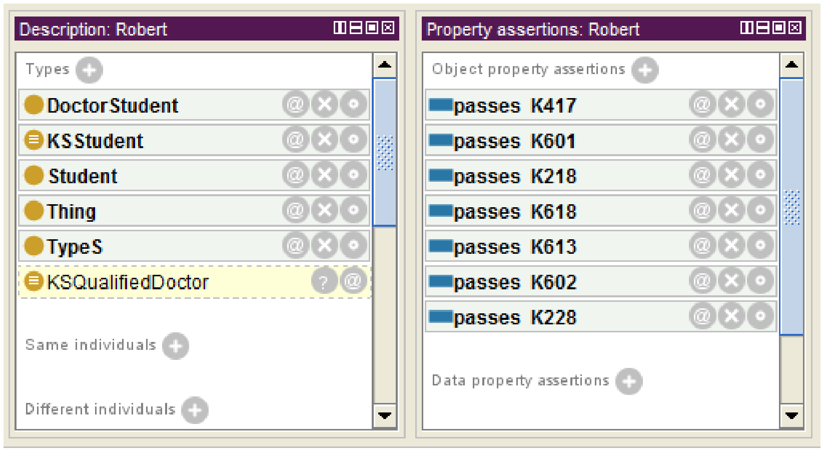
Data property assertions

Negative object property assertions

Negative data property assertions

Figure 5.5: Individual Descriptions for KS Ontology

an additional sub tab appears on the Entities tab to show the inferred class hierarchy and for every class (or individual), its inferred super-classes/equivalent classes (or types) are appeared as light color items, as shown in Figure 5.6. DL Query tab can be used to find additional inferred results such as inferred sub-classes or answer a specific query from the user. Unsatisfiable classes (if exist) appear in red under Nothing and everything else appears in the hierarchy under their inferred superclasses. In the ontology contains inconsistent individual, Protégé also shows a warming message to the user.


Description: Robert

Types
- DoctorStudent
- KSStudent
- Student
- Thing
- TypeS
- KSQualifiedDoctor

Same individuals

Different individuals

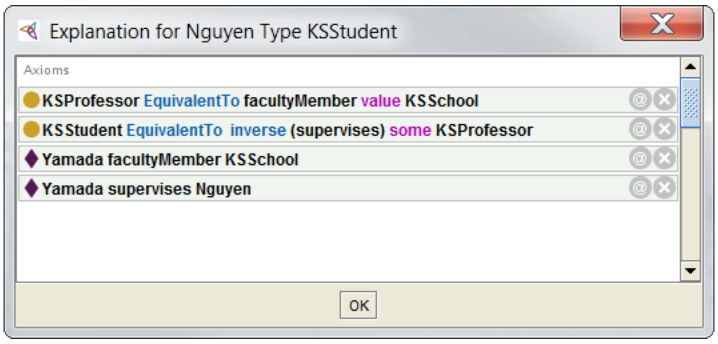
Property assertions: Robert

Object property assertions
- passes K417
- passes K601
- passes K218
- passes K618
- passes K613
- passes K602
- passes K228

Data property assertions

Figure 5.6: Inferred Type for Individual in KS Ontology

Modern ontology developing environment like Protégé also provides explanation service to help the user to understand the reasoning process to obtain the results. Explanation dialog appears when clicking the '?' mark beside the inferred result. Figure 5.7 shows an example how to infer that Nguyen is a student of School of Knowledge Science from the knowledge that he is supervised by Professor Yamada who is a faculty member of School of Knowledge Science.


Explanation for Nguyen Type KSStudent

Axioms

KSProfessor EquivalentTo facultyMember value KSSchool

KSStudent EquivalentTo inverse (supervises) some KSProfessor

Yamada facultyMember KSSchool

Yamada supervises Nguyen

Figure 5.7: An Explanation for Ontology Reasoning Result

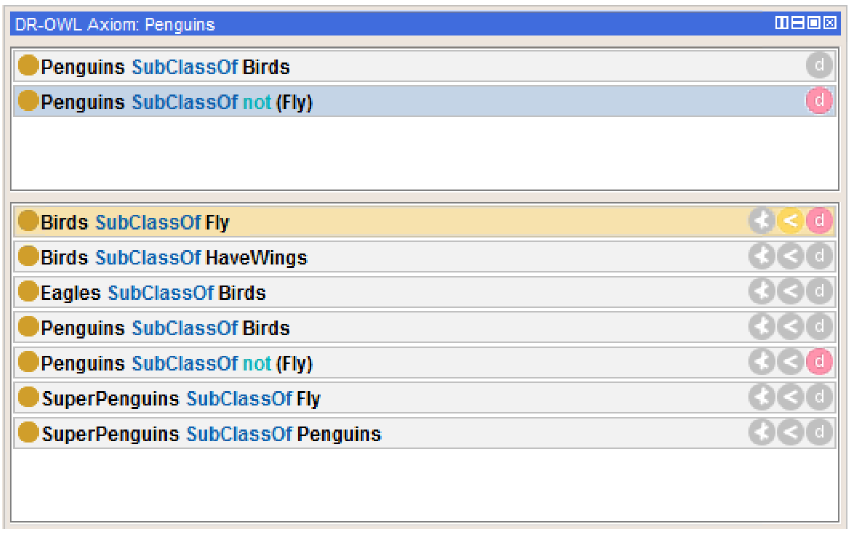
5.3 DR-OWL–Prototype of Defeasible Reasoning

We implement a prototype of defeasible reasoning approach as a Protégé plugins DR-OWL, which stands for Defeasible Reasoning OWL. DR-OWL plugins consists of two main views that is defeasible axiom view and defeasible query view.

Defeasible axiom view allows the ontology modeler to specify the defeasible and priority information of axioms. An example of defeasible axiom view for Bird ontology is depicted in Figure 5.8. This view includes two axiom lists. The upper list shows axioms which relate to a selected concept and the lower shows all axioms in the ontology. Modeler can toggle defeasibility of an axiom by clicking on the "d" button beside the axiom. To specify priority relation between axioms, we select an axiom in upper list and click "<" button of its inferior axiom in the lower list. For the Bird ontology in the Figure 5.8, there are two defeasible axioms that 'birds fly' and 'penguins do not fly'. And the axiom 'penguins do not fly' is stronger than 'birds fly'.

Defeasible and priority information is stored in the axiom annotations. Therefore, no explicit extension of current ontology specification is required and the logical meaning of the ontology is preserved. This information is used to perform defeasible reasoning but does not affect other reasoning process in Protégé. This technique of using axiom annotations for extending ontology representation has been utilized in several works such as fuzzy OWL [18] or preferential DL [84].

Query view provides two defeasible reasoning tasks: query checking and concept classification. Those reasoning tasks are performed by implementing linear reasoning strategy as discussed in Section 4.2. To perform reasoning tasks, user can select one of Protégé reasoners as the classical checker in the framework. In our experiments, we use HermiT [85, 37] as the classical reasoner. This reasoner implements a hyper-tableaux algorithm


DR-OWL Axiom: Penguins

Penguins SubClassOf Birds

Penguins SubClassOf not (Fly)

Birds SubClassOf Fly

Birds SubClassOf HaveWings

Eagles SubClassOf Birds

Penguins SubClassOf Birds

Penguins SubClassOf not (Fly)

SuperPenguins SubClassOf Fly

SuperPenguins SubClassOf Penguins

Figure 5.8: DR-OWL Axiom view

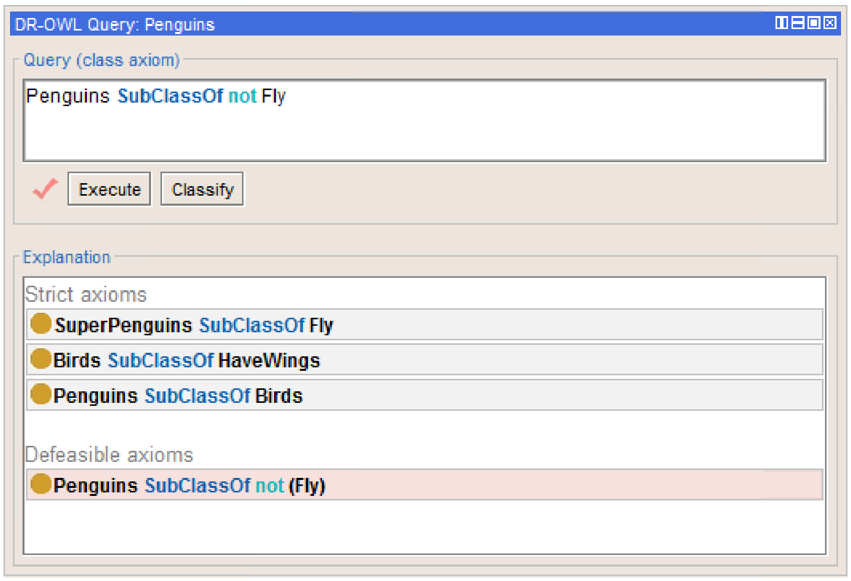
with several optimization heuristics for ontology classification. This reasoner is also compatible to most semantic features of OWL 2 ontologies and efficient for most ontology reasoning services.

For query checking task, an class axiom editor is used to input the query. The plugins then check whether the query is entailed from the ontology. The explanation panel shows all justified axioms with respect to the query. Those axioms are divided into strict and defeasible axiom groups. Figure 5.9 illustrates result of checking the query 'penguins do not fly' in Bird ontology. This query is defeasibly entailed by the ontology because there are both classical and defeasible axioms in the set of justified axioms with respect to the query. The other reasoning function is concept classification which is applied for class which is currently selected in Protégé class hierarchy. The explanation panel then shows all classical and defeasible superclasses of selected class.

5.4 Evaluation of DR-OWL

We evaluate DR-OWL plugins on some example ontologies including Bird and Student ontologies from the examples in previous sections and people+pets ontology [2], which is created from ISWC tutorial on ontology [14]. The last ontology contains a typical example of unsatisfiable concept named MadCow. In this ontology, MadCow is a Cow which eats brains of sheep, whereas a Cow is considered as a vegetarian which does not eat everything relating to animal. The interpretation of this example is as follows:

[2]`http://protege.cim3.net/file/pub/ontologies/people.pets/people+pets.owl`


DR-OWL Query: Penguins

Query (class axiom)

Penguins SubClassOf not Fly

Explanation

Strict axioms

● SuperPenguins SubClassOf Fly

● Birds SubClassOf HaveWings

● Penguins SubClassOf Birds

Defeasible axioms

● Penguins SubClassOf not (Fly)

Figure 5.9: DR-OWL Query View

$\mathsf{Cow} \sqsubseteq \mathsf{Vegetarian}$

$\mathsf{MadCow} \sqsubseteq \mathsf{Cow}$

$\mathsf{MadCow} \sqsubseteq \exists \mathsf{eats}.(\mathsf{Brain} \sqcap \exists \mathsf{part\_of}.\mathsf{Sheep})$

$\mathsf{Sheep} \sqsubseteq \mathsf{Animal}$

$\mathsf{Vegetarian} \sqsubseteq (\forall \mathsf{eats}.\neg \mathsf{Animal}) \sqcap (\forall \mathsf{eats}.(\neg \exists \mathsf{part\_of}.\mathsf{Animal}))$

In order to evaluate the reasoning framework for inconsistent ontologies, Huang et al. [56] proposes an intuitive test in which the answers provided by the system are compared to the intuitive answers of the users for a set of queries. Using a syntactical relevance function as selection function, Huang et al.'s system, named PION, can give matching answer for most of the queries except for one important query that 'is mad cow a vegetarian?' We expect that 'mad cow is not a vegetarian' but the system accepts 'mad cow is a vegetarian' instead. This is because the syntactical path of the latter answer (mad cod–cow–vegetarian) is shorter than of the former (mad cow–eats part of sheep–eats part of animal–not vegetarian). PION prefers the shorter syntactical path when resolving conflict.

Meanwhile, DR-OWL allows the modeler to flexibly resolve conflict by the means of defeasibility and priority relation. For people+pets ontology, we can express 'cows is typically a vegetarian' as a defeasible axiom. With this representation, the problem of unsatisfiable concept 'mad cow' is solved and the expected answer 'mad cow is not a vegetarian' is obtained. We apply a semantic approach to determine defeasible axiom in this example. For syntactic-based heuristic presented Section 4.4, the defeasible axioms and superiority relations are computed as follows:

Ontology	Queries	Defeasible axiom(s)	Priority relation(s)
Student	64	$\mathsf{Adult} \precsim \mathsf{PaysPension}$ $\mathsf{Student} \precsim \neg\mathsf{PaysPension}$ $\mathsf{GRPStudent} \precsim \neg\mathsf{PaysPension}$	 $> \mathsf{Adult} \precsim \mathsf{PaysPension}$ $> \mathsf{Student} \precsim \neg\mathsf{PaysPension}$
Bird	80	$\mathsf{Bird} \precsim \mathsf{Fly}$ $\mathsf{Penguin} \precsim \neg\mathsf{Fly}$	 $> \mathsf{Bird} \precsim \mathsf{Fly}$
people+pet	288	$\mathsf{Cow} \precsim \mathsf{Vegetarian}$	

Table 5.1: Settings of example ontologies to pass intuitive test

$CA : \mathsf{MadCow} \precsim \exists\mathsf{eats}.(\mathsf{Brain} \sqcap \exists\mathsf{part\_of}.\mathsf{Sheep})$

$VA : \mathsf{Vegetarian} \precsim \forall\mathsf{eats}.(\neg\exists\mathsf{part\_of}.\mathsf{Animal})$

$CA > VA$

Both approaches can address the conflict problem but for syntactic approach, semantically unexpected result such as 'mad cow is a vegetarian' is inferred. This example shows the differences among semantic approach, which reflects the actual meaning of the knowledge, and syntactic approach, which is only based on relationship among axioms. This result suggests that the heuristic to determine defeasible axiom and priority relations can be improved by taking into account the meanings of concepts and axioms.

Table 5.1 represents the settings of three example ontologies such that defeasible reasoning return correct answers in the intuitive test. For each ontology, a set of queries is generated and executed by the system. The total number of the queries for each ontology is shown in the table. An answer can be unsatisfied, classically satisfied and defeasibly satisfied. The answers returned by DR-OWL are compared to the intuitive answers, taking into account the logical meaning of the ontology. In our experiments, all the answers of the system are exact to the expected answers.

Regarding to the practical time complexity, performing defeasible reasoning takes under 1 second per query on a typical PC setting (with 2.4 GHz CPU, 4Gb RAM under Windows 7). This time complexity is acceptable for moderate ontology such as people+pets ontology which contains 60 concepts and about 400 axioms. The practical efficiency of reasoning framework comes from the combination between linear strategy and skeptical selection function. With skeptical selection function, no combination of axioms is needed to explore during the reasoning.

Chapter 6

Related Works

Since conflict knowledge appears in many real-world applications and situations, researches into representation and reasoning with conflict knowledge attract much attention in AI and Semantic Web communities. Those researches fall into two categories: (i) integrate non-monotonic reasoning with ontology reasoning and (ii) represent uncertainty knowledge in ontology. For the first category, the most common approach is to integrate defaults to DL ontology. Lexicographic entailment in default logic also influences other approaches in second categories of uncertainty representation in DL ontology. This section provides an overview of related works as the basis to compare our proposed approach.

6.1 Non-monotonic Approaches for DL Ontology

6.1.1 Default Logic

Default logic has been introduced by Reiter [99] as an approach to commonsense reasoning. It can be used to deal with the inability to fully describe the world and to provide more concise representations of knowledge due to the form of specifying exceptions to defaults. The notion of a default is introduced, which acts as an inference rule on the current set of beliefs.

A *default* is in the form $\frac{\alpha : \beta}{\gamma}$ where α, β and γ are well-formed formulae. α is the *prerequisite*, β is the *justification* and γ is the *consequent*. The default can be applied and the consequent inferred if the prerequisite can be proved and the justification is consistent with the current knowledge. We say that a default is closed if it contains no free variables.

A *normal default* is in the form $\frac{\alpha : \gamma}{\gamma}$ which is usually read as "Typically, if α then β" and abbreviated as $(\alpha : \gamma)$. Default reasoning mainly deals with normal defaults, thus we assumes hereafter the normal form of defaults without loss of generality.

A default theory is a pair $\Sigma = \langle W, \Gamma \rangle$, where W is a set of first-order logic formulae creating a world description and Γ is a set of defaults. The consequences of a default theory are defined using the notion of an extension, which is a set of deductively closed formulae derived from the application of defaults that does not cause any inconsistency. Due to the complex interaction among defaults, a default theory can have more than one extension or even no extensions. However, a default theory consisting only of normal defaults (normal default theory) guarantees having at least one extension. The following definition shows a non-deterministic iterative process of obtaining extensions of a default theory. In each step a default is used to add the consequent to the resulting set of formulae. An extension is defined by the fixed point of this process.

Let E be a set of closed formulae and $Th(E)$ we denote the deductive closure of a set of formulae E. $Th(E)$ is an *extension* of $\langle W, \Gamma \rangle$ iff

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i)$$

where

$$E_0 = W$$

$$E_{i+1} = E_i \cup \{\gamma | (\alpha : \gamma) \in \Gamma, \alpha \in Th(E_i) \text{ and } \neg\gamma \notin Th(E)\}$$

The consequences of a default theory can be defined by employing skeptical or credulous reasoning. Given the default theory Σ, a formula φ is a consequence of *skeptical reasoning* in Σ if it is in at least one extension, i.e. $\exists E \in ext(\Sigma), \varphi \in E$. A formula φ is a consequence of *credulous reasoning* in Σ if it is in all extensions, i.e. $\forall E \in ext(\Sigma), \varphi \in E$. Both approaches for default reasoning are NP-complete because the extensions are finite sets of formulas and the definition of extension is non-constructive.

6.1.2 Defaults with DL Ontology

Baader and Hollunder [10] show how defaults can be embedded into description logics. Based on Baader and Hollunder's formalism, Więch [115] propose a framework to perform default reasoning in a distributed environment where agents exchange not only reasoning results but also default justification to achieve the result. Default justification is help to resolve possible conflict among reasoning results of agents. Defaults for description logics are formed analogously to the original defaults syntax. Here, the formulas for the prerequisite, justification and conclusion are replaced by DL concepts. A default is in the form $C : D$ which expresses that it can be inferred that x is an instance of the concept D if x is an instance of C and it is consistent to assume that x is an instance of D.

A *default DL* ontology $\Sigma = \langle \mathcal{T}, \mathcal{D} \rangle$ consists of $\mathcal{T}$ a set of classical axioms and $\mathcal{D}$ a set of defaults. In order to answer a query $\varphi = C \sqsubseteq D$? from a default DL ontology, an default reasoning algorithm such as proposed by Risch and Schwind [101] is used to compute all extensions of the theory $\langle \mathcal{T} \cup \{C\}, \mathcal{D}\rangle$. The answer to the query is computed by investigating the extensions to look for D with one of two default reasoning rules, skeptical or credulous reasoning.

Example 6.1. Consider a default theory $\Sigma = \langle \mathcal{T}, \mathcal{D}\rangle$, where

$$\mathcal{T} = \{\{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Adult}, \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{Student}\}$$

$$\mathcal{D} = \{\mathsf{Adult} : \mathsf{PaysPension}, \mathsf{Student} : \neg\mathsf{PaysPension}\}$$

For the query $\varphi = \{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{PaysPension}$, there are two extensions, which corresponds to the application of one of two defaults respectively, as follows:

$$E_1 = \{\{\mathsf{SUZUKI}\}, \mathsf{Adult}, \mathsf{Student}, \mathsf{PaysPension}\}$$

$$E_2 = \{\{\mathsf{SUZUKI}\}, \mathsf{Adult}, \mathsf{Student}, \neg\mathsf{PaysPension}\}$$

Since $\mathsf{PaysPension}$ is in E_1 but not in E_2, φ is satisfied w.r.t. credulous reasoning but not satisfied w.r.t. skeptical reasoning.

Więch [115] points out two limitations of default reasoning on DL ontology. The first one is embedding defaults in description logics is not as straightforward defaults can only be applied to named individuals which already exist in the knowledge base. This problem is due to the incompatibility between default reasoning and description logic. The second problem of is inconsistent answers can be returned by default reasoning. For example, both query $\{\mathsf{SUZUKI}\} \sqsubseteq \mathsf{PaysPension}$ and $\{\mathsf{SUZUKI}\} \sqsubseteq \neg\mathsf{PaysPension}$ are satisfied by credulous reasoning from the default ontology in Example 6.1. This is due to the nature of default reasoning and can be addressed by posing a priority relationship among defaults. Lehmann proposes preferential entailment [70] and lexicographic entailment [69] which allow the use of logical specificity or exceptionality of defaults to handle the inconsistency in default reasoning.

6.1.3 Preferential Description Logics

Preferential DLs [20] handle incompatibility issue by adding defeasible axiom $C \sqsubseteq\!\!\!\sim D$, which corresponds to a normal default $(C : D)$, to DL ontology and lifting the semantics of preferential logic from propositional case [70] to description logic case and thus inherits the elegant and well-formed semantics of the original one. Application of this approach is also implemented as a plugins of common ontology editor Protégé [84].

Preferential reasoning utilizes *ranking* of the Σ which is computed as a collection of sets of sentences $\mathcal{D}$. Each sentence in a particular set from $\mathcal{D}$ shares the same level of exceptionality. Given the collection of sets $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}, \mathcal{D}_1$ represents the *lowest* rank containing the *least* exceptional (specific) sentences in the Σ. Then they are the ones which can be "disregarded" with the most confidence when contradicting information is found. Meanwhile, the highest ranking subset $\mathcal{D}_\infty$ contains all non-defeasible or classical axioms that will not be disregarded and should always remain in the Σ.

Example 6.2. Consider the following preferential DL ontology:

$$\Sigma = \left\{ \begin{array}{l} \mathsf{Bird} \sqsubsetsim \mathsf{Fly} \\ \mathsf{Bird} \sqsubsetsim \mathsf{HasWings} \\ \mathsf{Penguin} \sqsubseteq \mathsf{Bird} \\ \mathsf{Penguin} \sqsubsetsim \neg\mathsf{Fly} \end{array} \right\}$$

The ranking of Σ is as follows:

$$\mathcal{D}_\infty = \{\mathsf{Penguin} \sqsubseteq \mathsf{Bird}\}$$

$$\mathcal{D}_2 = \{\mathsf{Penguin} \sqsubsetsim \neg\mathsf{Fly}\}$$

$$\mathcal{D}_1 = \{\mathsf{Bird} \sqsubsetsim \mathsf{Fly}, \mathsf{Bird} \sqsubsetsim \mathsf{HasWings}\}$$

The ranking is computed based on *exceptionality* of the axioms. A classical axiom is always exceptional w.r.t. Σ while a defeasible axiom $C \sqsubsetsim D$ is exceptional w.r.t. Σ if its antecedence is not satisfied by Σ, i.e. $\Sigma \models C \sqsubseteq \bot$. These ranks is computed iteratively, starting from the lowest rank [84]: axioms that are *not* exceptional w.r.t. the current Σ are removed from the Σ and added to the current rank. The remaining KB is used to compute next higher ranks. The iteration continues until current KB is empty or all axioms are *exceptional* w.r.t to current KB.

For a given query, preferential DL reasoning procedure finds the maximal subset of the ranking that satisfies the antecedent and returning the reasoning result according to this subset. There are two strategies two find the maximal subset corresponding to two type of reasoning in preferential DL: prototypical and presumptive reasoning. *Prototypical reasoning* uses the original ranking. That means it will remove the *entire* rank when antecedent is not satisfiable. In Example 6.2, for the query $\mathsf{Penguin} \sqsubsetsim \mathsf{Fly}$, the maximal subset of $\mathcal{D}$ is $\mathcal{D}_\infty \cup \mathcal{D}_2$ because $\mathcal{D}_1$ is remove to avoid inconsistency. Therefore, the query is not satisfiable and does not conflict with knowledge that Penguin does not fly. However, $\mathsf{Penguin} \sqsubsetsim \mathsf{HasWings}$ is also not satisfied because of the removal.

Presumptive reasoning provides more lenient answer than prototypical reasoning by trying to remove *one by one* axiom before removing the whole rank. It performs this computation by adding some ranks above every rank $\mathcal{D}_i (i \neq \infty)$ in the prototypical ranking. Every additional rank corresponds to the union of subsets of $\mathcal{D}_i$ after removing one axiom. Example 6.3 shows the converted ranking for presumptive reasoning. The additional rank $\mathcal{D}'_{12}$ is added and it is the union of the subset of $\mathcal{D}_1$ after remove one axiom. With this converted ranking, $\mathsf{Penguin} \sqsubsetapprox \mathsf{HasWings}$ is satisfiable w.r.t presumptive reasoning.

Example 6.3. The covnerted ranking $\mathcal{D}'$ for presumptive reasoning is as follows:

$$\mathcal{D}'_{\infty} = \{\mathsf{Penguin} \sqsubseteq \mathsf{Bird}\}$$

$$\mathcal{D}'_{2} = \{\mathsf{Penguin} \sqsubsetapprox \neg\mathsf{Fly}\}$$
$$\mathcal{D}'_{12} = \{\mathsf{Bird} \sqsubsetapprox \mathsf{Fly} \sqcup \mathsf{HasWings}\}$$
$$\mathcal{D}'_{11} = \{\mathsf{Bird} \sqsubsetapprox \mathsf{Fly}, \mathsf{Bird} \sqsubsetapprox \mathsf{HasWings}\}$$

The problem of presumptive reasoning is the cost of generating all sub-rankings of a ranking. The computational complexity of presumptive reasoning is NEXP which easily becomes untraceable for a moderate preferential DL ontology.

Lexicographic entailment [69] for sets of defaults is the formalization of semantics of presumptive reasoning. It partition defaults into lexicographic structure *z-partition* based on the specificity to define preferred models among interpretations. A model is *lexicographically preferable* (or *lex-preferable*) to the other if it satisfies more defaults in the lexicographical structure and it is a *lexicographically minimal* (or *lex-minimal* model if and only if no model is lex-preferable to it. Given a query, the entailment is determined by the lex-minimal model that satisfies or falsifies it. Similar to presumptive reasoning, lexicographic entailment allows inheritance to exceptional subclasses, for example penguins can inherit properties of birds even though they do not fly, but it also suffers the high computational expense. Lexicographic entailment is applied in several formalisms as shown in Section 6.2.

6.1.4 Other non-monotonic approaches for DL Ontology

Beside above approaches and syntactical defeasible DL ontology approach that has been introduced in Section 3.6, there are some notable works that apply non-monotonic reasoning to DL ontology. Heymans and Vermeir (2002) extend the DL $\mathcal{SHOQ}(\mathbf{D})$ with a preference order on the axioms. With this strict partial order, certain axioms can be overruled if defeated with more preferred ones. They also impose a preferred model semantics, introducing non-monotonicity into $\mathcal{SHOQ}$ but the detail discussion about inference for this approach is not presented.

Gomez et. al. [38] present δ-ontologies, a framework for reasoning with inconsistent DL ontologies which expresses DL ontologies as defeasible logic programs (DeLP). Given a query posed w.r.t. an inconsistent ontology, a dialectical analysis will be performed on a DeLP program obtained from such an ontology, where all arguments in favor and against the final answer of the query will be taken into account. However, some expressiveness features of DL ontology, for example disjunction constructor and existence restriction in consequence of an axiom, is missing during the transformation because the non-overlapping between expressiveness of description logic and logic programming.

6.2 Representation of Uncertainty in DL Ontology and OWL

6.2.1 Probabilistic Description Logics

Probabilistic DL is based on probabilistic logic, a well-founded logic which has its origin back to philosophy. There is a wide spectrum of formal languages that have been explored in probabilistic logic, ranging from constraints for unconditional and conditional events to rich languages that specify linear inequalities over events. There are several proposals for probabilistic description logics such as in [49, 58, 30, 59, 74] among which Lukasiewicz's expressive probabilistic description logic [74] is the only approach so far for Semantic Web and it also supports both terminological and assertional probabilistic knowledge.

Probabilistic DL utilizes conditional constraints to express uncertain statements. A *conditional constraint* is an expression of the form $(D|C)[l, u]$, where C and D are DL concepts, and l and u are reals form [0,1]. Informally, $(D|C)[l, u]$ encodes that the probability of ψ given ϕ lies between l and u. It naturally interprets terminological and assertional probabilistic knowledge as statistical knowledge about concepts and roles and as degrees of belief about instances of concepts and roles, respectively, and allows for deriving both statistical knowledge and degrees of belief.

The semantics of probabilistic DL is based on the notion of lexicographic entailment in probabilistic default reasoning, which is a probabilistic generalization of the sophisticated notion of lexicographic entailment in default reasoning from conditional knowledge bases. This semantics implies that we can express default knowledge about concepts (as a special case of terminological probabilistic knowledge), which is semantically interpreted as in Lehmann's lexicographic default entailment. Therefore, probabilistic reasoning for DL ontology consists of two steps. Given a probabilistic DL ontology and a query written as $\varphi = (D|C)$, probabilistic reasoning first verifies the satisfiability of the default $C : D$ w.r.t. lexicographic entailment and then compute probability interval $[l, u]$ by a linear optimization technique [74]. The computational complexity of the each step is NEXP and $\mathrm{FP}^{\mathrm{NEXP}}$ [75], respectively, which cause much difficulty on implementing practical probabilistic DL systems.

6.2.2 Possibilistic Description Logics

Possibilistic DL [97] is based on possibilistic logic [28], which is a weighted logic where each classical logic formula is associated with a number in (0,1]. Possibilistic DL provides a simpler and more flexible treatment to compute uncertainty in DL ontology than probabilistic DL. Implementation and evaluation of various reasoning schemes in possibilistic DL using KAON2 is reported in [97, 96, 95, 94].

The syntax of possibilistic DL is based on the syntax of classical DL. A *possibilistic axiom* is a pair (ϕ, α) consisting of an axiom ϕ and a weight $\alpha \in (0, 1]$. A possiblistic TBox (resp., *ABox*) is a finite set of possibilistic axioms (ϕ, α), where ϕ is an TBox (resp., ABox) axiom. A possibilistic DL knowledge base $\Sigma = (\mathcal{T}, \mathcal{A})$ consists of a possiblistic TBox $\mathcal{T}$ and a possibilistic ABox $\mathcal{A}$. We use $\mathcal{T}^* = \{\phi_i : (\phi_i, \alpha_i) \in \mathcal{T}\}$ ($\mathcal{A}^*$ can be defined similarly). The classical base Σ^* of a possibilistic DL knowledge base is $\Sigma^* = (\mathcal{T}^*, \mathcal{A}^*)$.

The main *possibilistic inference* tasks are deciding whether a possibilistic DL ontology is satisfiable, deciding whether a possibilistic axiom is a logical consequence of a possibilistic DL ontology, and computing possibility of a classical description logic axiom. All those tasks can be reduced to the task of computing inconsistent degree of a possibilistic ontology [95] and based on the notion of α-cut of the ontology. Given a possibilistic DL ontology $\Sigma = (\mathcal{T}, \mathcal{A})$ and $\alpha \in (0, 1]$, the α-cut of $\mathcal{T}$ is $\mathcal{T}_{\geq\alpha} = \{\phi \in \Sigma^* \mid (\phi, \beta) \in \mathcal{T} \text{ and } \beta \geq \alpha\}$ (the α-cut of $\mathcal{A}$, denoted as $\mathcal{A}_{\geq\alpha}$, can be defined similarly). The strict α-cut of $\mathcal{T}$ (resp., $\mathcal{A}$) can be defined similarly as the strict cut in possibilistic logic. The α-cut (resp., strict α-cut) of Σ is $\Sigma_{\geq\alpha} = (\mathcal{T}_{\geq\alpha}, \mathcal{A}_{\geq\alpha})$ (resp., $\Sigma_{>\alpha} = (\mathcal{T}_{>\alpha}, \mathcal{A}_{>\alpha})$). The *inconsistent degree* of Σ, denoted $Inc(\Sigma)$, is defined as $Inc(\Sigma) = max\{\alpha_i : \Sigma_{\geq\alpha_i} \text{ is inconsistent}\}$. A axiom φ is satisfied by Σ to a degree α, written $\Sigma \models_\pi (\varphi, \alpha)$ if the following conditions hold: (1) $\Sigma_{\geq\alpha}$ is consistent, (2) $\Sigma_{\geq\alpha} \models \varphi$, (3) forall $\beta > \alpha, \Sigma \not\models \varphi$.

Example 6.4. Suppose we have a possibilistic DL knowledge base[1]

$$\Sigma = \left\{ \begin{array}{l} (\mathsf{Penguin} \sqsubseteq \mathsf{Bird}, 1), \\ (\mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly}, 0.95), \\ (\mathsf{Bird} \sqsubseteq \mathsf{HasWings}, 0.9), \\ (\mathsf{Bird} \sqsubseteq \mathsf{Fly}, 0.8), \\ (\mathsf{EatsFish} \sqsubseteq \mathsf{Swims}, 0.8), \\ (\mathsf{Penguin} \sqsubseteq \mathsf{EatsFish}, 0.7) \end{array} \right\}$$

For $\alpha = 0.8, \Sigma_{\geq 0.8} = \{(\mathsf{Penguin} \sqsubseteq \mathsf{Bird}, 1), (\mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly}, 0.95), (\mathsf{Bird} \sqsubseteq \mathsf{HasWings}, 0.9), (\mathsf{Bird} \sqsubseteq \mathsf{Fly}, 0.8), (\mathsf{EatsFish} \sqsubseteq \mathsf{Swims}, 0.8)\}$ is inconsistent.
For $\alpha = 0.9, \Sigma_{\geq 0.9} = \{(\mathsf{Penguin} \sqsubseteq \mathsf{Bird}, 1), (\mathsf{Penguin} \sqsubseteq \neg\mathsf{Fly}, 0.95), (\mathsf{Bird} \sqsubseteq \mathsf{HasWings}, 0.9)\}$ is consistent.

[1]We assume an empty ABox here.

Therefore, $Inc(\Sigma) = 0.8$. And since $\Sigma_{\geq 0.9} \models \mathsf{Penguin} \sqsubseteq \mathsf{HasWings}$ and $\Sigma_{\geq 0.95} \not\models \mathsf{Penguin} \sqsubseteq \mathsf{HasWings}$, we have $\Sigma_\pi \models (\mathsf{Penguin} \sqsubseteq \mathsf{HasWings}, 0.9)$

As noted earlier, possibilistic inference is performed by searching α such that $Inc(\Sigma) = \alpha$. The computational complexity of possibilistic inferebce linear w.r.t. ontology reasoning. However, possibilistic inference in possibilistic DL inherits the drowning effect of possibilistic inference in possibilistic logic which drops all axioms that have possibility lower than inconsistent degree.

Linear order inference can be adapted to deal with the drowning problem. It only drops the axioms in the most recent stratum, set of axioms that have same degree, that causes the inconsistency and keep the others. The weighted consequence of Σ in linear order inference is based on the possibilistic inference. For example, in Example 6.4, we only need to drop axioms that have degree of 0.8 to make the ontology consistent. Therefore, we have $\Sigma_{LO} \models (\mathsf{Penguin} \sqsubseteq \mathsf{EatsFish}, 0.7)$ but $\Sigma_\pi \not\models (\mathsf{Penguin} \sqsubseteq \mathsf{EatsFish}, 0.7)$

Linear order inference has stronger inferential power than possibilistic inference. However, it drops all axioms in one stratum even if only a few of them are involved in the inconsistency. *Lexicographic inference* is proposed to provide a more powerful inference mechanism [94]. The idea of lexicographic inference for possibilistic DL is the same to the one for default reasoning except for possibility is used to construct lexicographic order instead of specificity. With lexicographic inference, we can deduce detail consequence such as $\mathsf{Penguin} \sqsubseteq \mathsf{Swims}$ from the ontology in Example 6.4.

6.2.3 Other uncertainty representation approaches

The literature contains several probabilistic generalizations of web ontology languages. Many of these approaches focus especially on combining the web ontology language OWL with probabilistic formalisms based on Bayesian networks [75].

In particular, da Costa and Laskey [22] suggest a probabilistic generalization of OWL, called PR-OWL, whose probabilistic semantics is based on multi-entity Bayesian networks (MEBNs). Roughly speaking, PR-OWL represents knowledge as parameterized fragments of Bayesian networks. Hence, it can encode probability distributions on the interpretations of an associated first-order theory as well as repeated structure.

In [27], Ding et al. propose a probabilistic generalization of OWL, called BayesOWL, which is based on standard Bayesian networks. BayesOWL provides a set of rules and procedures for the direct translation of an OWL ontology into a Bayesian network, and it also provides a method for incorporating available probability constraints when constructing the Bayesian network. The generated Bayesian network, which preserves the semantics of the original ontology and which is consistent with all the given probability constraints, supports ontology reasoning, both within and across ontologies, as Bayesian inferences. In [90, 27], Ding et al. also describe an application of the BayesOWL approach in ontology mapping.

In closely related work, Mitra et al. [82] describe an implemented technique, called OMEN, to enhancing existing ontology mappings by using a Bayesian network to represent the influences between potential concept mappings across ontologies. More concretely, OMEN is based on a simple ontology model similar to RDF Schema. It uses a set of meta-rules that capture the inuence of the ontology structure and the semantics of ontology relations, and matches nodes that are neighbors of already matched nodes in the two ontologies.

Yang and Calmet [116] present an integration of the web ontology language OWL with Bayesian networks, called OntoBayes. The approach makes use of probability and dependency-annotated OWL to represent uncertain information in Bayesian networks.

Similar above approaches, Essaid and Yaghlane [31] proposes BeliefOWL which utilize a set of structural translation rules in OWL language to build evidential network. The evidential network is the realization of Dempster-Shafer theory [106] which is a more generalization of Bayesian theory of probability.

Chapter 7

Conclusion

In order to conclude this dissertation, we introduce Figure 7.1 to summarize the main features of our proposed defeasible DL and other related works. Compared to the others, our proposed approach has sufficient syntactical power, which can represent defeasible knowledge as well as preferential information about knowledge, model-based semantics, which is compatible to DL and extensible to most ontology languages, and efficient reasoning algorithms as the advantages. Defeasible DL does not overcome other approaches in all features but the combination of these advantages can provide a rational solution to the conflict knowledge issues.

Among related works, rule-based defeasible DL has polynomial reasoning complexity with respect to ontology size and it does not suffer theoretically exponential complexity of ontology reasoning since it does not use ontology reasoning but transforms ontology to logic program for reasoning. However, this transformation prevents rule-based defeasible DL from inheriting expressive power of ontology language. Transforming expressive features of ontology to logic program indeed results in a much higher computational expense.

Probabilistic DL and possibilistic DL provide more power extension than proposed defeasible DL because conditional constraints and possibilistic axioms define a global ordering among axioms. However, those methods tolerate a highly computational cost which originates from the lexicographic inference of default reasoning. Moreover, probabilistic DL also suffers a higher complexity issue of solving linear optimization problem to compute conditional intervals.

The comparison results show that defeasible DL is promising to handle conflict knowledge in practical applications of knowledge sharing and co-creation. However, there are several limitations in the proposed approach that can create new research directions.

The most important limitation derives from the property of defeasible logic that there is *no* universal preference among knowledge in defeasible DL like the global preference in probabilistic and possibilistic DL. In defeasible DL, defeasible axioms are globally weaker than classical axioms while priority relations are locally is defined among conflict axioms.

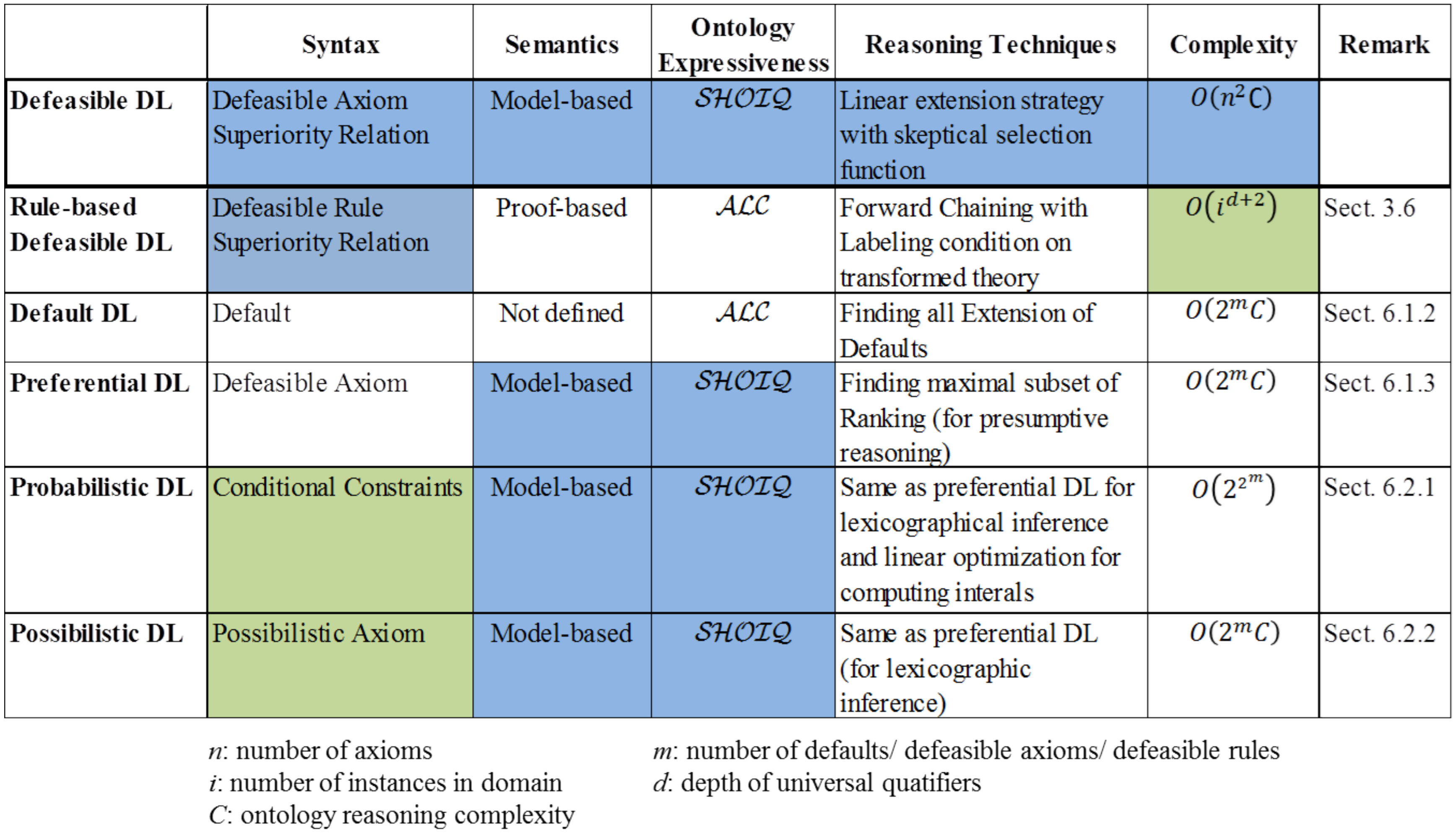
This limitation causes difficulty in computing defeasible axioms and priority relations as shown in evaluation section for the case of mad cow. The only heuristic is to use syntactical approach to determine defeasible axioms, global preference among axioms, and superiority relations, local preference among defeasible axioms. The syntax-based approach is clearly not suitable in a semantic-rich environment like knowledge sharing and co-creation application. Other approaches such as semantic or statistic measures can be used to compute global preferences among all axioms but the next problem is how to transform this preferential information to defeasible axioms and superiority relations. The answer for this question may not be realistic because of the property of defeasible logic as shown above. One alternative solution is to combine defeasible reasoning with methods that represent uncertainty in DL ontology like probabilistic logic or possibilistic logic to express the global preferential. As presented in related works, those approach have been integrated with lexicographic inference of default reasoning. Hence, the combination between those uncertainty representation methods with defeasible reasoning is a promising research direction to handle conflict knowledge in DL ontology.

The second limitation relates to reasoning technique of proposed defeasible DL which utilizes ontology reasoner as the basis for defeasible reasoning. Therefore, every iteration in defeasible reasoning increase computational complexity by an untraceable factors, i.e. the complexity of ontology reasoning. The other approach is to integrate principle of defeasible reasoning to ontology reasoning algorithms, hopefully that the integration does not increase computational complexity of defeasible reasoning for DL ontology like in the propositional case. Several researches have been introduced to modify tableau algorithm to combine possibilistic reasoning [96] or paraconsistent logic reasoning [118]. Thus, integrating defeasible reasoning and tableau algorithm and comparing between different approaches for defeasible ontology reasoning is a difficult but interesting problem for other research.

In summary, this dissertation demonstrates the expressiveness of DL ontology in knowledge sharing and co-creation problem, by introducing the KS ontology to help students to understand and utilize school's knowledge to create their own knowledge, and extends the ability of DL ontology to handle conflict knowledge that can be encountered in practical problems, by introducing defeasible DL and defeasible reasoning approach for ontology. With this encouraging results, we plan to develop methodology for knowledge sharing and co-creation process in real-world application. In detail, the future works for this research are as follow:

Evaluate defeasible DL on large ontologies and ontology integration: Even most of current common large ontologies are light-weight ontologies, they still contains potential inconsistencies, as indicated by [103]. For example, [73] has detected 75 inconsistencies in Gene Ontology. We are currently studying on some large ontologies to understand the inconsistencies in practical problem and discover how we can solve those inconsistencies by defeasible reasoning. This trend of research can show the applicability of proposed approach on real-world problems. Sharing and integrating knowledge from multiple sources is one of the main reasons to caused inconsistency because different sources usually contains conflicts. We plan to implement plugins feature to perform defeasible reasoning on ontology which imports knowledge from different sources and determine the preference among axioms from different ontologies.

Extend the proposal to handle contextual knowledge: Conflict knowledge is a special case of contextual knowledge. In a knowledge co-creation process, we assume that each participant in the process has a local context of viewpoint so the knowledge exchanged by a participant can be understood and utilized differently by the others. For example, students of different entrance years have different graduation requirements so they use the knowledge about educational program of School of Knowledge Science distinctly. We believe that sharing and reasoning with contextual knowledge is an interesting research question to understand how knowledge is created in a dynamic environment like our contemporary world.



	Syntax	Semantics	Ontology Expressiveness	Reasoning Techniques	Complexity	Remark
Defeasible DL	Defeasible Axiom Superiority Relation	Model-based	$\mathcal{SHOIQ}$	Linear extension strategy with skeptical selection function	$O(n^2C)$	
Rule-based Defeasible DL	Defeasible Rule Superiority Relation	Proof-based	$\mathcal{ALC}$	Forward Chaining with Labeling condition on transformed theory	$O(i^{d+2})$	Sect. 3.6
Default DL	Default	Not defined	$\mathcal{ALC}$	Finding all Extension of Defaults	$O(2^mC)$	Sect. 6.1.2
Preferential DL	Defeasible Axiom	Model-based	$\mathcal{SHOIQ}$	Finding maximal subset of Ranking (for presumptive reasoning)	$O(2^mC)$	Sect. 6.1.3
Probabilistic DL	Conditional Constraints	Model-based	$\mathcal{SHOIQ}$	Same as preferential DL for lexicographical inference and linear optimization for computing interals	$O(2^{2^m})$	Sect. 6.2.1
Possibilistic DL	Possibilistic Axiom	Model-based	$\mathcal{SHOIQ}$	Same as preferential DL (for lexicographic inference)	$O(2^mC)$	Sect. 6.2.2

n: number of axioms	m: number of defaults/ defeasible axioms/ defeasible rules
i: number of instances in domain	d: depth of universal quatifiers
C: ontology reasoning complexity	

Figure 7.1: Comparison between proposed defeasible DL and related works

Publications

[1] H-V. To, B. Le, and M. Ikeda: "Applying Hierarchical Information with Learning Approach for Activity Recognition", Proceedings of the International Symposium on Integrated Uncertainty In Knowledge Modeling and Decision Making 2011 (IUKM 2011), pp. 231-242, 2011.

[2] H-V. To, B. Le, and M. Ikeda: "On the Semantics of Defeasible Reasoning for Description Logic Ontologies", Proceedings of the Knowledge and Systems Engineering (KSE 2013), pp. 51-63, 2013.

[3] H-V. To, B. Le, and M. Ikeda: "An Ontological Approach for Representation of Educational Program", International Journal of Knowledge and Systems Science, Volume 5, Issue 3, 2014 (to appear).

[4] H-V. To, B. Le, and M. Ikeda: "Defeasible Reasoning Approach for Reasoning with Inconsistent Ontology", submitted to International Journal of Knowledge and Systems Science, 2014.

Bibliography

[1] Dean Allemang and James A. Hendler. *Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL, Second Edition.* Elsevier, 2011.

[2] Grigoris Antoniou. Nonmonotonic rule systems on top of ontology layers. In *Proceedings of the First International Semantic Web Conference on The Semantic Web*, ISWC '02, pages 394–398, London, UK, 2002. Springer-Verlag.

[3] Grigoris Antoniou and Antonis Bikakis. Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web. *IEEE Transactions on Knowledge and Data Engineering*, 19:233–245, 2007.

[4] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Representation results for defeasible logic. *ACM Trans. Comput. Logic*, 2(2):255–287, 2001.

[5] Grigoris Antoniou, David Billington, Guido Governatori, and Michael J. Maher. Embedding defeasible logic into logic programming. *Theory Pract. Log. Program.*, 6(6):703–735, November 2006.

[6] Grigoris Antoniou, Michael J. Maher, and David Billington. Defeasible logic versus logic programming without negation as failure. *The Journal of Logic Programming*, 42(1):47 – 57, 2000.

[7] Grigoris Antoniou and Frank van Harmelen. *A Semantic Web Primer.* MIT Press, Cambridge, MA, 2 edition, 2008.

[8] Grigoris Antoniou and Gerd Wagner. Rules and defeasible reasoning on the semantic web. In Michael Schröder and Gerd Wagner, editors, *Rules and Rule Markup Languages for the Semantic Web*, volume 2876 of *Lecture Notes in Computer Science*, pages 111–120. Springer Berlin, Heidelberg, 2003.

[9] Lora Aroyo and Darina Dicheva. The new challenges for e-learning: The educational semantic web. *J. Educational Technology & Society*, 7(4):59–69, 2004.

[10] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.

[11] Franz Baader, Ian Horrocks, and Ulrike Sattle. Description logics. In *Handbook of Knowledge Representation*, chapter 3, pages 135–179. Elsevier, 2008.

[12] Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.

[13] Nick Bassiliades, Grigoris Antoniou, and Ioannis Vlahavas. A defeasible logic reasoner for the semantic web. *International Journal of Semantic Web and Information Systems 2*, 1:1–41, 2006.

[14] Sean Bechhofer, Ian Horrocks, and Peter Patel-Schneider. Tutorial on owl, 2003. `http://www.cs.man.ac.uk/ horrocks/ISWC2003/Tutorial/`.

[15] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[16] David Billington. Defeasible logic is stable. *Journal of Logic and Computation*, pages 379–400, 1993.

[17] David Billington, Grigoris Antoniou, Guido Governatori, and Michael Maher. An inclusion theorem for defeasible logics. *ACM Trans. Comput. Logic*, 12(1):1–27, 2010.

[18] Fernando Bobillo and Umberto Straccia. Fuzzy ontology representation using owl 2. *Int. J. Approx. Reasoning 52*, 7:1073–1094, 2011.

[19] Dan Brickley and R.V Guha. Rdf schema 1.1. `http://www.w3.org/TR/rdf-schema/`, 2014.

[20] Katarina Britz, Thomas Meyer, and Ivan Varzinczak. Semantic foundation for preferential description logics. In *AI 2011: Advances in Artificial Intelligence*, pages 491–500. Springer, 2011.

[21] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Daml+oil (march 2001) reference description. `http://www.w3.org/TR/owl2-overview/`, 2001.

[22] Paulo C. G. Costa and Kathryn B. Laskey. Pr-owl: A framework for probabilistic ontologies. In *Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*, pages 237–249, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

[23] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. Owl 2: The next step for owl. *Journal of Web Semantics*, 6(4):309–322, 2008.

[24] Mike Dean and Guus Schreiber. Owl web ontology language reference. W3C Working Draft. `http://www.w3.org/TR/owl-ref/`, 2003.

[25] Vladan Devedzic. Web intelligence and artificial intelligence in education. *J. Educational Technology & Society*, 7(4):29–39, 2004.

[26] Darina Dicheva. Ontologies and semantic web for e-learning. In HeimoH. Adelsberger, Kinshuk, JanM. Pawlowski, and DemetriosG. Sampson, editors, *Handbook on Information Technologies for Education and Training*, International Handbooks on Information Systems, pages 47–65. Springer Berlin Heidelberg, 2008.

[27] Zhongli Ding, Yun Peng, Rong Pan, Zhongli Ding, Yun Peng, and Rong Pan. Bayesowl: Uncertainty modeling in semantic web ontologies. In *Soft Computing in Ontologies and Semantic Web, volume 204 of Studies in Fuzziness and Soft Computing*. Springer, 2005.

[28] Didier Dubois, Jérôme Lang, and Henri Prade. Possibilistic logic. In *Handbook of Logic in Artificial Inteligence and Logic Programming*, pages 439–513. Oxford University Press, 1994.

[29] 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.

[30] Michael Dürig and Thomas Studer. Probabilistic abox reasoning: Preliminary results. In Ian Horrocks, Ulrike Sattler, and Frank Wolter, editors, *Proceedings DL 2005*, volume 147 of *CEUR Workshop Proceedings*, pages 104–111, 2005.

[31] Amira Essaid and Boutheina Ben Yaghlane. Beliefowl: an evidential representation in owl ontology. In *Proceedings of the Fifth International Workshop on Uncertainty Reasoning for the Semantic Web (URSW2009)*, pages 77–80, 2009.

[32] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[33] Adam Farquhar, Richard Fikes, and James Rice. The ontolingua server: a tool for collaborative ontology construction. In *International Journal of Human-Computer Studies*, volume 46, pages 707–727, 1997.

[34] Dieter Fensel, Ian Horrocks, Frank Van Harmelen, Deborah McGuinness, and Peter F. Patel-Schneider. Oil: Ontology infrastructure to enable the semantic web. *IEEE Intelligent Systems*, 16:200–221, 2001.

[35] Fabian Gandon and Guus Schreiber. Rdf 1.1 xml syntax. `http://www.w3.org/TR/rdf-syntax-grammar/`, 2014.

[36] Michael Genesereth, Richard E. Fikes, Ronald Brachman, Thomas Gruber, Patrick Hayes, Reed Letsinger, Vladimir Lifschitz, Robert Macgregor, John Mccarthy, Peter Norvig, and Ramesh Patil. Knowledge interchange format version 3.0 reference manual. Technical report, Computer Science Department, Stanford University, 1992.

[37] Birte Glimm, Ian Horrocks, Boris Motik, and Giorgos Stoilos. Optimising ontology classification. In *Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I*, ISWC'10, pages 225–240, Berlin, Heidelberg, 2010. Springer-Verlag.

[38] Sergio Alejandro Gómez, Carlos Iván Chesñevar, and Guillermo Ricardo Simari. Reasoning with inconsistent ontologies through argumentation. *Applied Artificial Intelligence*, 24(1&2):102–148, 2010.

[39] Asunción Gómez-Pérez, Mariano Fernández-Lopez, and Oscar Corcho, editors. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web (2nd edition)*. Springer-Verlag, 2007.

[40] Guido Governatori. Defeasible description logics. In *Rules and Rule Markup Languages for the Semantic Web*, number 3323 in LNCS, pages 98–112, 2004.

[41] Guido Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14:181–216, 2005.

[42] Guido Governatori, Micheal J. Maher, Grigoris Antoniou, and David Billington. Argumentation semantics for defeasible logic. *Journal of Logic and Computation*, 14:675–702, 2004.

[43] Guido Governatori and Antonino Rotolo. Bio logical agents: Norms, beliefs, intentions in defeasible logic. *Autonomous Agents and Multi-Agent Systems*, 17(1):36–69, 2008.

[44] Guido Governatori, Antonino Rotolo, and Giovanni Sartor. Temporalised normative positions in defeasible logic. In *10th International Conference on Artificial Intelligence and Law (ICAIL2005)*, pages 25–34. ACM, 2005.

[45] Benjamin N. Grosof. Prioritized conflict handling for logic programs. In *Proceedings of the 1997 International Logic Programming Symposium*, pages 197–211. MIT Press, 1997.

[46] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[47] Nicola Guarino. Formal ontology and information systems. In *International Conference on Formal Ontology in Information Systems (FOIS'98)*, pages 3–15, Amsterdam, 1998. IOS Press.

[48] Weisen Guo and Steven B. Kraines. Semcl: A cross-language semantic model for knowledge sharing. *Int. J. Knowl. Syst. Sci.*, 1(3):1–19, July 2010.

[49] Jochen Heinsohn. Probabilistic description logics. In *Proceedings UAI-1994*, pages 311–318, 1994.

[50] S. Heymans and Dirk Vermeir. A defeasible ontology language. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002*, pages 1033–1046, London, UK, 2002. Springer-Verlag.

[51] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semantic Web*, 2(1):11–21, 2011.

[52] Matthew Horridge, Nick Drummond, John Goodwin, Alan Rector, and Hai H Wang. The manchester owl syntax. In *Proc. of the 2006 OWL Experiences and Directions Workshop (OWL-ED2006)*, 2006.

[53] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining inconsistencies in owl ontologies. In Llus Godo and Andrea Pugliese, editors, *Scalable Uncertainty Management*, volume 5785 of *Lecture Notes in Computer Science*, pages 124–137. Springer Berlin Heidelberg, 2009.

[54] Ian Horrocks. Ontologies and the semantic web. *Communications of the ACM*, 51(12):58–67, 2008.

[55] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI'05, pages 448–453, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[56] Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 454–459, Edinburgh, Scotland, 2005.

[57] Eero Hyvönen. Preventing ontology interoperability problems instead of solving them. *Journal of Semantic Web*, 1(1,2):33–37, April 2010.

[58] Manfred Jaeger. Probabilistic reasoning in terminological logics. In *Proceedings of KR-1994*, pages 305–316. Morgan Kaufmann, 1994.

[59] Manfred Jaeger. Probabilistic role models and the guarded fragment. In *Proceedings IPMU-2004*, pages 235–242, 2004.

[60] Aditya Kalyanpur, Bijan Parsia, Matthew Horridge, and Evren Sirin. Finding all justifications of owl dl entailments. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudr-Mauroux, editors, *Proceedings of 6th International and 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer Berlin Heidelberg, 2007.

[61] Norihiro Kamide. A comparison of paraconsistent description logics. *International Journal of Intelligence Science*, 3(2):99–109, 2013.

[62] Yevgeny Kazakov. An extension of complex role inclusion axioms in the description logic $\mathcal{SROIQ}$. In *Automated Reasoning*, pages 472–486. Springer, 2010.

[63] Mohammad Khodadadi, RenateA. Schmidt, and Dmitry Tishkovsky. A refined tableau calculus with controlled blocking for the description logic $\mathcal{SHOI}$. In Didier Galmiche and Dominique Larchey-Wendling, editors, *Automated Reasoning with Analytic Tableaux and Related Methods*, volume 8123 of *Lecture Notes in Computer Science*, pages 188–202. Springer Berlin Heidelberg, 2013.

[64] Je-Min Kim, Soon-Hyen Kwon, and Young-Tack Park. Enhanced search method for ontology classification. *Computing and Informatics*, 28(6):795–809, 2009.

[65] Patrick Koopmann and Renate A. Schmidt. Forgetting concept and role symbols in $\mathcal{ALCH}$-ontologies. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning*, volume 8312 of *Lecture Notes in Computer Science*, pages 552–567. Springer Berlin Heidelberg, 2013.

[66] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *CoRR*, abs/1201.4089, 2013.

[67] Ho-Pun Lam and Guido Governatori. The making of spindle. In *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, RuleML '09, pages 315–322, Berlin, Heidelberg, 2009. Springer-Verlag.

[68] Ho-Pun Lam and Guido Governatori. On the problem of computing ambiguity propagation and well-founded semantics in defeasible logic. In Mike Dean, John Hall, Antonino Rotolo, and Said Tabet, editors, *Semantic Web Rules*, volume 6403 of *Lecture Notes in Computer Science*, pages 119–127. Springer Berlin Heidelberg, 2010.

[69] Daniel Lehmann. Another perspective on default reasoning. *Annals of Mathematics and Artificial Intelligence*, 15(1):61–82, 1995.

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

[71] Domenico Lembo, Valerio Santarelli, and Domenico Fabio Savo. Graph-based ontology classification in owl 2 ql. In *The Semantic Web: Semantics and Big Data*, pages 320–334. Springer, 2013.

[72] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[73] Paea LePendu, Dejing Dou, and Doug Howe. Detecting inconsistencies in the gene ontology using ontology databases with not-gadgets. In *On the Move to Meaningful Internet Systems: OTM 2009*, volume 5871 of *Lecture Notes in Computer Science*, pages 948–965. Springer Berlin Heidelberg, 2009.

[74] Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852 – 883, 2008.

[75] Thomas Lukasiewicz and Umberto Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):291–308, 2008.

[76] Jenny Eriksson Lundström, Guido Governatori, Subhasis Thakur, and Vineet Padmanabhan. An asymmetric protocol for argumentation games in defeasible logic. In *In 10th Pacific Rim International Conference on Multi-Agents (PRIMA 2007)*, pages 219–231. Springer, 2007.

[77] M. J. Maher, A. Rock, G. Antoniou, D. Billington, and T. Miller. Efficient defeasible reasoning systems. In *Tools with Artificial Intelligence, 2000. ICTAI 2000. Proceedings. 12th IEEE International Conference on*, pages 384 –392, 2000.

[78] Michael J. Maher. Propositional defeasible logic has linear complexity. *Theory Pract. Log. Program.*, 1(6):691–711, 2001.

[79] Michael J. Maher. Relative expressiveness of defeasible logics. *Theory and Practice of Logic Programming*, 12:793–810, 2012.

[80] Frederick Maier and Donald Nute. Ambiguity propagating defeasible logic and the well-founded semantics. In Michael Fisher, Wiebe Hoek, Boris Konev, and Alexei Lisitsa, editors, *Logics in Artificial Intelligence*, volume 4160 of *Lecture Notes in Computer Science*, pages 306–318. Springer Berlin Heidelberg, 2006.

[81] Deborah L. McGuinness, Richard Fikes, Lynn Andrea Stein, and James A. Hendler. Daml-ont: An ontology language for the semantic web. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pages 65–94. The MIT Press, Cambridge, Massachusetts, 2003.

[82] Prasenjit Mitra, Natalya F. Noy, and Anuj R. Jaiswal. Omen: A probabilistic ontology mapping tool. In *In Workshop on Meaning Coordination and Negotiation at the Third International Conference on the Semantic Web (ISWC-2004). Hisroshima*, pages 537–547, 2004.

[83] Riichiro Mizoguchi and Jacqueline Bourdeau. Using ontological engineering to overcome common ai-ed problems. *International Journal of Artificial Intelligence in Education*, 11(2):107–121, 2000.

[84] Kody Moodley, Thomas Meyer, and Ivan José Varzinczak. A defeasible reasoning approach for description logic ontologies. In *Proceedings of the SAICSIT '12*, pages 69–78, New York, NY, USA, 2012. ACM.

[85] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau reasoning for description logics. *J. of Artificial Intelligence Research*, 36:165–228, 2009.

[86] NgocThanh Nguyen. Conflicts of ontologies – classification and consensus-based methods for resolving. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4252 of *Lecture Notes in Computer Science*, pages 267–274. Springer Berlin Heidelberg, 2006.

[87] Ikujirō Nonaka and Hirotaka Takeuchi. *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.

[88] Donald Nute. Defeasible logic. In *Handbook of Logic in Artificial Inteligence and Logic Programming*, pages 353–395. Oxford University Press, 1987.

[89] Donald Nute. Defeasible logic. In *Web Knowledge Management and Decision Support*, pages 151–169. Springer, 2003.

[90] Rong Pan, Zhongli Ding, Yang Yu, and Yun Peng. A bayesian network approach to ontology mapping. In Yolanda Gil, Enrico Motta, V.Richard Benjamins, and MarkA. Musen, editors, *The Semantic Web ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 563–577. Springer Berlin Heidelberg, 2005.

[91] Pakornpong Pothipruk and Guido Governatori. $\mathcal{ALE}$ defeasible description logic. In *AI 2006: Advances in Artificial Intelligence*, pages 110–119. Springer, 2006.

[92] Henry Prakken and Giovanni Sartor. Formalising arguments about the burden of persuasion. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, ICAIL '07, pages 97–106, New York, NY, USA, 2007. ACM.

[93] Graham Priest, Koji Tanaka, and Zach Weber. Paraconsistent logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2013 edition, 2013.

[94] Guilin Qi and Jianfeng Du. Reasoning with uncertain and inconsistent owl ontologies. In Thomas Eiter and Thomas Krennwallner, editors, *Reasoning Web. Semantic Technologies for Advanced Query Answering*, volume 7487 of *Lecture Notes in Computer Science*, pages 211–244. Springer Berlin Heidelberg, 2012.

[95] Guilin Qi, Qiu Ji, Jeff Z. Pan, and Jianfeng Du. Extending description logics with uncertainty reasoning in possibilistic logic. *International Journal of Intelligent Systems*, 26(4):353–381, 2011.

[96] Guilin Qi and Jeff Pan. A tableau algorithm for possibilistic description logic $\mathcal{ALC}+$. In Diego Calvanese and Georg Lausen, editors, *Web Reasoning and Rule Systems*, volume 5341 of *Lecture Notes in Computer Science*, pages 238–239. Springer Berlin / Heidelberg, 2008.

[97] Guilin Qi, Jeff Pan, and Qiu Ji. Extending description logics with uncertainty reasoning in possibilistic logic. In Khaled Mellouli, editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 4724 of *Lecture Notes in Computer Science*, pages 828–839. Springer Berlin / Heidelberg, 2007.

[98] Daniel M. Reeves, Michael P. Wellman, and Benjamin N. Grosof. Automated negotiation from declarative contract descriptions. *Computational Intelligence*, 18:482–500, 2002.

[99] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

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

[101] Vincent Risch and Camilla Schwind. Tableaux-based characterization and theorem proving for default logic. *J. Autom. Reasoning*, pages 223–242, 1994.

[102] Andrew Rock. Deimos: Query answering defeasible logic system, 2000. `http://www.ict.griffith.edu.au/arock/defeasible/Defeasible.cgi`.

[103] Stefan Schlobach. Debugging and semantic clarification by pinpointing. In *The Semantic Web: Research and Applications*, volume 3532 of *Lecture Notes in Computer Science*, pages 226–240. Springer Berlin Heidelberg, 2005.

[104] Manfred Schmidt-Schauss and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[105] Guus Schreiber and Yves Raimond. Rdf 1.1 primer. `http://www.w3.org/TR/rdf11-primer/`, 2014.

[106] Glenn Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, 1976.

[107] M. Sharples, N. Jeffery, J. B. H. Du Boulay, D. Teather, B. Teather, and G. H. Du Boulay. Socio-cognitive engineering: a methodology for the design of human-centered technology. *European Journal of Operational Research*, 132:310–323, 2002.

[108] Guillermo R. Simari and Ronald P. Loui. A mathematical treatment of defeasible reasoning and its implementation. *Artificial Intelligence*, 53(23):125–157, 1992.

[109] Michael Smith, Ian Horrocks, Markus Krötzsch, Birte Glimm, and eds. Owl 2 web ontology language: Conformance (second edition). `http://www.w3.org/TR/owl2-overview/`, 2012.

[110] Giorgos Stoilos, Birte Glimm, Ian Horrocks, Boris Motik, and Rob Shearer. A novel approach to ontology classification. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14:84–101, 2012.

[111] Rudi Studer, V.Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161 – 197, 1998.

[112] Paul Thagard. Cognitive science. The Stanford Encyclopedia of Philosophy (Fall 2008 Edition), 2008.

[113] Stephan Tobies. The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. *J. Artif. Int. Res.*, 12(1):199–217, 2000.

[114] Hai Wang, Matthew Horridge, Alan Rector, Nick Drummond, and Julian Seidenberg. Debugging owl-dl ontologies: A heuristic approach. In *The Semantic Web–ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 745–757. Springer, 2005.

[115] Premyłsaw Więch. *Distributed Default Reasoning in Semantic Web.* PhD thesis, Warsaw University of Technology, 2011.

[116] Yi Yang and Jacques Calmet. Ontobayes: An ontology-driven uncertainty model. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06) - Volume 01*, CIMCA '05, pages 457–463, Washington, DC, USA, 2005. IEEE Computer Society.

[117] Xiaowang Zhang, Zuoquan Lin, and Kewen Wang. A tableau algorithm for paraconsistent and nonmonotonic reasoning in description logic-based system. In Xiaoyong Du, Wenfei Fan, Jianmin Wang, Zhiyong Peng, and MohamedA. Sharaf, editors, *Web Technologies and Applications*, volume 6612 of *Lecture Notes in Computer Science*, pages 345–356. Springer Berlin Heidelberg, 2011.

[118] Xiaowang Zhang, Guohui Xiao, and Zuoquan Lin. A tableau algorithm for handling inconsistency in owl. In *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ESWC 2009 Heraklion, pages 399–413, Berlin, Heidelberg, 2009. Springer-Verlag.